



CONNECTED VEHICLE SECURITY METRICS AND THREAT INTELLIGENCE

BED34-977-01

FINAL REPORT

May 2024

Guillermo A. Francia, III
Hossain Shahriar
University of West Florida

Disclaimer

The opinions, findings, and conclusions expressed in this publication are those of the authors and not necessarily those of the Florida Department of Transportation.

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.	
4. Title and Subtitle Connected Vehicle Security Metrics and Threat Intelligence		5. Report Date May 2024	
		6. Performing Organization Code	
7. Author(s) Guillermo Francia, III, Ph.D. Hossain Shahriar, Ph.D.		8. Performing Organization Report No.	
9. Performing Organization Name and Address University of West Florida 11000 University Pkwy Pensacola, FL 32514		10. Work Unit No. (TRAIS)	
		11. Contract or Grant No. BED34-977-01	
12. Sponsoring Agency Name and Address Florida Department of Transportation 605 Suwannee Street, MS 30 Tallahassee, FL 32399		13. Type of Report and Period Covered Final Report 06/01/2024--5/31/2024	
		14. Sponsoring Agency Code	
15. Supplementary Notes			
16. Abstract <p>The sophistication of vehicle communication systems, found externally and internally, enables the expansion of the attack surface on connected and autonomous vehicles (CAVs). For that reason, we embarked on this three-pronged approach that has major implications for the safety and security of CAVs. Firstly, recognizing the need for vehicle security metrics and their continuous improvement, we designed applicable security metrics and developed their corresponding visualization systems. Secondly, with the emergence of machine learning (ML) systems, the opportunity to enhance and advance the security of connected vehicle messaging systems using artificial intelligence becomes more efficient and plausible. We explored this prospect by developing prototype ML systems utilizing data collected from Roadside Units (RSUs) as input. Finally, with the overarching goal of promoting vehicle safety, this project offers a seminal work on collecting, processing, and storing vehicle threat intelligence. The principal objective of such a system is to enable immediate access to vehicle vulnerability information to researchers, operators, manufacturers, supply chain, and the public in general. This enabler provides proactive means to mitigate risks and thereby promotes vehicle and personal safety.</p>			
17. Key Word Connected Vehicles, Cyber Security, Threat Intelligence, Security Metrics, Visualization, Vulnerability, Risk		18. Distribution Statement No restrictions.	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 393	22. Price

Form DOT F 1700.7 (8-72)

Executive Summary

The rapid advancement of connected and autonomous vehicles (CAVs) created new challenges for security and safety professionals. The sophistication of vehicle communication systems, found externally and internally, provides an added complexity to the issue. In security parlance, this is an expansion of the attack surface on vehicles. These challenges prompted the enhancement of existing, and the development of new, safety and security standards initiated by government, industry, and trade organizations. These initiatives clearly underscore the need to examine the state of connected vehicle security. For that reason, we embarked on this three-pronged approach that has major implications for the safety and security of connected vehicles. Firstly, we recognize the need for the development of vehicle security metrics and their continuous improvement. As a major component of continuous improvement, quantitative and qualitative measures must be devised to be able to fully appreciate the process. Consequently, we designed applicable security metrics and developed their corresponding visualization systems. Secondly, with the emergence of machine learning (ML) systems, the opportunity to enhance and advance the security of connected vehicle messaging system using artificial intelligence becomes more efficient and plausible. We explored this prospect by developing prototype ML systems utilizing data collected from Roadside Units (RSUs) as input. Finally, with the overarching goal of promoting vehicle safety, this project offers a seminal work on collecting, processing, and storing vehicle threat intelligence. The principal objective of such a system is to enable immediate access to vehicle vulnerability information to researchers, operators, manufacturers, supply chain, and the public in general. This enabler provides proactive means to mitigate risks and thereby promotes vehicle and personal safety.

Table of Contents

Disclaimer	ii
Technical Report Documentation Page.....	iii
Executive Summary	iv
List of Tables	viii
List of Figures.....	viii
List of Abbreviations and Acronyms.....	x
Chapter 1. Introduction.....	1
1.1 <i>Project Objectives</i>	1
1.2 <i>Major Accomplishments</i>	1
Chapter 2. Development of Vehicle Security Metrics.....	3
2.1. <i>Common Vulnerability Scoring System (CVSS)</i>	3
2.2. <i>Common Methodology for IT Security Evaluation (CEM)</i>	3
2.3. <i>Threats on Assets</i>	3
2.4. <i>Common Weakness Scoring System</i>	4
2.5. <i>Operational Safety Assessment Metrics</i>	4
2.5.1 <i>Cybersecurity Metrics for Operational Safety</i>	4
2.6. <i>Security Vulnerability Metrics for Connected Vehicles</i>	6
2.6.1 <i>Electronic Control Unit (ECU) Coupling Risk</i>	6
2.6.2 <i>Communication Channel Risk</i>	6
2.6.3 <i>Complexity Risk</i>	7
2.6.4 <i>Input and Output Data Risk</i>	7
2.6.5 <i>History of Security Issues</i>	7
2.6.6 <i>Overall Security Vulnerability Metric</i>	8
2.7. <i>Vehicle Security Best Practices Assessment Metrics</i>	8
2.7.1. <i>Vehicle Security Best Practices Assessment Metrics</i>	8
Chapter 3. Design and Implementation of the Vehicle Security Metrics Visualization System (VSMVS)	11
3.1. <i>Visualization of CVSS Vector and Metrics</i>	11
3.2. <i>Visualization of Attack Potential of Threats on Vehicle Assets</i>	13
3.3. <i>Visualization of Common Weakness Scoring System</i>	14
3.4. <i>Visualization of Security Metrics for Operational Safety</i>	15
3.5. <i>Visualization of Security Vulnerability Metrics</i>	16
3.6. <i>Visualization of Vehicle Security Best Practices Assessment Metrics</i>	18
Chapter 4. Testing and Deployment of the VSMVS	20
4.1. <i>Test Framework Description</i>	20

4.2. VSMVS Unit Tests	20
4.2.1. TreeMap Tests	20
4.2.2. Common Weakness Scoring System (CWSS) Tests	20
4.2.3. Security Metrics Operational Safety (SMOS) Tests	20
4.2.4. Security Vulnerability Metrics for Connected Vehicles (SVMCV) Tests	20
4.2.5. Vehicle Security Best Practices Assessment Metrics (VSBPAM) Tests	20
Chapter 5. Data Ingestion, Cleansing, Normalization, and Data Mutation of Roadside Unit Data	21
5.1. Roadside Unit Dataset.....	21
5.2. Basic Safety Message (BSM)	21
5.3. Synthetic Dataset	22
5.4. Malicious Dataset.....	23
5.4.1. Brake System Anomaly	23
5.4.2. Transmission System Anomaly	23
5.4.3. Longitudinal Acceleration Anomaly	24
5.4.4. Hard Braking Anomaly.....	24
5.4.5. Speed Anomaly	24
Chapter 6. Design and Implementation of a Machine Learning System for Vehicle Security	25
Chapter 7. Design and Implementation of a Vehicle Threat Modeling Engine (VTME).....	26
Chapter 8. Design and Implementation of a Vehicle Threat Collection System (VTCS).....	28
8.1. The VTCS Interfaces	28
Chapter 9. Design and Implementation of a Vehicle Threat Database System (VTDS)	30
9.1. Entity Relationship (ER) Diagrams.....	30
Chapter 10. Design and Implementation of a Vehicle Threat Information Portal (VTIP).....	32
10.1. VTIP Threat Record Interface	32
10.2. VTIP Threat Record Search and Edit Interface	32
Chapter 11. System Integration, Testing, and Deployment	34
11.1 The VSSI Interface.....	34
Chapter 12. Continuous Improvement Process.....	36
12.1 List of Improvements and Rationale.....	36
Chapter 13. Conclusion and Future Works.....	37
References	38
Appendices	41
Appendix I. Technical Report UWF-TR-FDOT-001-01.....	42
Appendix II. Technical Report UWF-TR-FDOT-002-01	63
Appendix III. Technical Report UWF-TR-FDOT-002-02	102

Appendix IV. Technical Report UWF-TR-FDOT-003-01	140
Appendix V. BSM Dataset Attributes	145
Appendix VI. Technical Report UWF-TR-FDOT-003-02	149
Appendix VII. Technical Report UWF-TR-FDOT-004-01	180
Appendix VIII. Technical Report UWF-TR-FDOT-005-01	199
Appendix IX. Technical Report UWF-TR-FDOT-006-01	229
Appendix X. Technical Report UWF-TR-FDOT-006-02	248
Appendix XI. Technical Report UWF-TR-FDOT-006-03	252
Appendix XII. Technical Report UWF-TR-FDOT-007-01	258
Appendix XIII. Technical Report UWF-TR-FDOT-007-02	279
Appendix XIV. Technical Report UWF-TR-FDOT-007-03	284
Appendix XV. Technical Report UWF-TR-FDOT-008-01	287
Appendix XVI. Technical Report UWF-TR-FDOT-008-02	309
Appendix XVII. Technical Report UWF-TR-FDOT-008-03	312
Appendix XVIII. Technical Report UWF-TR-FDOT-009-01	319
Appendix XIX. Technical Report UWF-TR-FDOT-009-02	345
Appendix XX. Technical Report UWF-TR-FDOT-009-03	349
Appendix XXI. Technical Report UWF-TR-FDOT-010-01	357
Appendix XXII. Technical Report UWF-TR-FDOT-010-02	371
Appendix XXIII. Data Processing Algorithm	383
<i>Algorithm 1: RSU Data extraction and Cleansing</i>	383
Appendix XXIV. Continuous Integration/Continuous Deployment Pipeline	385
<i>Pipeline Steps</i>	385
<i>Self-Hosted Runner</i>	386
<i>GitHub Actions Runners</i>	386
<i>Setting Up the GitHub Actions Self-hosted Runner</i>	386
What you will need:.....	386
Setup Procedure	387
<i>Configuring the SQL Server</i>	389
<i>Running the Pipeline</i>	392

List of Tables

Table 1: Vehicle Security Best Practices Assessment Checklist 9
Table 2: BSM Dataset Attributes 25
Table 3: Summary of ML Model Validation and Testing Results..... 25
Table 4: BSM Coredata Characteristics..... 145

List of Figures

Figure 1: The Connected Vehicle Security System Architecture 2
Figure 2: The Landing Page of the VSMVS..... 11
Figure 3: The CVSS Vector Interface for the VSMVS 12
Figure 4: The CVSS Vector Scoring System..... 12
Figure 5: The Attack Potential of Threats on Vehicle Assets..... 13
Figure 6: The Tree Map Visualization of Vehicle Attack Potential 13
Figure 7: The Common Weakness Scoring System User Interface..... 14
Figure 8: The Visualization for the Common Weakness Scoring System Metrics 15
Figure 9: Data Entry Interface for Security Metrics for Operational Safety 15
Figure 10: Calculated Security Metrics for Operational Safety 16
Figure 11: Data Entry Interface for Security Vulnerability Metrics 17
Figure 12: Calculated Security Vulnerability Metrics 17
Figure 13: Vehicle Security Best Practices Assessment Metrics..... 18
Figure 14: Lockheed Martin’s Cyber Kill Chain 26
Figure 15: Query Page Viewer Prototype 28
Figure 16: Prototype VTCS Record Viewer Interface..... 29
Figure 17: Entity Relationship (ER) Diagram for the VTDS 30
Figure 18: ER Diagram for the User and Transaction Log Entities 30
Figure 19: VTIP Interface 32
Figure 20: VTIP Threat Record Search and Edit Interface..... 33
Figure 21: The VSSI Landing Page Interface 34
Figure 22: The Workflow 385
Figure 23: Runner Setup 386
Figure 24: Creating New Self-hosted Runner..... 387
Figure 25: Windows PowerShell 387
Figure 26: New Self-hosted Runner 388
Figure 27: Log-on Configuration..... 388
Figure 28: Configuring the SQL Server..... 389
Figure 29: Configuring the SQL Server Authentication..... 390
Figure 30: Creating a New Login Account..... 390
Figure 31: Setting Authentication Properties..... 391
Figure 32: Setting Server Roles 392
Figure 33: Manually Running a Pipeline 392
Figure 34: Running a Pipeline Automatically 393
Figure 35: Action Secrets and Variables 393

List of Abbreviations and Acronyms

API	Application Program Interface
ATT&CK	Adversarial Tactics, Techniques, and Common Knowledge
AWS	Amazon Web Services
BSM	Basic Set Message
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
CWE	Common Weakness Enumeration
EC2	Elastic Compute Cloud
ER	Entity Relationship
GUI	Graphical User Interface
IIS	Internet Information Services
IoC	Indicators of Compromise
ML	Machine Learning
.NET	A cross-platform, open-source developer platform created by Microsoft
OSINT	Open-Source Intelligence
OTX	Open Threat Exchange
TTPs	Tactics, Techniques and Procedures
VSMVS	Vehicle Security Metrics Visualization System
VSSI	Vehicle Security System Integration
VTCS	Vehicle Threat Collection System
VTDS	Vehicle Threat Database System
VTIP	Vehicle Threat Intelligence Portal
VTME	Vehicle Threat Modeling Engine

Chapter 1. Introduction

The emergence of connected and autonomous vehicles at an unprecedented pace ushered in several government-sponsored initiatives to start planning and building a transportation information network that utilizes intelligent sensors and sophisticated communication systems. Peripheral sensors that are used to assist the human operator in lane changing, obstacle avoidance, and parking are slowly being integrated in modern automotive vehicles. Although this newly found convenience is a boon to society, both socially and economically, it presents security challenges that are endemic to connected technologies. These challenges underscore the need to examine the state of connected vehicle security and design an effective threat intelligence portal.

1.1 Project Objectives

The objectives of the proposed project are as follows:

1. Design and develop a comprehensive set of security metrics for connected vehicles;
2. Implement a prototype machine learning (ML) system that will process sensor data for intelligent security analytics; and
3. Create a Web-based threat intelligence portal for connected vehicles.

1.2 Major Accomplishments

The major accomplishments of the project were delineated by the tasks and the associated deliverables for each task. These tasks enable the realization of all the project objectives as stipulated in the above.

Figure 1 illustrates the relationships and interactions of the major components of the project. The interactions between components are as follow:

- *VTCS-VirusTotal*. The Vehicle Threat Collection System (VTCS) queries the VirusTotal repository for specific vehicle threat intelligence through the system Application Program Interface (API). The threat data that were returned by the query are deposited into the Vehicle Threat Database System (VTDS).
- *VTCS-OTX*. The VTCS queries the Open Threat Exchange (OTX) for specific vehicle threat intelligence through the system API. Similarly, the data that were gathered are deposited in the VTDS.
- *VTCS-CVE*. The VTCS queries the Common Vulnerability Enumeration (CVE) web portal through the system API. Likewise, the data that were gathered are deposited in the VTDS.
- *VTIP-VTDS*. The Vehicle Threat Information Portal (VTIP) is the main user interface to the VTDS. A user may retrieve any available threat information that pertains to a specific vehicle using the interface. The VTDS responds appropriately to the query depending on the availability of threat information that was collected by the VTCS.
- *VTIP-MISP*. The VTIP provides an interface with the Malware Information Sharing Platform (MISP) server for a sharing, storing and correlating Indicators of Compromise of targeted attacks, threat intelligence, and vulnerability information on vehicles. The currently installed MISP server is a proof-of-concept implementation.

- *VTDS-VSMVS*. The interaction between the VTDS and the Vehicle Security Metrics Visualization System (VSMVS) is not implemented due to the incompatibility of the data objects representing threats and vulnerability.
- *VTDS-VSML*. The interaction between the VTDS and the Vehicle Security Machine Learning (VSML) system is not implemented due to the incompatibility of the data objects representing threats and the SAE J2735 message format (Basic Safety Message (BSM)).

Each component is fully described in the task subsection in which it is closely associated with.

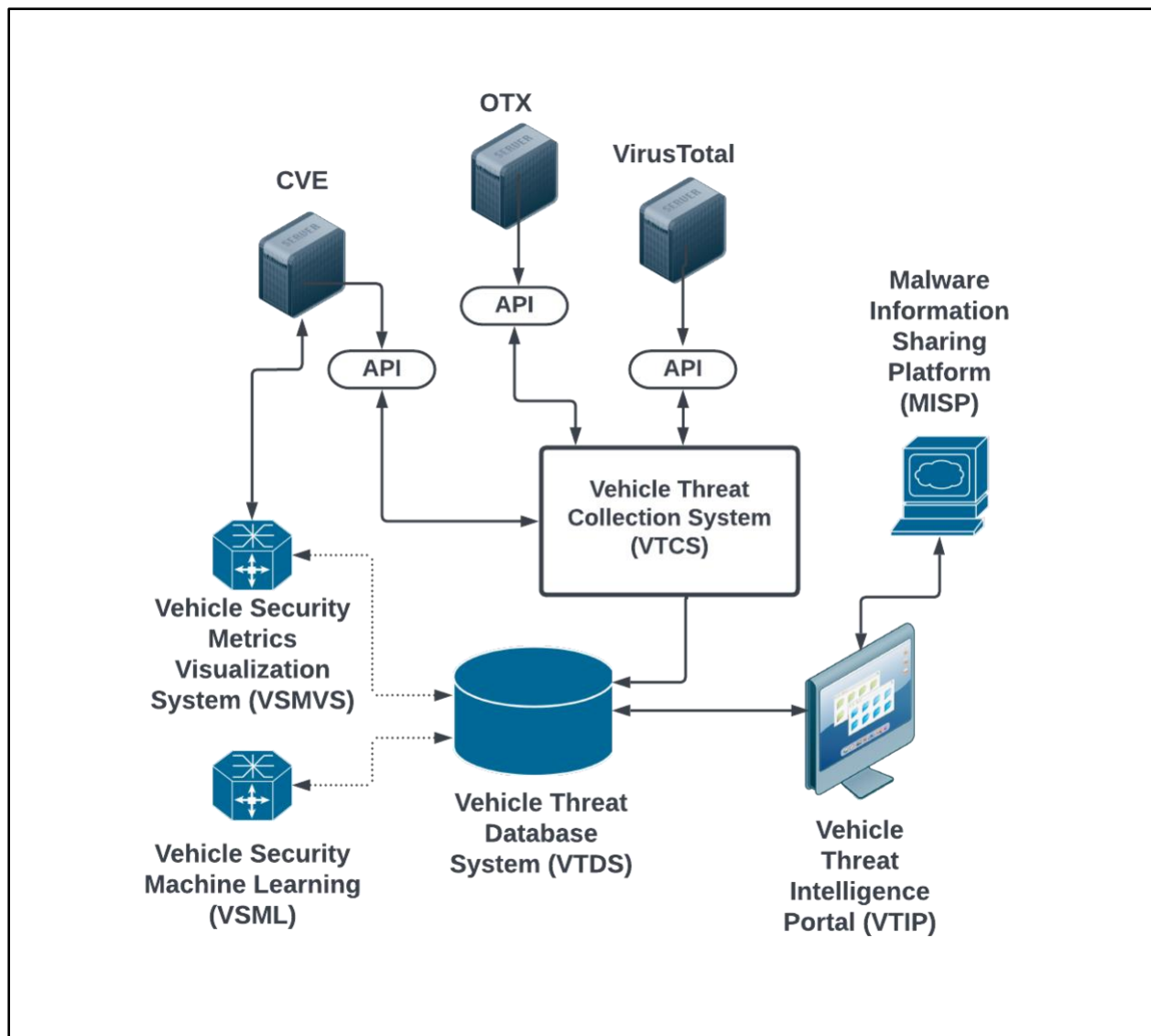


Figure 1: The Connected Vehicle Security System Architecture

Chapter 2. Development of Vehicle Security Metrics

The first task of the project called for the research and development vehicle security metrics based on published literature and standards. The purpose of developing security metrics is for continuous improvement, i.e. any improvement process needs an established benchmark.

The following section is an excerpt from Technical Report UWF-TR-FDOT-001-01, which contains the details of each derived security metric. The complete report can be found in Appendix I.

2.1. Common Vulnerability Scoring System (CVSS)

CVSS is an open framework for communicating the characteristics and severity of software vulnerabilities. It consists of three metric groups: Base, Temporal, and Environmental. Details on these metrics can be found in [1]. In [2], an illustration of the application of these metrics on the vulnerability of the Tesla Model S/X vehicles manufactured before March 2018 [3] and the vulnerability in the infotainment component of several BMW Series vehicles (CVE-2018-9322) [4] are illustrated.

White paper provides another use case to illustrate the applicability of this metric. A copy of the white paper can be found in Appendix III.

2.2. Common Methodology for IT Security Evaluation (CEM)

The CEM [5] is a companion document to the Common Criteria for Information Technology Security Evaluation (CC). It defines the minimum actions to be taken by an evaluator conducting a CC evaluation, utilizing the criteria and evidence as stated in the CC.

We specifically examined the attack potential on an automotive vehicle based on the following factors: elapsed time, specialist expertise, knowledge of the target, window of opportunity, and IT hardware/software or other equipment. We provided a detailed description of each of these factors in a white paper found in Appendix III.

2.3. Threats on Assets

We identify the following threats on vehicle assets and derive the attack potential metrics. We apply the previously defined factors for analysis and aggregate the metrics. The threats on vehicle assets are the following: false data from ECU, blocking of CAN bus, malicious software, denial of telematics service, unauthorized access, command injection, masquerading, data tampering. We shall provide a detailed description of each of these threats in a white paper found in Appendix III.

The total attack potential for each threat is simply a summation of the value assigned to each of the attributes of a successful attack. These results can be utilized during the decision-making process of cybersecurity asset allocation towards risk mitigation or prevention.

2.4. Common Weakness Scoring System

The Common Weakness Scoring System (CWSS) [6] is a mechanism for evaluating software weaknesses in a consistent, flexible, open manner. It is a community-based undertaking which addresses the need for prioritizing the software vulnerability issues. The measurements are organized into three metric groups: base finding, attack surface, and environmental. The groups, including their subgroups, as described in [6], are fully expounded in a white paper found in Appendix III.

2.5. Operational Safety Assessment Metrics

We next turn our attention to the impact of cybersecurity to operational safety assessment (OSA). There exist several OSA metrics that have been proposed, adopted, and studied [7] [8]. SAE J3237 [9], Driving Assessment (DA) Metrics for Automated Driving Systems (ADS), is currently under development. The SAE report provides definitions and a lexicon for describing operational safety metrics for ADS vehicles. The characteristics of the listed metrics include the following: definition, data source, subjectivity, observable variable, formulation, subjective assumptions and thresholds, and origin. A related work by the SAE V&V Task Force is the development of a proposed taxonomy for a recommended practice on operational safety metrics [10]. At the classification level of the proposed taxonomy are the operational safety metrics [10]. These operational safety metrics, which were fully covered in a white paper (see Appendix III), will become the foundation of the OSA metrics that we have derived and augmented with the following:

- **Authentication Metric.** This OSA metric measures the quality of the authentication system deployed in the vehicle. This is extremely useful in modern vehicles that rely on communications such as those in V2V or V2I environment.
- **Physical Access Metric.** This OSA metric measures the strength of physical access protection of vehicle controls. An example is the unsecured physical access to an OBD port which could compromise the vehicle's CAN bus.
- **Communication Channel Metric.** This OSA metric pertains to the quality of the communication channel used by the vehicle.

2.5.1 Cybersecurity Metrics for Operational Safety

We investigated the impact of cybersecurity on operational safety. In doing so, we devised cybersecurity metrics that have close affinities with OSA metrics. These cybersecurity metrics for operational safety are described in the following:

- **Safety Envelope Metric.** This cybersecurity metric measures the security resiliency of a connected vehicle to be able to maintain a safe boundary amidst a cyber intrusion incident. An example is a vehicle's capability in preventing malicious manipulation of the control and sensing systems that enable driving at a safe distance from other vehicles. Values range from 0.0 for least resilient to 1.0 for most security resilient.

- Behavioral Metric. This cybersecurity metric measures the vehicle's capability to protect against a cyber-attack that enables the improper behavior of the subject vehicle. An example of such attack is the manipulation of the vehicle cruise control mechanism. Values range from 0.0 for least capable to 1.0 for most capable.
- Component Metric. This is a measure of the susceptibility of the vehicle components to cyber-attack. For example, an Electronic Control Unit (ECU) device originating from an unverifiable supply chain may be highly susceptible to cyber-attack. Values range from 0.0 for most susceptible resilient to 1.0 for least susceptible.
- Sensing Metric. This cybersecurity metric pertains to the integrity and accuracy of data collected by the vehicle sensors. Roadside Units (RSUs) that are not properly secured may produce inaccurate or tampered data. Values range from 0.0 for least reliable data to 1.0 for most reliable data.
- Perception Metric. This cybersecurity metric pertains to the security of the system that provides for the interpretation of environment data collected by the vehicle sensors. For example, an insecure image processing system that is highly susceptible to an attack may provide inaccurate interpretation of traffic signs or signals. Values range from 0.0 for least secure to 1.0 for most secure.
- Planning Metric. This cybersecurity metric measures the vulnerability of the trajectory planning system to malicious intrusion. Values range from 0.0 for most vulnerable to 1.0 for least vulnerable.
- Control Metric. This cybersecurity metric measures the vulnerability of the vehicle's control system to malicious intrusion. Values range from 0.0 for most vulnerable to 1.0 for least vulnerable.
- Authentication Metric. This cybersecurity metric measures the security posture of the authentication system deployed in the vehicle. Values range from 0.0 for least secure to 1.0 for most secure.
- Physical Access Metric. This cybersecurity metric measures the strength of physical access protection of vehicle controls. Values range from 0.0 for least physically secure to 1.0 for most physically secure.
- Communication Channel Metric. This cybersecurity metric pertains to the level of protection of the communication channel used by the vehicle. Security characteristics of data transmission such as encryption, authentication, and attribution are pertinent concerns in this metric. Values range from 0.0 for least secure to 1.0 for most secure.

Emulating the evaluation methodology of the OSA metrics that was introduced by Wishart et.al. [11], we presented four evaluation factors for the formulation of the aggregation of cybersecurity metrics. The four evaluation factors are described in the following:

- Reliability. This factor quantifies the fidelity of the sources of measurement data. For instance, data originating from actual events carry a higher value than those from simulated events. Values range from 0.1 for less reliable to 1.0 for most reliable.
- Relevance. This factor quantifies the relevance of the measurement to a subject vehicle. This value may vary according to the specificity of data such as make and model of the subject vehicle. Data for a Honda CRV is more specific than data that refers to Honda vehicles in general. Values range from 0.1 for least relevant to 1.0 for most relevant.

- Extent. This factor quantifies the scope or extensiveness of the measurement data. The value ranges from 0.1 for least extensive to 1.0 for most extensive.
- Criticality. This factor quantifies the gravity of a specific metric. For instance, security measurement on control will carry a heavier weight than that on safety envelope. The value ranges from 0.1 for least critical to 1.0 for most critical.

The Aggregate Security Metric (ASM) for a specific vehicle is calculated as

$$\text{ASM} = \text{Reliability} \times \text{Relevance} \times \text{Extent} \times \left(\sum_{k=1}^N \text{Criticality}_k \times \text{Security_Metric}_k \right)$$

The ASM value will range from 0 to 10.

2.6. Security Vulnerability Metrics for Connected Vehicles

The purpose of security vulnerability metrics is to provide guidance to security engineers and testers using security vulnerability metrics that measure weak or vulnerable features in the software system of connected vehicles. These metrics are based on the seminal work of Moukahal and Zulkernine [12]. We describe each of the following risks on connected vehicles that may eventually contribute to the likelihood of vulnerability exploitation.

2.6.1 Electronic Control Unit (ECU) Coupling Risk

This risk is manifested by level of interconnection among ECU components. This means the higher the coupling value the higher is the probability for vulnerabilities. Thus, for every functionality, F, for all ECUs, N, and for all communication links between ECU j and ECU k, the ECU coupling risk, R_{EC} , is calculated as

$$R_{EC}(F) = \sum_{j=1, k=1}^N C_{jk}$$

$C_{jk}=1$ if there is at least one information transfer between ECU j and ECU k; 0 otherwise.

$$\text{Max} (R_{EC}(F)) = N$$

2.6.2 Communication Channel Risk

This risk is based on the communication channel types that are available for connected vehicles: vehicle to vehicle (V2V), vehicle to infrastructure (V2I), user to vehicle (U2V), and intra-vehicle (IV). The communication risk, for each functionality, F, is calculated according to the following formula:

$$R_{CC}(F) = \sum_{j=1}^N w_j C_j$$

Where N is the number of communication links, w_j the weight of a specific communication channel type based on its propensity to vulnerability, and C_j is 1 if the functionality uses the channel; 0 otherwise.

$$\text{Max (R}_{CC} \text{ (F))} = \text{Total number of all communication channels}$$

2.6.3 Complexity Risk

This risk is associated with the number of defects in software used in automotive vehicles. The complexity metric in software is an excellent indicator of vulnerabilities. The Halstead Complexity measure is a standard way of deriving the complexity of software. Thus, for calculating the complexity of the functionalities in connected vehicle, we use the formula:

$$\text{R}_{SC} \text{ (F)} = \text{SLOC} + a \text{ (Nesting)}$$

Where SLOC is the Source Line of Code, Nesting is the number of control structures, and a is the weight, with value over one, indicating complexity of the nesting structure.

$$\text{Max (R}_{SC} \text{ (F))} = \text{SLOC} + 10 \text{ (Nesting)}$$

2.6.4 Input and Output Data Risk

This risk involves the input and output data in a connected vehicle. The metric distinguishes between a Fixed Input (FI) from a Fluctuating Input (LI). It also distinguishes an Insensitive Output (IO) from a Sensitive Output (SO). Weights (a, b) are added to highlight the significance of the Fluctuating Input and the Sensitive Output. To calculate the Input and Output Data Risk, we use the formula:

$$\text{R}_{DIO} \text{ (F)} = \text{FI} + a \text{ (LI)} + \text{IO} + b \text{ (SO)}$$

$$\text{Max (R}_{DIO} \text{ (F))} = \text{FI} + 5 \text{ (LI)} + \text{IO} + 5 \text{ (SO)}$$

2.6.5 History of Security Issues

This risk considers the past security issues of a certain vehicle functionality. Given Y as the total number of years since the first car attack and a_y as the number of attacks that occurred in year y. A forgetting factor, l, is introduced to provide relevancy to the attacks that occurred in more recent years, where $0 \leq l \leq 1$. To calculate the risk of a vehicle functionality using the history of security issues, we use the formula:

$$\text{R}_{HS} \text{ (F)} = \sum_{y=1}^Y \alpha_y \lambda^{Y-y}$$

For a 2-year comparison, the calculation simply boils down to

$$l = 1 - (a_1 / a_2) \quad \leftarrow \text{the forgetting factor}$$

$$\text{R}_{HS} \text{ (F)} = a_1 (l)^{Y-1} + a_2$$

$$\text{Max (R}_{HS} (F)) = a_1 + a_2$$

2.6.6 Overall Security Vulnerability Metric

The overall security vulnerability metric of a certain functionality in a connected vehicle is calculated by first normalizing the values of each of the metrics and applying a weighting factor (a, b, g, d, f), which indicates its significance to the overall scheme. The metrics are added to obtain the overall value, which is in direct correlation with the vulnerability level of the functionality. The formula is shown as follow:

$$\begin{aligned} OSV = & \alpha \left[\frac{R_{EC}(F)}{\max(R_{EC}(F))} \right] + \beta \left[\frac{R_{CC}(F)}{\max(R_{CC}(F))} \right] + \gamma \left[\frac{R_{SC}(F)}{\max(R_{SC}(F))} \right] \\ & + \delta \left[\frac{R_{DIO}(F)}{\max(R_{DIO}(F))} \right] + \varphi \left[\frac{R_{HS}(F)}{\max(R_{HS}(F))} \right] \end{aligned}$$

2.7. Vehicle Security Best Practices Assessment Metrics

Vehicle Security Best Practices Assessment Metrics are designed based on the National Highway Traffic Safety Administration (NHTSA) Report DOT HS812 075 [13]. The report contains a review and analysis of cybersecurity best practices involving automotive vehicles.

The study utilizes the iterative Information Security Life Cycle divided into four phases and processes. The four phases and processes are described in the following:

- *Assessment Phase.* This phase includes the development and implementation of security policies, the evaluation of system security, and the processes of risk assessment.
- *Design Phase.* This phase entails the prioritization of systems and resources applicable to security and design and analysis of the system's security architecture.
- *Implementation Phase.* This phase covers the steps taken in vulnerability remediation and the processes in security testing and evaluation.
- *Operation Phase.* This phase includes the security awareness training for all personnel, customers, and other stakeholders. It also includes continuous security monitoring, intrusion detection and response.

The four phases were described in the white paper.

2.7.1. Vehicle Security Best Practices Assessment Metrics

We propose the following vehicle security best practice assessment metrics based on the Information Security Life Cycle described above. The metrics are built by a self-assessment form, shown on Table 1, which consists of a checklist of the status of each of the four phases.

Table 1: Vehicle Security Best Practices Assessment Checklist

Process	Checklists	Status
Security Policy	<p>Are security policies established?</p> <p>Are security policies properly documented and widely disseminated within the organization?</p> <p>Are security policies strictly enforced?</p> <p>Are security policies periodically reviewed/updated?</p>	
Data Security & Privacy	<p>Is collected/stored data protected/encrypted?</p> <p>Is transmitted data encrypted?</p> <p>Is there a control mechanism for sharing data?</p> <p>Does the site comply with data protection standards and regulations (e.g., ISO/IEC 27001 certification, GDPR)?</p>	
Risk Assessment	<p>Do you conduct a periodic risk assessment of vehicle cybersecurity?</p> <p>Is there a developed and implemented organization-wide risk management strategy?</p> <p>Is there a Supply Chain Risk Management (SCRM) policy?</p> <p>Are security controls in place and periodically evaluated and/or enhanced?</p>	
System Protection & Prioritization	<p>Have you implemented security-by-design principles during the vehicle design phase?</p> <p>Have you implemented domain separation for in-vehicle networks (i.e., limiting the communication between the safety-critical and non-safety critical domains)?</p> <p>Does the organization triage the identified risks according to priority for resource allocation?</p> <p>Do you have a comprehensive system security test plan?</p>	
Security Architecture	<p>Have you implemented a layered approach to vehicle security (ECU level, in-vehicle network level, V2V level, V2X level)?</p> <p>Is there a periodic evaluation of the system's security architecture?</p>	

	<p>Do you maintain an inventory of operational software components used in each automotive ECU and assembled vehicle?</p> <p>Have you considered the risks and vulnerabilities associated with vehicle sensor devices?</p>	
Remediation & Implementation	<p>Are there established mechanisms to update vehicle software and firmware remotely and securely?</p> <p>Are appropriate security controls implemented and are in place?</p> <p>Do you have an established remediation process?</p> <p>Is the remediation plan evaluated and implemented?</p>	
Security Test & Evaluation	<p>Have you conducted a thorough code review on the vehicle software?</p> <p>Have you conducted penetration testing on connected vehicle communication systems before deployment?</p> <p>Are security controls tested and evaluated for compliance with security performance specifications?</p> <p>Do you conform with secure software development best practices as outlined in NIST 8151 and ISO/SAE 21434?</p>	
Awareness & Security Training	<p>Is there a periodic security awareness training program for the entire workforce?</p> <p>Is security risk and mitigation disclosure available to the consumer and other stakeholders?</p> <p>Do you evaluate the effectiveness of the security awareness training program and introduce improvements if needed?</p> <p>Do you collect, maintain, analyze, and share information related to cybersecurity through the Automotive Information Sharing and Analysis Center (Auto-ISAC)?</p>	
Intrusion Detection & Response	<p>Is there an Incident Response Plan (IRP) in place?</p> <p>Is the IRP periodically tested, evaluated, and updated?</p> <p>Do you have a systematic process for continuous risk and security monitoring?</p> <p>Are security incidents properly documented and reported?</p>	

The development of the vehicle security metrics was successfully completed and documented. These security metrics were derived from published literature to provide benchmarks for security improvement processes. A technical memorandum was submitted to and approved by the FDOT Research Office in December 2022 and January 2023, respectively. The document is attached as Appendix I.

Chapter 3. Design and Implementation of the Vehicle Security Metrics Visualization System (VSMVS)

The second project task entails the design and implementation of web-enabled Vehicle Security Metrics Visualization System (VSMVS). The landing page of the VSMVS implementation is depicted in Figure 2.

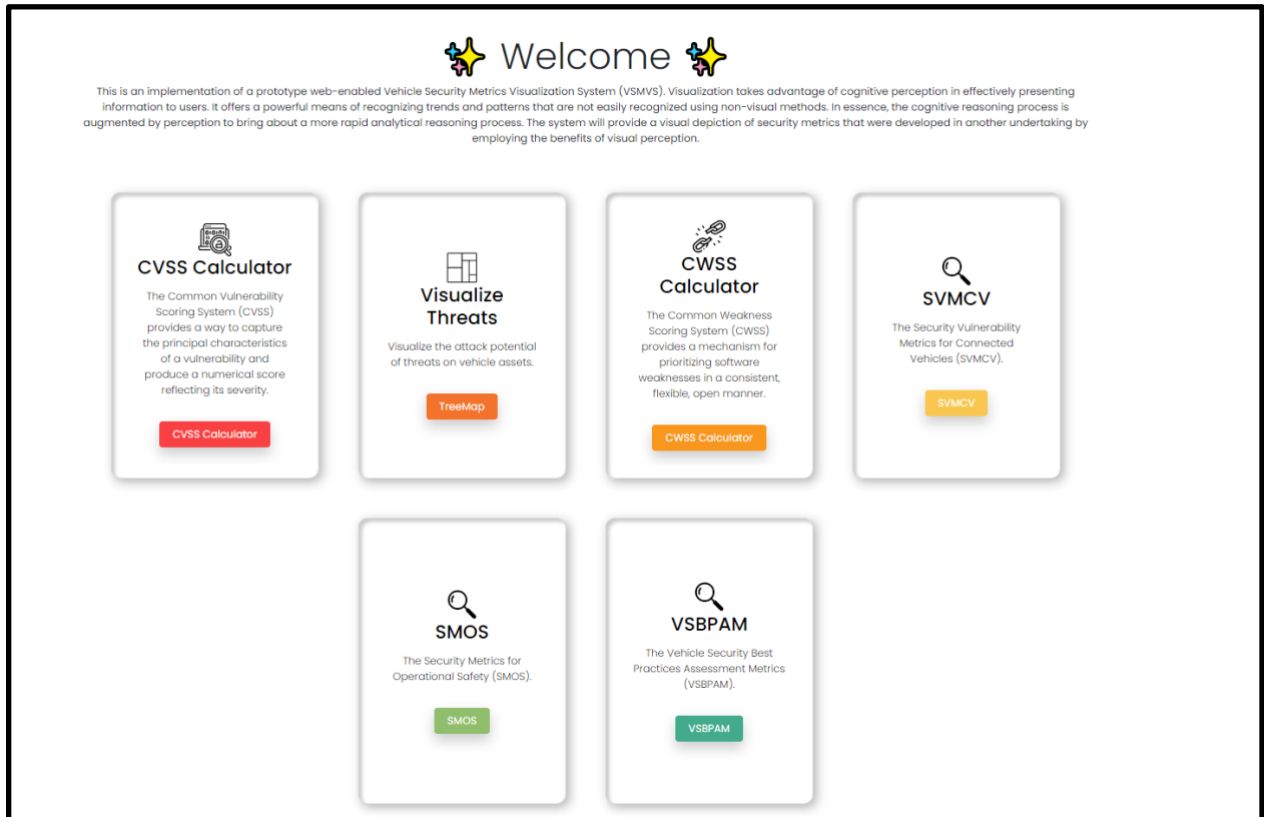


Figure 2: The Landing Page of the VSMVS

The following discussions provide an overview of the web interfaces of the VSMVS. For an in-depth treatment, the system requirements and specification document can be located at the project's Github repository at URL: <https://github.com/UWF-CfC-FDOT/VSMS>.

3.1. Visualization of CVSS Vector and Metrics

The VSMVS provides a Graphical User Interface (GUI) for data entry of CVSS vector and metrics information. The CVSS GUI is shown on Figure 3.

CVSS vector

Results
AV:N/AC:H/PR:H/UI:R/S:U/C:H/I:H/A:H

Base Score Metrics

Exploitability Metrics

Attack Vector (AV)

Attack Complexity (AC)

Privileges Required (PR)

User Interaction (UI)

Impact Metrics

Scope (s)

Confidentiality Impact (C)

Integrity Impact (I)

Availability Impact (A)

Temporal Score Metrics

Exploit Code Maturity (E)

Remediation Level (RL)

Report Confidence (RC)

Figure 3: The CVSS Vector Interface for the VSMVS

The VSMVS provides a corresponding visualization for the CVSS Vector Scoring System as shown on Figure 4.

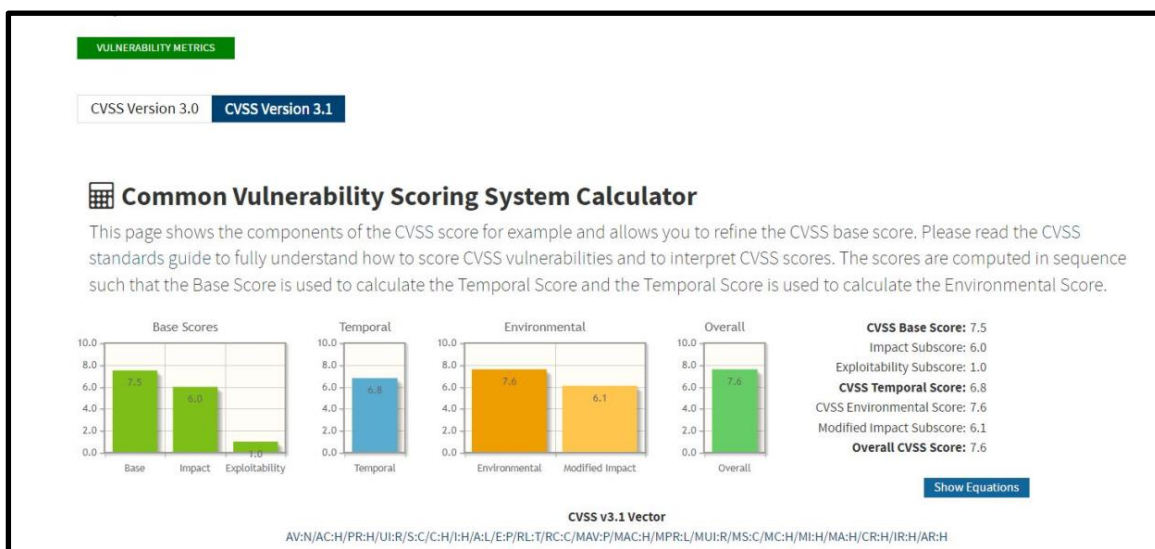


Figure 4: The CVSS Vector Scoring System

3.2. Visualization of Attack Potential of Threats on Vehicle Assets

The VSMVS provides a GUI for the data entry of attack potential of threats on vehicle assets as shown on Figure 5.

Threat on Assets	Elapsed Time	Specialist Expertise	Knowledge of Target	Window of Opportunity	Equipment/Software	Total
False Data from ECU ⓘ	Immediate (20) ▾	novice (15) ▾	public knowledge (20) ▾	unlimited (20) ▾	minimal (20) ▾	95
Blocking of CAN Bus ⓘ	one week (10) ▾	proficient (10) ▾	public knowledge (20) ▾	unlimited (20) ▾	minimal (20) ▾	80
Malicious Software ⓘ	one month (5) ▾	expert (5) ▾	sensitive information (10) ▾	difficult (5) ▾	minimal (20) ▾	45
Denial of Telematics Service ⓘ	one week (10) ▾	expert (5) ▾	critical information (5) ▾	easy (15) ▾	specialized(10) ▾	45
Unauthorized Access ⓘ	one week (10) ▾	proficient (10) ▾	public knowledge (20) ▾	easy (15) ▾	specialized(10) ▾	65
Command Injection ⓘ	one week (10) ▾	expert (5) ▾	public knowledge (20) ▾	moderate (10) ▾	highly specialized(5) ▾	50
Masquerading ⓘ	one week (10) ▾	multiple experts (1) ▾	sensitive information (10) ▾	difficult (5) ▾	multi-specialized (1) ▾	27
Data Tampering ⓘ	one year (1) ▾	expert (5) ▾	critical information (5) ▾	moderate (10) ▾	highly specialized(5) ▾	26

[Visualize](#)

Figure 5: The Attack Potential of Threats on Vehicle Assets

The VSMVS provides a corresponding visualization for the calculated attack potential of threats on vehicle assets as shown on Figure 6.

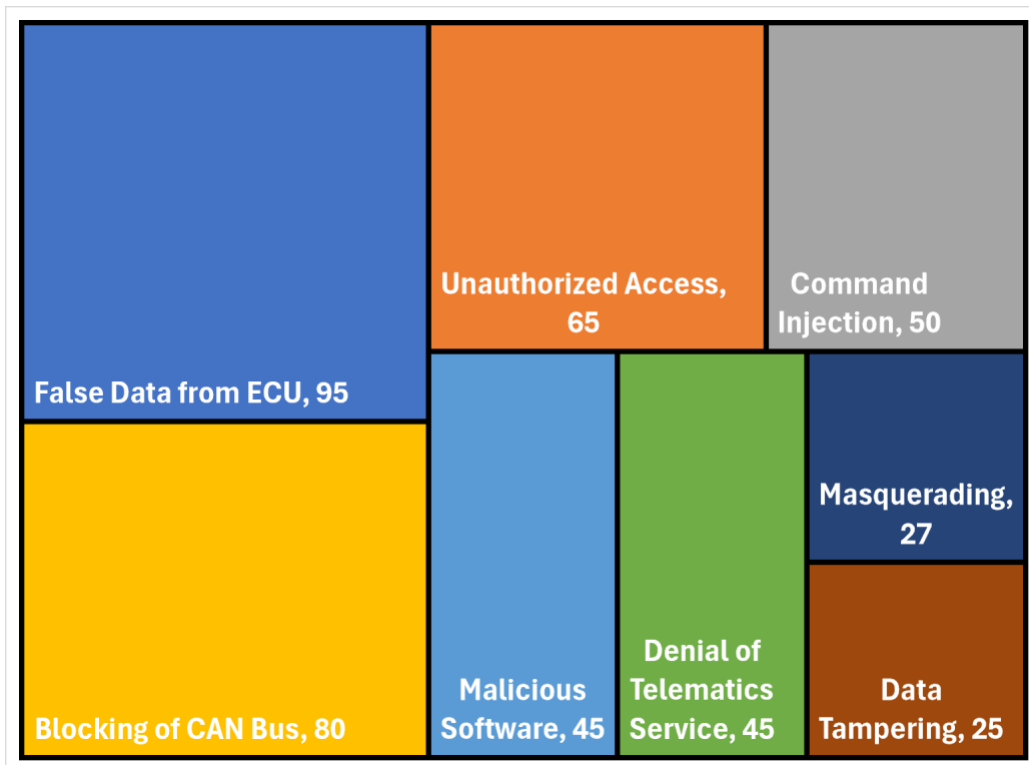


Figure 6: The Tree Map Visualization of Vehicle Attack Potential

3.3. Visualization of Common Weakness Scoring System

The VSMVS provides a GUI for data entry of CWSS vector and metrics information as shown on Figure 7.

Common Weakness Scoring System (CWSS)

CWSS vector: [TI:1/AP:0.9/AL:0.7/IC:0.7/FC:0.8]
CWSS score: 67.3 Visualize

Base Finding

Technical Impact: Critical
Acquired Privilege: Administrator
Acquired Priv Layer: Application
Int Control Effectivene: None
Finding Confidence: Proven True

Attack Surface

Required Privilege: None
Req Privilege Layer: Application
Access Vector: Internet
Authentication Strength: Strong
Level Interaction: Automated
Deployment Scope: All

Environmental

Business Impact: Critical
Likelihood Discovery: High
Likelihood Exploit: High
Ext Control Effect: None
Prevalence: Widespread

Figure 7: The Common Weakness Scoring System User Interface

The corresponding visualization for the Common Weakness Scoring System (CWSS) metrics is shown in Figure 8.

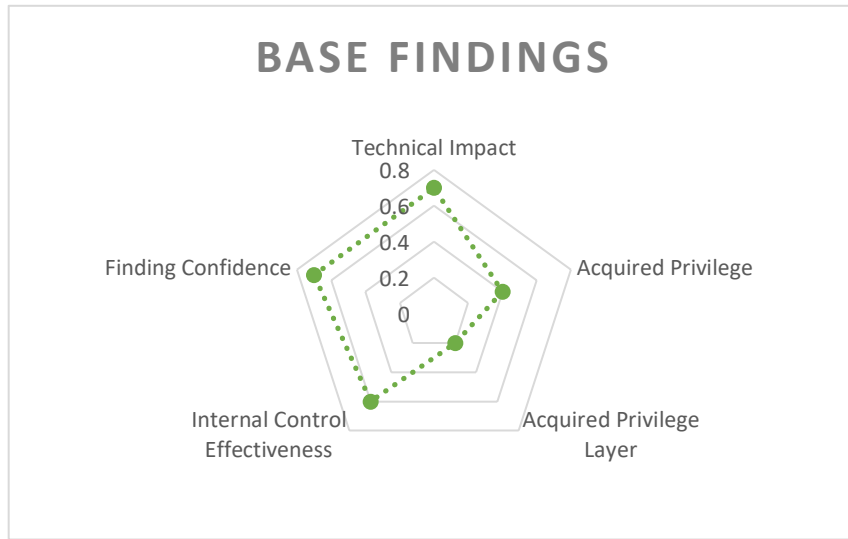


Figure 8: The Visualization for the Common Weakness Scoring System Metrics

3.4. Visualization of Security Metrics for Operational Safety

The VSMVS provides a data entry interface for the Operational Safety Assessment (OSA) metrics and the corresponding visualization are shown in Figure 9 and Figure 10, respectively.

Figure 9: Data Entry Interface for Security Metrics for Operational Safety

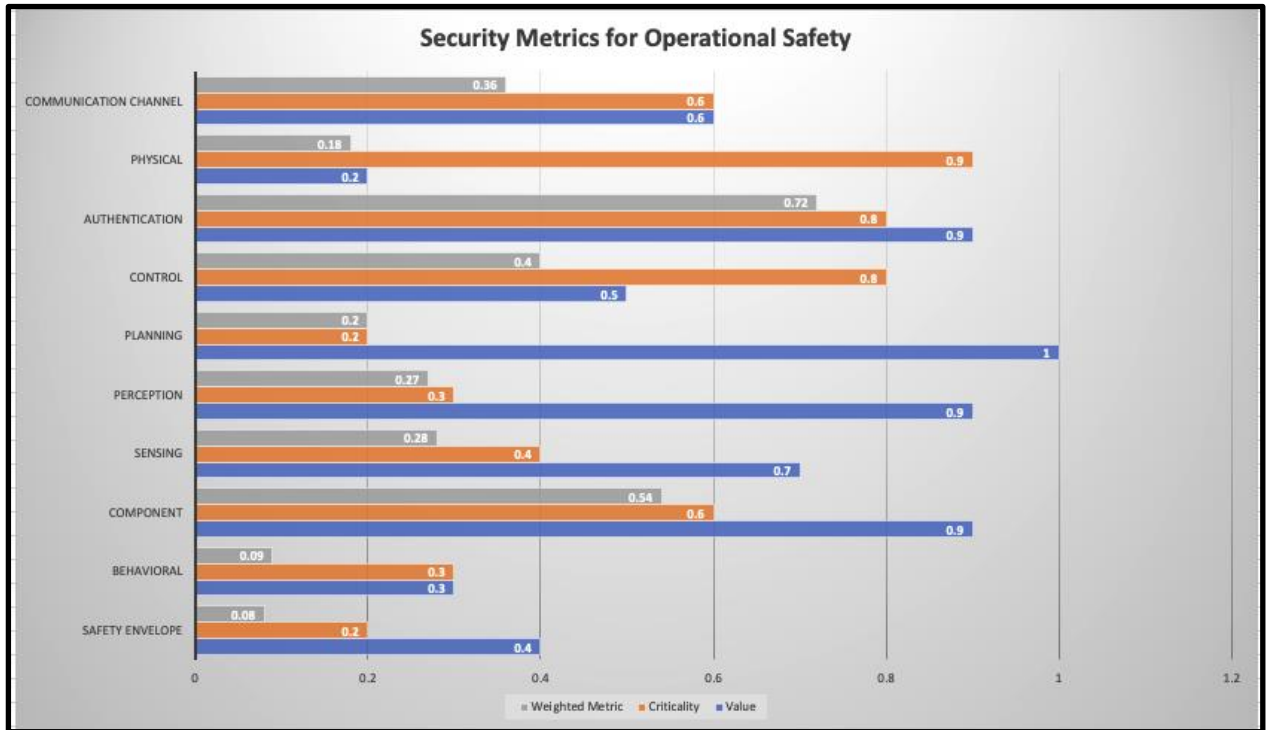


Figure 10: Calculated Security Metrics for Operational Safety

3.5. Visualization of Security Vulnerability Metrics

The VSMVS provides a data entry interface for the Security Vulnerability Metrics for connected vehicles and the corresponding visualization are shown in Figure 11 and Figure 12, respectively.

Security Vulnerability Metrics for Connected Vehicles

<p>ECU Coupling Risk ? Weight 1</p> <p>Active ECU Links <input type="text" value="0"/></p> <p>Total ECU Links <input type="text" value="0"/></p> <p>Complexity Risk ? Weight 1</p> <p>SLOCs <input type="text" value="0"/></p> <p>Number of Nestings <input type="text" value="0"/></p> <p>Weight of Nestings <input type="text" value="1"/></p>	<p>Communication Channel Risk ? Weight 1</p> <p>Active V2V Links <input type="text" value="0"/></p> <p>Total V2V Links <input type="text" value="0"/></p> <p>V2V Weight <input type="text" value="0.1"/></p> <p>Active U2V Links <input type="text" value="0"/></p> <p>Total U2V Links <input type="text" value="0"/></p> <p>U2V Weight <input type="text" value="0.1"/></p> <p>Active V2I Links <input type="text" value="0"/></p> <p>Total V2I Links <input type="text" value="0"/></p> <p>V2I Weight <input type="text" value="0.1"/></p> <p>Active IV Links <input type="text" value="0"/></p> <p>Total IV Links <input type="text" value="0"/></p> <p>IV Weight <input type="text" value="0.1"/></p>
<p>History of Security Issue Risk ? Weight 1</p> <p>Number of Years Since First Attack <input type="text" value="0"/></p> <p>Number of First Attack <input type="text" value="0"/></p> <p>Number of Second Attack <input type="text" value="0"/></p> <p>Forgetting Factor 0.2</p>	<p>I/O Data Risk ? Weight 1</p> <p>Number of Fixed Input <input type="text" value="0"/></p> <p>Number of Fluctuating Input <input type="text" value="0"/></p> <p>FI Weight <input type="text" value="1"/></p> <p>Number of Insensitive Output <input type="text" value="0"/></p> <p>Number of Sensitive Output <input type="text" value="0"/></p> <p>SO Weight <input type="text" value="1"/></p>

2.05

Figure 11: Data Entry Interface for Security Vulnerability Metrics

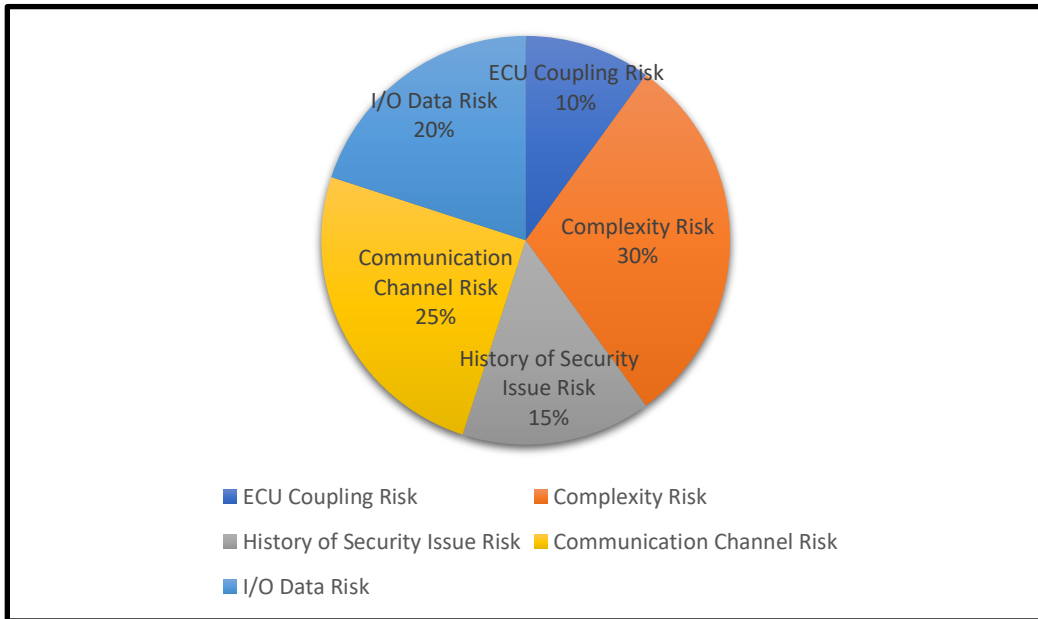


Figure 12: Calculated Security Vulnerability Metrics

3.6. Visualization of Vehicle Security Best Practices Assessment Metrics

VSMVS provides a data entry interface for Vehicle Security Best Practices Assessment Metrics for Connected Vehicles information as shown in Figure 13.

Phase	Process	Checklist	Response	
			Yes	No
Assessment	Security Policy	Security policies established	X	
		Security policies properly documented and widely disseminated within the organization?	X	
		Security policies strictly enforced	X	
		Security policies periodically reviewed/updated		X
	Data Security & Privacy	Collected/stored data protected/encrypted	X	
		Transmitted data encrypted	X	
		A control mechanism for sharing data	X	
		Compliant with data protection standards and regulations (e.g., ISO/IEC 27001 certification, GDPR)?	X	
	Risk Assessment	Conduct a periodic risk assessment of vehicle cybersecurity	X	
		Developed and implemented organization-wide risk management strategy	X	
A Supply Chain Risk Management (SCRM) policy		X		
Security controls in place and periodically evaluated and/or enhanced		X		
Design	System Protection and Prioritization	Implemented security-by-design principles during the vehicle design phase		X
		Implemented domain separation for in-vehicle networks (i.e. limiting the communication between the safety-critical and non-safety critical domains)		X
		Process to triage the identified risks according to priority for resource allocation	X	
		Comprehensive system security test plan		X
	Security Architecture	Implemented a layered approach to vehicle security (ECU level, in-vehicle network level, V2V level, V2X level)?	X	
		Periodic evaluation of the system's security architecture		X
		Maintain an inventory of operational software components used in each automotive ECU and assembled vehicle	X	
		Considered the risks and vulnerabilities associated with vehicle sensor devices	X	
Implementation	Remediation and Implementation	Established mechanisms to update vehicle software and firmware remotely and securely	X	
		Appropriate security controls implemented and are in place	X	
		Established a remediation process	X	
		Remediation plan evaluated and implemented	X	
	Security Test and Evaluation	Conducted a thorough code review on the vehicle software	X	
		Conducted penetration testing on connected vehicle communication systems before deployment	X	
		Security controls tested and evaluated for compliance with security performance specifications	X	
		Conformance with secure software development best practices as outlined in NIST 8151 and ISO/SAE 21434		X
Operation	Awareness and Security Training	Periodic security awareness training program for the entire workforce	X	
		Security risk and mitigation disclosure available to the consumer and other stakeholders	X	
		Evaluate the effectiveness of the security awareness training program and introduce improvements if needed	X	
		Collect, maintain, analyze, and share information related to cybersecurity through the Automotive Information Sharing and Analysis Center (Auto-ISAC)		X
	Intrusion Detection and Response	An Incident Response Plan (IRP) in place	X	
		The IRP periodically tested, evaluated, and updated		X
		A systematic process for continuous risk and security monitoring		X
		Security incidents properly documented and reported		X

Figure 13: Vehicle Security Best Practices Assessment Metrics

The project task to design and implement a vehicle security visualization system was successfully completed. The purpose of such system is to provide an immediate visual analysis of the security posture of a vehicle. Technical memoranda consisting of the design specification and requirements, a white paper on vehicle security research, and implementation code and support documents, and were submitted to and approved by the FDOT Research Office in January 2023. These documents are found in Appendix II, Appendix III, and Appendix IV, respectively.

Chapter 4. Testing and Deployment of the VSMVS

The third project task focused on the testing and deployment of the Web-enabled VSMVS.

4.1. Test Framework Description

The VSMVS project was developed using the C# language and the standard MSTest framework for unit testing. MSTest is a native unit testing library that comes with Microsoft's Visual Studio.

4.2. VSMVS Unit Tests

4.2.1. TreeMap Tests

The TreeMapTests.cs file contains unit test coverage of the /Pages/TreeMapModel.cshtml.cs class. It performs validation of the score calculation method for the Tree Map page.

4.2.2. Common Weakness Scoring System (CWSS) Tests

The CWSSTests.cs file contains unit test coverage of the /Pages/CWSSModel.cshtml.cs class. It performs validation of the scoring calculations for the CWSS page.

4.2.3. Security Metrics Operational Safety (SMOS) Tests

The SMOSTests.cs file unit test coverage of the /Pages/SMOSModel.cshtml.cs class. It performs validation of the score calculation methods for the SMOS page.

4.2.4. Security Vulnerability Metrics for Connected Vehicles (SVMCV) Tests

The SVMCVTests.cs file unit test coverage of the /Pages/SVMCVModel.cshtml.cs class. It performs validation of the score calculation method for the SVMCV page.

4.2.5. Vehicle Security Best Practices Assessment Metrics (VSBPAM) Tests

VSBPAMTests.cs file unit test coverage of the /Pages/VSBPAMModel.cshtml.cs class. It performs validation of the total score calculation method for the VSBPAM page.

The project task on visualization system testing was successfully completed. The purpose of the task is to ensure that all functional system components and features are working as intended and comply with the requirements. A technical memorandum was submitted to and approved by the FDOT Research Office in May 2023 and June 2023, respectively. This document is attached as Appendix IV. Also, a Vehicle Security Metrics Visualization System User Manual was included in the submission and can be found attached as Appendix VI. The source code repository is located at this URL: <https://github.com/UWF-CfC-FDOT/VSMS>.

Chapter 5. Data Ingestion, Cleansing, Normalization, and Data Mutation of Roadside Unit Data

The fourth project task utilized the dataset, in SAE J2735 message format, that was previously collected by RSUs in Gainesville, FL. The data were preprocessed, normalized, and mutated for the development of machine learning systems.

Data cleansing for machine learning systems can include the careful processing of missing data values or noise. Care must be taken not to cleanse a valid outlier data value. Data cleansing also includes the omission of some data elements that are not needed for training and testing of the machine learning model. One example would be the Vehicle ID, which is a unique identifier and will not contribute to a generalized model. Data conversion includes the transformation from XML or JSON to a single CSV format. Data conversion also includes using consistent units of measurement across all data sources, such as, time in milliseconds, speed in mph, distance in miles, etc. Normalization includes the introduction of binary flags to indicate the presence or absence of hard braking, the presence or absence of malicious information, or the introduction of limits on speed and acceleration to name a few. Normalization can also take the form of a min-max normalization on all data columns so columns with the highest or lowest numeric magnitude do not bias the machine learning model.

To gain a better understanding of CAV datasets, we started by scrutinizing the RSU datasets that were collected and shared by the research team in the FDOT pilot site at the University of Florida.

5.1. Roadside Unit Dataset

The dataset, in SAE J2735 message format, was collected by RSUs in Gainesville, FL, and can be availed through an AWS Simple Storage Service (S3) repository. The raw dataset, in compressed XML format, was converted to a readable comma-separated value (csv) format in preparation for data cleansing. We focused on the Basic Safety Message (BSM) component of the RSU and provide a brief description in the following.

5.2. Basic Safety Message (BSM)

The *Basic Safety Message (BSM)* is used to exchange safety data and consists of two parts:

- The mandatory part, also called BSMcoreData, is typically described in Abstract Syntax Notation One (ASN.1) [14] format, a formal notation to describe data transmitted by telecommunication protocols. Instead of describing the BSMcoreData in ASN.1 notation, we present each item in detail using a tabular format as shown in Appendix I. The data characteristics presented on the table are excerpted from [15] .
- The optional part of the BSM includes the Vehicle Safety Extension consisting of event flags, path history, path prediction, and the Radio Technical Commission for Maritime Services (RTCM) package.

5.3. Synthetic Dataset

After gaining an understanding of the BSM dataset characteristics, we investigated the viability of generating synthetic datasets that depict a typical vehicle moving in a straight-line trajectory without regard to traffic condition. The motivation behind this initiative is three-fold:

- to derive CAV datasets that can be used to study the application of Machine Learning (ML) algorithms to identify and classify malicious messages;
- to be able to create additional malicious datasets that will test attack scenarios for each of the V2V safety applications described in [16]; and
- to be able to perform a comparative study on the speed of convergence of various ML training algorithms on disparate datasets.

The synthetic BSM coreData generation proceeds using the following assumptions and constraints:

- Starting geolocation (Central Florida) with coordinates:
 - latitude: 28.890658
 - longitude: -82.097812
- Data collection time interval: 20 sec. Note that SAE J2945/1 Standard [16] requires vehicles to transmit 10 BSMs per second.
- Travel is on a straight line towards north
- Acceleration is in units of 0.0328 ft/sec²
- Acceleration (deceleration) is randomly applied every 10 minutes. The random value will range from -10 to 10 mph. This represents lateral acceleration (deceleration)
- Angle steering is 127 (unavailable)
- Brake system status: 0 during acceleration; 5 during deceleration (assuming front wheel brakes). The brake system status is adjusted to reflect the acceleration or deceleration condition
- Elevation: 61440 (unknown)
- Heading: 28800 (unavailable)
- Latitude is calculated using equations (1)-(5)
 - earth_radius = 6371 km (1)
 - current_lat= math.radians(initial_lat) (2)
 - # distance after 20 seconds of travel
 - dist_meters = speed_mph * 0.44704 * 20 (3)
 - # change in latitude
 - delta_lat = dist_meters / earth_radius (4)
 - # new latitude is calculated as
 - new_lat=math.degrees(current_lat+delta_lat) (5)
- Longitude is assumed constant at -82.097812
- Msg Count: 0 (unavailable)
- Vehicle ID: 0 (unavailable)
- SecMark : Calculated elapsed time in milliseconds
- Transmission: 2 (forward gear)
- Hard Braking: 0 (no), 1 (yes). Deceleration of 8 mph in 1 second (23.09 ft/sec² when applying the unit measure of 0.01 meters/sec²) while traveling at a speed of > 25 mph indicates hard braking.

- Vehicle size:
 - Length = 21 ft
 - Width = 7 ft
- Starting speed is 25 mph
- The brake status is adjusted to reflect the acceleration or deceleration condition
- The speed is calculated using equation (6)

$$V1 = V0 + a * t \tag{6}$$

where

V1 is the current speed, miles/hr

V0 is the initial speed, miles/hr

a is the acceleration, miles/hr²

t is the elapsed time, hr

- The data record is terminated with a status flag: 1 for normal and 0 for abnormal (malicious).

5.4. Malicious Dataset

After obtaining datasets from two sources: the RSU datasets for the Gainesville pilot site and the synthetically generated dataset, we implemented an application to inject malicious data records into each of the datasets. This process, in essence, produces two mutated datasets. The mutation process is described in the following.

Using the assumption for a typical acceleration of 0.577 ft/sec² (derived from 17.6 ft/sec² and applying the unit measure of 0.01 meters/sec²), malicious BSMcoreData test datasets are generated according to the following parameters. Note that for each data record, the status flag value is 1 to indicate an abnormal (malicious) data record.

5.4.1. Brake System Anomaly

1. To simulate an accelerating vehicle while brakes are engaged:
Randomly generate a BSMcoreData record with Brake System Status = 1111 (decimal value 15; brakes applied) and Longitudinal Acceleration = 5.364 m/sec².
2. To simulate an accelerating vehicle in reverse while the brakes are engaged:
Randomly generate a BSMcoreData record with Brake System Status = 0000 (decimal value = 0; all brakes not engaged), Longitudinal Acceleration = 5.364 m/sec² and Transmission Status=011 (decimal value 3; reverse gear).

5.4.2. Transmission System Anomaly

1. To simulate a speeding vehicle with transmission system in neutral: randomly generate a BSM data record with speed = 15 mph and Transmission Status = 000 (decimal value 0).
2. To simulate a speeding vehicle with transmission system in park: randomly generate a BSM data record with speed = 25 mph and Transmission Status = 001 (decimal value 1).
3. To simulate a speeding vehicle with transmission system in reverse gear: randomly generate a BSM data record with speed = 55 mph and Transmission Status = 011 (decimal value 3).

5.4.3. Longitudinal Acceleration Anomaly

1. To simulate an accelerating vehicle with transmission gear in neutral: randomly generate a BSM data record with Longitudinal Acceleration = 5.364 m/sec^2 and Transmission Status=000.
2. To simulate a decelerating vehicle with the transmission gear in park: randomly generate a BSM data record with Longitudinal Acceleration = -5.364 m/sec^2 and Transmission Status=001.
3. To simulate an accelerating vehicle with the transmission gear in reverse: randomly generate a BSM data record with Longitudinal acceleration = 5.364 m/sec^2 and Transmission Status=011.

5.4.4. Hard Braking Anomaly

To simulate malicious hard braking with the transmission in forward gear: randomly generate a BSM data record with Longitudinal Deceleration = -25.59 ft/sec^2 (2.5 times the value of deceleration rate considered as hard braking) and Transmission Status=010 (decimal value 2).

5.4.5. Speed Anomaly

To simulate malicious speed value with transmission in forward gear: randomly generate a BSM data record with speed 3 times the speed value of the previous speed reading and Transmission Status=010 (decimal value 2).

This task on data cleansing, pre-processing, normalization, and mutation was successfully completed. These data processing steps were made to ensure that the data that is consumed by the machine learning system is untainted and reliable to produce an effective system. A technical memorandum containing dataset attributes, details of pre-processing, normalization, and injection processes, and the data attributes and indicators of simulated cyber threats and attacks. The technical memorandum was submitted to and approved by the FDOT Research Office in November 2023 and December 2023, respectively. This document is attached as Appendix VII.

Chapter 6. Design and Implementation of a Machine Learning System for Vehicle Security

The fifth project task complemented the fourth task by utilizing the datasets derived to realize a prototype machine learning (ML) system for vehicle security. Table 2 shows the dataset attributes of the Basic Safety Message (BSM) dataset. Table 3 summarizes the machine learning model validation and testing results.

Table 2: BSM Dataset Attributes

Validation Observations	Testing Observations	Number of Predictors	Response Classes
2283	253	12	2

Table 3: Summary of ML Model Validation and Testing Results

Machine Learning Model	Validation Accuracy, %	Test Accuracy, %
Neural Network	100.0	100.0
Decision Tree	99.7	99.6
Optimizable Ensemble	99.9	99.6
K-Nearest Neighbor (KNN)	99.2	99.2
Logistic Regression	80.0	79.8
Support Vector Machine (SVM)	80.0	79.8

This task to design and implement a prototype machine learning system was successfully completed. The purpose of such system is to demonstrate the viability of constructing an intelligent system that can recognize malicious vehicle data. This type of system can be very useful in securing vehicles from malicious data introduced by attacks on vehicle communication systems. A technical memorandum containing the design specification and requirements of the prototype ML system, the implementation of the prototype ML system, the performance indicators gathered during the application of the ML system on the modified RSU dataset, and the documents containing the plans, scenarios, and cases used for testing. The technical memorandum, attached as Appendix VIII, was submitted to and approved by the FDOT Research Office in May 2023 and June 2023, respectively. The development team maintained a source code repository and version control on GitHub. The GitHub project URL is <https://github.com/UWF-CfC-FDOT/VMLFramework>.

Chapter 7. Design and Implementation of a Vehicle Threat Modeling Engine (VTME)

This sixth project task builds cyber threat models based on the stages identified in the slightly modified Lockheed Martin Cyber Kill Chain® [17], shown in Figure 14. Specific Tactics, Techniques and Procedures (TTPs), endemic to connected vehicle systems, are initially populated with information from MITRE’s Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) matrix. Specific Indicators of Compromise (IoCs) associated with TTPs are obtained from sources such as Open Threat Exchange (OTX) and VirusTotal.

For each stage of the Kill Chain, the VTME creates a Threat Model using the information gathered from the MITRE ATT&CK Framework, the Common Vulnerability Enumeration (CVE) and the Common Weakness Enumeration (CWE) sources. For instance, for the first stage, Reconnaissance, the system will collect the threat information, the vulnerability, and mitigation associated with that threat and upload it in the threat database. The other stages are weaponization, delivery, exploitation, installation, command and control, and actions on objectives. Note that some of those may not have any information available. Also, that information will be labeled according to which vehicle manufacturer it applies (e.g. Tesla, Honda, Kia, Ford, etc.).



Figure 14: Lockheed Martin's Cyber Kill Chain

The project task on the design and implementation of the vehicle threat modeling engine was successfully completed. The purpose of the vehicle threat modeling engine is to establish a archetypal system with which subsequent vehicle threat schemes can build on. A technical memoranda consisting of the system requirements, Ontology of the CVE-Kill Chain Mapping, and the Unit Test Overview. The technical memoranda were submitted to and approved by the FDOT Research Office in December 2022 and January 2023, respectively. The documents are attached as Appendix IX, Appendix X, and Appendix XI, respectively. The development team maintained a source code repository and version control on GitHub. The GitHub project URL is <https://github.com/UWF-CfC-FDOT/VTMECS>.

Chapter 8. Design and Implementation of a Vehicle Threat Collection System (VTCS)

The seventh project task called for the design and implementation of a Vehicle Threat Collection System (VTCS). It focuses on the design and development of an automated system whose purpose is to collect threat intelligence feeds from publicly available Open-Source Intelligence (OSINT) sites such as Open Threat Exchange (OTX) and VirusTotal, both of which offer Application Programming Interfaces (APIs) for automated queries.

8.1. The VTCS Interfaces

The Query Interface is used to provide the VTCS with vehicle search parameters. Figure 15 depicts a prototype of the interface where the user can submit one or more vehicles for processing. Using the Threat Source drop-down selection box, the user is provided an option to select a specific threat intelligence source such as OTX, VirusTotal, or ALL in an attempt to query all sources. The user can also filter the information to collect by specifying the Year, Make, and Model of the vehicle through drop-down selection boxes. The Add button will take the options selected by the user in the Year, Make, and Model selections and place the vehicle into the Query Interfaces Vehicle Selection table at the bottom of the page. Once a vehicle is added to the selection table the Year, Make, and Model drop-down selection boxes are reset to their default option and the user can add another vehicle. When the user is satisfied with the data source and vehicle(s) selected the Search button will initiate the VTCS to run the specified user query using the API of the selected open threat intelligence source. Clicking the Search button will also bring the user to the Record Viewer Interface. Figure 16 depicts the Graphical User Interface (GUI) of the VTCS record viewer.

Vehicle Threat Collection System (VTCS)

Query Page

Threat Source

OTX

Vehicle Selection

Year Make Model

2021 Toyota RAV4
2019 Ford Explorer

Add Search

Figure 15: Query Page Viewer Prototype

VTCS Results Page

2021 Toyota RAV4 2023-29389

Toyota RAV4 2021 vehicles automatically trust messages from other ECUs on a CAN bus, which allows physically proximate attackers to drive a vehicle by accessing the control CAN bus after pulling the bumper away and reaching the headlight connector, and then sending forged "Key is validated" messages via CAN Injection, as exploited in the wild in (for example) July 2022.

First Reported	Last Updated	Related CVEs	Associated CWEs	Data Source(s)	Indicators of Compromise	CVSS Score / Severity
2023/04/05 1615	2023/04/14 1603	CVE 2023-29389 CVE 2023-19365 CVE 2022-12980	CWE-74 CWE-115 CWE-43	OTX/MISP	N/A	6.8 / Medium

All records successfully added to VTDBS

VTCS Generated Records

Record Title	Data Source Hits	CVE Hits	CWE ID(s)	Indicators of Compromise	CVSS Score/Severity
2021 Toyota RAV4 2023-29389	2	3	74, 115, 43	0	6.8 / Medium
2019 Ford Explorer 2022-12980	3	1	22	15	8.8 / High

Figure 16: Prototype VTCS Record Viewer Interface

This task was successfully completed as evidenced by technical memoranda consisting of the system requirements (Appendix XII), the test plan overview (Appendix XIII), and the requirements traceability verification matrix (Appendix XIV). The technical memoranda were submitted to and approved by the FDOT Research Office in June 2023 and August 2023, respectively. The development team maintained a source code repository and version control on GitHub. The GitHub project URL is <https://github.com/UWF-CfC-FDOT/VTMECS>.

Chapter 9. Design and Implementation of a Vehicle Threat Database System (VTDS)

The eighth project task calls for the design and implementation of a Vehicle Threat Database System (VTDS). It focuses on the design, implementation, and deployment of a VTDS whose purpose is to function as an efficient threat data repository and processing system.

9.1. Entity Relationship (ER) Diagrams

The VTDS was designed using the Entity Relationship (ER) Diagram shown in Figure 17. The ER diagram depicted in Figure 18 shows the relationship between the User entity with the Transaction entity.

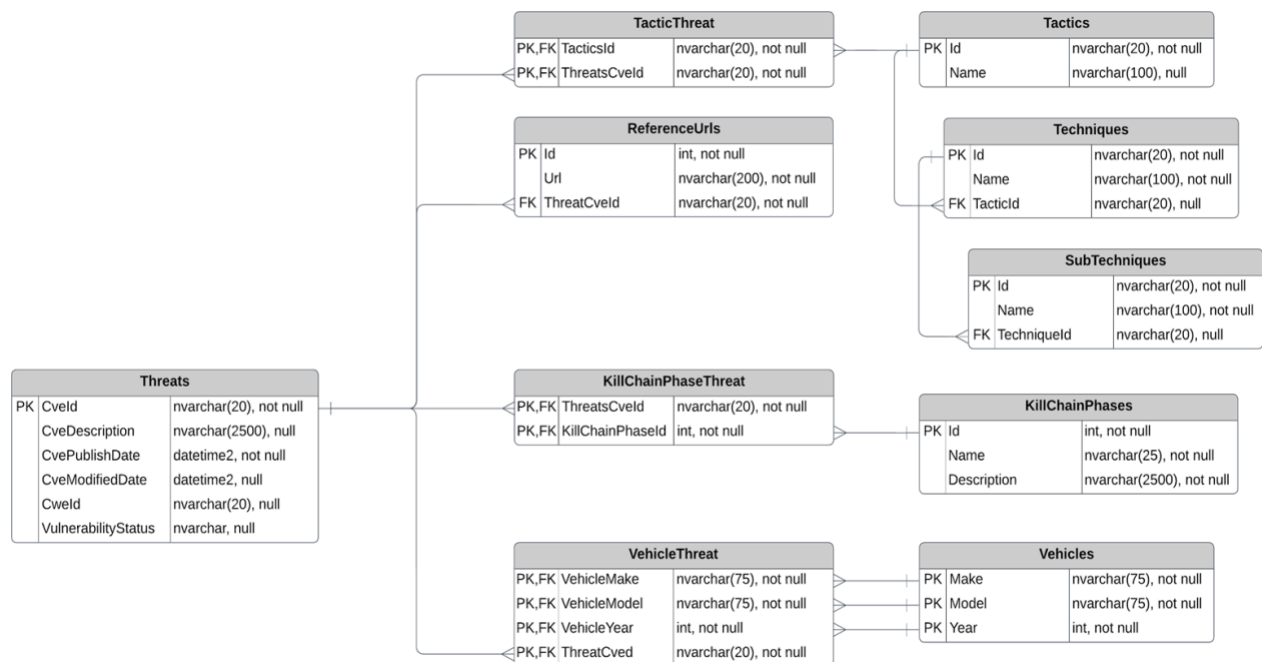


Figure 17: Entity Relationship (ER) Diagram for the VTDS

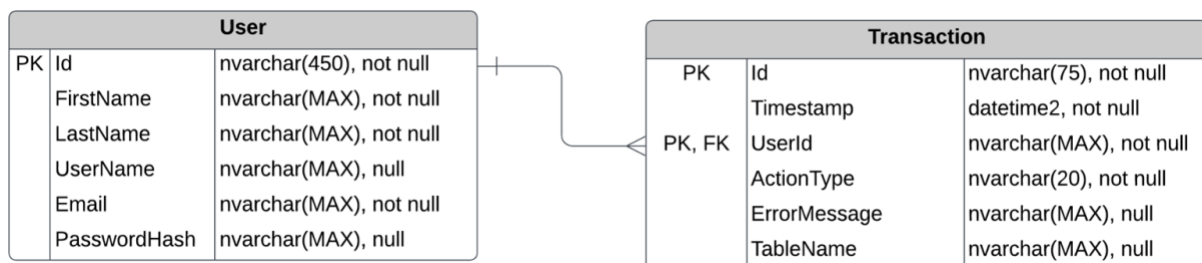


Figure 18: ER Diagram for the User and Transaction Log Entities

The project task on vehicle threat collection system was successfully completed. The system is intended to automatically collect threat information from disparate sources. The collection process is facilitated by the Application Program Interfaces (APIs), which are provided and enabled by the source provider. Technical memoranda consisting of the system design and requirements (Appendix XV), the test plan overview (Appendix XVI), and the requirements traceability verification matrix (Appendix XVII). The technical memoranda were submitted to and approved by the FDOT Research Office in November 2023 and January 2024, respectively.

Chapter 10. Design and Implementation of a Vehicle Threat Information Portal (VTIP)

The ninth project task calls for the design, development, and deployment of a Vehicle Threat Information Portal (VTIP), which facilitates the secure access to the functionalities of the connected vehicle threat information system. This portal provides a threat sharing functionality and connectivity to a Malware Information Sharing Platform (MISP).

10.1. VTIP Threat Record Interface

The VTIP Threat Record Interface, depicted in Figure 19, is used to display the records generated by querying the VTDS. The upper portion of the GUI displays a threat record search functionality where users may query the VTIP by date range, vehicle information, CVE ID, and/or keyword. The center and lower portions of the GUI display the threat record information to include the CVE ID, Threat record publish information, CWE information, threat record description, CVSS information, vehicle information, and Indicators of Compromise (IOC).

The screenshot shows the 'VTIP Interface Prototype: Search and Display Threat Record' interface. At the top, there is a search bar with fields for 'CVE ID', 'Date Start', 'Date End', and 'Keyword', followed by a 'Search' button. Below the search bar is a section titled 'Threat Record Information' which contains a table with the following data:

CVE ID:	Publish Info:	CWE Information:
CVE-0001 dig	09/16/2017	74 88
CVSS Information: Base Score: 6.8 MEDIUM Attack Vector (AV): Physical	Impact Score: 5.9 Attack Complexity (AC): Low	Exploitability Score: 0.9 Privileges Required (PR): None

Below the table is a 'Threat Description' section with the text: 'Toyota RAV4 2021 vehicles automatically trust messages from other ECUs on a CAN bus, which allows physically proximate attackers to drive a vehicle by accessing the control CAN bus after pulling the bumper away and reaching the headlight connector, and then sending forged "Key is validated" messages via CAN Injection, as exploited in the wild in (for example) July 2022.' Below this is a 'Vehicle Information' section with an 'Add' button, and an 'IOCs' section with an 'Edit' button. At the bottom of the interface, there are three buttons: 'Update Record', 'Access MISP', and 'Export Record'. Below these buttons is a section titled 'VTIP Records' with a list of CVE IDs: CVE-0001, CVE-0002, CVE-0003, CVE-0004, and CVE-0005. At the very bottom, there are 'Previous' and 'Next' buttons.

Figure 19: VTIP Interface

10.2. VTIP Threat Record Search and Edit Interface

The VTIP Threat Record Interface, depicted in Figure 20, will provide an edit mechanism allowing users to make changes to threat record data in the VTDS stemming from the information derived by the data analyst. This includes the ability to add/remove/change data such as vehicle information or IOCs.

VTIP Interface Prototype: Search and Display Threat Record

CVE ID: Date Start: Date End: Keyword:

Threat Record Information		« January 2024 »							CWE Information:	
CVE ID:	Publish Info:	Su	Mo	Tu	We	Th	Fr	Sa	74 88	
CVE-0001	09/16/2017	31	1	2	3	4	5	6		
CVSS Information:		7	8	9	10	11	12	13		
Base Score: 6.8 MEDIUM	Impact Score: 5	14	15	16	17	18	19	20	Exploitability Score: 0	
Attack Vector (AV): Physical	Attack Comple:	21	22	23	24	25	26	27	Privileges Required (P)	
		28	29	30	31	1	2	3		

Figure 20: VTIP Threat Record Search and Edit Interface

The project task on the design and implementation of the vehicle threat information portal was successfully completed. The purpose of the portal is to provide a user-friendly interface for users to be able to query the vehicle threat database system for threat information on specific vehicles. Technical memoranda consisting of the system design and requirements (Appendix XVIII), the test plan overview (Appendix XIX), and the requirements traceability verification matrix (Appendix XX). The technical memoranda were submitted to the FDOT Research Office in January 2024 and approved in February 2024.

Chapter 11. System Integration, Testing, and Deployment

The tenth task calls for the integration, testing, and deployment of all system components. Integration was accomplished through the Vehicle Security System Integration (VSSI). The VSSI provides different interface cards to enable access to the Vehicle Threat Intelligence Portal (VTIP), the Vehicle Threat Collection System (VTCS), the Vehicle Security Metrics Visualization System (VSMVS), and the Malware Information Sharing Platform (MISP). This interface provides the integration of the various subsystems using a common entry point and collectively binds them to function with a common purpose, i.e., supporting the security of CAVs.

11.1 The VSSI Interface

The landing page, as depicted in Figure 21, contains interface cards for each subsystem in the system, including the VSMVS, VTCS, VTIP, and MISP. Each card contains the name of the subsystem, an associated image, and a button to navigate to the associated subsystem.

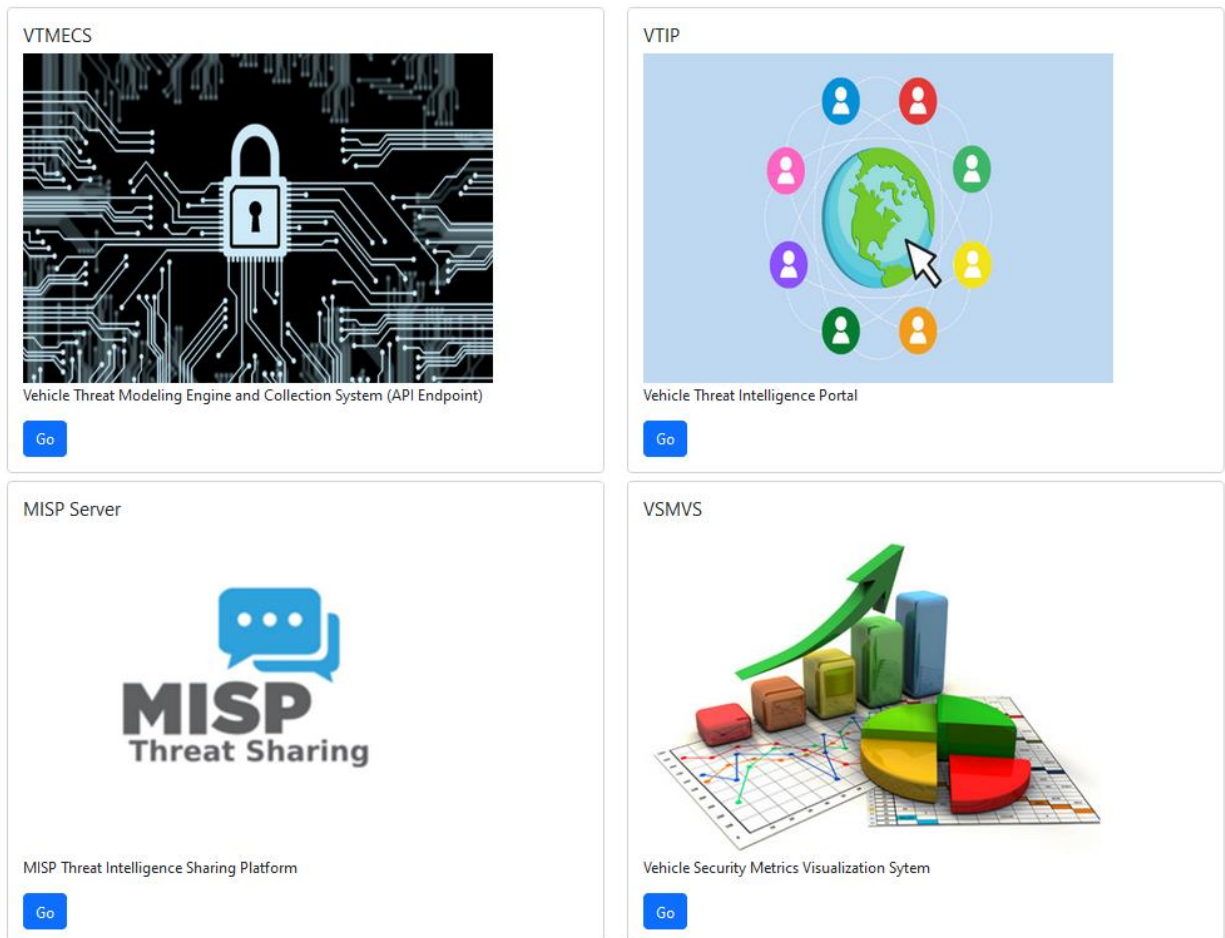


Figure 21: The VSSI Landing Page Interface

The project task on building an integrated vehicle security system was successfully. This integrated system provides one-stop landing pages for all the sub-systems developed by the project. A technical memorandum that consists of a collation of documents on system design and requirements, requirements traceability verification matrix, and test plan overview is included in this report as Appendix XXI. A user manual (see Appendix XXII) for the VSSI subsystem is also provided as a project deliverable.

Chapter 12. Continuous Improvement Process

Toward the end of the project execution, improvements were identified and acted upon. These actions are part of the continuous improvement process that led to the enhancement of the subsystems. What is presented herein is a list of the initial improvements that the project personnel have accomplished up until the project closeout meeting on March 31, 2024.

12.1 List of Improvements and Rationale

1. Developed the Vehicle Threat Database System (VTDS) controllers.
The VTDS controllers are the mechanisms that determine the purpose and the means by which threat data are processed. We need these controllers to streamline data processing and make the database system more efficient.
2. Ported the project to a web app for API access across the entire system.
The web app was created to provide an Application Program Interface (API) for an efficient access for all sub systems.
3. Created/Modified several database tables to support the rest of the team and the project's needs.
The Entity-Relationships (ERs) were expanded to accommodate the requirements of the other subsystems. These requirements were not considered in the initial design of the database system.
4. Designed and implemented over forty (40) Unit tests for the VTDS.
Additional unit tests were designed and implemented for the VTDS for the validation of the newly added requirements.
5. Developed API controllers in the VTIP to access the VTDS in a testable manner.
API controllers were added to the Vehicle Threat Intelligence Portal (VTIP) to allow access to the VTDS and enable automated testing.
6. Designed and implemented over twenty-five (25) unit tests for the VTIP and MISP integration.
Additional twenty-five (25) unit test cases were designed and implemented for the integration of the VTIP and the Malware Information Sharing Platform (MISP).
7. Incorporated the search functionality into the MISP API.
The search functionality was included into the MISP API to facilitate a thorough inspection of vehicle threats.
8. The Structured Threat Information eXchange (STIX) object submission was added on the MISP search interface after completing the interface with the VTIP.
Since most threat information are encoded using the STIX format, it would be an added advantage if the MISP is allowed to accept and process threat information in that format.

Chapter 13. Conclusion and Future Works

The project threaded together three very important and emerging security aspects of connected vehicles, namely security metrics, machine learning (ML), and threat intelligence. We designed applicable security metrics and developed their corresponding visualization systems. We envision these measurement and visual analytics tools to significantly advance the continuous improvement of vehicle security. With ML systems gaining wide acceptance and becoming almost ubiquitous, we seized the opportunity to enhance and to advance the security of connected vehicle messaging systems using artificial intelligence. We explored this prospect by developing prototype ML systems utilizing data collected from RSUs as input. We mutated the data by introducing synthetic malicious information to test the viability of the ML system in classifying benign and malicious communication traffic. The tests yielded very promising results. The third thrust of the project produced a prototypical vehicle threat intelligence system. This system demonstrates that security can be enhanced by enabling immediate access to vehicle vulnerability information to researchers, operators, manufacturers, supply chain, and the public in general. This enabler provides proactive means to mitigate risks and thereby promotes vehicle and personal safety.

Although the project objectives were successfully achieved and the feasibility of the proof-of-concept systems adequately demonstrated, there are opportunities for advancement that remain to be explored. The following suggested topics for future enhancement are worthy of vigorous pursuit:

1. Expand the Vehicle Threat Intelligence System to a fully functional and automated system;
2. Explore the feasibility of commercializing the Vehicle Threat Intelligence System;
3. Enhance and test the robustness of the Vehicle Security Metrics and Visualization System;
4. Offer the Vehicle Security Metrics and Visualization System as an open-source application and solicit contributions from application developers;
5. Investigate adversarial attacks on autonomous vehicle image and communication systems; and
6. Continue to explore the use of Machine Learning in augmenting the security of vehicle communication systems.

References

- [1] Forum of Incident Response and Security Teams (FIRST), "Common Vulnerability Scoring System version 3.1: Specification Document," June 2019. [Online]. Available: <https://www.first.org/cvss/specification-document>. [Accessed 13 February 2020].
- [2] G. A. Francia, "Connected Vehicle Security," in *15th International Conference on Cyber Warfare and Security (ICCWS 2020)*, Norfolk, VA, 2020.
- [3] NIST, "CSV-2019-13582 Detail," 15 November 2019. [Online]. Available: <https://nvd.nist.gov/vuln/detail/CVE-2019-13582>. [Accessed 20 February 2023].
- [4] Common Vulnerabilities and Exposure, "CVE-2018-9322," 31 May 2018. [Online]. Available: <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-9322>. [Accessed 13 February 2020].
- [5] Common Criteria Portal, "Common Criteria for Information Technology Security Evaluation," April 2017. [Online]. Available: <https://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R5.pdf>. [Accessed 24 February 2020].
- [6] MITRE Corporation, "Common Weakness Scoring System (CWSS)," 2 April 2018. [Online]. Available: https://cwe.mitre.org/cwss/cwss_v1.0.1.html.
- [7] Automated Vehicle Safety Consortium, "Best Practice for Metrics and Methods for Assessing Safety Performance of Automated Driving Systems (ADS)," SAE Industry Technologies Consortium, March 2021.
- [8] M. Elli, J. Wishart, S. Como, S. Dhakshinamoorthy and J. Weast, "Evaluation of Operational Safety Assessment (OSA) Metrics for Automated Vehicles in Simulation," SAE, 2021.
- [9] SAE, "Operational Safety Metrics for Verification and Validation (V&V) of Automated Driving Systems (ADS) J3237," SAE International, September 2020.
- [10] SAE, "Taxonomy and Definitions of ADS V&V J3208," SAE International, August 2019.
- [11] J. Wishart, Y. Chen, S. Como, N. Kidambi, D. Lu and Y. Yang, *Fundamentals of Connected and Automated Vehicles*, Warrendale, PA: SAE International, 2022.
- [12] L. Moukahal and M. Zulkernine, "Security Vulnerability Metrics for Connected Vehicles," in *2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, Sofia, Bulgaria, 2019.
- [13] C. McCarthy, K. Harnett and A. Carter, "A Summary of Cybersecurity Best Practices," US Department of Transportation (USDOT), Washington, D.C., 2014.
- [14] International Telecommunication Union (ITU), "Introduction to ASN.1," 2023. [Online]. Available: <https://www.itu.int/en/ITU-T/asn1/Pages/introduction.aspx>. [Accessed 17 May 2023].
- [15] SAE International, "DSRC Implementation Guide. A Guide to Users of SAE J2735 Message Sets over DSRC," SAE International, 2008.
- [16] SAE International, "On-Board System Requirements for V2V Safety Communications J2945/1_202004," 30 April 2020. [Online]. Available: https://www.sae.org/standards/content/j2945/1_202004. [Accessed 20 May 2023].

[17] Lockheed Martin, "The Cyber Kill Chain," 2024. [Online]. Available: <https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html>. [Accessed 7 February 2024].

APPENDICES

Connected Vehicle Security Metrics and Visualization

Research Methodology and Activities

Contract Number: BED34. Task Order: 977-01
Florida Department of Transportation (FDOT)

Guillermo A. Francia, III, Principal Investigator
Center for Cybersecurity
University of West Florida, USA
gfranciaiii@uwf.edu

Preface. The rapid advancement of connected and autonomous vehicles created new challenges for security and safety professionals. The sophistication of vehicle communication systems, found externally and internally, provides an added complexity to the issue. In security parlance, this is an expansion of the attack surface on vehicles. These challenges prompted the enhancement of existing and the development of new safety and security standards initiated by government, industry, and trade organizations. These initiatives clearly underscore the need to examine the state of connected vehicle security. For that reason, security metrics must be developed. As a major component of continuous improvement, quantitative and qualitative measures must be devised to be able to make a full appreciation of the process. This document presents our research methodology and pertinent activities in delineating what is already known and currently emerging and in adopting, enhancing, and creating security metrics for connected vehicles. An associated white paper is currently being compiled to provide a comprehensive exposition of this research.

I. RESEARCH METHODOLOGY

There are four research categories—theoretical, observational, experimental, and applied. Theoretical research focuses on formal methods with interrelated concepts, definitions, and propositions to develop model relationships and outcomes [18]. Observational research method observes subjects and phenomena in an environment expected by the study. Experimental or empirical research gathers or produces data in a controlled environment. Applied research focuses on the practical application of scientific methods to an identified problem such as the need for metrics to guide the continuous improvement of the security of connected vehicles. Our research method is predominantly that on the applied research category.

There three main types of applied research—evaluation research, research and development, and action research [19]. Evaluation research performs data analyses to arrive at outcomes that can be used for informed decision-making. Research and development research is that type of research that focuses on the development of solutions and services for some specific demand or need. Action research steers an organization to a particular business objective(s). Our research

methodology is inclined towards the research and development type. We delve deep into the literature looking for existing security metrics that can be adopted and formulate new metrics that are applicable to connected vehicle security.

II. LITERATURE REVIEW

The unprecedented advancement of technologies in both internal and external communication of connected vehicles imposes unwarranted consequences on their security and safety. Fortunately, the eagerness to deploy these technologies is tempered by government regulations. It is imperative that sound regulatory framework be put in place to ensure the security and safety of modern vehicles.

The ISO/SAE 21434 [20] came about when two organizations: ISO 26262 and SAE J3061 realized a common goal, i.e., automotive safety and security related standards. The ISO/SAE 21434 document titled “Road vehicles—Cybersecurity Engineering” established an effective global standard for automotive cybersecurity [21] [22]. The document provides the necessary vocabulary, objectives, requirements, and guidelines that are pertinent to cybersecurity engineering. Essentially, it enables organizations to define cybersecurity policies and processes, manage risks, and foster cybersecurity culture awareness [23].

The internal communication among electronic control units (ECUs), which are embedded devices that controls and automates a vehicle operation and performance, goes through inherently insecure channels, such as the Controller Area Network (CAN). For instance, a vulnerability, described by Trend Micro [24], enables a stealthy denial-of-service attack that practically works for every automotive vendor. This Exploitable hardware design flaws in some capacitive micro-electromechanical system (MEMS) accelerometer sensors produced by prominent automobile parts manufacturers were reported in another ICS-CERT alert: ICS_ALERT-17-073-01A in early 2017.

External communication systems in vehicles enable access convenience and online services [25]. Vehicle external communication can be classified into four main categories: vehicle-device (V2D) communication, vehicle-vehicle (V2V) communication, vehicle-infrastructure (V2I) communication, and vehicle-pedestrian (V2P) communication [2]. These are supported by cellular-vehicle-everything (C-V2X) [26] to enable a more efficient transportation ecosystem. With the advent of Electric Vehicles (EVs), a new type has emerged--vehicle-grid (V2G) communication [27].

II. CONNECTED VEHICLE SECURITY

The literature review steered the research to first focus on connected vehicle security specifically on threats, vulnerabilities, and attacks. We summarize in the following several compelling reasons for our ultimate objective, i.e., the derivation of vehicle security metrics.

1. Vehicle Data, Devices and Communication

Cybersecurity policies protect data, devices, and communication channels. Connected vehicles operate utilizing those three entities. The confidentiality, integrity, and availability of connected vehicle data is paramount to their safe operation. These data are generated and

consumed by devices in vehicles and are transmitted through various communication channel types as described above. Existing connected vehicle data sources, as described by Otonomo in [28]. Vehicle data provide several benefits such as acquisition cost reduction, lower operating cost, efficient logistics, effective data utilization, and expanded services [28]. However, these benefits introduce security tradeoffs which are discussed in subsequent sections.

2. Connected Vehicle Threats and Vulnerabilities

The Car Hacking Village, a conference track and interactive event at DEF CON, featured a study by NDIAS—a Japan-based automotive cybersecurity assessment group. In their study, the group tested over 40 ECUs provided by multiple manufacturers (15 in-vehicle infotainment units, 8 telematic control units, 8 gateways, advanced driver-assistance systems, smart key units, and electric vehicle chargers). The results of the study revealed more than 300 vulnerabilities in software and hardware components [29].

Threats and vulnerabilities in connected vehicles have been discovered in recent years and continue to proliferate. Prominent among these are the vulnerability of Tesla’s touch screen infotainment system [30], the vulnerability of the remote keyless system on Renault ZOE 2021 vehicles [31], the rolling-PWN replay vulnerability of keyless entry system on Honda vehicles [32], and the vulnerability in SiriusXM [33].

3. Vehicle Security Attacks and Mitigations

The inherently insecure Controller Area Network (CAN) in most connected vehicles is the focus of several studies [34] [35]. To alleviate those issues, research works, such as those using techniques such as Machine Learning [36] [37], authentication code [38], and clock skew signature [39], started to proliferate.

Other notable documented attack models and vulnerability mitigations include the works of Petit, Feiri, and Kargl [40] which was extended by Monteuis, et al. [41], the design, implementation, and evaluation of a hardware security module for a modern automotive vehicle [42], and the work of Lokman, et al. [43] on a systematic review of Intrusion Detection Systems (IDS) for automotive CAN bus system.

3. Vehicle Attack Surfaces

The inherently insecure Controller Area Network (CAN) in most connected vehicles is the focus of several studies [34] [35]. The lack of message authentication and the absence of data encryption are the main enabler of malicious activities in this network protocol. To alleviate those issues, research works, such as those using techniques such as Machine Learning [36] [37], authentication code [38], and clock skew signature [39], started to proliferate.

Other notable documented attack models and vulnerability mitigations such as those works by Petit, Feiri, and Kargl [40], Monteuis, et al. [41], Wolf and Gendrullis [42], Lokman et al. [43].

III. INDUSTRY AND GOVERNMENT INITIATIVES AND STANDARDS

There have been several initiatives towards the protection of a vehicle’s electronic control units. Notable examples are the E-safety Vehicle Intrusion Protected Application (EVITA) Project

[44], the Preparing Secure Vehicle-to-X Communication Systems (PRESERVE) Project [45], Secure Vehicular Communication (SeVeCom) Project [46], and the Society of Automotive Engineers (SAE) J3061 Guidebook [47]. In a very recent work by Bauer and Schartner [48], an illustration depicting attack surfaces and the classification of attack potential according to common criteria is presented. The presentation includes information on the difficulty and the impact of a certain exploit to an asset. Further, the work introduced a novel solution towards a realistic assessment of the integration of specialized countermeasures into the design of vehicular cybersecurity concepts.

The U.S. Government Accountability Office (GAO) Report on Vehicle Cybersecurity [49] contains, among others, the key security vulnerabilities in modern vehicles, the key practices and technologies to mitigate vehicle cybersecurity vulnerabilities, the challenges facing stakeholders, and the Department of Transportation's (DOT) efforts in addressing the issues in vehicle cybersecurity.

The Society of Automotive Engineers (SAE) Cybersecurity Guidebook for Cyber-Physical Vehicle Systems [47] describes a cybersecurity process framework from which an organization can develop processes to design and build cybersecurity in vehicular systems. The process framework covers the entire product lifecycle, including postproduction aspects with respect to service, incident monitoring, incident response, etc.

The National Highway Traffic Safety Administration (NHTSA) Automotive Security Best Practices for Modern Vehicles [50] presents the results and analysis of a review of best practices and observations in the field of cybersecurity involving electronic control systems across a variety of industry segments where the safety-of-life is concerned.

IV. VEHICLE SECURITY METRICS

There exists notable works on automotive vehicle security metrics. In [12], a set of security metrics for the software system in a connected vehicle is proposed. The set of metrics provides a quantitative indicator of the security vulnerability of the following risks on the system software: ECU coupling, communication, complexity, input and output data, and past security issues. The ECU coupling metric is based on the connectivity of the ECUs. Simply put, the risk is proportional to the extent of the connectivity of the ECUs. This proposed metric failed to take into account the fact that most vehicle networks are using the bus topology for interconnection. The communication risk metric is based on the number of communication technologies that are enabled on-board the vehicle. These are further normalized by the level of risk assigned to each of those technologies. The issue with this metric is that the assignment of risk level is quite arbitrary. The metric on input and output data risk takes into account the number of input data, the fixed and fluctuating properties of the input data, and the sensitivity level of output data. The authors argue that fluctuating input data and sensitive output data are more significant and should be given more emphasis in the calculation of security vulnerability. This metric failed to account the level of security testing that was applied to the vehicle's embedded system before deployment. Finally, the metric on security history utilizes the number of past attacks that occurred on the vehicle. This metric appears to assume the recurrence of an attack and that the vulnerability was never fixed. With system patches actively being carried out during vehicle recalls, this assumption is rather weak.

Use cases of Automotive Security Threats are described in [51]. The use cases include, among others, brake disconnect, horn activation, engine halt air bag, portable device injection, key fob cloning, cellular attack, and malware download. The threat matrix on each of these use cases includes attributes such as exploitable vulnerability, difficulty of implementation, resources needed, attack scenario, and outcome.

A Bayesian Network (BN) for connected and autonomous vehicle cyber-risk classification was developed by Sheehan, et al. [52]. The BN model uses the Common Vulnerability Scoring System (CVSS) software vulnerability risk-scoring framework for input parameters specifically on the Global Positioning System (GPS) jamming and spoofing.

In the following section, we present a collection of connected vehicle security metrics that we derived using a similar work on critical infrastructure and industrial controls systems security [53] [54].

1. Common Vulnerability Scoring System (CVSS)

CVSS is an open framework for communicating the characteristics and severity of software vulnerabilities. It consists of three metric groups: Base, Temporal, and Environmental. Details on these metrics can be found in [1].

In [2], an illustration on the application of these metrics on the vulnerability of the Tesla Model S/X vehicles manufactured before March, 2018 [3] and the vulnerability in the infotainment component of BMW Series vehicles (CVE-2018-9322) [4] are illustrated.

In the accompanying white paper, we shall provide another use case to illustrate the applicability of this metric.

2. Common Methodology for IT Security Evaluation (CEM) [5]

The CEM is a companion document to the Common Criteria for Information Technology Security Evaluation (CC). It defines the minimum actions to be taken by an evaluator conducting a CC evaluation utilizing the criteria and evidence as stated in the CC.

We specifically examined the attack potential on an automotive vehicle based on the following factors: Elapsed time, Specialist expertise, Knowledge of the target, Window of opportunity, and IT hardware/software or other equipment. We shall provide a detailed description of each of these factors in the accompanying white paper.

3. Threats on Assets

We identify the following threats on vehicle assets and derive the attack potential metrics. We apply the previously defined factors for analysis and aggregate the metrics. The threats on vehicle assets are the following: False Data from ECU, Blocking of CAN Bus, Malicious Software, Denial of Telematics Service, Unauthorized Access, Command Injection, Masquerading, Data Tampering. We shall provide a detailed description of each of these threats in the accompanying white paper.

The total attack potential for each threat is simply a summation of the value assigned to each of the attribute of a successful attack. These results can be utilized during the decision-making process of cybersecurity asset allocation towards risk mitigation or prevention.

4. Common Weakness Scoring System [6]

The Common Weakness Scoring System (CWSS) is a mechanism for evaluating software weaknesses in a consistent, flexible, open manner. It is a community-based undertaking which addresses the need for prioritizing the software vulnerability issues. The measurements are organized into three metric groups: Base Finding, Attack Surface, and Environmental. The groups, including their subgroups, as described in [6], will be fully expounded in a forthcoming white paper.

5. Operational Safety Assessment Metrics

We next turn our attention on the impact of cybersecurity to operational safety assessment (OSA). There exists several OSA metrics that have been proposed, adopted, and studied [7] [8]. SAE J3237 [9], a work in progress information report, is currently being developed. This report provides definitions and lexicon for describing operational safety metrics for ADS vehicles. The characteristics of the listed metrics include the following: definition, data source, subjectivity, observable variable, formulation, subjective assumptions and thresholds, and origin. A related work by the SAE V&V Task Force is the development of a proposed taxonomy for a Recommended Practice on Operational Safety metrics [10]. At the classification level of the proposed taxonomy are the operational safety metrics [10]. These operational safety metrics, fully covered in the forthcoming white paper, will become the foundation of the OSA metrics that we have derived and augmented with the following:

- *Authentication Metric.* This OSA metric measures the quality of the authentication system deployed in the vehicle. This is extremely useful in modern vehicles that rely on communications such as those in V2V or V2I environment.
- *Physical Access Metric.* This OSA metric measures the strength of physical access protection of vehicle controls. An example is the unsecured physical access to an OBD port which could compromise the vehicle's CAN bus.
- *Communication Channel Metric.* This OSA metric pertains to the quality of the communication channel used by the vehicle.

5.1 Cybersecurity Metrics for Operational Safety

We investigate the impact of cybersecurity to operational safety. In doing so, we devise cybersecurity metrics that have close affinities with OSA metrics. These cybersecurity metrics for operational safety are described in the following:

- *Safety Envelope Metric.* This cybersecurity metric measures the security resiliency of a connected vehicle to be able to maintain a safe boundary amidst a cyber intrusion incident. An example is a vehicle's capability in preventing malicious manipulation of the control and sensing systems that enable safe distance driving operation. Values range from 0.0 for least resilient to 1.0 for most security resilient.

- *Behavioral Metric.* This cybersecurity metric measures the vehicle's capability to protect against a cyber-attack that enables the improper behavior of the subject vehicle. An example of such attack is the manipulation of the vehicle cruise control mechanism. Values range from 0.0 for least capable to 1.0 for most capable.
- *Component Metric.* This is a measure of the susceptibility of the vehicle components to cyber-attack. For example, an Electronic Control Unit (ECU) device originating from an unverifiable supply chain may be highly susceptible to cyber-attack. Values range from 0.0 for most susceptible resilient to 1.0 for least susceptible.
- *Sensing Metric.* This cybersecurity metric pertains to the integrity and accuracy of data collected by the vehicle sensors. Roadside Units (RSUs) that are not properly secured may produce inaccurate or tampered data. Values range from 0.0 for least reliable data to 1.0 for most reliable data.
- *Perception Metric.* This cybersecurity metric pertains to the security of the system that provides for the interpretation of environment data collected by the vehicle sensors. For example, an insecure image processing system that is highly susceptible to an attack may provide inaccurate interpretation of traffic signs or signals. Values range from 0.0 for least secure to 1.0 for most secure.
- *Planning Metric.* This cybersecurity metric measures the vulnerability of the trajectory planning system to malicious intrusion. Values range from 0.0 for most vulnerable to 1.0 for least vulnerable.
- *Control Metric.* This cybersecurity metric measures the vulnerability of the vehicle's control system to malicious intrusion. Values range from 0.0 for most vulnerable to 1.0 for least vulnerable.
- *Authentication Metric.* This cybersecurity metric measures the security posture of the authentication system deployed in the vehicle. Values range from 0.0 for least secure to 1.0 for most secure.
- *Physical Access Metric.* This cybersecurity metric measures the strength of physical access protection of vehicle controls. Values range from 0.0 for least physically secure to 1.0 for most physically secure.
- *Communication Channel Metric.* This cybersecurity metric pertains to the level of protection of the communication channel used by the vehicle. Security characteristics of data transmission such as encryption, authentication, and attribution are pertinent concerns in this metric. Values range from 0.0 for least secure to 1.0 for most secure.

Emulating the evaluation methodology of the OSA metrics that was introduced by Wishart, et.al. [11], we present four evaluation factors for the formulation of the aggregation of cybersecurity metrics. The four evaluation factors are described in the following:

- *Reliability.* This factor quantifies the fidelity of the sources of measurement data. For instance, data originating from actual events carry a higher value than those from simulated events. Values range from 0.1 for less reliable to 1.0 for most reliable.
- *Relevance.* This factor quantifies the relevance of the measurement to a subject vehicle. This value may vary according to the specificity of data such as make and model of the subject vehicle. Data for a Honda CRV is more specific than data that refers to Honda vehicles in general. Values range from 0.1 for least relevant to 1.0 for most relevant.
- *Extent.* This factor quantifies the scope or extensiveness of the measurement data. The value ranges from 0.1 for least extensive to 1.0 for most extensive.

- *Criticality*. This factor quantifies the gravity of a specific metric. For instance, security measurement on control will carry a heavier weight than that on safety envelope. The value ranges from 0.1 for least critical to 1.0 for most critical.

The Aggregate Security Metric (ASM) for a specific vehicle is calculated as

$$ASM = Reliability \times Relevance \times Extent \times \left(\sum_{k=1}^N Criticality_k \times Security_Metric_k \right)$$

The ASM value will range from 0 to 10.

6. Security Vulnerability Metrics for Connected Vehicles

The purpose of security vulnerability metrics is to provide guidance to security engineers and testers using security vulnerability metrics that measure weak or vulnerable features in the software system of connected vehicles. These metrics are based on the seminal work of Moukahal and Zulkernine [12]. We describe each of the following risks on connected vehicles that may eventually contribute to the likelihood of vulnerability exploitation.

6.1 ECU Coupling Risk

This risk is manifested by level of interconnection among ECU components. This means the higher the coupling value the higher is the probability for vulnerabilities. Thus, for every functionality, F, for all ECUs, N, and for all communication links between ECU j and ECU k, the ECU coupling risk, R_{EC} , is calculated as

$$R_{EC}(F) = \sum_{j=1, k=1}^N C_{jk}$$

$C_{jk}=1$ if there is at least one information transfer between ECU j and ECU k; 0 otherwise.

$$\text{Max}(R_{EC}(F)) = N$$

6.2 Communication Channel Risk

This risk is based on the communication channel types that are available for connected vehicles: vehicle to vehicle (V2V), vehicle to infrastructure (V2I), user to vehicle (U2V), and intra-vehicle (IV). The communication risk, for each functionality, F, is calculated according to the following formula:

$$R_{CC}(F) = \sum_{j=1}^N w_j C_j$$

Where N is the number of communication links, w_j the weight of a specific communication channel type based on its propensity to vulnerability, and C_j is 1 if the functionality uses the channel; 0 otherwise.

$$\text{Max}(R_{CC}(F)) = \text{Total number of all communication channels}$$

6.3 Complexity Risk

This risk is associated with the number of defects in software used in automotive vehicles. The complexity metric in software is an excellent indicator of vulnerabilities. The Halstead Complexity measure is a standard way of deriving the complexity of software. Thus, for calculating the complexity of the functionalities in connected vehicle, we use the formula:

$$R_{SC} (F) = SLOC + \alpha (Nesting)$$

Where SLOC is the Source Line of Code, Nesting is the number of control structures, and α is the weight, with value over one, indicating complexity of the nesting structure.

$$\text{Max} (R_{SC} (F)) = SLOC + 10 (Nesting)$$

6.4 Input and Output Data Risk

This risk involves the input and output data in a connected vehicle. The metric distinguishes between a Fixed Input (FI) from a Fluctuating Input (LI). It also distinguishes an Insensitive Output (IO) from a Sensitive Output (SO). Weights (α , β) are added to highlight the significance of the Fluctuating Input and the Sensitive Output. To calculate the Input and Output Data Risk, we use the formula:

$$R_{DIO} (F) = FI + \alpha (LI) + IO + \beta (SO)$$

$$\text{Max} (R_{DIO} (F)) = FI + 5(LI) + IO + 5(SO)$$

6.5 History of Security Issues

This risk considers the past security issues of a certain vehicle functionality. Given Y as the total number of years since the first car attack and α_y as the number of attacks that occurred in year y. A forgetting factor, λ , is introduced to provide relevancy to the attacks that occurred in more recent years, where $0 \leq \lambda \leq 1$. To calculate the risk of a vehicle functionality using the history of security issues, we use the formula:

$$R_{HS} (F) = \sum_{y=1}^Y \alpha_y \lambda^{Y-y}$$

For a 2-year comparison, the calculation simply boils down to

$$\lambda = 1 - (\alpha_1 / \alpha_2) \quad \leftarrow \text{the forgetting factor}$$

$$R_{HS} (F) = \alpha_1 (\lambda)^{Y-1} + \alpha_2$$

$$\text{Max} (R_{HS} (F)) = \alpha_1 + \alpha_2$$

6.6 Overall Security Vulnerability Metric

The overall security vulnerability metric of a certain functionality in a connected vehicle is calculated by first normalizing the values of each of the metrics and applying a weighting factor

($\alpha, \beta, \gamma, \delta, \phi$), which indicates its significance to the overall scheme. The metrics are added to obtain the overall value, which is in direct correlation with the vulnerability level of the functionality. The formula is shown as follow:

$$OSV = \alpha \left[\frac{R_{EC}(F)}{\max(R_{EC}(F))} \right] + \beta \left[\frac{R_{CC}(F)}{\max(R_{CC}(F))} \right] + \gamma \left[\frac{R_{SC}(F)}{\max(R_{SC}(F))} \right] \\ + \delta \left[\frac{R_{DIO}(F)}{\max(R_{DIO}(F))} \right] + \phi \left[\frac{R_{HS}(F)}{\max(R_{HS}(F))} \right]$$

7. Vehicle Security Best Practices Assessment Metrics

Vehicle Security Best Practices Assessment Metrics are designed based on the National Highway Traffic Safety Administration (NHTSA) Report DOT HS812 075 [13]. The report contains a review and analysis of cybersecurity best practices involving automotive vehicles.

The study utilizes the iterative Information Security Life Cycle divided into four phases and processes. The four phases and processes are described in the following.

Assessment Phase. This phase includes the development and implementation of security policies, the evaluation of system security, and the processes of risk assessment.

Design Phase. This phase entails the prioritization of systems and resources applicable to security and design and analysis of the system’s security architecture.

Implementation Phase. This phase covers the steps taken in vulnerability remediation and the processes in security testing and evaluation.

Operation Phase. This phase includes the security awareness training for all personnel, customers, and other stakeholders. It also includes continuous security monitoring, intrusion detection and response.

The four phases will be fully described in the forthcoming white paper.

Vehicle Security Best Practices Assessment Metrics

We propose the following vehicle security best practice assessment metrics based on the Information Security Life Cycle described above. The metrics are built by a self-assessment form, shown on Table 3, which consists of a checklist of the status of each of the four phases.

Table 3. Vehicle Security Best Practices Assessment Checklist

Process	Checklists	Status
Security Policy	Are security policies established?	
	Are security policies properly documented and widely disseminated within the organization?	
	Are security policies strictly enforced?	
	Are security policies periodically reviewed/updated?	
Data Security & Privacy	Is collected/stored data protected/encrypted?	
	Is transmitted data encrypted?	
	Is there a control mechanism for sharing data?	

	Does the site comply with data protection standards and regulations (e.g., ISO/IEC 27001 certification, GDPR)?	
Risk Assessment	Do you conduct a periodic risk assessment of vehicle cybersecurity? Is there a developed and implemented organization-wide risk management strategy? Is there a Supply Chain Risk Management (SCRM) policy? Are security controls in place and periodically evaluated and/or enhanced?	
System Protection & Prioritization	Have you implemented security-by-design principles during the vehicle design phase? Have you implemented domain separation for in-vehicle networks (i.e. limiting the communication between the safety-critical and non-safety critical domains)? Does the organization triage the identified risks according to priority for resource allocation? Do you have a comprehensive system security test plan?	
Security Architecture	Have you implemented a layered approach to vehicle security (ECU level, in-vehicle network level, V2V level, V2X level)? Is there a periodic evaluation of the system's security architecture? Do you maintain an inventory of operational software components used in each automotive ECU and assembled vehicle? Have you considered the risks and vulnerabilities associated with vehicle sensor devices?	
Remediation & Implementation	Are there established mechanisms to update vehicle software and firmware remotely and securely? Are appropriate security controls implemented and are in place? Do you have an established remediation process? Is the remediation plan evaluated and implemented?	
Security Test & Evaluation	Have you conducted a thorough code review on the vehicle software? Have you conducted penetration testing on connected vehicle communication systems before deployment? Are security controls tested and evaluated for compliance with security performance specifications? Do you conform with secure software development best practices as outlined in NIST 8151 and ISO/SAE 21434?	
Awareness & Security Training	Is there a periodic security awareness training program for the entire workforce? Is security risk and mitigation disclosure available to the consumer and other stakeholders?	

	<p>Do you evaluate the effectiveness of the security awareness training program and introduce improvements if needed?</p> <p>Do you collect, maintain, analyze, and share information related to cybersecurity through the Automotive Information Sharing and Analysis Center (Auto-ISAC)?</p>	
Intrusion Detection & Response	<p>Is there an Incident Response Plan (IRP) in place?</p> <p>Is the IRP periodically tested, evaluated, and updated?</p> <p>Do you have a systematic process for continuous risk and security monitoring?</p> <p>Are security incidents properly documented and reported?</p>	

V. CONNECTED VEHICLE SECURITY METRICS VISUALIZATION

Visualization takes advantage of cognitive perception in effectively presenting information to users. It offers a powerful means of recognizing trends and patterns that are not easily recognized using non-visual methods. In essence, the cognitive reasoning process is augmented by perception to bring about a more rapid analytical reasoning process [55]. There exist numerous works on information security visualization, e.g. [56], [57].

As an extension to this research, we ventured on vehicle security metrics visualization. We designed and are in the process of implementing a visualization system for each of the security metrics that are described in the preceding sections. A detailed description of each of the visualization component is found in another manuscript: the Vehicle Security Metrics Visualization System Specification, Design, and Implementation Document. The visualization system design prototypes will be discussed and shown in the forthcoming white paper.

VI. ACKNOWLEDGEMENT

This research is partially funded by a grant from the Florida Department of Transportation (FDOT). It is intended to support FDOT’s mission to provide a safe transportation system to ensure the mobility of people and goods. To illustrate its direct support to this mission, the impact of cybersecurity on operational safety is presented. Henceforth, cybersecurity metrics for operational safety are derived and developed (see Section IV.5).

The research presents a holistic treatment of the security of connected vehicles. It covers both the intranet (internal connectivity) and internet (external connectivity) systems of connected vehicles. The derived security metrics both implicitly and explicitly support the operational safety of connected vehicles—the primary concern of FDOT. We describe various risks found on connected vehicles that may eventually contribute to the likelihood of vulnerability exploitation. A successful attack on communication channels (V2V or V2X), ECU couplings, Input and Output Data, Supply Chain, or other security vulnerabilities could easily be leveraged to attack the entire connected vehicle ecosystem.

As a stretch objective, we conclude the report with vehicle security best practice assessment metrics. These security metrics provide a significant impact on the safety of connected vehicles.

Auto-ISAC, in their Best Practices Guides, recognized the proactive collaboration of various organizations and the automotive industry in protecting consumer safety through a robust vehicle cybersecurity [58].

This report is intended for security practitioners, designers, manufacturers, technology providers, service providers, infrastructure owner-operators, and transportation agencies and regulators.

VII. ADDITIONAL RESOURCES

IEEE 1609.2.1-2020. *IEEE Standard for Wireless Access in Vehicular Environments (WAVE)—Certificate Management Interfaces for End Entities*. URL: <https://standards.ieee.org/ieee/1609.2.1/10172/>. Last Access: December 1, 2022.

National Electrical Manufacturers Association (NEMA). *Cyber and Physical Security for Intelligent Transportation Systems (ITS)*. NEMA TS 8-2018. URL: <https://www.nema.org/Standards/view/Cyber-and-Physical-Security-for-Intelligent-Transportation-Systems-ITS>. Last Access: December 4, 2022.

Society of Automotive Engineers (SAE), (2020). *Service Specific Permissions and Security Guidelines for Connected Vehicle Applications*. J2945/5_202002. URL: https://www.sae.org/standards/content/j2945/5_202002/. Last Access: December 1, 2022.

US Department of Transportation (USDOT), (2020). *Module CSE202: Introduction to Cybersecurity for Transportation Agencies*. URL: <https://www.pcb.its.dot.gov/StandardsTraining/mod64/ppt/m64ppt.pdf>. Last Access: December 1, 2022.

US Department of Transportation (USDOT), (2022). *ARC-IT 9.1. The National ITS Reference Architecture: Security*. URL: <https://www.arc-it.net/html/security/security.html>.

Xiong, W., Legrand, E., Aberg, O., & Lagerstrom, R., Cyber Security Threat Modeling Based on the MITRE Enterprise ATT&CK Matrix. *Software and Systems Modeling* (2022) 21:157-177.

VII. REFERENCES

- [1] Forum of Incident Response and Security Teams (FIRST), "Common Vulnerability Scoring System version 3.1: Specification Document," June 2019. [Online]. Available: <https://www.first.org/cvss/specification-document>. [Accessed 13 February 2020].
- [2] G. A. Francia, "Connected Vehicle Security," in *15th International Conference on Cyber Warfare and Security (ICWS 2020)*, Norfolk, VA, 2020.
- [3] NIST, "CSV-2019-13582 Detail," 15 November 2019. [Online]. Available: <https://nvd.nist.gov/vuln/detail/CVE-2019-13582>. [Accessed 20 February 2023].

- [4] Common Vulnerabilities and Exposure, "CVE-2018-9322," 31 May 2018. [Online]. Available: <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-9322>. [Accessed 13 February 2020].
- [5] Common Criteria Portal, "Common Criteria for Information Technology Security Evaluation," April 2017. [Online]. Available: <https://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R5.pdf>. [Accessed 24 February 2020].
- [6] MITRE Corporation, "Common Weakness Scoring System (CWSS)," 2 April 2018. [Online]. Available: https://cwe.mitre.org/cwss/cwss_v1.0.1.html.
- [7] Automated Vehicle Safety Consortium, "Best Practice for Metrics and Methods for Assessing Safety Performance of Automated Driving Systems (ADS)," SAE Industry Technologies Consortium, March 2021.
- [8] M. Elli, J. Wishart, S. Como, S. Dhakshinamoorthy and J. Weast, "Evaluation of Operational Safety Assessment (OSA) Metrics for Automated Vehicles in Simulation," SAE, 2021.
- [9] SAE, "Operational Safety Metrics for Verification and Validation (V&V) of Automated Driving Systems (ADS) J3237," SAE International, September 2020.
- [1] SAE, "Taxonomy and Definitions of ADS V&V J3208," SAE International, August 2019.
- [0]
- [1] J. Wishart, Y. Chen, S. Como, N. Kidambi, D. Lu and Y. Yang, *Fundamentals of Connected and Automated Vehicles*, Warrendale, PA: SAE International, 2022.
- [1] L. Moukahal and M. Zulkernine, "Security Vulnerability Metrics for Connected Vehicles,"
- [2] in *2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, Sofia, Bulgaria, 2019.
- [1] C. McCarthy, K. Harnett and A. Carter, "A Summary of Cybersecurity Best Practices," US
- [3] Department of Transportation (USDOT), Washington, D.C., 2014.
- [1] International Telecommunication Union (ITU), "Introduction to ASN.1," 2023. [Online].
- [4] Available: <https://www.itu.int/en/ITU-T/asn1/Pages/introduction.aspx>. [Accessed 17 May 2023].
- [1] SAE International, "DSRC Implementation Guide. A Guide to Users of SAE J2735 Message
- [5] Sets over DSRC," SAE International, 2008.
- [1] SAE International, "On-Board System Requirements for V2V Safety Communications
- [6] J2945/1_202004," 30 April 2020. [Online]. Available: https://www.sae.org/standards/content/j2945/1_202004. [Accessed 20 May 2023].
- [1] Lockheed Martin, "The Cyber Kill Chain," 2024. [Online]. Available:
- [7] <https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html>. [Accessed 7 February 2024].
- [1] T. W. Edgar and D. O. Manz, "Chapter 7 - Theoretical Research," in *Research Methods for*
- [8] *Cyber Security*, Syngress, 2017, pp. 177-192.
- [1] Formplus, "What is Applied Research? +[Types, Examples & Methods]," 22 July 2022.
- [9] [Online]. Available: <https://www.formpl.us/blog/applied-research>.

- [2 International Organization for Standardization (ISO), "ISO/SAE 21434:2021 Road vehicles
0] — Cybersecurity engineering," August 2021. [Online]. Available:
<https://www.iso.org/standard/70918.html>.
- [2 Upstream Security Ltd., "ISO/SAE 21434: Setting the Standard for Automotive
1] Cybersecurity," 2020. [Online]. Available:
https://info.upstream.auto/hubfs/White_papers/Upstream_Security_Setting_the_Standard_for_Automotive_Cybersecurity_WP.pdf?_hsmi=87208721&_hsenc=p2ANqtz-8ke_6RWU7hkISDBzRoHFeUhfbaRRQ7E9-Z2bvc4YMIP3JNvc42_oh1ZxJ5jtWQOUItehUaSmp7MfNDcwzbzUWoZjrGHw.
[Accessed 5 November 2020].
- [2 C. Schmittner, G. Griessnig and Z. Ma, "Status of the Development of ISO/SAE 21434," in
2] *Proc of the 25th European Conference, EuroSPI 2018*, Bilbao, Spain, 2018.
- [2 ISO/SAE, "ISO/SAE 21434:2021(en) Road vehicles--Cybersecurity engineering," 2021.
3] [Online]. Available: <https://www.iso.org/obp/ui/#iso:std:iso-sae:21434:ed-1:v1:en>.
- [2 Trend Micro, "A Vulnerability in Modern Automotive Standards and How We Exploited It,"
4] July 2017. [Online]. Available: <https://documents.trendmicro.com/assets/A-Vulnerability-In-Modern-Automotive-Standards-and-How-We-Exploited-It.pdf>. [Accessed November 2018].
- [2 A. Karahasanovic, "Automotive Cyber Security," Chalmers University of Technology
5] University of Gothenburg, Gotehnborg, Sweden, 2016.
- [2 Qualcomm, Inc., "C-V2X: A new era of smart transporation in the United States," 17
6] February 2023. [Online]. Available: https://www.qualcomm.com/content/dam/qcomm-martech/dm-assets/documents/C-V2X_Whitepaper.pdf.
- [2 IEEE, "Vehicle to Grid (V2G) Technology," 18 February 2023. [Online]. Available:
7] <https://innovationatwork.ieee.org/vehicle-to-grid-v2g-technology/>.
- [2 Otonomo, "The Promise of Connected Vehicle Data," 16 February 2023. [Online].
8] Available: <https://info.otonomo.io/hubfs/PDF/OOOO-Smart-Cities-Survey-Promise-of-data.pdf>.
- [2 J. Tyrrell, "Trends in ECU vulnerabilities highlighted at DEF CON 2020," 20 August 2020.
9] [Online]. Available: <https://www.securecav.com/trends-in-ecu-vulnerabilities-highlighted-at-def-con-2020/>. [Accessed February 2023].
- [3 D. Pauli, "Hackers Hijack Tesla Model S from Afar, While the Cars are Moving," 16
0] September 2016. [Online]. Available:
https://www.theregister.co.uk/2016/09/20/tesla_model_s_hijacked_remotely/. [Accessed October 2019].
- [3 NIST-NVD, "National Vulnerability Database," 10 January 2023. [Online]. Available:
1] <https://nvd.nist.gov/vuln/detail/CVE-2022-38766>.
- [3 B. Toulas, "Hackers can unlock Honda cars remotely in Rolling-PWN attacks," 13 July
2] 2022. [Online]. Available: <https://www.bleepingcomputer.com/news/security/hackers-can-unlock-honda-cars-remotely-in-rolling-pwn-attacks/>. [Accessed February 2023].
- [3 Upstream Security, Inc., "Security researchers manage to control multiple vehicles from
3] various OEMs by exploiting an API based vulnerability in telematics service provider,"

November 2022. [Online]. Available: <https://upstream.auto/research/automotive-cybersecurity/?id=12360>.

- [3 L. Pan, X. Zheng, H. X. Chen, T. Luan, H. Bootwala and L. Batten, "Cyber security attacks
4] to modern vehicular systems," *J. Inf. Secur. Appl.*, vol. 36, pp. 90-100, October 2017.
- [3 S. Woo, H. J. Jo and D. H. Lee, "A practical wireless attack on the connected car and
5] security protocol for in-vehicle CAN," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 993-1006, April 2015.
- [3 M. J. Kang and J. W. Kang, "Intrusion detection system using deep neural network for in-
6] vehicle network security," *PLoS ONE*, vol. 11, no. 6, 2016.
- [3 O. Avatefipour, A. S. Al-Sumaiti, A. M. El-Sherbeeney, E. M. Awwad, A. Elmeligy, M. A.
7] Mohamed and H. Malik, "An Intelligent Secured Framework for Cyberattack Detection in Electric Vehicles' CAN Bus Using Machine Learning," *IEEE Access*, vol. 7, 2019.
- [3 Q. Wang and S. Sawhney, "VeCure: A Practical Security Framework to Protect the CAN
8] Bus of Vehicles," in *International Conference on the Internet of Things (IOT)*, Cambridge, MA, 2014.
- [3 K.-T. Cho and K. G. Shin, "Fingerprinting Electronic Control Units for Vehicle Intrusion
9] Detection," in *Proceedings of the 25th USENIX Security Symposium (USENIX Security 16)*, 2016.
- [4 J. Petit, M. Feiri and F. Kargl, "Revisiting attacker model for smart vehicles," in *2014 IEEE
0] 6th International Symposium on Wireless Vehicular Communications, WiVec 2014 Proceedings*, 2014.
- [4 J.-P. Monteuis, J. Petit, J. Zhang, H. Labiod, S. Mafrica and A. Serval, "Attacker Model for
1] Connected and AUtomed Vehicles," in *ACM Computer Science in Cars Symposium (CSCS'18)*, Berlin, Germany, 2018.
- [4 M. Wolf and T. Gendrullis, "Design, Implementation, and Evaluation of a Vehicular
2] Hardware Security Module," in *14th International Conference on Information Security and Cryptology*, Seoul, South Korea, 2011.
- [4 S. Lokman, T. Othman and M. Abu-Bakar, "Intrusion Detection System for Automotive
3] Controller Area Network (CAN) Bus System: a Review," *EURASIP Journal on Wireless Communications and Networking*, vol. 184, 2019.
- [4 EVITA Project, "EVITA E-Safety Vehicle Intrusion Protected Applications," 01 December
4] 2011. [Online]. Available: <https://www.evita-project.org/>. [Accessed 13 November 2018].
- [4 PRESERVE, "About the Project," June 2015. [Online]. Available: [https://preserve-](https://preserve-project.eu/about)
5] [project.eu/about](https://preserve-project.eu/about). [Accessed 12 October 2019].
- [4 SeVeCom, "Security on the Road," 2008. [Online]. Available: <https://www.sevecom.eu/>.
6] [Accessed 13 October 2019].
- [4 Society of Automotive Engineers (SAE), "Cybersecurity Guidebook for Cyber-Physical
7] Vehicle Systems J3061," 12 January 2012. [Online]. Available: <https://www.sae.org/standards/content/j3061/>. [Accessed 13 October 2019].
- [4 S. Bauer and P. Schartner, "Reducing Risk Potential by Evaluating Specialized
8] Countermeasures for Electronic Control Units," in *17th escar Europe conference 2019*, Stuttgart, Germany, 2019.

- [4] Government Accountability Office (GAO), United States, "Vehicle Cybersecurity: DOT and Industry Have Efforts Under Way, but DOT Needs to Define Its Role in Responding to a Real-world Attack. GAO Report 16-350.," 2016. [Online]. Available: <https://www.gao.gov/assets/680/676064.pdf>. [Accessed 14 November 2018].
- [5] C. McCarty, K. Harnett and A. Carter, "A Summary of Cybersecurity Best Practices," 0] National Highway Traffic Safety Administration, Washington, DC, 2014.
- [5] C. McCarthy, K. Harnett and A. Carter, "Characterization of Potential Security Threats in 1] Modern Automobiles: A Composite Modeling Approach.," October 2014. [Online]. Available: <https://rosap.nhtl.bts.gov/view/dot/12119>. [Accessed 25 February 2020].
- [5] B. Sheehan, F. Murphy, M. Mullins and C. Ryan, "Connected and autonomous vehicles: A 2] cyber-risk classification framework," *Transportation Research Part A*, vol. 124, pp. 523-536, 2019.
- [5] G. A. Francia and X. P. Francia, "Critical Infrastructure Protection and Security 3] Benchmarks," in *Encyclopedia of Information Science and Technology, 3rd Edition*, Hershey, PA, IGI Global, 2015, pp. 4267-4278.
- [5] G. Francia, "Baseline Operational Security Metrics for Industrial Control Systems," in 4] *International Conference on Security and Management*, Las Vegas, NV, 2016.
- [5] G. A. Francia and S. Jarupathirun, "Security Metrics-Review and Research Directions," in 5] *Proceedings of the 2009 International Conference on Security and Management*, Las Vegas, NV, 2009.
- [5] G. Conti, M. Ahamad and J. Stasko, "Attacking Information Visualization System Usability 6] Overloading and Deceiving the Human," in *SOUPS 2005*, Pittsburgh, PA, 2005.
- [5] H. Hochheiser and B. Schneiderman, "Using Interactive Visualizations of WWW Log Data 7] to Characterize Access Patterns and Inform Site Design," *Journal of the American Society for Information Science and Technology*, vol. 52, no. 4, pp. 331-343, 2001.
- [5] Auto-ISAC, Inc., "Best Practices," 16 February 2023. [Online]. Available: 8] <https://automotiveisac.com/best-practices/>.
- [5] Upstream Security, Inc., "Recent spike in car thefts prompts South Korean OEMs to offer 9] security kits in the US," September 2022. [Online]. Available: <https://upstream.auto/research/automotive-cybersecurity/?id=12080>.
- [6] G. A. Francia, III, "Vehicle Network Security Metrics," in *Advances in Cybersecurity 0] Management*, Cham, Switzerland, Springer Nature, 2021, pp. 55-73.
- [6] Fortinet, "What Is An Attack Surface?," 2023. [Online]. Available: 1] <https://www.fortinet.com/resources/cyberglossary/attack-surface>. [Accessed February 2023].
- [6] C. Maple, M. Bradbury, A. T. Le and K. Ghirardello, "A Connected and Autonomous 2] Vehicle Reference Architecture for Attack Surface Analysis," *Appl. Sci.*, vol. 9, no. 23, 2019.
- [6] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. 3] Kantor, D. Anderson, H. Shacham and S. Savage, "Experimental Security Analysis of a Modern Automobile," in *2010 IEEE Symposium on Security and Privacy*, Berkeley/Oakland, CA, 2010.
- [6] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, N. Shacham, S. Savage, K. Koscher, A. 4] Czeskis, F. Roesner and T. Kohno, "Comprehensive experimental analyses of automotive

- attack surfaces," in *20th USENIX Conference on Security (SEC'11)*, San Francisco, CA, 2011.
- [6 US Department of Transportation, "Cybersecurity Best Practices for the Safety of Modern
5] Vehicles," 9 September 2022. [Online]. Available:
<https://www.govinfo.gov/content/pkg/FR-2022-09-09/pdf/2022-19507.pdf>. [Accessed 18
February 2023].
- [6 Society of Automotive Engineers (SAE) International, "Hardware Protected Security for
6] Ground Vehicles," 10 February 2020. [Online]. Available:
https://www.sae.org/standards/content/j3101_202002/. [Accessed 12 November 2020].
- [6 British Standard Institution, "IATF 16949:2016 Automotive Quality Management," 2020.
7] [Online]. Available: [https://www.bsigroup.com/en-US/iatf-16949-automotive/introduction-
to-iatf-16949/](https://www.bsigroup.com/en-US/iatf-16949-automotive/introduction-to-iatf-16949/). [Accessed 12 November 2020].
- [6 American National Standards Institute (ANSI), "ISO/IEC/IEEE 29119-1:2013," 2020.
8] [Online]. Available:
[https://webstore.ansi.org/Standards/ISO/ISOIECIEEEE291192013?gclid=CjwKCAiA17P9B
RB2EiwAMvwNyKt4mT9KW0hN-
taVxEzZBa7nN5sfZQzDV6HdWGRQddq5dVFT6Pv8LxoCQrEQAvD_BwE](https://webstore.ansi.org/Standards/ISO/ISOIECIEEEE291192013?gclid=CjwKCAiA17P9BRB2EiwAMvwNyKt4mT9KW0hN-taVxEzZBa7nN5sfZQzDV6HdWGRQddq5dVFT6Pv8LxoCQrEQAvD_BwE). [Accessed 12
November 2020].
- [6 S. Saydjari, "Is Risk a Good Security Metric?," in *Proceedings of the 2nd ACM Workshop
9] on Quality of Protection*, 2006.
- [7 S. Schechter, "Toward Econometric Models of Security Risk from Remote Attack," *IEEE
0] Security and Privacy*, pp. 40-44, January-February 2005.
- [7 P. Manadhata and J. Wing, "An Attack Surface Metric--CMU-CS-05-155," Carnegie Mellon
1] University, Pittsburgh, PA, 2005.
- [7 T. W. Moore, C. W. Probst, K. Rannenber and M. van Eeten, "Assessing ICT Security
2] Risks in Socio-Technical Systems," 13-18 November 2016. [Online]. Available:
[https://drops.dagstuhl.de/opus/volltexte/2017/7039/pdf/dagrep_v006_i011_p063_s16461.pd
f](https://drops.dagstuhl.de/opus/volltexte/2017/7039/pdf/dagrep_v006_i011_p063_s16461.pdf).
- [7 D. Gollman, C. Herley, V. Koenig, W. Pieters and M. A. Sasse, "Socio-Technical Security
3] Metrics," *Dagstuhl Reports*, vol. 4, no. 12, pp. 1-28, 3 March 2015.
- [7 MITRE Corporation, "CVE-2022-37305," 01 August 2022. [Online]. Available:
4] <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-37305>. [Accessed February
2023].
- [7 G. A. Francia, III, D. Snider and B. Cyphers, "Basic Safety Message (BSM) Test Data
5] Generation for Vehicle Security Machine Learning Systems," in *Proc. of the 2023
International Conference on Security and Management (SAM'23)*, Las Vegas, NV, 2023.
- [7 R. W. van der Heijden, T. Lukaseder and F. Kargl, "VeReMi: A Dataset for Comparable
6] Evaluation of Misbehavior Detection in VANETs," in *Security and Privacy in
Communication Networks*, New York, NY, Springer, 2018, pp. 318-337.
- [7 H. Song, J. Woo and H. K. Kim, "Car-Hacking Dataset," 2020. [Online]. Available:
7] <https://ocslab.hksecurity.net/Datasets/CAN-intrusion-dataset>. [Accessed 20 May 2023].

- [7] U.S. Department of Transportation (DOT), "Connected Vehicle Pilot (CVP) Open Data," [Online]. Available: <https://data.transportation.gov/stories/s/Connected-Vehicle-Pilot-Sandbox/hr8h-ufhq#cv-pilot-data-sandbox>. [Accessed 20 May 2023].
- [7] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan and T. Darrell, [8] "BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [8] Society of Automotive Engineers (SAE), "J2735 Surface Vehicle Standard V2X [0] Communications Message Set Dictionary," 2023. [Online]. Available: https://www.sae.org/standards/content/j2735_202309. [Accessed 15 October 2023].
- [8] SAE International, "On-Board System Requirements for V2V Safety Communications [1] J2945/1_202004," 30 April 2020. [Online]. Available: https://www.sae.org/standards/content/j2945/1_202004/. [Accessed 20 May 2023].
- [8] G. Francia III, "Technical Report UWF-TR-FDOT-002-01," University of West Florida [2] Center for Cybersecurity, Pensacola, FL , 2023.
- [8] G. Francia III, "Technical Report UWF-TR-FDOT-002-02," University of West Florida [3] Center for Cybersecurity, Pensacola, FL, 2023.
- [8] G. Francia III, "Technical Report UWF-TR-FDOT-001-01," University of West Florida [4] Center for Cybersecurity, Pensacola, FL, 2022.
- [8] G. Francia III, "Technical Report UWF-TR-FDOT-003-01," University of West Florida, [5] Pensacola, FL, 2023.
- [8] G. Francia III, "Technical Report UWF-TR-FDOT-005-01," University of West Florida [6] Center for Cybersecurity, Pensacola, FL, 2023.
- [8] G. Francia III, "Technical Report UWF-TR-FDOT-006-01," University of West Florida [7] Center for Cybersecurity, Pensacola, FL, 2022.
- [8] G. Francia III, "Technical Report UWF-TR-FDOT-006-02," University of West Florida [8] Center for Cybersecurity, Pensacola, FL, 2022.
- [8] G. Francia III, "Technical Report UWF-TR-FDOT-006-03," University of West Florida [9] Center for Cybersecurity, Pensacola, FL, 2022.
- [9] G. Francia III, "Technical Report UWF-TR-FDOT-007-01," University of West Florida [0] Center for Cybersecurity, Pensacola, FL, 2023.
- [9] G. Francia III, "Technical Report UWF-TR-FDOT-007-02," University of West Florida [1] Center for Cybersecurity, Pensacola, FL, 2023.
- [9] G. Francia III, "Technical Report UWF-TR-FDOT-007-03," University of West Florida [2] Center for Cybersecurity, Pensacola, FL, 2023.
- [9] G. Francia III, "Technical Report UWF-TR-FDOT-008-01," University of West Florida [3] Center for Cybersecurity, Pensacola, FL, 2023.
- [9] G. Francia III, "Technical Report UWF-TR-FDOT-008-02," University of West Florida [4] Center for Cybersecurity, Pensacola, FL, 2023.
- [9] G. Francia III, "Technical Report UWF-TR-FDOT-008-03," University of West Florida [5] Center for Cybersecurity, Pensacola, FL, 2023.

[9 G. Francia III, "Technical Report UWF-TR-FDOT-009-01," University of West Florida,
6] Center for Cybersecurity, Pensacola, FL, 2024.

[9 G. Francia III, "Technical Report UWF-TR-FDOT-009-02," University of West Florida
7] Center for Cybersecurity, Pensacola, FL, 2024.

[9 G. Francia III, "Technical Report UWF-TR-FDOT-009-03," University of West Florida
8] Center for Cybersecurity, Pensacola, FL, 2024.

[9 J. Patterson and D. King, "Vehicle Security Metrics Visualization System," 2023. [Online].
9] Available: <https://github.com/UWF-CfC-FDOT/VSMVS>. [Accessed December 2023].

[1 G. Francia III, "Technical Report UWF-TR-FDOT-003-02," University of West Florida
00] Center for Cybersecurity, Pensacola, FL, 2023.
]

[1 G. Francia III, "Technical Report UWF-TR-FDOT-003-01," University of West Florida
01] Center for Cybersecurity, Pensacola, FL, 2023.
]

[1 G. Francia III, "Technical Report UWF-TR-FDOT-010-01," University of West Florida
02] Center for Cybersecurity, Pensacola, FL, 2024.
]

[1 G. A. Francia III, "Technical Report UWF-TR-FDOT-010-03 Continuous Improvement
03] Report--TR," University of West Florida, Pensacola, FL, 2024.
]

[1 S. Payne, "A Guide to Security Metrics," SANS Institute, 19 June 2006. [Online]. Available:
04] <http://www.sans.org/readingroom/papers/5/55.pdf>.
]

[1 K. Kark, P. Stamp, J. Penn, S. Bernhardt and A. Dill, "Defining An Effective Security
05] Metrics Program," 16 May 2007. [Online]. Available:
] <https://www.forrester.com/report/Defining+An+Effective+Security+Metrics+Program/-/E-RES42354#>. [Accessed February 2020].

[1 SAE International, "CAN Specification 2.0: Protocol and Implementations," 01 August
06] 1998. [Online]. Available: [https://www.sae.org/publications/technical-](https://www.sae.org/publications/technical-papers/content/921603/)
] [papers/content/921603/](https://www.sae.org/publications/technical-papers/content/921603/). [Accessed 13 October 2019].

[1 Gemalto, "Securing Vehicle to Everything," 2018. [Online]. Available:
07] <https://www.gemalto.com/brochures-site/download-site/Documents/auto-V2X.pdf>.
] [Accessed 13 April 2020].

[1 C. McCarthy, K. Harnett and A. Carter, "Characterization of potential security threats in
08] modern automobiles: A composite modeling approach," Washington, D.C., September,
] 2014.

[1 National Institute of Standards and Technology, "CVE-2019-13582 Detail," 15 November
09] 2019. [Online]. Available: <https://nvd.nist.gov/vuln/detail/CVE-2019-13582>. [Accessed 13
] February 2020].

[1 MITRE Corporation, "CWE-787: Out-of-bounds Write," 20 August 2020. [Online].
10] Available: <http://cwe.mitre.org/data/definitions/787.html>.
]

Software Requirements Specification

for

Vehicle Security Metrics Visualization System (VSMVS)

Version 0.9 approved

Prepared by Guillermo Francia, III

Contributors: Alessandro Amato

**The University of West Florida
Florida Department of Transportation**

August 20th, 2022

November 4, 2022 (Revision 1)

November 11, 2022 (Revision 2)

November 28, 2022 (Revision 3)

December 3, 2022 (Revision 4)

December 7, 2022 (Revision 5)

December 21, 2022 (Revision 6)

January 3, 2023 (Revision 7)

January 5, 2023 (Revision 8)

August 2, 2023 (Revision 9)

Table of Contents

1. Introduction	67
1.1 Purpose	67
1.2 Document Conventions	67
1.3 Scope	67
1.4 References	68
1.5 Document Revisions Table	68
2. Overview of Product.....	68
2.1 Vehicle Security Metrics Visualization System (VSMVS)	68
2.2 Hosting	69
2.3 Software Architecture and Use Cases	70
2.4 User Classes and Characteristics.....	71
2.5 Operating Environment.....	71
2.6 Design and Implementation Constraints	71
2.7 Assumptions and Dependencies	72
3. Interface Requirements.....	72
4. Functional Requirements.....	80
4.1 FR-1: Data Entry and Storage.....	80
4.1.1 Description and Priority	80
4.1.2 Related User Classes	81
4.1.3 Functional Requirements	81
4.2 FR-2: Security Metrics Calculators.....	82
4.2.1 Description and Priority	82
4.2.2 Related User Classes	82
4.2.3 Functional Requirements	82
4.3 FR-3: Security Metrics Visualizations.....	89
4.3.1 Description and Priority	89
4.3.2 Related User Classes	89
4.3.3 Functional Requirements	89
5. Test Requirements.....	90
5.1 T-1: Unit Tests	90
5.2 T-2: Integration Tests.....	90
5.3 T-3: Test Report.....	90
6. Non-Functional Requirements	90
6.1 NF-1: Portability	90
6.2 NF-2: Usability.....	91

6.3	<i>NF-3: Speed</i>	91
7.	Quality Attributes	91
8.	Source Code Repository and Version Control Requirement	91
8.1	<i>SC-1: Source Code Repository and Control</i>	91
Appendix A: Requirements Table		91
Appendix B: Requirements Traceability Matrix		98
Appendix C: Glossary		101

Introduction

Purpose

This undertaking entails the design and implementation of a prototype web-enabled Vehicle Security Metrics Visualization System (VSMVS).

Visualization takes advantage of cognitive perception in effectively presenting information to users. It offers a powerful means of recognizing trends and patterns that are not easily recognized using non-visual methods.

In essence, the cognitive reasoning process is augmented by perception to bring about a more rapid analytical reasoning process. The system will provide a visual depiction of security metrics that were developed in another undertaking by employing the benefits of visual perception. The implementation of this system will utilize the tools and facilities that are available through subscriptions provided by Amazon Web Services (AWS).

Document Conventions

This document is based on the IEEE 830 Standards and the Florida Department of Transportation Requirements Standards. Specific conventions used in this document are listed below:

- Priorities are indicated for each feature as well as in the Requirements Table. A green highlighting indicates must have features, while a yellow highlight represents a should have feature.
- Requirements follow the form of <TAG>-#.#.# where a tag indicates a category of requirements. And the # represents the id of the requirement in a hierarchical fashion.

Scope

This document contains a complete description of the design of the Vehicle Security Metrics Visualization System (VSMVS).

The basic architecture is a web server from a client server paradigm.

The webpages will be created using the ASP.Net framework with Blazor.

References

The following references were used in the creation of this document:

- IEEE 830 Standards on Software Requirement Specifications
- UWF Scope of Service Document for the Connected Vehicle Security Metrics and Threat Intelligence Project

[IEEE] The applicable IEEE standards are published in “IEEE Standards Collection,” 2001 edition.

[Bruade] The principal source of textbook material is “Software Engineering: An Object-Oriented Perspective” by Eric J. Bruade (Wiley 2001).

Document Revisions Table

Revisor	Revision Date	Reason
Guillermo Francia, III	November 2, 2022	Added the CWSS function
Guillermo Francia, III	November 11, 2022	Revised the Threats on Assets requirements
Guillermo Francia, III	November 28, 2022	Added the Operational Security Assessment Metrics requirements
Guillermo Francia, III	December 3, 2022	Added the Security Vulnerability of Connected Vehicles Metrics requirements
Guillermo Francia, III	December 7, 2022	Added the Security Best Practices Assessment Metrics requirements
Guillermo Francia, III	December 21, 2022	Updated the Functional Requirements
Guillermo Francia, III	January 3, 2023	Updated the Requirements Traceability Matrix
Guillermo Francia, III	January 5, 2023	Revised the TOC and the set of figures
Guillermo Francia, III	August 2, 2023	Added the revisions suggested by the FDOT Research office

Overview of Product

Vehicle Security Metrics Visualization System (VSMVS)

This Vehicle Security Metrics Visualization System (VSMVS) provides a visual depiction of the security metrics that are defined for the automotive vehicle.

Hosting

The system will be hosted inside an AWS EC2 instance reachable at <https://183.73.161.243>. This IP address will be changed with a more user-friendly name once the domain name is decided.

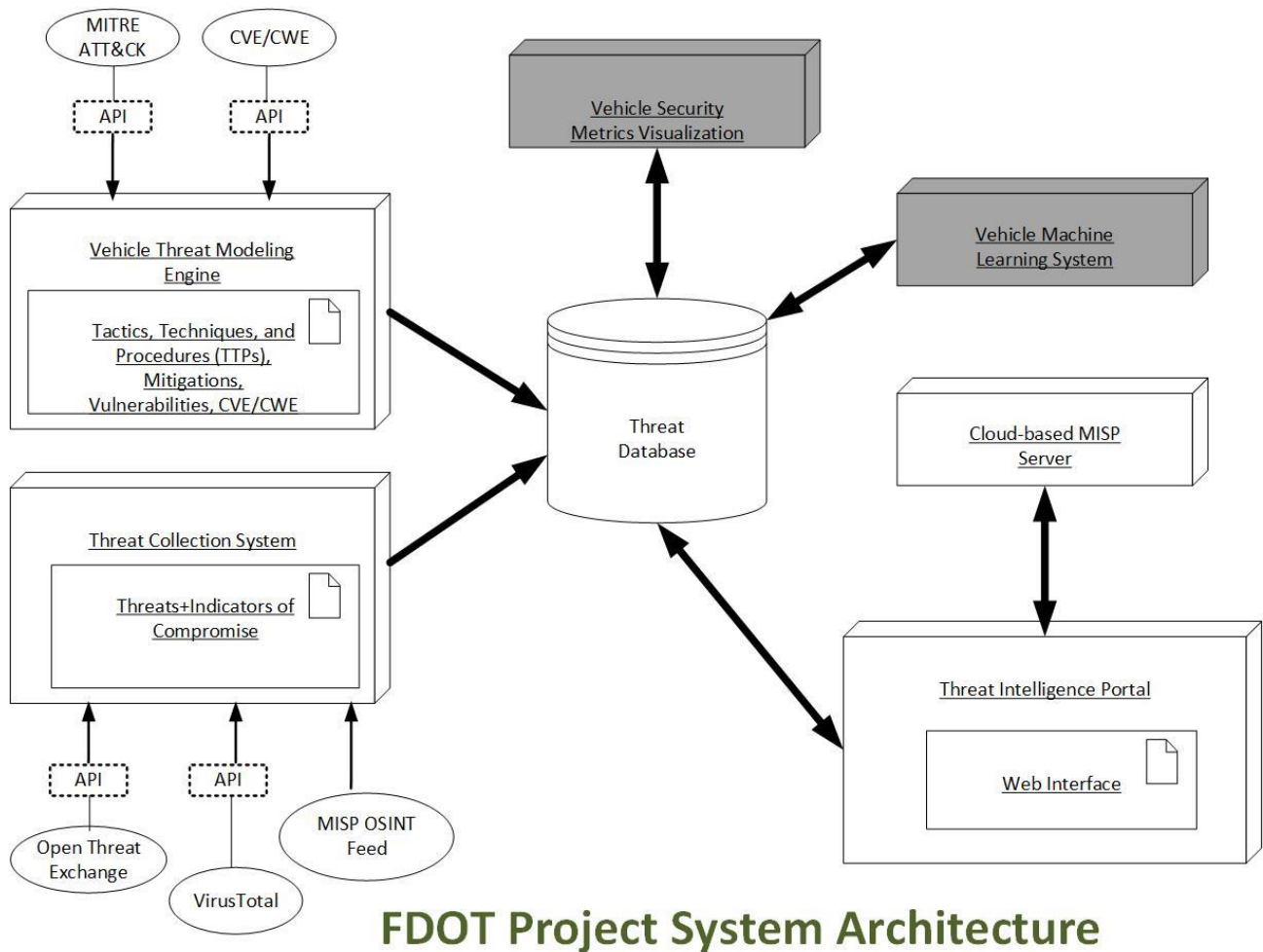


Figure 1. Vehicle Security and Threat Modeling System Architecture

Software Architecture and Use Cases

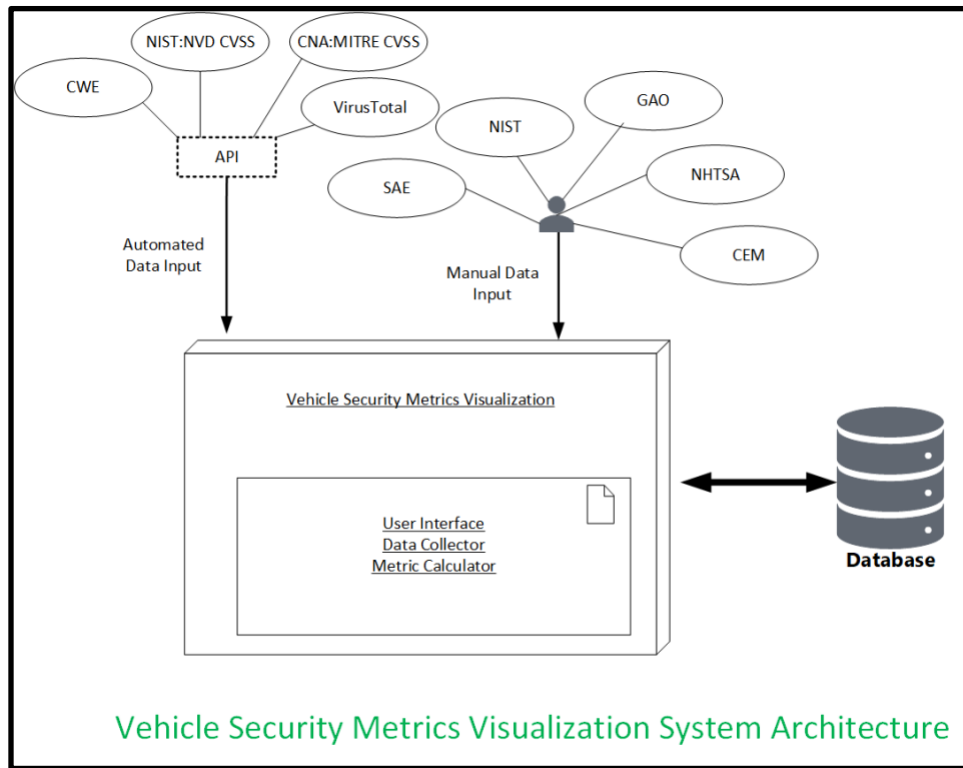


Figure 2. Vehicle Security Metrics Visualization System Architecture

User Classes and Characteristics

User Class	Characteristics
Admin User	This user is responsible for the overall administration of the system
Regular User	This user provides the data for generating the various vehicle security metrics and visualization
Database Administrator	This user administers the threat and metrics database system

Operating Environment

The VSMVS operating environment is defined by the following:

OE-1: The VSMVS shall run within an Amazon Elastic Compute Cloud (EC2) Web Service utilizing a Windows Server environment.

OE-2: The VSMVS shall run as a .NET application on an Internet Information Services (IIS) on a Windows Server within the AWS EC2 instance.

OE-3: The AWS EC2 instance shall be configured with type t2.xlarge having 4 vCPU and 16 GB of memory.

OE-4: The VSMVS shall interact with a Vehicle Threat Database System (VTDBS) backend. The VTDBS will be designed and implemented as a major deliverable of the project.

OE-5: The VTDBS shall be configured using MS SQL Server 2018.

Design and Implementation Constraints

DIC-1: The VSMVS shall be developed using Microsoft Visual Studio 2022 or Visual Studio Code and

DIC-2: The VSMVS shall be developed using the C# programming language and

DIC-3: The VSMVS shall be developed using .NET Core

DIC-4: The VSMVS will be constrained by the limitations of the data source APIs

DIC-5: The VSMVS will be designed and implemented as a working prototype capable of future expansion

DIC-6: The initial iterations of the VSMVS will be limited to the open frameworks CVE and CWE data sources

DIC-7: All visualizations shall be based on the derived vehicle security metrics that are fully defined in an accompanying white paper.

Assumptions and Dependencies

Assumptions and dependencies for the VSMVS implementation include the following:

ASS-1: The VSMVS assumes the availability of information for some vehicle security metric attributes.

ASS-2: The VSMVS assumes the availability of domain experts to provide reliable information for some vehicle security metric attributes.

DEP-1: The VSMVS shall be dependent on the accuracy and currency of the CVE and CWE information.

Interface Requirements

The VSMVS will require input from the user. In addition, the results of calculated and derived metrics must be displayed. These interface requirements are described in the following.

INT-1: The VSMVS will periodically collect information from threats using the Common Vulnerability Enumeration (CVE) Application Program Interfaces (APIs) from the National Vulnerability Database (NVD). The CVE information shall be used to derive a Common Vulnerability Scoring System (CVSS) vector which, in turn, provides a corresponding vehicle security metrics.

INT-2: The VSMVS shall provide a Graphical User Interface (GUI) for data entry of CVSS vector and metrics information. The CVSS GUI shall resemble the following:

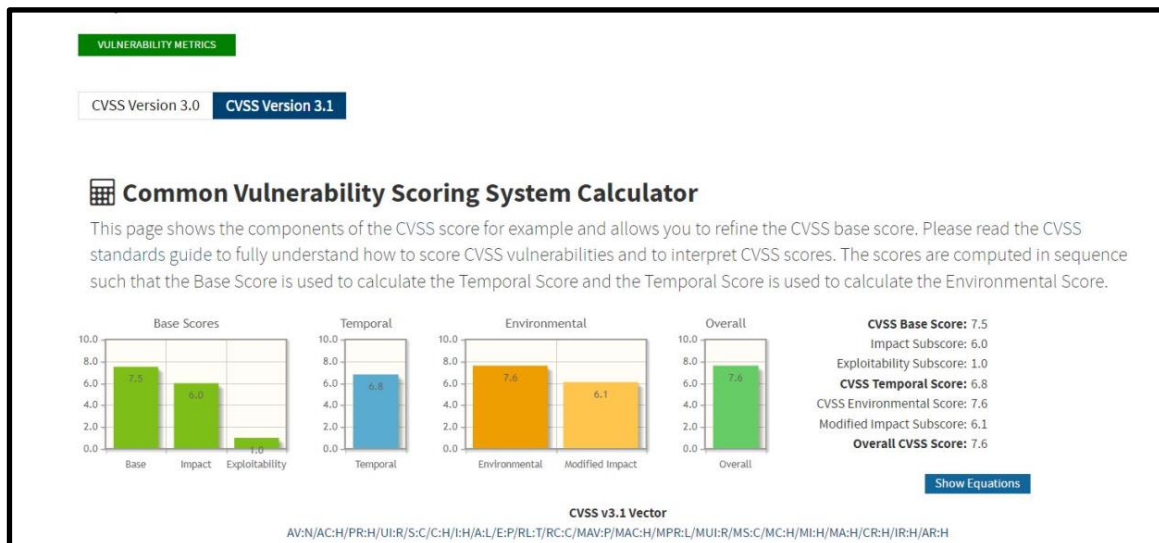
The screenshot displays a web interface for entering CVSS metrics. At the top, it shows the 'CVSS vector' results as **AV:N/AC:H/PR:H/UI:R/S:U/C:H/I:H/A:H**. Below this, the 'Base Score Metrics' section is divided into two columns:

- Exploitability Metrics:**
 - Attack Vector (AV): Network (selected), Adjacent Network, Local, Physical
 - Attack Complexity (AC): Low (selected), High
 - Privileges Required (PR): None, Low, High (selected)
 - User Interaction (UI): None, Required (selected)
- Impact Metrics:**
 - Scope (S): Unchanged (selected), Changed
 - Confidentiality Impact (C): None, Low, High (selected)
 - Integrity Impact (I): None, Low, High (selected)
 - Availability Impact (A): None, Low, High (selected)

The 'Temporal Score Metrics' section includes:

- Exploit Code Maturity (E): Not Defined, Unproven that exploit exists, Proof of concept code, Functional exploit exists, High (selected)
- Remediation Level (RL): Not Defined, Official fix, Temporary fix, Workaround, Unavailable
- Report Confidence (RC): Not Defined, Unknown, Reasonable, Confirmed

INT-3: The VSMVS shall provide a corresponding visualization for the CVSS vector and metrics resembling the following:



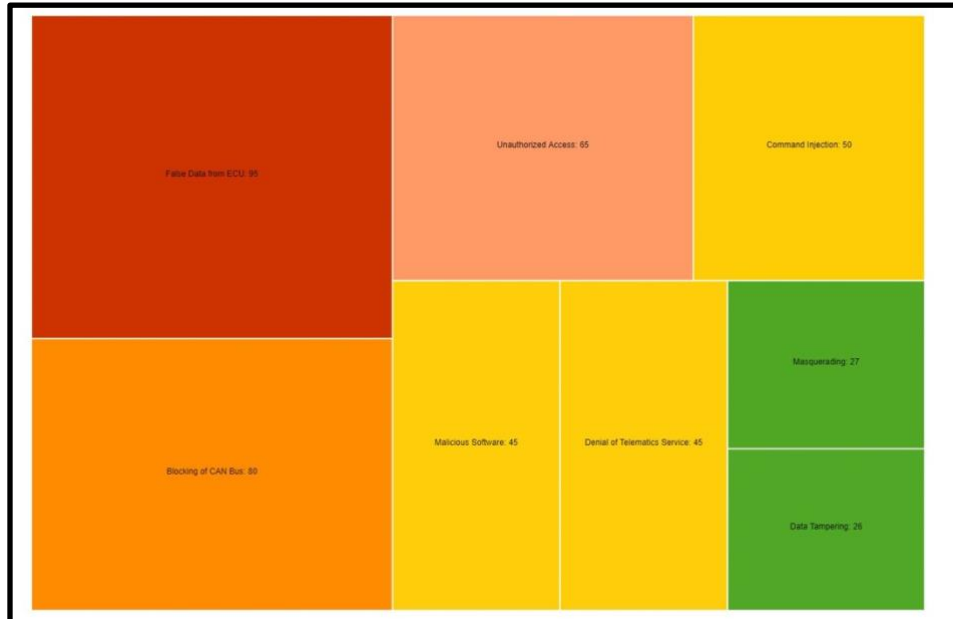
INT-4: The VSMVS shall provide a Graphical User Interface (GUI) for the data entry of attack potential of threats on vehicle assets. The GUI shall resemble the following:

Attack Potential of Threats on Vehicle Assets

Threat on Assets	Elapsed Time	Specialist Expertise	Knowledge of Target	Window of Opportunity	Equipment/Software	Total
False Data from ECU ⓘ	Immediate (20) ▾	novice (15) ▾	public knowledge (20) ▾	unlimited (20) ▾	minimal (20) ▾	95
Blocking of CAN Bus ⓘ	one week (10) ▾	proficient (10) ▾	public knowledge (20) ▾	unlimited (20) ▾	minimal (20) ▾	80
Malicious Software ⓘ	one month (5) ▾	expert (5) ▾	sensitive information (10) ▾	difficult (5) ▾	minimal (20) ▾	45
Denial of Telematics Service ⓘ	one week (10) ▾	expert (5) ▾	critical information (5) ▾	easy (15) ▾	specialized(10) ▾	45
Unauthorized Access ⓘ	one week (10) ▾	proficient (10) ▾	public knowledge (20) ▾	easy (15) ▾	specialized(10) ▾	65
Command Injection ⓘ	one week (10) ▾	expert (5) ▾	public knowledge (20) ▾	moderate (10) ▾	highly specialized(5) ▾	50
Masquerading ⓘ	one week (10) ▾	multiple experts (1) ▾	sensitive information (10) ▾	difficult (5) ▾	multi-specialized (1) ▾	27
Data Tampering ⓘ	one year (1) ▾	expert (5) ▾	critical information (5) ▾	moderate (10) ▾	highly specialized(5) ▾	26

[Visualize](#)

INT-5: A corresponding visualization for the calculated attack potential of threats on vehicle asset shall resemble the following:



INT-6: The VSMVS shall provide a GUI for data entry of CWSS vector and metrics information. The CWSS data entry GUI shall resemble the following:

Common Weakness Scoring System (CWSS)

CWSS vector: [TI:1/AP:0.9/AL:0.7/IC:0.7/FC:0.8]
 CWSS score: 67.3 Visualize

Base Finding

Technical Impact Critical	Acquired Privilege Administrator
Acquired Priv Layer Application	Int Control Effectivene None
Finding Confidence Proven True	

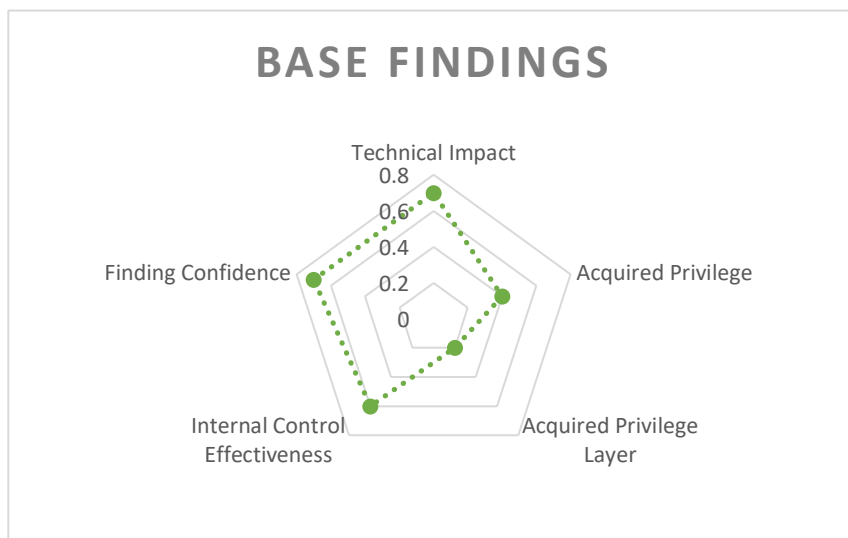
Attack Surface

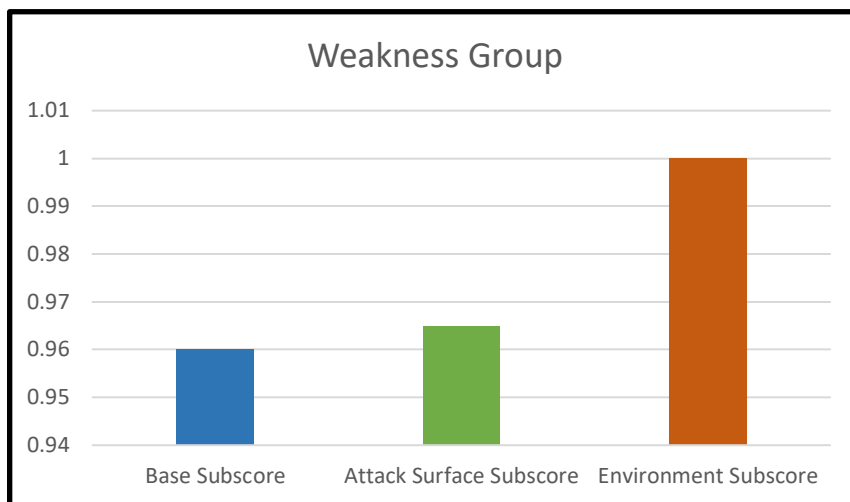
Required Privilege None	Req Privilege Layer Application
Access Vector Internet	Authentication Strength Strong
Level Interaction Automated	Deployment Scope All

Environmental

Business Impact Critical	Likelihood Discovery High
Likelihood Exploit High	Ext Control Effect None
Prevalence Widespread	

INT-7: The VSMVS shall provide a corresponding visualization for the CWSS vector and metrics resembling the following:





INT-8: The VSMVS shall provide a GUI for data entry for Operational Safety Assessment (OSA) metrics information. The OSA metrics data entry GUI shall resemble the following:

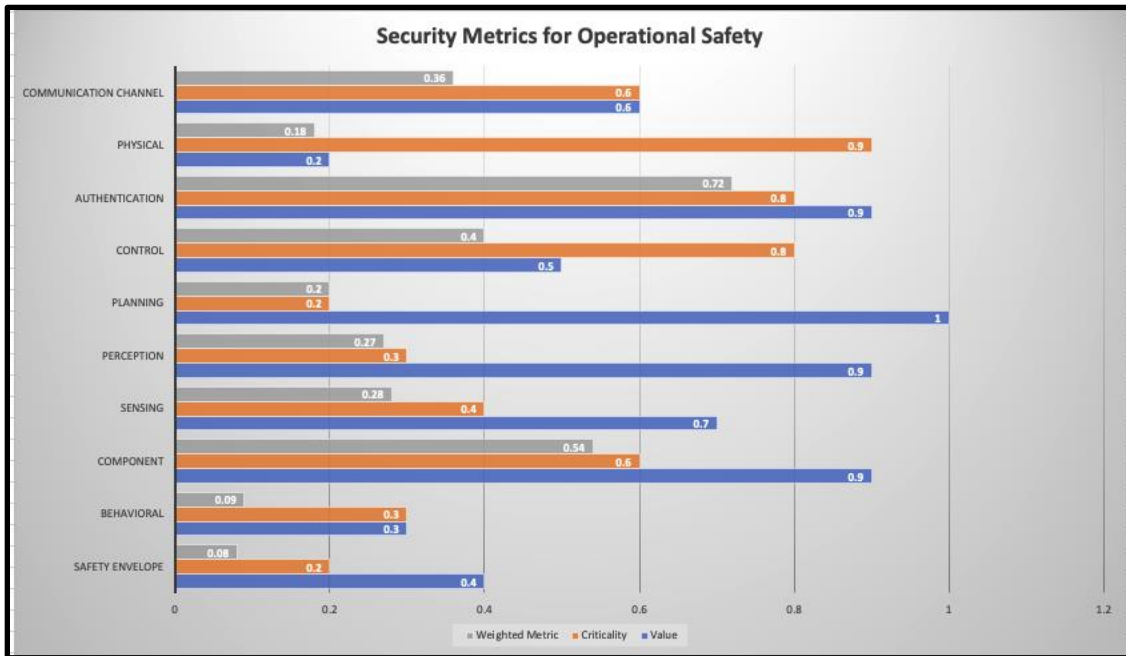
The screenshot shows a web-based interface for entering OSA metrics. It features a grid of input fields for various categories, each with a 'Value' and 'Criticality' dropdown menu. At the bottom, there are buttons for 'Calculate Aggregate Security', 'Visualize', and 'Reset'. The calculated aggregate security score is displayed as 2.02.

Category	Value	Criticality
Safety Envelope	0.0	0.1
Behavioral	0.0	0.1
Component	0.0	0.1
Sensing	0.0	0.1
Perception	0.0	0.1
Planning	0.0	0.1
Control	0.0	0.1
Authentication	0.0	0.1
Physical	0.0	0.1
Communication	0.0	0.1
Reliability	0.1	
Relevance	0.1	
Extent	0.1	

Calculate Aggregate Security: **2.02**

Buttons: Calculate Aggregate Security, Visualize, Reset

INT-9: The VSMVS shall provide a corresponding visualization for the OSA metrics resembling the following:



INT-10: The VSMVS shall provide a GUI for data entry for Security Vulnerability Metrics for Connected Vehicles information. The metrics data entry GUI shall resemble the following:

Security Vulnerability Metrics for Connected Vehicles

ECU Coupling Risk ? Weight 1

Active ECU Links: 0
Total ECU Links: 0

Complexity Risk ? Weight 1

SLOCs: 0
Number of Nestings: 0
Weight of Nestings: 1

History of Security Issue Risk ? Weight 1

Number of Years Since First Attack: 0
Number of First Attack: 0
Number of Second Attack: 0
Forgetting Factor: 0.2

Communication Channel Risk ? Weight 1

Active V2V Links: 0
Total V2V Links: 0
V2V Weight: 0.1

Active V2I Links: 0
Total V2I Links: 0
V2I Weight: 0.1

Active U2V Links: 0
Total U2V Links: 0
U2V Weight: 0.1

Active IV Links: 0
Total IV Links: 0
IV Weight: 0.1

I/O Data Risk ? Weight 1

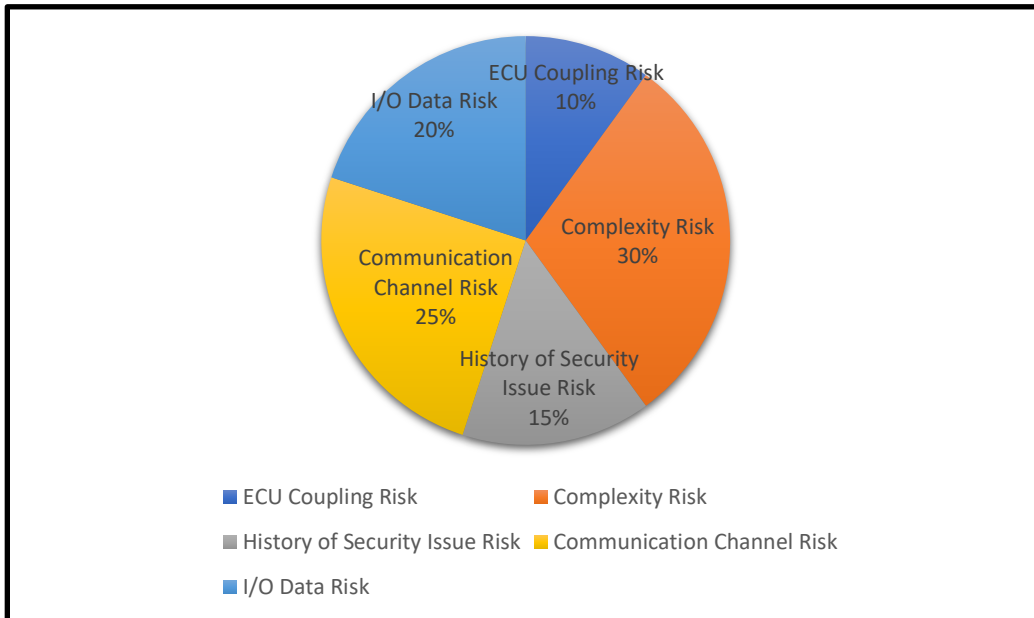
Number of Fixed Input: 0
Number of Fluctuating Input: 0
FI Weight: 1

Number of Insensitive Output: 0
Number of Sensitive Output: 0
SO Weight: 1

Calculate Total Risk 2.05

Visualize **Reset**

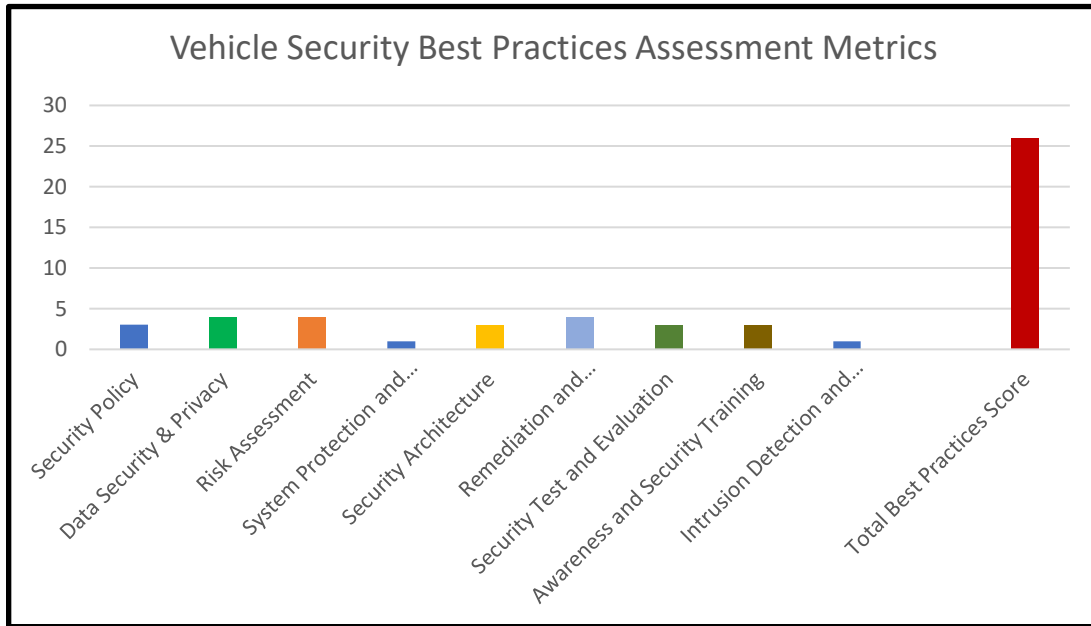
INT-11: The VSMVS shall provide a corresponding visualization for the Security Vulnerability Metrics for Connected Vehicles resembling the following:



INT-12: The VSMVS shall provide a GUI for data entry for Vehicle Security Best Practices Assessment Metrics for Connected Vehicles information. The metrics data entry GUI shall resemble the following:

Phase	Process	Checklist	Response		
			Yes	No	
Assessment	Security Policy	Security policies established	X		
		Security policies properly documented and widely disseminated within the organization?	X		
		Security policies strictly enforced	X		
		Security policies periodically reviewed/updated		X	
	Data Security & Privacy	Collected/stored data protected/encrypted	X		
		Transmitted data encrypted	X		
		A control mechanism for sharing data	X		
		Compliant with data protection standards and regulations (e.g., ISO/IEC 27001 certification, GDPR)?	X		
	Risk Assessment	Conduct a periodic risk assessment of vehicle cybersecurity	X		
		Developed and implemented organization-wide risk management strategy	X		
A Supply Chain Risk Management (SCRM) policy		X			
Security controls in place and periodically evaluated and/or enhanced		X			
Design	System Protection and Prioritization	Implemented security-by-design principles during the vehicle design phase		X	
		Implemented domain separation for in-vehicle networks (i.e. limiting the communication between the safety-critical and non-safety critical domains)		X	
		Process to triage the identified risks according to priority for resource allocation	X		
		Comprehensive system security test plan		X	
	Security Architecture	Implemented a layered approach to vehicle security (ECU level, in-vehicle network level, V2V level, V2X level)?	X		
		Periodic evaluation of the system's security architecture		X	
		Maintain an inventory of operational software components used in each automotive ECU and assembled vehicle	X		
Implementation	Remediation and Implementation	Established mechanisms to update vehicle software and firmware remotely and securely	X		
		Appropriate security controls implemented and are in place	X		
		Established a remediation process	X		
		Remediation plan evaluated and implemented	X		
	Security Test and Evaluation	Conducted a thorough code review on the vehicle software	X		
		Conducted penetration testing on connected vehicle communication systems before deployment	X		
		Security controls tested and evaluated for compliance with security performance specifications	X		
		Conformance with secure software development best practices as outlined in NIST 8151 and ISO/SAE 21434		X	
	Operation	Awareness and Security Training	Periodic security awareness training program for the entire workforce	X	
			Security risk and mitigation disclosure available to the consumer and other stakeholders	X	
Evaluate the effectiveness of the security awareness training program and introduce improvements if needed			X		
Collect, maintain, analyze, and share information related to cybersecurity through the Automotive Information Sharing and Analysis Center (Auto-ISAC)				X	
Intrusion Detection and Response		An Incident Response Plan (IRP) in place	X		
		The IRP periodically tested, evaluated, and updated		X	
		A systematic process for continuous risk and security monitoring		X	
		Security incidents properly documented and reported		X	

INT-13: The VSMVS shall provide a corresponding visualization for the Security Vulnerability Metrics for Connected Vehicles resembling the following:



INT-14: The VSMVS will log any interactions with external APIs in a format containing the time, message type, sent request, response code, and response.

INT-15: The VSMVS will log any interactions with the threat database in a format containing the time, query type, query contents, and response.

INT-16: The VSMVS will log errors or exceptions in a format containing the time of the event and a stack trace.

Functional Requirements

The major features that the VSMVS will need to deliver are the data entry interfaces, the metrics calculators, and the visualization of the generated metrics. VSMVS is a proof of concept and, as such, limitations in its implementation will be identified.

4.1 FR-1: Data Entry and Storage Description and Priority

The VSMVS shall facilitate user input of metrics information. In addition, it should provide the storage of information in a central database.

Data entry functions shall be provided for the following metrics.

Priority: Must Have

Related User Classes
All Users

Functional Requirements

FR-1.1 Common Vulnerability Scoring System (CVSS) Metrics

CVSS is an open framework for communicating the characteristics and severity of software vulnerabilities. It consists of three metric groups: Base, Temporal, and Environmental. The Base group characterizes the static intrinsic qualities of vulnerability; the Temporal group represents the vulnerability as it evolves over time; and the Environmental group depicts the characteristics of the vulnerability that are endemic to the user's environment. The third group of metrics lends itself perfectly with that of an automotive vehicle system. Data input and storage shall be implemented for the CVSS metrics.

FR-1.2 Attack Potential on Vehicle Assets Metrics

The attack potential for each vehicle asset is calculated based on the input provided by the user for each of the following factors: elapsed time, specialist expertise, knowledge of the target, window of opportunity, and IT hardware and/or software availability. Data input and storage shall be implemented for the attack potential on vehicle assets.

FR-1.3 Common Weakness Scoring System (CWSS) Metrics

The Common Weakness Scoring System (CWSS) is a mechanism for evaluating software weaknesses in a consistent, flexible, open manner. It is a community-based undertaking which addresses the need for prioritizing the software vulnerability issues. The measurements are organized into three metric groups: Base Finding, Attack Surface, and Environmental. Data input and storage shall be implemented for the CWSS metric groups.

FR-1.4 Operational Safety Assessment (OSA) Cybersecurity Metrics

The Operational Safety Assessment (OSA) Cybersecurity Metrics are defined and implemented based on the OSA metrics that are augmented by authentication metric, physical access metric, and communication channel metric. The OSA metrics that are proposed by SAE

include the following: safety envelope metric, behavioral metric, component metric, sensing metric, perception metric, planning metric, and control metric. Data input and storage shall be implemented for the OSA Cybersecurity Metrics.

FR-1.5 Security Vulnerability Metrics for Connected Vehicles

Security vulnerability metrics for connected vehicles measure weak or vulnerable features in the software system of connected vehicles. These security vulnerabilities are associated with the following risks found in connected vehicles: ECU Coupling risk, Communication Channel risk, Complexity risk, Input and Output Data risk, and History of Security Issues risk. Data input and storage shall be implemented for the security vulnerability metrics.

FR-1.6 Vehicle Security Best Practices Assessment Metrics

The Vehicle Security Best Practices Assessment Metrics are designed based on the National Highway Traffic Safety Administration (NHTSA) Report DOT HS812 075. These metrics are based on the four phases of the Information Security Life Cycle: Assessment Phase, Design Phase, Implementation Phase, and Operation Phase. Data input and storage shall be implemented for the vehicle security best practices assessment metrics.

4.2 FR-2: Security Metrics Calculators

4.2.1 Description and Priority

The VSMVS shall provide calculators that will process user input data to derive the security metrics value. Calculations shall use the given formula for each defined security metric and will be performed by backend processes. The results will be displayed as part of the visualization of the associated metric.

Priority: Must Have

4.2.2 Related User Classes

All Users

4.2.3 Functional Requirements

FR-2.1 Common Vulnerability Scoring System (CVSS) Calculator

This CVSS Base Score is calculated based on a table of metric values and the following formula found in CVSS v3.1 Specification Document (see <https://www.first.org/cvss/specification-document>). The formula for the Temporal and Environmental metrics are found in the same document.

$$BaseScore = \begin{cases} 0 & \text{if } Impact \leq 0 \\ [(Min[(Impact + Exploitability), 10])] & \text{if } Scope \text{ is } Unchanged \\ [(Min[(1.08 * (Impact + Exploitability), 10))] & \text{if } Scope \text{ is } Changed \end{cases}$$

Where

$$Impact = \begin{cases} 6.42 * ISS & \text{if } Scope \text{ is } Unchanged \\ 7.52 * (ISS - 0.029) - 3.25 * (ISS - 0.02)^{15} & \text{if } Scope \text{ is } Changed \end{cases}$$

$$Exploitability = 8.22 * AttackVector * AttackComplexity * PrivilegesRequired * UserInteraction$$

$$ISS = 1 - [(1 - Confidentiality) * (1 - Integrity) * (1 - Availability)]$$

FR-2.2 Attack Potential on Vehicle Assets Metrics Calculator

The attack potential for each vehicle asset is calculated based on the input provided by the user for each asset. The calculator will use the following formula.

$$TA_k = \sum_{n=1}^5 F_n$$

Where TA_k is the k th threat and F_n is the n th factor.

Priority: Must Have

FR-2.3 Common Weakness Scoring System (CWSS) Metrics Calculator

The CWSS scoring system is depicted in Figure 4.1.

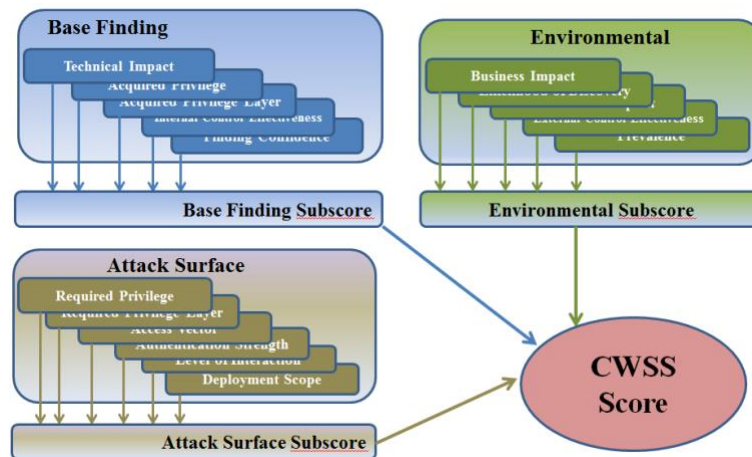


Figure 4.1 The CWSS Scoring System (Source: https://cwe.mitre.org/cwss/cwss_v1.0.1.html)

A summary of the CWSS factors categorized by metric group is shown in Table 4.1.

Table 4.1 The CWSS Scoring Factors by Metric Group (Source: https://cwe.mitre.org/cwss/cwss_v1.0.1.html)

Group	Name	Summary
Base Finding	Technical Impact (TI)	The potential result that can be produced by the weakness, assuming that the weakness can be successfully reached and exploited.
Base Finding	Acquired Privilege (AP)	The type of privileges that are obtained by an attacker who can successfully exploit the weakness.
Base Finding	Acquired Privilege Layer (AL)	The operational layer to which the attacker gains privileges by successfully exploiting the weakness.
Base Finding	Internal Control Effectiveness (IC)	the ability of the control to render the weakness unable to be exploited by an attacker.
Base Finding	Finding Confidence (FC)	the confidence that the reported issue is a weakness that can be utilized by an attacker
Attack Surface	Required Privilege (RP)	The type of privileges that an attacker must already have in order to reach the code/functionality that contains the weakness.
Attack Surface	Required Privilege Layer (RL)	The operational layer to which the attacker must have privileges in order to attempt to attack the weakness.
Attack Surface	Access Vector (AV)	The channel through which an attacker must communicate to reach the code or functionality that contains the weakness.
Attack Surface	Authentication Strength (AS)	The strength of the authentication routine that protects the code/functionality that contains the weakness.

Group	Name	Summary
Attack Surface	Level of Interaction (IN)	the actions that are required by the human victim(s) to enable a successful attack to take place.
Attack Surface	Deployment Scope (SC)	Whether the weakness is present in all deployable instances of the software, or if it is limited to a subset of platforms and/or configurations.
Environmental	Business Impact (BI)	The potential impact to the business or mission if the weakness can be successfully exploited.
Environmental	Likelihood of Discovery (DI)	The likelihood that an attacker can discover the weakness
Environmental	Likelihood of Exploit (EX)	the likelihood that, if the weakness is discovered, an attacker with the required privileges/authentication/access would be able to successfully exploit it.
Environmental	External Control Effectiveness (EC)	the capability of controls or mitigations outside of the software that may render the weakness more difficult for an attacker to reach and/or trigger.
Environmental	Prevalence (P)	How frequently this type of weakness appears in software.

The CWSS Score, with value between 0 and 100, is calculated using the following formula:

$$\text{Base_Finding_Subscore} * \text{Attack_Surface_Subscore} * \text{Environment_Subscore}$$

where the Base_Finding_Subscore, with value between 0 and 100, is calculated using the following formula:

$$\text{Base} = [(10 * \text{TechnicalImpact} + 5 * (\text{AcquiredPrivilege} + \text{AcquiredPrivilegeLayer}) + 5 * \text{FindingConfidence}) * f(\text{TechnicalImpact}) * \text{InternalControlEffectiveness}] * 4.0$$

$$f(\text{TechnicalImpact}) = 0 \text{ if } \text{TechnicalImpact} = 0; \text{ otherwise } f(\text{TechnicalImpact}) = 1.$$

the Attack_Surface_Subscore, with value between 0 and 1, is calculated using the formula:

$$[20 * (\text{RequiredPrivilege} + \text{RequiredPrivilegeLayer} + \text{AccessVector}) + 20 * \text{DeploymentScope} + 15 * \text{LevelOfInteraction} + 5 * \text{AuthenticationStrength}] / 100.0$$

the Environment_Subscore, with value between 0 and 1, is calculated using the formula:

$$[(10 * \text{BusinessImpact} + 3 * \text{LikelihoodOfDiscovery} + 4 * \text{LikelihoodOfExploit} + 3 * \text{Prevalence}) * f(\text{BusinessImpact}) * \text{ExternalControlEffectiveness}] / 20.0$$

$$f(\text{BusinessImpact}) = 0 \text{ if } \text{BusinessImpact} == 0; \text{ otherwise } f(\text{BusinessImpact}) = 1$$

FR-2.4 Operational Safety Assessment (OSA) Security Metrics Calculator

The OSA Aggregate Security Metric (ASM) is calculated based on the input data provided by the user and applying the following formula:

$$ASM = Reliability \times Relevance \times Extent \times \left(\sum_{k=1}^N Criticality_k \times Security_Metric_k \right)$$

Where

- Reliability. Factor for quantifying the fidelity of the sources of measurement data. For instance, data originating from actual events carry a higher value than those from simulated events. Values range from 0.1 for less reliable to 1.0 for most reliable.
- Relevance. Factor for quantifying the relevance of the measurement to a subject vehicle. This value may vary according to the specificity of data such as make and model of the subject vehicle. Data for a Honda CRV is more specific than data that refers to Honda vehicles in general. Values range from 0.1 for least relevant to 1.0 for most relevant.
- Extent. Factor for quantifying the scope or extensiveness of the measurement data. The value ranges from 0.1 for least extensive to 1.0 for most extensive.
- Criticality. Factor for quantifying the gravity of a specific metric. For instance, security measurement on control will carry a heavier weight than that on safety envelope. The value ranges from 0.1 for least critical to 1.0 for most critical.
- N is the number cybersecurity metrics for operational safety.

FR-2.5 Security Vulnerability Metrics for Connected Vehicles Calculator

Security vulnerability metrics measure weak or vulnerable features in the software system of connected vehicles. Each of these metrics is calculated using the risks on connected vehicles that may eventually contribute to the likelihood of exploiting a vulnerability.

ECU Coupling Risk

This risk is manifested by level of interconnection among ECU components. This means the higher the coupling value the higher is the probability for vulnerabilities. Thus, for every functionality, F, for all ECUs, N, and for all communication links between ECU j and ECU k, the ECU coupling risk, R_{EC} , is calculated as

$$R_{EC}(F) = \sum_{j=1, k=1}^N C_{jk}$$

$C_{jk}=1$ if there is at least one information transfer between ECU j and ECU k; 0 otherwise.

$$\text{Max} (R_{EC}(F)) = N$$

Communication Channel Risk

This risk is based on the communication channel types that are available for connected vehicles: vehicle to vehicle (V2V), vehicle to infrastructure (V2I), user to vehicle (U2V), and intra-vehicle (IV). The communication risk, for each functionality, F, is calculated according to the following formula:

$$R_{CC} (F) = \sum_{j=1}^N w_j C_j$$

Where N is the number of communication links, w_j the weight of a specific communication channel type based on its propensity to vulnerability, and C_j is 1 if the functionality uses the channel; 0 otherwise.

$$\text{Max} (R_{CC} (F)) = \text{Total number of all communication channels}$$

Complexity Risk

This risk is associated with the number of defects in software used in automotive vehicles. The complexity metric in software is an excellent indicator of vulnerabilities. The Halstead Complexity measure is a standard way of deriving the complexity of software. Thus, for calculating the complexity of the functionalities in connected vehicle, we use the formula:

$$R_{SC} (F) = SLOC + a (Nesting)$$

Where SLOC is the Source Line of Code, Nesting is the number of control structures, and a is the weight, with value over one, indicating complexity of the nesting structure.

$$\text{Max } (R_{SC} (F)) = SLOC + 10 (Nesting)$$

Input and Output Data Risk

This risk involves the input and output data in a connected vehicle. The metric distinguishes between a Fixed Input (FI) from a Fluctuating Input (LI). It also distinguishes an Insensitive Output (IO) from a Sensitive Output (SO). Weights (a, b) are added to highlight the significance of the Fluctuating Input and the Sensitive Output. To calculate the Input and Output Data Risk, we use the formula:

$$R_{DIO} (F) = FI + a (LI) + IO + b (SO)$$

$$\text{Max } (R_{DIO} (F)) = FI + 5(LI) + IO + 5(SO)$$

History of Security Issues

This risk considers the past security issues of a certain vehicle functionality. Given Y as the total number of years since the first car attack and α_y as the number of attacks that occurred in year y. A forgetting factor, l, is introduced to provide relevancy to the attacks that occurred in more recent years, where $0 \leq l \leq 1$. To calculate the risk of a vehicle functionality using the history of security issues, we use the formula:

$$R_{HS} (F) = \sum_{y=1}^Y \alpha_y \lambda^{Y-y}$$

For a 2-year comparison, the calculation simply boils down to

$$l = 1 - (\alpha_1 / \alpha_2) \quad \leftarrow \text{the forgetting factor}$$

$$R_{HS} (F) = \alpha_1 (l)^{Y-1} + \alpha_2$$

$$\text{Max} (R_{HS} (F)) = a_1 + a_2$$

Overall Security Vulnerability Metric

The overall security vulnerability metric of a certain functionality in a connected vehicle is calculated by first normalizing the values of each of the metrics and applying a weighting factor (a, b, g, d, f), which indicates its significance to the overall scheme. The metrics are added to obtain the overall value, which is in direct correlation with the vulnerability level of the functionality. The formula is shown as follow:

$$\begin{aligned} OSV = & \alpha \left[\frac{R_{EC}(F)}{\max(R_{EC}(F))} \right] + \beta \left[\frac{R_{CC}(F)}{\max(R_{CC}(F))} \right] + \gamma \left[\frac{R_{SC}(F)}{\max(R_{SC}(F))} \right] \\ & + \delta \left[\frac{R_{DIO}(F)}{\max(R_{DIO}(F))} \right] + \varphi \left[\frac{R_{HS}(F)}{\max(R_{HS}(F))} \right] \end{aligned}$$

4.3 FR-3: Security Metrics Visualizations

4.3.1 Description and Priority

The VSMVS shall provide a visualization depicting each of the derived or calculated metrics.

Priority: Must Have

4.3.2 Related User Classes

All Users

4.3.3 Functional Requirements

FR-3.1 Common Vulnerability Scoring System (CVSS) Metrics Visualization

The VSMVS shall provide a CVSS metrics visualization as described in interface requirement INT-3.

FR-3.2 Attack Potential on Vehicle Assets Metrics

The VSMVS shall provide an Attack Potential on Vehicle Assets metrics visualization as described in interface requirement INT-5.

FR-3.3 Common Weakness Scoring System (CWSS) Metrics Visualization

The VSMVS shall provide a CWSS metrics visualization as described in interface requirement INT-7.

FR-3.4 Operational Safety Assessment (OSA) Cybersecurity Metrics Visualization

The VSMVS shall provide an Operational Safety Assessment (OSA) Cybersecurity metrics visualization as described in interface requirement INT-9.

FR-3.5 Security Vulnerability Metrics for Connected Vehicles Visualization

The VSMVS shall provide a Security Vulnerability metrics visualization as described in interface requirement INT-11.

FR-3.6 Vehicle Security Best Practices Assessment Metrics Visualization

The VSMVS shall provide a Vehicle Security Best Practices Assessment metrics visualization as described in interface requirement INT-13.

Test Requirements

The VSMVS requires testing and validation of the main application functionalities.

T-1: Unit Tests

Unit system testing shall be conducted for all functional system components. Unit tests shall be integrated and documented in the source code. The GitHub repository is found at this URL:
<https://github.com/UWF-CfC-FDOT/VSMVS>.

T-2: Integration Tests

System integration testing is not within the scope of the VSMVS system.

T-3: Test Report

An associated documentation of all system testing activities shall be provided. The Unit Test Overview document will be submitted as a separate deliverable.

Non-Functional Requirements

Non-functional requirements for the VSMVS are system attributes that are desired but not required.

The following are the non-functional requirements for the VSMVS:

NF-1: Portability

The development team will attempt to make the web enabled VSMVS system portable across multiple computing form factors.

NF-2: Usability

The development team will attempt to satisfy system usability features such as navigation, performance quality, and intuitiveness of interfaces.

NF-3: Speed

The development team will attempt to enhance the VSMVS system responsiveness to user interactions and database transactions.

Quality Attributes

The VSMVS is a proof of concept and not meant for a production release. As such, traditional quality attributes such as availability, security, robustness, etc. are not as relevant.

Source Code Repository and Version Control Requirement

The development team shall facilitate a source code repository and version control for the project.

SC-1: Source Code Repository and Control

The development team shall maintain a source code repository and version control on GitHub. The GitHub project URL is <https://github.com/UWF-CfC-FDOT/VSMVS>

Appendix A: Requirements Table

Requirement ID	Requirement Type	Requirement Name	Requirement Description	Priority
S1	Scope	System Scope	Defines the scope of the system	Must have
FR-1.1	Functional	CVSS Metrics	The VSMVS shall facilitate data input for the CVSS metrics	Must have
FR-1.2	Functional	Attack Potential on Vehicle Assets Metrics	Data input and storage shall be implemented for the attack potential on vehicle assets	Must have

FR-1.3	Functional	Common Weakness Scoring System (CWSS) Metrics	Data input and storage shall be implemented for the CWSS metric groups.	Must have
FR-1.4	Functional	Operational Safety Assessment (OSA) Cybersecurity Metrics	Data input and storage shall be implemented for the OSA Cybersecurity metrics.	Must have
FR-1.5	Functional	Security Vulnerability Metrics for Connected Vehicles	Data input and storage shall be implemented for the security vulnerability metrics.	Must have
FR-1.6	Functional	Vehicle Security Best Practices Assessment Metrics	Data input and storage shall be implemented for the security best practices assessment metrics.	Must have
FR-2.1	Functional	CVSS Metrics Calculator	The VSMVS shall provide a metrics calculator using the established formulae for the CVSS metrics	Must have
FR-2.2	Functional	Attack Potential on Vehicle Assets Metrics Calculator	The VSMVS shall provide a metrics calculator using the established formulae for the attack potential on vehicle assets metrics	Must have

FR-2.3	Functional	Common Weakness Scoring System (CWSS) Metrics Calculator	The VSMVS shall provide a metrics calculator using the established formulae for the CWSS metrics	Must have
FR-2.4	Functional	Operational Safety Assessment (OSA) Security Metrics Calculator	The VSMVS shall provide a metrics calculator using the established formulae for the OSA security metrics	Must have
FR-2.5	Functional	Security Vulnerability Metrics for Connected Vehicles Calculator	The VSMVS shall provide a metrics calculator using the established formulae for the security vulnerability metrics for connected vehicles	Must have
FR-3.1	Functional	CVSS Metrics Visualization	The VSMVS shall provide a CVSS metrics visualization as described in interface requirement INT-3	Must have
FR-3.2	Functional	Attack Potential on Vehicle Assets Metrics Visualization	The VSMVS shall provide an Attack Potential on Vehicle Assets metrics visualization as described in interface requirement INT-5	Must have
FR-3.3	Functional	Common Weakness Scoring System (CWSS) Metrics Visualization	The VSMVS shall provide a CWSS metrics visualization as described in interface requirement INT-7	Must have

FR-3.4	Functional	Operational Safety Assessment (OSA) Cybersecurity Metrics Visualization	The VSMVS shall provide an Operational Safety Assessment (OSA) Cybersecurity metrics visualization as described in interface requirement INT-9	Must have
FR-3.5	Functional	Security Vulnerability Metrics for Connected Vehicles Visualization	The VSMVS shall provide a Security Vulnerability metrics visualization as described in interface requirement INT-11	Must have
FR-3.6	Functional	Vehicle Security Best Practices Assessment Metrics Visualization	The VSMVS shall provide a Vehicle Security Best Practices Assessment metrics visualization as described in interface requirement INT-13	Must have
INT -1	Interface	CVE Data Collection Interface	The VSMVS shall provide a Graphical User Interface to be able to periodically collect information from threats using the NVD and CVE API	Must have
INT-2	Interface	CVSS Vector Data Entry Interface	The VSMVS shall provide a Graphical User Interface (GUI) for data entry of CVSS vector and metrics information	Must have
INT-3	Interface	CVSS Visual Interface	The VSMVS shall provide a corresponding visualization for the CVSS vector and metrics	Must have

INT-4	Interface	Attack Potential Data Entry Interface	The VSMVS shall provide a GUI for the data entry of attack potential of threats on vehicle assets	Must have
INT-5	Interface	Attack Potential Visual Interface	The VSMVS shall provide a GUI for the visualization of attack potential of threats on vehicle assets	Must have
INT-6	Interface	CWSS Vector Data Entry Interface	The VSMVS shall provide a GUI for data entry of CWSS vector and metrics information	Must have
INT-7	Interface	CWSS Visual Interface	The VSMVS shall provide a GUI for the visualization of CWSS vector and metrics information	Must have
INT-8	Interface	Operational Safety Assessment Data Entry Interface	The VSMVS shall provide a GUI for data entry of OSA metrics information	Must have
INT-9	Interface	Operational Safety Assessment Visual Interface	The VSMVS shall provide a GUI for visualization of OSA metrics information	Must have
INT-10	Interface	Security Vulnerability Data Entry Interface	The VSMVS shall provide a GUI for data entry for Security Vulnerability Metrics for Connected Vehicles information	Must have

INT-11	Interface	Security Vulnerability Visual Interface	The VSMVS shall provide a GUI for the visualization of Security Vulnerability Metrics for Connected Vehicles	Must have
INT-12	Interface	Vehicle Security Best Practices Assessment Data Entry Interface	The VSMVS shall provide a GUI for data entry for Vehicle Security Best Practices Assessment Metrics for Connected Vehicles information	Must have
INT-13	Interface	Vehicle Security Best Practices Assessment Visual Interface	The VSMVS shall provide a GUI for the visualization of Vehicle Security Best Practices Assessment Metrics for Connected Vehicles information	Must have
INT-14	Interface	API Interaction Logger	The VSMVS shall log any interactions with external APIs in a format containing the time, message type, sent request, response code, and response	Must have
INT-15	Interface	DBMS Transaction Logger	The VSMVS shall log all threat database transactions in a format containing the time, query type, query contents, and response	Must have
INT-16	Interface	Error/Exception Logger	The VSMVS shall log errors or exceptions in a format containing the time of the event and a stack trace	Must have

T-1	Test	Unit Test	Unit system testing shall be conducted for all functional system components	Must have
T-2	Test	Integration Test	System integration testing is not within the scope of the VTME system	Out of scope
T-3	Test	Test Report	An associated documentation of all system test activities shall be provided	Must have
NF-1	Non-functional	Portability	The development team will attempt to make the web enabled VTME system portable across multiple computing form factors	Could have
NF-2	Non-functional	Usability	The development team will attempt to satisfy system usability features such as navigation, performance quality, and intuitiveness of interfaces	Could have
NF-3	Non-functional	Speed	The development team will attempt to enhance the VTME system responsiveness to user interactions and database transactions	Should have
SC-1	Source Code	Source Code Control	The development team shall maintain a source code repository and version control on GitHub	Should have

Appendix B: Requirements Traceability Matrix

Requirement ID	Requirement Description	Test Case	Status
S1	Defines the scope of the system	N/A	N/A
T-1	Unit system testing shall be conducted for all functional system components	Multiple Test Cases	Not Started
T-2	System integration testing is not within the scope of the VTME system	N/A	N/A
T-3	An associated documentation of all system test activities shall be provided	N/A	Not Started
NF-1	The development team will attempt to make the web enabled VTME system portable across multiple computing form factors	N/A	N/A
NF-2	The development team will attempt to satisfy system usability features such as navigation, performance quality, and intuitiveness of interfaces	N/A	N/A
NF-3	The development team will attempt to enhance the VTME system responsiveness to user interactions and database transactions.	N/A	N/A
SC-1	The development team shall maintain a source code repository and version control on GitHub	N/A	N/A
FR-1.1	The VSMVS shall facilitate data input for the CVSS metrics	Not Started	Not Started
FR-1.2	Data input and storage shall be implemented for the attack potential on vehicle assets	Not Started	Not Started

FR-1.3	Data input and storage shall be implemented for the CWSS metric groups.	Not Started	Not Started
FR-1.4	Data input and storage shall be implemented for the OSA Cybersecurity metrics.	Not Started	Not Started
FR-1.5	Data input and storage shall be implemented for the security vulnerability metrics.	Not Started	Not Started
FR-1.6	Data input and storage shall be implemented for the security best practices assessment metrics.	Not Started	Not Started
FR-2.1	The VSMVS shall provide a metrics calculator using the established formulae for the CVSS metrics	Not Started	Not Started
FR-2.2	The VSMVS shall provide a metrics calculator using the established formulae for the attack potential on vehicle assets metrics	Not Started	Not Started
FR-2.3	The VSMVS shall provide a metrics calculator using the established formulae for the CWSS metrics	Not Started	Not Started
FR-2.4	The VSMVS shall provide a metrics calculator using the established formulae for the OSA cybersecurity metrics	Not Started	Not Started
FR-2.5	The VSMVS shall provide a metrics calculator using the established formulae for the security vulnerability metrics for connected vehicles	Not Started	Not Started
FR-3.1	The VSMVS shall provide a CVSS metrics visualization as described in interface requirement INT-3	Not Started	Not Started

FR-3.2	The VSMVS shall provide an Attack Potential on Vehicle Assets metrics visualization as described in interface requirement INT-5	Not Started	Not Started
FR-3.3	The VSMVS shall provide a CWSS metrics visualization as described in interface requirement INT-7	Not Started	Not Started
FR-3.4	The VSMVS shall provide an Operational Safety Assessment (OSA) Cybersecurity metrics visualization as described in interface requirement INT-9	Not Started	Not Started
FR-3.5	The VSMVS shall provide a Security Vulnerability metrics visualization as described in interface requirement INT-11	Not Started	Not Started
FR-3.6	The VSMVS shall provide a Vehicle Security Best Practices Assessment metrics visualization as described in interface requirement INT-13	Not Started	Not Started

Appendix C: Glossary

Term	Description
API	Application Program Interface
ATT&CK	Adversarial Tactics, Techniques, and Common Knowledge
ATT&CK Framework	A knowledge base of adversary tactics and techniques based on real-world observations.
AWS	Amazon Web Services
CVE	Common Vulnerabilities and Exposures
CWE	Common Weakness Enumeration
Cyber Kill Chain	A model developed by Lockheed Martin® used for the identification and prevention of cyber intrusions.
EC2	Elastic Compute Cloud
GUI	Graphical User Interface
IIS	Internet Information Services
IoC	Indicators of Compromise
.NET	A cross-platform, open-source developer platform created by Microsoft
OSINT	Open-Source Intelligence
OTX	Open Threat Exchange
Tactics, Techniques and Procedures (TTPs)	Activities and methods used by an adversary to carry out a cyber attack
VSMVS	Vehicle Security Metrics Visualization System
VTCS	Vehicle Threat Collection System
VTDBS	Vehicle Threat Database System
VTME	Vehicle Threat Modeling Engine

Appendix III. Technical Report UWF-TR-FDOT-002-02

Connected Vehicle Security Metrics and Visualization

A White Paper

Technical Report UWF-TR-FDOT-002-02

Contract Number: BED34. Task Order: 977-01
Florida Department of Transportation (FDOT)

Guillermo A. Francia, III, Principal Investigator
Center for Cybersecurity
University of West Florida, USA
gfranciaiii@uwf.edu

Abstract. The rapid advancement of connected and autonomous vehicles created new challenges for security and safety professionals. The sophistication of vehicle communication systems, found externally and internally, provides an added complexity to the issue. In security parlance, this is an expansion of the attack surface on vehicles. These challenges prompted the enhancement of existing and the development of new safety and security standards initiated by government, industry, and trade organizations. These initiatives clearly underscore the need to examine the state of connected vehicle security. For that reason, security metrics must be developed. As a major component of continuous improvement, quantitative and qualitative measures must be devised to be able to make a full appreciation of the process. This white paper describes updates on previous works by the PI on connected vehicle security metrics, offers new metrics, illustrates the applicability of the metrics through sample calculations, and proposes visualization systems to enhance their utilization.

Keywords. Common Vulnerability Scoring System (CVSS), Common Weakness Scoring System (CWSS), Operational Safety, Security Attacks, Security Metrics, Threats, Vulnerabilities, Connected Vehicle, Security Visualization

I. INTRODUCTION

The unprecedented advancement of technologies in both internal and external communication of connected vehicles imposes unwarranted consequences on their security and safety. Fortunately, the eagerness to deploy these technologies is tempered by government regulations. It is imperative that sound regulatory framework be put in place to ensure the security and safety of modern vehicles.

The ISO/SAE 21434 [20] came about when two organizations: ISO 26262 and SAE J3061 realized a common goal, i.e., automotive safety and security related standards. The ISO/SAE 21434 document titled “Road vehicles—Cybersecurity Engineering” established an effective global standard for automotive cybersecurity [21] [22]. The document provides the necessary vocabulary, objectives, requirements, and guidelines that are pertinent to cybersecurity engineering. Essentially, it enables organizations to define cybersecurity policies and processes, manage risks, and foster cybersecurity culture awareness [23].

The internal communication among electronic control units (ECUs), which are embedded devices that controls and automates a vehicle operation and performance, goes through inherently insecure channels, such as the Controller Area Network (CAN). For instance, a vulnerability, described by Trend Micro [24], enables a stealthy denial-of-service attack that practically works for every automotive vendor. This Exploitable hardware design flaws in some capacitive micro-electromechanical system (MEMS) accelerometer sensors produced by prominent automobile parts manufacturers were reported in another ICS-CERT alert: ICS_ALERT-17-073-01A in early 2017.

External communication systems in vehicles enable access convenience and online services [25]. Vehicle external communication can be classified into four main categories: vehicle-device (V2D) communication, vehicle-vehicle (V2V) communication, vehicle-infrastructure (V2I) communication, and vehicle-pedestrian (V2P) communication [2]. These are supported by cellular-vehicle-everything (C-V2X) [26] to enable a more efficient transportation ecosystem. With the advent of Electric Vehicles (EVs), a new type has emerged--vehicle-grid (V2G) communication [27].

II. CONNECTED VEHICLE SECURITY

1. Vehicle Data, Devices and Communication

Cybersecurity policies protect data, devices, and communication channels. Connected vehicles operate utilizing those three entities. The confidentiality, integrity, and availability of connected vehicle data is paramount to their safe operation. These data are generated and consumed by devices in vehicles and are transmitted through various communication channel types as described above. Existing connected vehicle data sources, as described by Otonomo [28], include the following:

- *Traffic sensors.* Data collected by these devices include vehicle speed, direction, location, and weight.
- *Cameras.* These can be CCTV systems or in-vehicle surveillance systems.
- *Mobile phones.* The data collected by these devices include geolocations and other mobility data while walking, riding a bike, on a public transport system, etc.
- *Human Surveillance.* Data collected through surveys.
- *Lidar/Radar/Sonar.* These devices, mounted on vehicles, collect and/or generate data that can be used for autonomous or semi-autonomous operations.

Vehicle data provide several benefits such as acquisition cost reduction, lower operating cost, efficient logistics, effective data utilization, and expanded services [28]. However, these benefits introduce security tradeoffs which are discussed in subsequent sections.

2. Connected Vehicle Threats and Vulnerabilities

The Car Hacking Village, a conference track and interactive event at DEF CON, featured a study by NDIAS—a Japan-based automotive cybersecurity assessment group. In their study, the group tested over 40 ECUs provided by multiple manufacturers (15 in-vehicle infotainment units,

8 telematic control units, 8 gateways, advanced driver-assistance systems, smart key units, and electric vehicle chargers). The results of the study revealed more than 300 vulnerabilities in software and hardware components [29].

Threats and vulnerabilities in connected vehicles have been discovered in recent years and continue to proliferate. Prominent among these are the following:

- The vulnerability of Tesla’s touch screen infotainment system was exploited and used as a gateway to manipulate the driver’s seat motor, the windshield wipers, the turn indicators, and the sunroof from a distance while the car was in motion [30].
- The remote keyless system on Renault ZOE 2021 vehicles sends 433.92 MHz RF signals from the same Rolling Codes set for each door-open request, which allows for a replay attack [31].
- The lack of keyless ignition hardware made Hyundai and Kia vehicles (2016-2021 models) target of car thieves. Hyundai came up with a \$170 security kit to deal with the issue [59].
- The rolling-PWN replay vulnerability of keyless entry system on Honda vehicles [32].
- Security researchers found a vulnerability in SiriusXM, a 3rd party services-provider that powers telematics and infotainment systems for connected vehicles, by sending API requests with the VIN on a unique ID field [33].

In McCarthy, et al. [51], use cases of Automotive Security Threats are described. The use cases include, among others, brake disconnect, horn activation, engine halt air bag, portable device injection, key fob cloning, cellular attack, and malware download. In addition, vehicle threat matrix development and matrix population using use cases are demonstrated by the authors. In [60], a threat matrix on a Denial-of-Service threat on CAN protocol use case is shown. The threat matrix includes categories of severity, sophistication level, and likelihood. Further, the likelihood is assessed by an expert as either high, medium, or low. In this whitepaper, we apply this concept by building a threat matrix on a vehicle remote keyless entry system use case that was documented in CVE-2022-38766 [31]. The populated threat matrix is depicted on Table 1.

Table 1. A Populated Threat Matrix on a Remote Keyless Entry System

Matrix Category	Category Description	Category Options
ID Number	ID for the attack	0002
Attacked Safety and Non-safety Zone Groups	Components and systems that are targeted or used as support	Remote key fob
Attacked Zone Safety	Safety related functions	Yes
Component/System	Component or system under attack	RF-signal activated door
Exploitable Vulnerability	RF Signal reuse	433.92 MHz RF signals
Attack Vector	Replay of rolling code set	RF receiver/transmitter
Access Method	Remote wireless	Replay of RF rolling code set
Attack Type	Type of attack	Replay attack

Resources Required	Resources needed to carry out the attack	RF Signal capture and generator
Severity	Degree of severity	Medium
Trip Phase	Vehicle movement status	Parked
Loss of Privacy	Privacy compromised	No
Sophistication Level	Complexity of potential attack	Medium-need to know how to use RF signal capture and generator device
Difficulty of Implementation	How difficult is it to implement	Medium
Likelihood	Likelihood of a potential attack to be carried out	Medium

3. Vehicle Security Attacks and Mitigations

The inherently insecure Controller Area Network (CAN) in most connected vehicles is the focus of several studies [34] [35]. The lack of message authentication and the absence of data encryption are the main enabler of malicious activities in this network protocol. To alleviate those issues, research works, such as those using techniques such as Machine Learning [36] [37], authentication code [38], and clock skew signature [39], started to proliferate.

Other notable documented attack models and vulnerability mitigations include the following:

- Petit, Feiri, and Kargl [40] described an abstract model of attack surfaces on the vehicular communication domain. The attack model considers the sensor data in various stages: acquisition, processing, storing, and transmission. This seminal work has been extended by Monteuis, et al. [41] with the notion of a secured automotive perception consisting of two main components: Objects and Data Stages.
- In [42], the design, implementation, and evaluation of a hardware security module for a modern automotive vehicle is presented.
- Lokman et al. conducted a systematic review of Intrusion Detection Systems (IDS) for automotive CAN bus system based on detection approaches, deployment strategies, attacking techniques and technical challenges [43]

4. Vehicle Attack Surfaces

An attack surface is the number of possible points or attack vectors where a malicious user can initiate an unauthorized access to the system to manipulate the system or extract data [61]. The objective is to minimize this surface to reduce the risk of a successful cyberattack.

In Bauer and Schartner [48], a table depicting attack surfaces and the classification of attack potential according to common criteria is presented. The table includes information on the difficulty and the impact of a certain exploit to an asset.

Maple, et al. [62] propose a reference architecture using a hybrid Functional-Communication viewpoint for attack surface analysis of Connected Autonomous Vehicles (CAVs), including the Device Edge and Cloud systems CAVs.

Various attack surfaces on vehicles ranging from the OBD port to the infotainment system were examined by Koscher, et al. [63].

Checkoway, et al. demonstrated the feasibility of external attacks on modern automobiles through a systematic analysis of external attack vectors [64]. Their study is focused on three main areas: threat model characterization, vulnerability analysis, and threat assessment. On threat model characterization, the feasibility of multiple I/O channels, on indirect physical access channels, short-range wireless access, and long-range wireless access, to deliver malicious payload is demonstrated. On vulnerability analysis, the study revealed the existence of exploitable vulnerabilities without requiring physical access. Finally, on threat assessment, the study puts forth the arguments on the utility of these attacks [64].

As the number of connected vehicles worldwide continue to grow at an almost exponential rate, the attack surfaces on connected vehicles cannot be ignored. Vehicle security attack surfaces include the following [2]:

- Telematics servers. These servers act as remote command and controls, which not only collect data but also send remote commands such as locking and unlocking doors, switching engines on and off, etc.
- Onboard Diagnostic (OBD) Ports. These ports can be remotely reached with OBD devices that are configured for Wi-Fi or cellular communications.
- Mobile Device Apps. Mobile device applications are increasingly used to communicate with connected vehicles via centralized application servers. When these servers get compromised, so goes all vehicles that are dependent on them.
- Wi-Fi devices. Several car manufacturers equip vehicles with built-in Wi-Fi access points for immediate connection on the Internet. These devices, when left unsecured, can easily become entry points for an attack.
- Telematics Control Units. These devices are usually found on OBD ports as wireless dongles. Their main purposes are for data collection for insurance, fleet management, location tracking, and performance monitoring.

III. INDUSTRY AND GOVERNMENT INITIATIVES AND STANDARDS

In [2], several initiatives geared toward the protection of a vehicle's electronic control units have been reported. Notable examples are the E-safety Vehicle Intrusion Protected Application (EVITA) Project [44], the Preparing Secure Vehicle-to-X Communication Systems (PRESERVE) Project [45], Secure Vehicular Communication (SeVeCom) Project [46], and the Society of Automotive Engineers (SAE) J3061 Guidebook [47].

The National Highway Traffic Safety Administration (NHTSA) document titled "Automotive Security Best Practices for Modern Vehicles" [50] presents the results and analysis of a review of best practices and observations in the field of cybersecurity involving electronic control systems

across a variety of industry segments where the safety-of-life is concerned. The document was updated in September 2022 [65]. It builds upon emerging voluntary industry standards such as the ISO/SAE 21434, “Road Vehicles—Cybersecurity Engineering” [23] and other best practice documents such as that produces by the Automotive Information Sharing and Analysis Center (Auto-ISAC) [58].

We close this section by enumerating the following additional standards. This list could be used as a guidance or a starting point for further exploration of applicable instruments in the design and implementation of tools and processes in connected vehicle security.

- SAE J3101 [66]. This standard describes the requirements for hardware protected security for ground vehicle applications. Use cases include, among others, creation of key fob, re-flashing of the ECU firmware, reading and exporting of PII out of the ECU, service activities on the ECU, etc.
- International Automotive Task Force (IATF) 16949:2016 [67] provides guidelines on common processes and procedures for the automotive industry. Certification to this standard is required throughout the automotive supply chain.
- ISO/IEC/IEEE 29119-1:2013 Software and Systems Engineering—Software Testing-Part 1: Concepts and Definitions [68]. This set of standards defines an internationally-agreed set of standards for software testing. This is an indispensable tool for the design and development of system and application software for the automotive vehicle.

IV. VEHICLE SECURITY METRICS

Continuous improvement is a practice that seeks to enhance operations. Security is a process and not a goal and should adapt into continuous improvement progression. The manner to measure this progression is through security metrics. To this end, we propose vehicle security metrics that are practical and relevant.

A good metric should measure the relevant data that satisfy the needs of decision makers and should be quantitatively measurable, accurate, validated on a solid base, inexpensive to execute, able to be verified independently, repeatable, and scalable to a larger scale [69]. By adapting security risk regression that is successful in predicting attacks from simple security threats, Schechter [70] concludes that security strength is a key indicator of security risks for more complex security threats in information systems. In congruence, Manadhata and Wing propose the attackability of a system as an indicator of security strength [71]. Their security metric is based on the notion of attack surface by comparing attackability of systems along three abstract dimensions: method, data, and channel. The attackability of a system is a cost-benefit ratio between efforts of gaining access and potential impacts of security failure among the three dimensions [54].

Human factor, the weakest link in security, adds additional complication in safety and security procedures alike. Therefore, in order to properly measure security at the level of socio-technical

system in which technical design decisions are influenced by human factors, it is imperative that social behavior needs to be taken into consideration [72] [73].

There exists notable works on automotive vehicle security metrics. In [12], a set of security metrics for the software system in a connected vehicle is proposed. The set of metrics provides a quantitative indicator of the security vulnerability of the following risks on the system software: ECU coupling, communication, complexity, input and output data, and past security issues. For a detailed discussion on this set of metrics, the reader is referred to [2].

A Bayesian Network (BN) for connected and autonomous vehicle cyber-risk classification was developed by Sheehan, et al. [52]. The BN model uses the Common Vulnerability Scoring System (CVSS) software vulnerability risk-scoring framework for input parameters specifically on the Global Positioning System (GPS) jamming and spoofing.

In the following section, we present a collection of vehicle security metrics similar to those in an earlier work on critical infrastructure and industrial controls systems security [53] [54].

1. Common Vulnerability Scoring System (CVSS)

CVSS is an open framework for communicating the characteristics and severity of software vulnerabilities. It consists of three metric groups: Base, Temporal, and Environmental. Details on these metrics can be found in [1].

In [2], an illustration on the application of these metrics on the vulnerability of the Tesla Model S/X vehicles manufactured before March, 2018 [3] and the vulnerability in the infotainment component of BMW Series vehicles (CVE-2018-9322) [4] are illustrated.

In this white paper, we apply the metrics using a very recent use case on a popular models of Honda vehicles. This Remote Keyless Entry (RKE) vulnerability is documented in [74]. The vulnerability allows attackers to perform unlock operations and force a resynchronization after capturing five consecutive and valid RKE signals. It has the following CVSS v3.1 Base vector:

AV:A/AC:H/PR:N/UI:R/S:U/C:N/I:H/A:H

This translates to Adjacent for the Attack Vector (AV), High for Attack Complexity (AC), None for Privileges Required (PR), Required for User Interaction (UI), Unchanged for Scope, None for Confidentiality (C), High for Integrity (I), and High for Availability. The CVSS score for this Base vector is 6.4, which is categorized as Medium. This CVSS Base Score is calculated based on a table of metric values and the formulae found in CVSS v3.1 Specification Document [1]:

2. Common Methodology for IT Security Evaluation (CEM) [5]

The CEM is a companion document to the Common Criteria for Information Technology Security Evaluation (CC). It defines the minimum actions to be taken by an evaluator conducting a CC evaluation utilizing the criteria and evidence as stated in the CC.

In this white paper, we specifically examine the attack potential on an automotive vehicle. The following factors need to be considered when performing an analysis of an attack potential:

- Elapsed Time. Time taken by an attacker to identify a potential vulnerability, to develop an attack method, and to sustain effort required to execute the attack. Value ranges from 1 day to more than 6 months.
- Specialist expertise. Describes the level of sophistication of the attacker. Levels include laymen, proficient personnel, expert, and multiple experts.
- Knowledge of the target. Refers to the familiarity of the attacker on the target. Levels include public knowledge availability, restricted information, sensitive information, and critical information.
- Window of opportunity. This refers to the duration of time in which the vulnerability is exploitable. Window of opportunity includes unlimited, easy, moderate, difficult, and none.
- IT hardware/software or other equipment. This refers to the availability and the level of complexity of equipment/software needed to identify or exploit a vulnerability. Classes of equipment/software include standard, specialized, highly specialized, and multi-specialized.

Levels in each factor are assigned corresponding numeric values and illustrated in the Common Criteria Portal [5]. Bauer & Schartner demonstrate sample calculations of attack potential [48] on generic threat assets in an automotive vehicle. The table is augmented by our own analysis of threats that are prevalent on connected automotive vehicles. Those last 5 rows represent denial of telematics service, unauthorized access, command injection, identity masquerading and unauthorized data tampering. An excerpt of those calculations is shown on the first 3 rows of Table 2.

An unauthorized access may originate locally, such as a cloning of key fob, or remotely through an internetwork communication channel. Time factor could be between one day to one week (1); expertise factor requires at least at the proficient level; knowledge of the vehicle assets will be, most likely, at the restricted level; the window of opportunity is unlimited; and the attack may not need specialized equipment.

The time to accomplish identity masquerading in connected vehicles may take a bit longer compared to unauthorized access; expertise factor requires at least at the proficient level; knowledge of the vehicle assets will be, most likely, at the sensitive level; the window of opportunity is very limited; and the attack may need some specialized equipment.

Data tampering can be accomplished by the widespread Man-In-The-Middle (MITM) attack tools. The time to accomplish such attack can take place very quickly; expertise factor requires at least at the semi-proficient level; knowledge of the vehicle assets will be most likely at the

familiarity level; the window of opportunity is somehow large; and the attack may not need specialized equipment.

2.1 Threats on Assets

We identify the following threats on vehicle assets and derive the attack potential metrics. We apply the previously defined factors for analysis and aggregate the metrics. A sample attack potential metrics derivation is shown on Table 2. The threats on vehicle assets are described in the following.

1. *False Data from ECU.* This type of attack occurs when a vehicle device or Electronic Control Unit (ECU), connected to the CAN bus, sends false information to other ECUs to induce them to behave abnormally.
2. *Blocking of CAN Bus.* The CAN protocol has an inherent vulnerability that can easily be exploited. The Arbitration ID field determines who has preference in using the bus. The node that has lowest Arbitration ID field value gets preference over the other nodes. An attack can be realized by the continuous transmission of CAN messages having very low Arbitration ID field values, which essentially blocks the CAN bus to other traffic with higher Arbitration ID.
3. *Malicious Software.* This is a common attack that is enabled by malicious software originating from the supply chain, the vehicle system patches, the telematics channel, or the On-Board Device (OBD) port.
4. *Denial of Telematics Service.* This is a special case of the denial-of-service attack in which the telematics communication channel is overwhelmed by excessive unwanted traffic to prevent the legitimate transfer of information to materialize.
5. *Unauthorized Access.* An unauthorized access may originate locally, such as from a cloned key fob, or remotely through an internetwork communication channel such as WiFi or 5G broadband.
6. *Command Injection.* In this type of attack, malicious commands are injected into the vehicle intranet to induce an abnormal behavior.
7. *Masquerading.* This attack potential is realized by impersonating an authorized system or user to take control of the vehicle.
8. *Data Tampering.* This attack potential can be realized by the widespread Man-In-The-Middle (MITM) attack tools. The time to accomplish such attack can take place very quickly.

Table 2. Attack Potential Calculation

THREAT on ASSETS	TIM E	E X P E R T I S E	K N O W L E D G E	O P P O R T U N I T Y	E Q U I P M E N T	TOTAL
False Data from ECU	10	6	3	1	4	24
Blocking of CAN Bus	8	3	3	1	4	19
Malicious Software	7	5	3	1	4	20
Denial of Telematics Service	1	3	4	2	2	12
Unauthorized Access	8	5	3	1	2	19
Command Injection	2	5	4	1	2	14
Masquerading	4	3	5	5	5	22
Data Tampering	1	2	4	1	2	10

In Table 2, the total attack potential for each threat is simply a summation of the value assigned to each of the attribute of a successful attack. These results can be utilized during the decision-making process of cybersecurity asset allocation towards risk mitigation or prevention.

3. Common Weakness Scoring System [6]

The Common Weakness Scoring System (CWSS) is a mechanism for evaluating software weaknesses in a consistent, flexible, open manner. It is a community-based undertaking which addresses the need for prioritizing the software vulnerability issues. The measurements are organized into three metric groups: Base Finding, Attack Surface, and Environmental. The groups, including their subgroups, as described in [6] are as follow:

- **Base Finding.** This group represents the inherent risk of the weakness, confidence in the accuracy of the finding, and strength of controls.
 - *Technical Impact (TI).* The potential result of the weakness.
 - *Acquired Privilege (AP).* The type of privilege acquired after successfully exploiting the weakness.
 - *Acquired Privilege Layer (AL).* The operational layer acquired after successfully exploiting the weakness.
 - *Internal Control Effectiveness (IC).* The ability of the control to prevent the exploitation of the weakness.

- *Finding Confidence (FC)*. The confidence that reported issue is an exploitable weakness.
- **Attack Surface**. This group represents the barrier in the weakness' exploitation.
 - *Required Privilege (RP)*. The type of privilege needed to be able to reach the code/functionality to exploit the weakness.
 - *Required Privilege Layer (RL)*. The operational layer with which the attacker must have the privilege to reach to exploit the weakness.
 - *Access Vector (AV)*. The channel through which the attacker must communicate to reach the code/functionality that contains the weakness.
 - *Authentication Strength (AS)*. The strength of authentication protecting the weakness.
 - *Level of Interaction (IN)*. The interactions needed to exploit the attack.
 - *Deployment Scope (SC)*. The extent of the weakness on all the software/hardware deployment.
- **Environmental**. This represents the attributes of the weakness that are specific to a particular environment or operational context.
 - *Business Impact (BI)*. *The potential impact to the organization by the successful exploitation of the weakness.*
 - *Likelihood of Discovery (DI)*. *The likelihood that the attacker can discover the weakness.*
 - *Likelihood of Exploit (EX)*. *The likelihood that weakness, if discovered, can be exploited.*
 - *External Control Effectiveness (EC)*. *The capability of external control to prevent the exploitation of the weakness.*
 - *Prevalence (P)*. *The frequency of occurrence of the weakness in the software*

3.1 Group Scoring

3.1.1 Base Finding Subscore

The Base Finding Subscore is calculated as follows:

$$\text{Base} = [(10 * \text{TechnicalImpact} + 5 * (\text{AcquiredPrivilege} + \text{AcquiredPrivilegeLayer}) + 5 * \text{FindingConfidence}) * f(\text{TechnicalImpact}) * \text{InternalControlEffectiveness}] * 4.0$$

$$f(\text{TechnicalImpact}) = 0 \text{ if } \text{TechnicalImpact} = 0; \text{ otherwise}$$

$$f(\text{TechnicalImpact}) = 1.$$

The maximum potential Base Finding Subscore is 100.

3.1.2 Attack Surface Subscore

The Attack Surface Subscore is calculated as:

```
AttackSurface = [ 20*(RequiredPrivilege + RequiredPrivilegeLayer +
AccessVector) + 20*DeploymentScope + 15*LevelOfInteraction +
5*AuthenticationStrength ] / 100.0
```

The formula indicates that the required privileges and access makes up 60% of the Attack Surface subscore; while the deployment scope and the interaction receive weights of 20% and 15% respectively, authentication receives a minor focus of just 5%.

The Attack Surface subscore ranges between 0 and 100, which is divided by 100.

3.1.3 Environmental Subscore

The Environmental Subscore is calculated as:

```
Environmental = [ (10*BusinessImpact + 3*LikelihoodOfDiscovery +
4*LikelihoodOfExploit + 3*Prevalence) * f(BusinessImpact) *
ExternalControlEffectiveness ] / 20.0
```

```
f(BusinessImpact) = 0 if BusinessImpact == 0; otherwise
f(BusinessImpact) = 1
```

BusinessImpact accounts for 50% of the environmental score. ExternalControlEffectiveness is always non-zero (to account for the risk that it can be inadvertently removed if the environment changes). It can have major impact on the final score. The combination of LikelihoodOfDiscovery and LikelihoodOfExploit accounts for 35% of the score, and Prevalence at 15%.

3.1.4 CWSS Score

The Common Weakness Scoring System is calculated by combining the three subscores. Thus,

```
CWSS = Base * AttackSurface * Environmental
```

4. Operational Safety Assessment Metrics

The purpose of this section is to provide an insight on the impact of cybersecurity to operational safety assessment (OSA). There exists several OSA metrics that have been proposed, adopted, and studied [7] [8]. SAE J3237 [9], a work in progress information report, is currently being developed. This report provides definitions and lexicon for describing operational safety metrics for ADS vehicles. The characteristics of the listed metrics include the following: definition, data source, subjectivity, observable variable, formulation, subjective assumptions and thresholds, and origin. A related work by the SAE V&V Task Force is the development of a proposed taxonomy for a Recommended Practice on Operational Safety metrics [10]. At the classification level of the proposed taxonomy are the following operational safety metrics [10]:

- *Safety Envelope Metric.* This is an OSA measure of the connected vehicle's maintenance of safe boundary. An example is the vehicle's capability of maintaining the safe distance driving rule.
- *Behavioral Metric.* This OSA measure indicates the improper behavior of the subject vehicle. Examples include speeding and sudden or hard braking.
- *Component Metric.* This is an OSA measure of the performance of the vehicle components under normal operating condition. For example, an Electronic Control Unit (ECU) operating according to its specification.
- *Sensing Metric.* This is an OSA measure of the accuracy of data collected by the CAV sensors. An example is the data collected by Roadside Units (RSUs). For non-autonomous vehicles, it is the measure of the quality of data transmitted or collected by vehicles participating in a vehicle-to-infrastructure (V2I) environment.
- *Perception Metric.* This OSA measure pertains to the quality of interpretation of environment data collected by the CAV sensors. An example would be the automated recognition of traffic signs and signals.
- *Planning Metric.* This OSA metric measures the ability of the CAV to devise a suitable trajectory through the CAV environment. An example is the quality of the CAV's planned trajectory in collision avoidance. For non-autonomous vehicles, it is the measure of the quality of maintaining control of the vehicle in the presence of an unexpected obstacle.
- *Control Metric.* This OSA metric measures the ability of the CAV to execute the planned route. An example is the ability of the CAV to stay on the predetermined route. For non-autonomous vehicles, it is the measure of the quality of maintaining control of the vehicle navigation.

We augment these OSA metrics with the following:

- *Authentication Metric.* This OSA metric measures the quality of the authentication system deployed in the vehicle. This is extremely useful in modern vehicles that rely on communications such as those in V2V or V2I environment.
- *Physical Access Metric.* This OSA metric measures the strength of physical access protection of vehicle controls. An example is the unsecured physical access to an OBD port which could compromise the vehicle's CAN bus.
- *Communication Channel Metric.* This OSA metric pertains to the quality of the communication channel used by the vehicle.

4.1 Cybersecurity Metrics for Operational Safety

We investigate the impact of cybersecurity to operational safety. In doing so, we devise cybersecurity metrics that have close affinities with OSA metrics. These cybersecurity metrics for operational safety are described in the following:

- *Safety Envelope Metric.* This cybersecurity metric measures the security resiliency of a connected vehicle to be able to maintain a safe boundary amidst a cyber intrusion incident. An example is a vehicle's capability in preventing malicious manipulation of the control and sensing systems that enable safe distance driving operation. Values range from 0.0 for least resilient to 1.0 for most security resilient.
- *Behavioral Metric.* This cybersecurity metric measures the vehicle's capability to protect against a cyber-attack that enables the improper behavior of the subject vehicle. An

example of such attack is the manipulation of the vehicle cruise control mechanism. Values range from 0.0 for least capable to 1.0 for most capable.

- *Component Metric.* This is a measure of the susceptibility of the vehicle components to cyber-attack. For example, an Electronic Control Unit (ECU) device originating from an unverifiable supply chain may be highly susceptible to cyber-attack. Values range from 0.0 for most susceptible resilient to 1.0 for least susceptible.
- *Sensing Metric.* This cybersecurity metric pertains to the integrity and accuracy of data collected by the vehicle sensors. Roadside Units (RSUs) that are not properly secured may produce inaccurate or tampered data. Values range from 0.0 for least reliable data to 1.0 for most reliable data.
- *Perception Metric.* This cybersecurity metric pertains to the security of the system that provides for the interpretation of environment data collected by the vehicle sensors. For example, an insecure image processing system that is highly susceptible to an attack may provide inaccurate interpretation of traffic signs or signals. Values range from 0.0 for least secure to 1.0 for most secure.
- *Planning Metric.* This cybersecurity metric measures the vulnerability of the trajectory planning system to malicious intrusion. Values range from 0.0 for most vulnerable to 1.0 for least vulnerable.
- *Control Metric.* This cybersecurity metric measures the vulnerability of the vehicle's control system to malicious intrusion. Values range from 0.0 for most vulnerable to 1.0 for least vulnerable.
- *Authentication Metric.* This cybersecurity metric measures the security posture of the authentication system deployed in the vehicle. Values range from 0.0 for least secure to 1.0 for most secure.
- *Physical Access Metric.* This cybersecurity metric measures the strength of physical access protection of vehicle controls. Values range from 0.0 for least physically secure to 1.0 for most physically secure.
- *Communication Channel Metric.* This cybersecurity metric pertains to the level of protection of the communication channel used by the vehicle. Security characteristics of data transmission such as encryption, authentication, and attribution are pertinent concerns in this metric. Values range from 0.0 for least secure to 1.0 for most secure.

Emulating the evaluation methodology of the OSA metrics that was introduced by Wishart, et.al. [11], we present four evaluation factors for the formulation of the aggregation of cybersecurity metrics. The four evaluation factors are described in the following:

- *Reliability.* This factor quantifies the fidelity of the sources of measurement data. For instance, data originating from actual events carry a higher value than those from simulated events. Values range from 0.1 for less reliable to 1.0 for most reliable.
- *Relevance.* This factor quantifies the relevance of the measurement to a subject vehicle. This value may vary according to the specificity of data such as make and model of the subject vehicle. Data for a Honda CRV is more specific than data that refers to Honda vehicles in general. Values range from 0.1 for least relevant to 1.0 for most relevant.
- *Extent.* This factor quantifies the scope or extensiveness of the measurement data. The value ranges from 0.1 for least extensive to 1.0 for most extensive.

- *Criticality*. This factor quantifies the gravity of a specific metric. For instance, security measurement on control will carry a heavier weight than that on safety envelope. The value ranges from 0.1 for least critical to 1.0 for most critical.

The Aggregate Security Metric (ASM) for a specific vehicle is calculated as

$$ASM = Reliability \times Relevance \times Extent \times \left(\sum_{k=1}^N Criticality_k \times Security_Metric_k \right)$$

The ASM value will range from 0 to 10.

5. Security Vulnerability Metrics for Connected Vehicles

The purpose of this section is to provide guidance to security engineers and testers using security vulnerability metrics that measure weak or vulnerable features in the software system of connected vehicles. These metrics are based on the seminal work of Moukahal and Zulkernine [12]. We describe each of the following risks on connected vehicles that may eventually contribute to the likelihood of exploiting a vulnerability.

5.1 ECU Coupling Risk

This risk is manifested by level of interconnection among ECU components. This means the higher the coupling value the higher is the probability for vulnerabilities. Thus, for every functionality, F, for all ECUs, N, and for all communication links between ECU j and ECU k, the ECU coupling risk, R_{EC} , is calculated as

$$R_{EC}(F) = \sum_{j=1, k=1}^N C_{jk}$$

$C_{jk}=1$ if there is at least one information transfer between ECU j and ECU k; 0 otherwise.

$$\text{Max} (R_{EC}(F)) = N$$

5.2 Communication Channel Risk

This risk is based on the communication channel types that are available for connected vehicles: vehicle to vehicle (V2V), vehicle to infrastructure (V2I), user to vehicle (U2V), and intra-vehicle (IV). The communication risk, for each functionality, F, is calculated according to the following formula:

$$R_{CC} (F) = \sum_{j=1}^N w_j C_j$$

Where N is the number of communication links, w_j the weight of a specific communication channel type based on its propensity to vulnerability, and C_j is 1 if the functionality uses the channel; 0 otherwise.

$$\text{Max} (R_{CC} (F)) = \text{Total number of all communication channels}$$

5.3 Complexity Risk

This risk is associated with the number of defects in software used in automotive vehicles. The complexity metric in software is an excellent indicator of vulnerabilities. The Halstead Complexity measure is a standard way of deriving the complexity of software. Thus, for calculating the complexity of the functionalities in connected vehicle, we use the formula:

$$R_{SC} (F) = SLOC + \alpha (Nesting)$$

Where SLOC is the Source Line of Code, Nesting is the number of control structures, and α is the weight, with value over one, indicating complexity of the nesting structure.

$$\text{Max} (R_{SC} (F)) = SLOC + 10 (Nesting)$$

5.4 Input and Output Data Risk

This risk involves the input and output data in a connected vehicle. The metric distinguishes between a Fixed Input (FI) from a Fluctuating Input (LI). It also distinguishes an Insensitive Output (IO) from a Sensitive Output (SO). Weights (α , β) are added to highlight the significance of the Fluctuating Input and the Sensitive Output. To calculate the Input and Output Data Risk, we use the formula:

$$R_{DIO} (F) = FI + \alpha (LI) + IO + \beta (SO)$$

$$\text{Max} (R_{DIO} (F)) = FI + 5(LI) + IO + 5(SO)$$

5.5 History of Security Issues

This risk considers the past security issues of a certain vehicle functionality. Given Y as the total number of years since the first car attack and α_y as the number of attacks that occurred in year y. A forgetting factor, λ , is introduced to provide relevancy to the attacks that occurred in more recent years, where $0 \leq \lambda \leq 1$. To calculate the risk of a vehicle functionality using the history of security issues, we use the formula:

$$R_{HS} (F) = \sum_{y=1}^Y \alpha_y \lambda^{Y-y}$$

For a 2-year comparison, the calculation simply boils down to

$$\lambda = 1 - (\alpha_1 / \alpha_2) \quad \leftarrow \text{the forgetting factor}$$

$$R_{HS} (F) = \alpha_1 (\lambda)^{Y-1} + \alpha_2$$

$$\text{Max} (R_{HS} (F)) = \alpha_1 + \alpha_2$$

5.6 Overall Security Vulnerability Metric

The overall security vulnerability metric of a certain functionality in a connected vehicle is calculated by first normalizing the values of each of the metrics and applying a weighting factor

$(\alpha, \beta, \gamma, \delta, \phi)$, which indicates its significance to the overall scheme. The metrics are added to obtain the overall value, which is in direct correlation with the vulnerability level of the functionality. The formula is shown as follow:

$$OSV = \alpha \left[\frac{R_{EC}(F)}{\max(R_{EC}(F))} \right] + \beta \left[\frac{R_{CC}(F)}{\max(R_{CC}(F))} \right] + \gamma \left[\frac{R_{SC}(F)}{\max(R_{SC}(F))} \right] \\ + \delta \left[\frac{R_{DIO}(F)}{\max(R_{DIO}(F))} \right] + \phi \left[\frac{R_{HS}(F)}{\max(R_{HS}(F))} \right]$$

6. Vehicle Security Best Practices Assessment Metrics

The Vehicle Security Best Practices Assessment Metrics are designed based on the National Highway Traffic Safety Administration (NHTSA) Report DOT HS812 075 [13]. The report contains a review and analysis of cybersecurity best practices involving automotive vehicles.

The study utilizes the iterative Information Security Life Cycle divide into four phases and processes as shown in Figure 1.

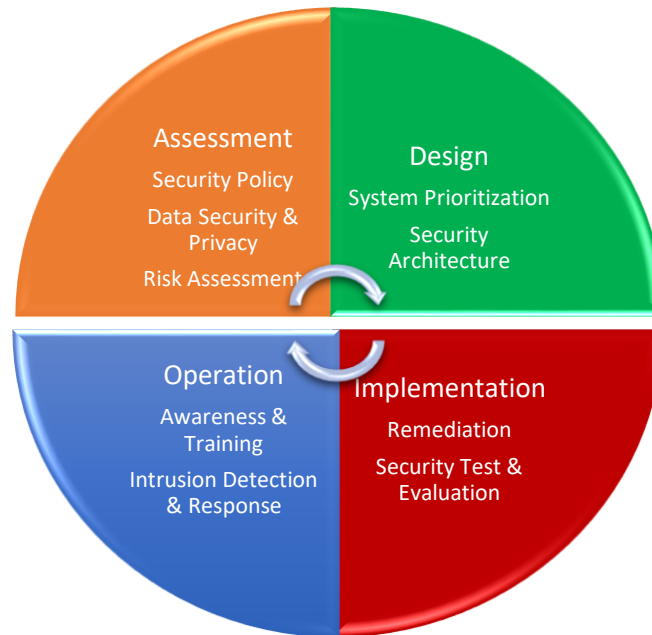


Figure 1. Information Security Life Cycle

The four phases and processes are described in the following.

Assessment Phase. This phase includes the development and implementation of security policies, the evaluation of system security, and the processes of risk assessment.

- *Establishing Security Policy.* A robust security policy must be developed, implemented, and strictly enforced. It also needs to be periodically reviewed and revised as needed.
- *Data Security and Privacy.* Data security must be examined and evaluated towards meeting standards, regulations, best practices, and organizational needs.
- *Risk Assessment.* Iterative risk assessment processes must be regularly conducted to assess and mitigate existing and emerging vulnerabilities.

Design Phase. This phase entails the prioritization of systems and resources applicable to security and design and analysis of the system’s security architecture.

- *System Protection and Prioritization.* In this process, the prioritization of resources is made to ensure that the identified risks are effectively and efficiently addressed.
- *Security Architecture.* The systems’ security architecture must be examined to complete the assessment phase and to initiate risk mitigation.

Implementation Phase. This phase covers the steps taken in vulnerability remediation and the processes in security testing and evaluation.

- *Remediation and Implementation.* In this process, vulnerability remediation must be implemented with robust security controls.
- *Security Test and Evaluation.* It is imperative that a robust conformance testing must be conducted to validate the security controls that are implemented. Further, a certification plan must also be developed in this stage.

Operation Phase. This phase includes the security awareness training for all personnel, customers, and other stakeholders. It also includes continuous security monitoring, intrusion detection and response.

- *Awareness and Security Training.* A periodic security awareness training must be conducted for all personnel and other stakeholders.
- *Intrusion Detection and Response.* In addition to continuous security monitoring, intrusion detection and response to identify successful exploitation of vulnerabilities and to effectively respond to such attack.

The Automotive Information Sharing and Analysis Center (Auto-ISAC) is another organization that is proactively working on Best Practices Guides to protect consumer safety through vehicle cybersecurity [58].

Vehicle Security Best Practices Assessment Metrics

We propose the following vehicle security best practice assessment metrics based on the Information Security Life Cycle described above. The metrics are built by a self-assessment form, shown on Table 3, which consists of a checklist of the status of each of the four phases. This form is converted into an interactive and tabular user interface as shown on Table 4.

Table 3. Vehicle Security Best Practices Assessment Checklist

Process	Checklists	Status
Security Policy	Are security policies established? Are security policies properly documented and widely disseminated within the organization?	

	<p>Are security policies strictly enforced?</p> <p>Are security policies periodically reviewed/updated?</p>	
Data Security & Privacy	<p>Is collected/stored data protected/encrypted?</p> <p>Is transmitted data encrypted?</p> <p>Is there a control mechanism for sharing data?</p> <p>Does the site comply with data protection standards and regulations (e.g., ISO/IEC 27001 certification, GDPR)?</p>	
Risk Assessment	<p>Do you conduct a periodic risk assessment of vehicle cybersecurity?</p> <p>Is there a developed and implemented organization-wide risk management strategy?</p> <p>Is there a Supply Chain Risk Management (SCRM) policy?</p> <p>Are security controls in place and periodically evaluated and/or enhanced?</p>	
System Protection & Prioritization	<p>Have you implemented security-by-design principles during the vehicle design phase?</p> <p>Have you implemented domain separation for in-vehicle networks (i.e. limiting the communication between the safety-critical and non-safety critical domains)?</p> <p>Does the organization triage the identified risks according to priority for resource allocation?</p> <p>Do you have a comprehensive system security test plan?</p>	
Security Architecture	<p>Have you implemented a layered approach to vehicle security (ECU level, in-vehicle network level, V2V level, V2X level)?</p> <p>Is there a periodic evaluation of the system's security architecture?</p> <p>Do you maintain an inventory of operational software components used in each automotive ECU and assembled vehicle?</p> <p>Have you considered the risks and vulnerabilities associated with vehicle sensor devices?</p>	
Remediation & Implementation	<p>Are there established mechanisms to update vehicle software and firmware remotely and securely?</p> <p>Are appropriate security controls implemented and are in place?</p> <p>Do you have an established remediation process?</p> <p>Is the remediation plan evaluated and implemented?</p>	
Security Test & Evaluation	<p>Have you conducted a thorough code review on the vehicle software?</p> <p>Have you conducted penetration testing on connected vehicle communication systems before deployment?</p> <p>Are security controls tested and evaluated for compliance with security performance specifications?</p> <p>Do you conform with secure software development best practices as outlined in NIST 8151 and ISO/SAE 21434?</p>	

Awareness & Security Training	<p>Is there a periodic security awareness training program for the entire workforce?</p> <p>Is security risk and mitigation disclosure available to the consumer and other stakeholders?</p> <p>Do you evaluate the effectiveness of the security awareness training program and introduce improvements if needed?</p> <p>Do you collect, maintain, analyze, and share information related to cybersecurity through the Automotive Information Sharing and Analysis Center (Auto-ISAC)?</p>	
Intrusion Detection & Response	<p>Is there an Incident Response Plan (IRP) in place?</p> <p>Is the IRP periodically tested, evaluated, and updated?</p> <p>Do you have a systematic process for continuous risk and security monitoring?</p> <p>Are security incidents properly documented and reported?</p>	

Table 4. Vehicle Security Best Practices Assessment User Interface

Phase	Process	Checklist	Response	
			Yes	No
Assessment	Security Policy	Security policies established	X	
		Security policies properly documented and widely disseminated within the organization?	X	
		Security policies strictly enforced	X	
		Security policies periodically reviewed/updated		X
	Data Security & Privacy	Collected/stored data protected/encrypted	X	
		Transmitted data encrypted	X	
		A control mechanism for sharing data	X	
		Compliant with data protection standards and regulations (e.g., ISO/IEC 27001 certification, GDPR)?	X	
	Risk Assessment	Conduct a periodic risk assessment of vehicle cybersecurity	X	
		Developed and implemented organization-wide risk management strategy	X	
A Supply Chain Risk Management (SCRM) policy		X		
Security controls in place and periodically evaluated and/or enhanced		X		
Design	System Protection and Prioritization	Implemented security-by-design principles during the vehicle design phase		X
		Implemented domain separation for in-vehicle networks (i.e. limiting the communication between the safety-critical and non-safety critical domains)		X
		Process to triage the identified risks according to priority for resource allocation	X	
		Comprehensive system security test plan		X
	Security Architecture	Implemented a layered approach to vehicle security (ECU level, in-vehicle network level, V2V level, V2X level)?	X	
		Periodic evaluation of the system's security architecture		X
		Maintain an inventory of operational software components used in each automotive ECU and assembled vehicle	X	
		Considered the risks and vulnerabilities associated with vehicle sensor devices	X	
Implementation	Remediation and Implementation	Established mechanisms to update vehicle software and firmware remotely and securely	X	
		Appropriate security controls implemented and are in place	X	
		Established a remediation process	X	
		Remediation plan evaluated and implemented	X	
	Security Test and Evaluation	Conducted a thorough code review on the vehicle software	X	
		Conducted penetration testing on connected vehicle communication systems before deployment	X	
	Security controls tested and evaluated for compliance with security performance specifications	X		
	Conformance with secure software development best practices as outlined in NIST 8151 and ISO/SAE 21434		X	
Operation	Awareness and Security Training	Periodic security awareness training program for the entire workforce	X	
		Security risk and mitigation disclosure available to the consumer and other stakeholders	X	
		Evaluate the effectiveness of the security awareness training program and introduce improvements if needed	X	
		Collect, maintain, analyze, and share information related to cybersecurity through the Automotive Information Sharing and Analysis Center (Auto-ISAC)		X
	Intrusion Detection and Response	An Incident Response Plan (IRP) in place	X	
		The IRP periodically tested, evaluated, and updated		X
		A systematic process for continuous risk and security monitoring		X
	Security incidents properly documented and reported		X	

V. CONNECTED VEHICLE SECURITY METRICS VISUALIZATION

Visualization takes advantage of cognitive perception in effectively presenting information to users. It offers a powerful means of recognizing trends and patterns that are not easily recognized using non-visual methods. In essence, the cognitive reasoning process is augmented by perception to bring about a more rapid analytical reasoning process [55]. There exist numerous works on information security visualization, e.g. [56], [57].

As an extension to this research, we ventured on vehicle security metrics visualization. We designed and are in the process of implementing a visualization system for each of the security

metrics that are described in the preceding sections. A detailed description of each of the visualization component is found in another manuscript, the Vehicle Security Metrics Visualization System Specification, Design, and Implementation Document. The visualization system prototypes are shown in the following sections.

1. Common Vulnerability Scoring System (CVSS)

The user interface, shown in Figure 2, provides data input controls to facilitate the assembly of the CVSS vector. After the CVSS vector has been created, the user clicks on the CVSS vector itself. The user is then taken to the CVSS calculator at the National Institute of Standards and Technology (NIST) website for the calculation and visualization. A sample visualization for the CVSS result is shown in Figure 3.

The screenshot displays a web-based interface for configuring a CVSS vector. At the top, the title 'CVSS vector' is centered. Below it, the word 'Results' is followed by the generated vector: **AV:N/AC:H/PR:H/UI:R/S:U/C:H/I:H/A:H**. The interface is organized into two main sections: 'Base Score Metrics' and 'Temporal Score Metrics'.
Base Score Metrics is further divided into two columns:
- **Exploitability Metrics**: Includes 'Attack Vector (AV)' with buttons for Network, Adjacent Network, Local, and Physical; 'Attack Complexity (AC)' with Low and High; 'Privileges Required (PR)' with None, Low, and High; and 'User Interaction (UI)' with None and Required.
- **Impact Metrics**: Includes 'Scope (s)' with Unchanged and Changed; 'Confidentiality Impact (c)' with None, Low, and High; 'Integrity Impact (i)' with None, Low, and High; and 'Availability Impact (A)' with None, Low, and High.
Temporal Score Metrics includes:
- 'Exploit Code Maturity (E)' with buttons for Not Defined, Unproven that exploit exists, Proof of concept code, Functional exploit exists, and Hight.
- 'Remediation Level (RL)' with buttons for Not Defined, Official fix, Temporary fix, Workaround, and Unavailable.
- 'Report Confidence (RC)' with buttons for Not Defined, Unknown, Reasonable, and Confirmed.

Figure 2. The Data Input for the CVSS Vector

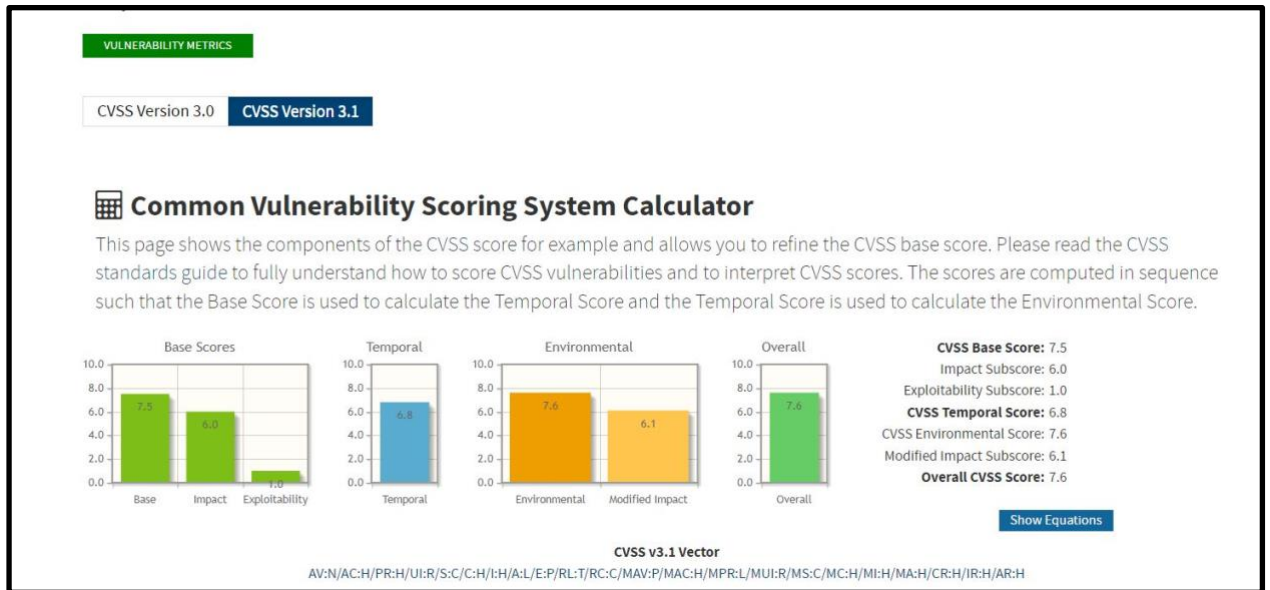


Figure 3. The CVSS Calculator Result and Visualization

2. Common Methodology for IT Security Evaluation (CEM)

The attack potential of threats on vehicle assets is derived from the CEM. The interactive data input interface is shown in Figure 4. A sample visualization for the attack potential metrics is shown in Figure 5.

Attack Potential of Threats on Vehicle Assets

Threat on Assets	Elapsed Time	Specialist Expertise	Knowledge of Target	Window of Opportunity	Equipment/Software	Total
False Data from ECU ⓘ	Immediate (20)	novice (15)	public knowledge (20)	unlimited (20)	minimal (20)	95
Blocking of CAN Bus ⓘ	one week (10)	proficient (10)	public knowledge (20)	unlimited (20)	minimal (20)	80
Malicious Software ⓘ	one month (5)	expert (5)	sensitive information (10)	difficult (5)	minimal (20)	45
Denial of Telematics Service ⓘ	one week (10)	expert (5)	critical information (5)	easy (15)	specialized(10)	45
Unauthorized Access ⓘ	one week (10)	proficient (10)	public knowledge (20)	easy (15)	specialized(10)	65
Command Injection ⓘ	one week (10)	expert (5)	public knowledge (20)	moderate (10)	highly specialized(5)	50
Masquerading ⓘ	one week (10)	multiple experts (1)	sensitive information (10)	difficult (5)	multi-specialized (1)	27
Data Tampering ⓘ	one year (1)	expert (5)	critical information (5)	moderate (10)	highly specialized(5)	26

[Visualize](#)

Figure 4. The Data Input Interface for the Attack Potential Threats on Vehicle Assets

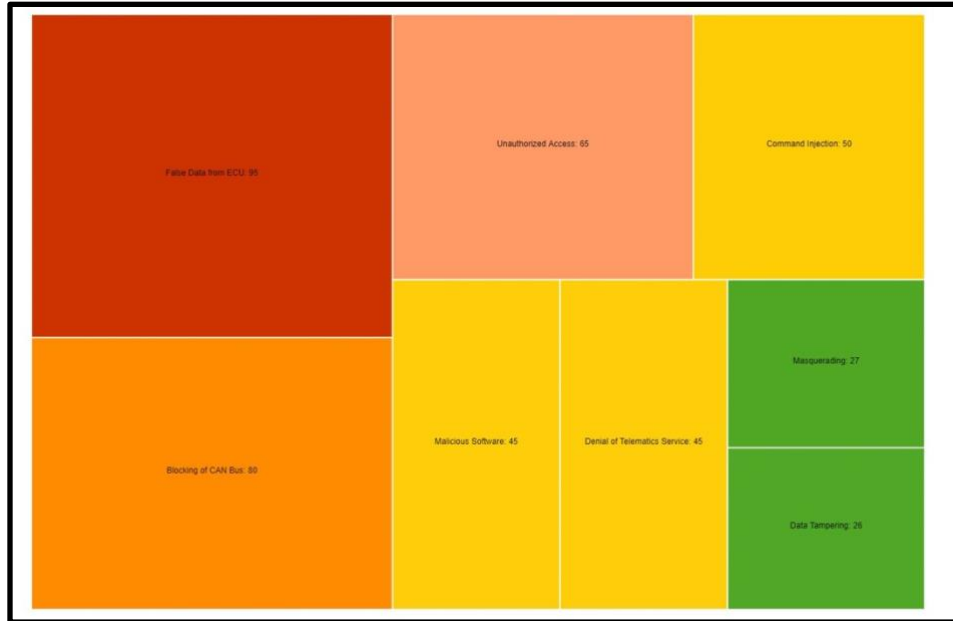


Figure 5. Attack Potential Threats on Vehicle Assets

3. Common Weakness Scoring System (CWSS)

Figure 6 depicts the CWSS user input interface. Calculations for the final CWSS score is done on a backend system. The visualization output of the CWSS components is depicted in Figure 7.

Common Weakness Scoring System (CWSS)

CWSS vector: [TI:1/AP:0.9/AL:0.7/IC:0.7/FC:0.8]
 CWSS score: 67.3 [Visualize](#)

Base Finding

- Technical Impact: Critical
- Acquired Privilege: Administrator
- Acquired Priv Layer: Application
- Int Control Effectiveness: None
- Finding Confidence: Proven True

Attack Surface

- Required Privilege: None
- Req Privilege Layer: Application
- Access Vector: Internet
- Authentication Strength: Strong
- Level Interaction: Automated
- Deployment Scope: All

Environmental

- Business Impact: Critical
- Likelihood Discovery: High
- Likelihood Exploit: High
- Ext Control Effect: None
- Prevalence: Widespread

Figure 6. The CWSS Data Input Interface

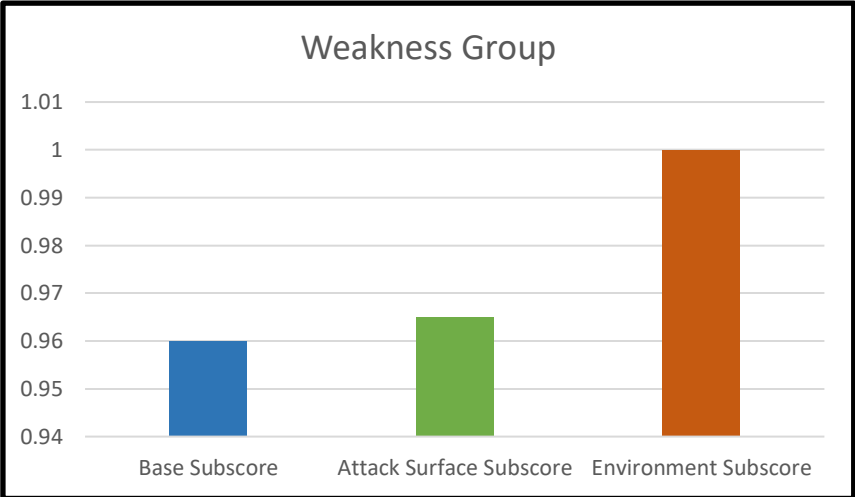
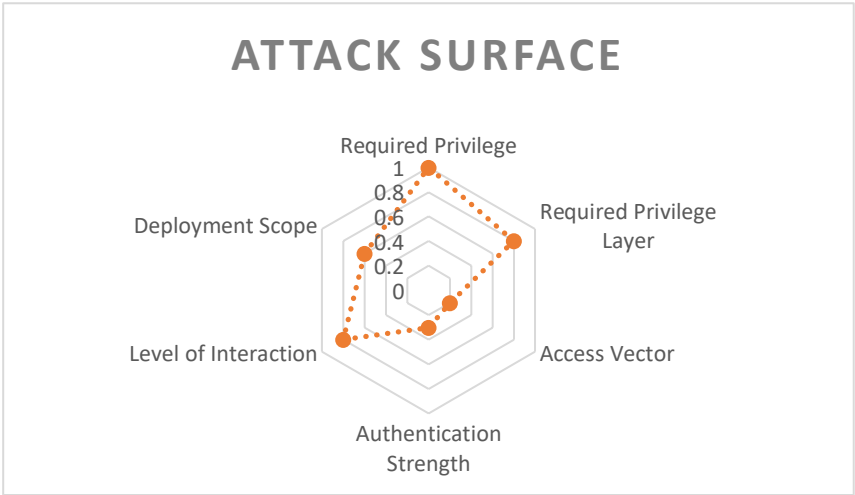
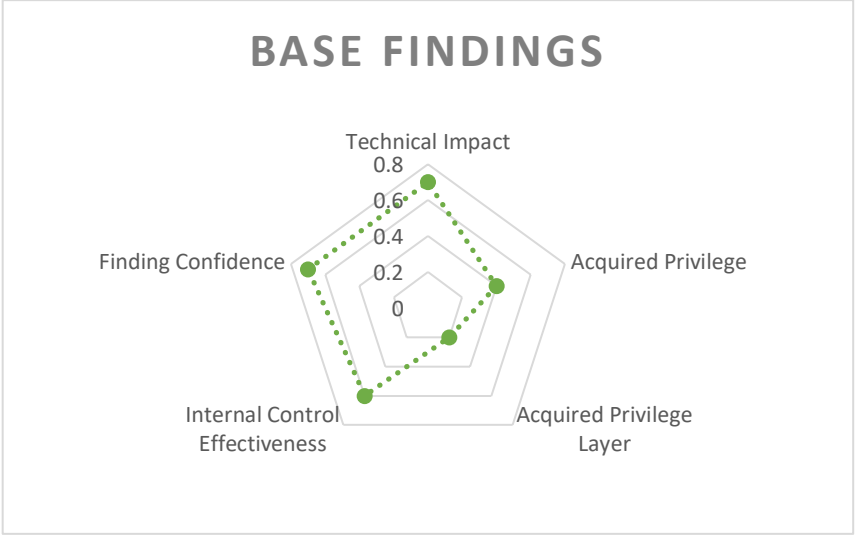
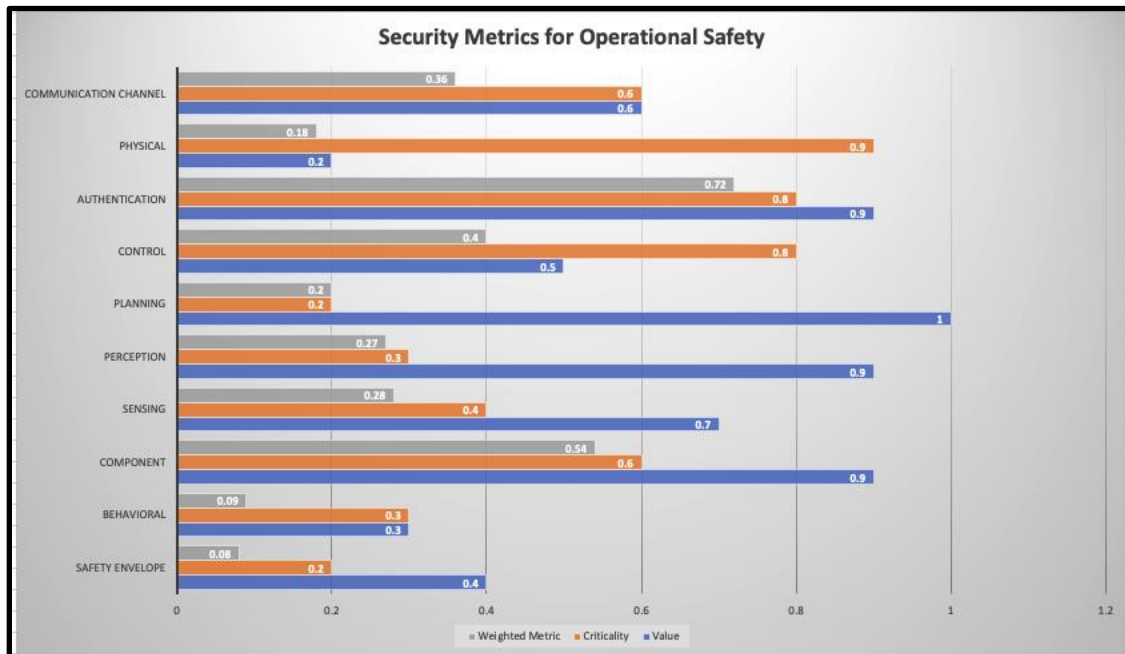


Figure 7. The CWSS Visualization

4. Operational Safety Assessment Metrics

The User Interface for the Security Metrics for the Operational Safety Assessment is depicted on Figure 8. The corresponding visualization for the input data is shown in Figure 9.

Figure 8. User Interface for the Security Metrics for Operational Safety



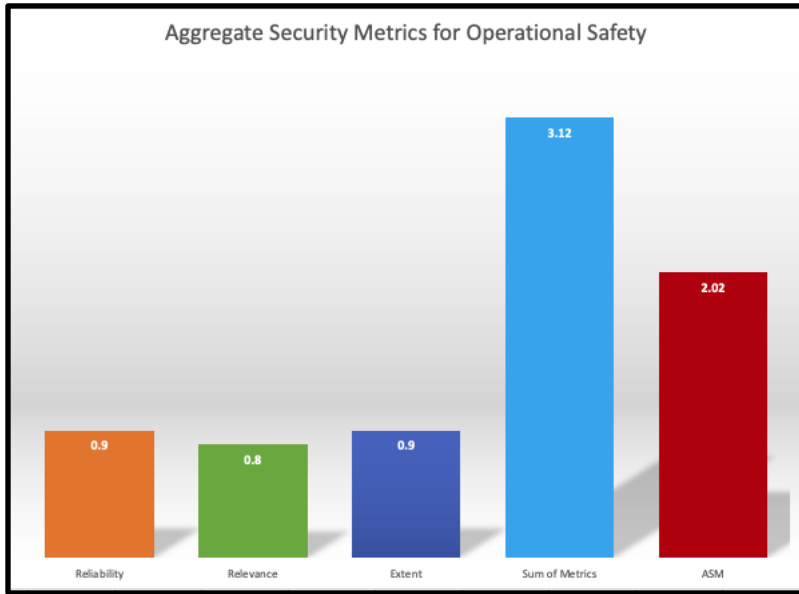


Figure 9. Visualization for the Security Metrics for Operational Safety Assessment

5. Security Vulnerability Metrics for Connected Vehicles

The Interactive Input interface for Security Vulnerability Metrics for Connected Vehicles is depicted on Figure 10. The corresponding visualization for the input data is shown in Figure 11.

Security Vulnerability Metrics for Connected Vehicles

Risk Category	Sub-Metric	Value	Weight
ECU Coupling Risk	Active ECU Links	0	1
	Total ECU Links	0	
	Complexity Risk		1
	SLOCs	0	
Communication Channel Risk	Active V2V Links	0	1
	Total V2V Links	0	
	V2V Weight	0.1	
	Active U2V Links	0	
History of Security Issue Risk	Number of Years Since First Attack	0	1
	Number of First Attack	0	
	Number of Second Attack	0	
	Forgetting Factor	0.2	
I/O Data Risk	Number of Fixed Input	0	1
	Number of Fluctuating Input	0	
	Number of Insensitive Output	0	
	Number of Sensitive Output	0	

Calculate Total Risk: 2.05

Buttons: Visualize, Reset

Figure 10. The User Interface for the Visualization of Security Vulnerability Metrics

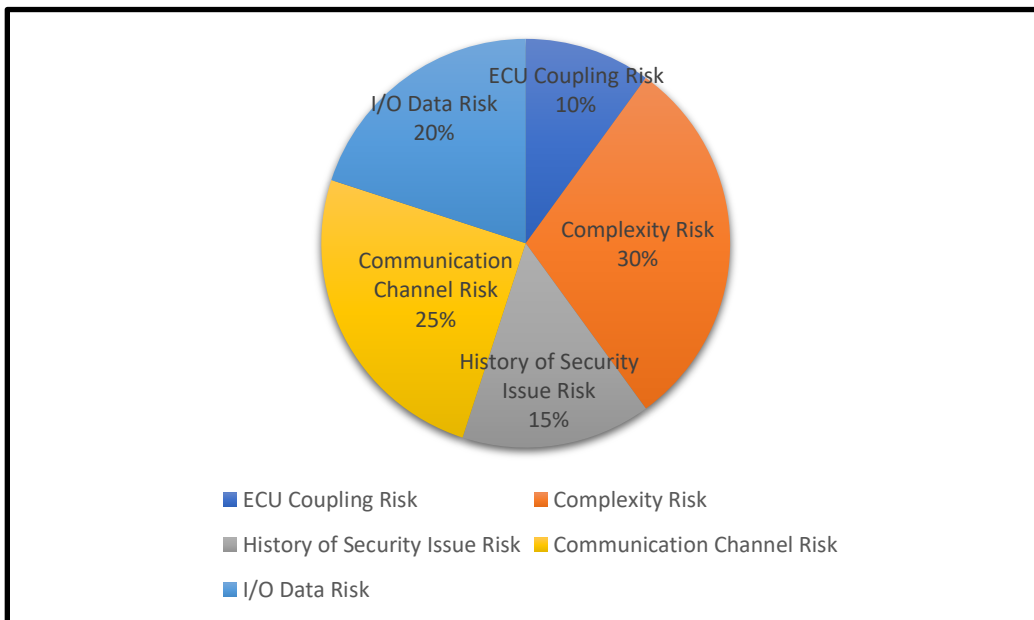
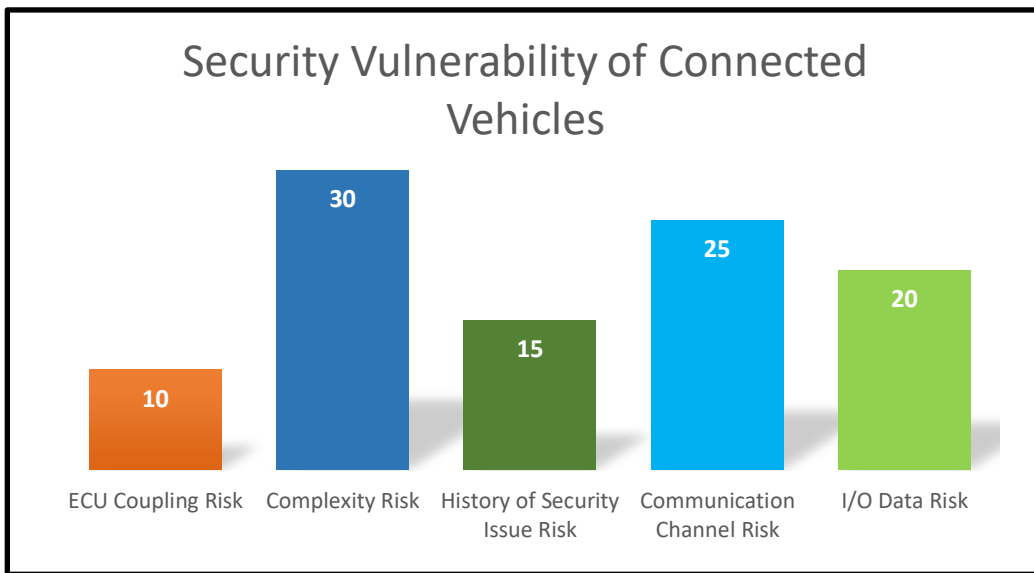


Figure 11. Visualization for the Security Metrics for Operational Safety Assessment

6. Vehicle Security Best Practices Assessment Metrics

The User Input interface for the Vehicle Security Best Practices Assessment Metrics for Connected Vehicles is depicted on Figure 12. The corresponding visualization for the input data is shown in Figure 13.

Phase	Process	Checklist	Response	
			Yes	No
Assessment	Security Policy	Security policies established	X	
		Security policies properly documented and widely disseminated within the organization?	X	
		Security policies strictly enforced	X	
		Security policies periodically reviewed/updated		X
	Data Security & Privacy	Collected/stored data protected/encrypted	X	
		Transmitted data encrypted	X	
		A control mechanism for sharing data	X	
		Compliant with data protection standards and regulations (e.g., ISO/IEC 27001 certification, GDPR)?	X	
	Risk Assessment	Conduct a periodic risk assessment of vehicle cybersecurity	X	
		Developed and implemented organization-wide risk management strategy	X	
A Supply Chain Risk Management (SCRM) policy		X		
Security controls in place and periodically evaluated and/or enhanced		X		
Design	System Protection and Prioritization	Implemented security-by-design principles during the vehicle design phase		X
		Implemented domain separation for in-vehicle networks (i.e. limiting the communication between the safety-critical and non-safety critical domains)		X
		Process to triage the identified risks according to priority for resource allocation	X	
		Comprehensive system security test plan		X
	Security Architecture	Implemented a layered approach to vehicle security (ECU level, in-vehicle network level, V2V level, V2X level)?	X	
		Periodic evaluation of the system's security architecture		X
		Maintain an inventory of operational software components used in each automotive ECU and assembled vehicle	X	
Implementation	Remediation and Implementation	Established mechanisms to update vehicle software and firmware remotely and securely	X	
		Appropriate security controls implemented and are in place	X	
		Established a remediation process	X	
		Remediation plan evaluated and implemented	X	
	Security Test and Evaluation	Conducted a thorough code review on the vehicle software	X	
		Conducted penetration testing on connected vehicle communication systems before deployment	X	
		Security controls tested and evaluated for compliance with security performance specifications	X	
		Conformance with secure software development best practices as outlined in NIST 8151 and ISO/SAE 21434		X
Operation	Awareness and Security Training	Periodic security awareness training program for the entire workforce	X	
		Security risk and mitigation disclosure available to the consumer and other stakeholders	X	
		Evaluate the effectiveness of the security awareness training program and introduce improvements if needed	X	
		Collect, maintain, analyze, and share information related to cybersecurity through the Automotive Information Sharing and Analysis Center (Auto-ISAC)		X
	Intrusion Detection and Response	An Incident Response Plan (IRP) in place	X	
		The IRP periodically tested, evaluated, and updated		X
		A systematic process for continuous risk and security monitoring		X
		Security incidents properly documented and reported		X

Figure 12. The User Interface for the Vehicle Security Best Practices Assessment Metrics

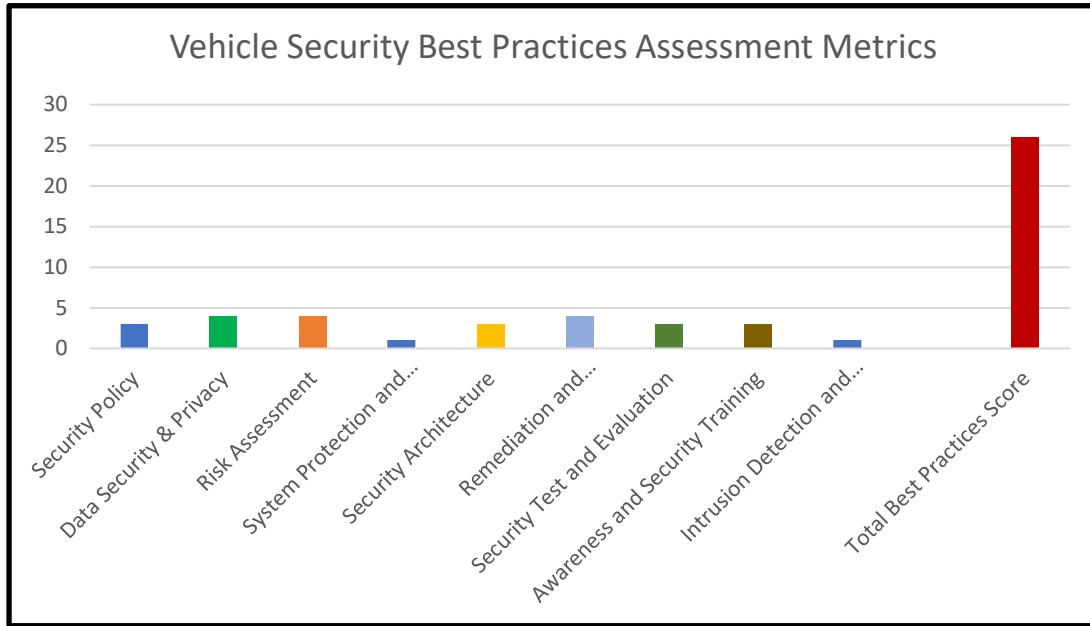


Figure 13. The Visualization for the Vehicle Security Best Practices Assessment Metrics

VI. CONCLUSION AND FUTURE RESEARCH DIRECTIONS

The rapid advancement of connected vehicle technology enabled the proliferation of newly found vulnerabilities in automotive vehicle systems. These vulnerabilities underscore the importance of paying close attention to the state of automotive vehicle security. Likewise, it is imperative that the security processes and tools be unceasingly improved and monitored. As a major component of continuous improvement, quantitative and qualitative measures must be devised to be able to make a full appreciation.

This white paper presents a comprehensive review of connected vehicle security threats, risks, and vulnerabilities. To accentuate the significance of continuous improvement process to connected vehicle security, we adapted and expanded widely recognized security metrics and derived novel security metrics to cover security best practices and emerging threats and vulnerabilities such as those found in sensors and communication links. Sample metric calculations and visualizations are illustrated to emphasize the significance of the security metrics.

With the preceding discussions in mind, we offer the following future research directions:

- development of Key Performance Indicators (KPI) for vehicle security metrics;
- elaboration of the visualization system for vehicle security metrics via the addition of analytics;
- extension of the defined metrics through the inclusion of quantifiable attack surfaces and threat likelihood;
- development of a unified automotive vehicle security metrics framework that incorporates both the CVSS framework and the Common Criteria for Information Security Evaluation; and
- the utilization of Machine Learning techniques to predict the status of automotive vehicle security based on known vulnerability attributes.

VII. ACKNOWLEDGEMENT

This research paper is partially funded by a grant from the Florida Department of Transportation (FDOT). It is intended to support FDOT's mission to provide a safe transportation system to ensure the mobility of people and goods. To illustrate its direct support to this mission, the impact of cybersecurity on operational safety is presented. Henceforth, cybersecurity metrics for operational safety are derived and developed (see Section IV.4.1).

The research presents a holistic treatment of the security of connected vehicles. It covers both the intranet (internal connectivity) and internet (external connectivity) systems of connected vehicles. The derived security metrics both implicitly and explicitly support the operational safety of connected vehicles—the primary concern of FDOT. We describe various risks found on connected vehicles that may eventually contribute to the likelihood of vulnerability exploitation. A successful attack on communication channels (V2V or V2X), ECU couplings, Input and Output Data, Supply Chain, or other security vulnerabilities could easily be leveraged to attack the entire connected vehicle ecosystem.

As a stretch objective, we conclude the research paper with vehicle security best practice assessment metrics. These security metrics provide a significant impact on the safety of connected vehicles. Auto-ISAC, in their Best Practices Guides, recognized the proactive collaboration of various organizations and the automotive industry in protecting consumer safety through a robust vehicle cybersecurity [58].

This white paper/report is intended for security practitioners, designers, manufacturers, technology providers, service providers, infrastructure owner-operators, and transportation agencies and regulators.

VIII. REFERENCES

- [1] Forum of Incident Response and Security Teams (FIRST), "Common Vulnerability Scoring System version 3.1: Specification Document," June 2019. [Online]. Available: <https://www.first.org/cvss/specification-document>. [Accessed 13 February 2020].
- [2] G. A. Francia, "Connected Vehicle Security," in *15th International Conference on Cyber Warfare and Security (ICWS 2020)*, Norfolk, VA, 2020.
- [3] NIST, "CSV-2019-13582 Detail," 15 November 2019. [Online]. Available: <https://nvd.nist.gov/vuln/detail/CVE-2019-13582>. [Accessed 20 February 2023].
- [4] Common Vulnerabilities and Exposure, "CVE-2018-9322," 31 May 2018. [Online]. Available: <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-9322>. [Accessed 13 February 2020].
- [5] Common Criteria Portal, "Common Criteria for Information Technology Security Evaluation," April 2017. [Online]. Available: <https://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R5.pdf>. [Accessed 24 February 2020].

- [6] MITRE Corporation, "Common Weakness Scoring System (CWSS)," 2 April 2018. [Online]. Available: https://cwe.mitre.org/cwss/cwss_v1.0.1.html.
- [7] Automated Vehicle Safety Consortium, "Best Practice for Metrics and Methods for Assessing Safety Performance of Automated Driving Systems (ADS)," SAE Industry Technologies Consortium, March 2021.
- [8] M. Elli, J. Wishart, S. Como, S. Dhakshinamoorthy and J. Weast, "Evaluation of Operational Safety Assessment (OSA) Metrics for Automated Vehicles in Simulation," SAE, 2021.
- [9] SAE, "Operational Safety Metrics for Verification and Validation (V&V) of Automated Driving Systems (ADS) J3237," SAE International, September 2020.
- [1] SAE, "Taxonomy and Definitions of ADS V&V J3208," SAE International, August 2019.
- [1] J. Wishart, Y. Chen, S. Como, N. Kidambi, D. Lu and Y. Yang, *Fundamentals of Connected and Automated Vehicles*, Warrendale, PA: SAE International, 2022.
- [1] L. Moukahal and M. Zulkernine, "Security Vulnerability Metrics for Connected Vehicles," in *2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, Sofia, Bulgaria, 2019.
- [1] C. McCarthy, K. Harnett and A. Carter, "A Summary of Cybersecurity Best Practices," US Department of Transportation (USDOT), Washington, D.C., 2014.
- [1] International Telecommunication Union (ITU), "Introduction to ASN.1," 2023. [Online]. Available: <https://www.itu.int/en/ITU-T/asn1/Pages/introduction.aspx>. [Accessed 17 May 2023].
- [1] SAE International, "DSRC Implementation Guide. A Guide to Users of SAE J2735 Message Sets over DSRC," SAE International, 2008.
- [1] SAE International, "On-Board System Requirements for V2V Safety Communications J2945/1_202004," 30 April 2020. [Online]. Available: https://www.sae.org/standards/content/j2945/1_202004. [Accessed 20 May 2023].
- [1] Lockheed Martin, "The Cyber Kill Chain," 2024. [Online]. Available: <https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html>. [Accessed 7 February 2024].
- [1] T. W. Edgar and D. O. Manz, "Chapter 7 - Theoretical Research," in *Research Methods for Cyber Security*, Syngress, 2017, pp. 177-192.
- [1] Formplus, "What is Applied Research? +[Types, Examples & Methods]," 22 July 2022. [Online]. Available: <https://www.formpl.us/blog/applied-research>.
- [2] International Organization for Standardization (ISO), "ISO/SAE 21434:2021 Road vehicles — Cybersecurity engineering," August 2021. [Online]. Available: <https://www.iso.org/standard/70918.html>.
- [2] Upstream Security Ltd., "ISO/SAE 21434: Setting the Standard for Automotive Cybersecurity," 2020. [Online]. Available: https://info.upstream.auto/hubfs/White_papers/Upstream_Security_Setting_the_Standard_for_Automotive_Cybersecurity_WP.pdf?_hsmi=87208721&_hsenc=p2ANqtz-8ke_6RWU7hkISDBzRoHFeUhfbaRRQ7E9-

Z2bvc4YMIP3JNvc42_oh1ZxJ5jtWQOUItehUaSmp7MfNDcwzbzUWoZjrGHw.
[Accessed 5 November 2020].

- [2] C. Schmittner, G. Griessnig and Z. Ma, "Status of the Development of ISO/SAE 21434," in 2] *Proc of the 25th European Conference, EuroSPI 2018*, Bilbao, Spain, 2018.
- [2] ISO/SAE, "ISO/SAE 21434:2021(en) Road vehicles--Cybersecurity engineering," 2021.
- 3] [Online]. Available: <https://www.iso.org/obp/ui/#iso:std:iso-sae:21434:ed-1:v1:en>.
- [2] Trend Micro, "A Vulnerability in Modern Automotive Standards and How We Exploited It," 4] July 2017. [Online]. Available: <https://documents.trendmicro.com/assets/A-Vulnerability-In-Modern-Automotive-Standards-and-How-We-Exploited-It.pdf>. [Accessed November 2018].
- [2] A. Karahasanovic, "Automotive Cyber Security," Chalmers University of Technology 5] University of Gothenburg, Gotehnborg, Sweden, 2016.
- [2] Qualcomm, Inc., "C-V2X: A new era of smart transporation in the United States," 17 6] February 2023. [Online]. Available: https://www.qualcomm.com/content/dam/qcomm-martech/dm-assets/documents/C-V2X_Whitepaper.pdf.
- [2] IEEE, "Vehicle to Grid (V2G) Technology," 18 February 2023. [Online]. Available: 7] <https://innovationatwork.ieee.org/vehicle-to-grid-v2g-technology/>.
- [2] Otonomo, "The Promise of Connected Vehicle Data," 16 February 2023. [Online]. 8] Available: <https://info.otonomo.io/hubfs/PDF/OOOO-Smart-Cities-Survey-Promise-of-data.pdf>.
- [2] J. Tyrrell, "Trends in ECU vulnerabilities highlighted at DEF CON 2020," 20 August 2020. 9] [Online]. Available: <https://www.securecav.com/trends-in-ecu-vulnerabilities-highlighted-at-def-con-2020/>. [Accessed February 2023].
- [3] D. Pauli, "Hackers Hijack Tesla Model S from Afar, While the Cars are Moving," 16 0] September 2016. [Online]. Available: https://www.theregister.co.uk/2016/09/20/tesla_model_s_hijacked_remotely/. [Accessed October 2019].
- [3] NIST-NVD, "National Vulnerability Database," 10 January 2023. [Online]. Available: 1] <https://nvd.nist.gov/vuln/detail/CVE-2022-38766>.
- [3] B. Toulas, "Hackers can unlock Honda cars remotely in Rolling-PWN attacks," 13 July 2] 2022. [Online]. Available: <https://www.bleepingcomputer.com/news/security/hackers-can-unlock-honda-cars-remotely-in-rolling-pwn-attacks/>. [Accessed February 2023].
- [3] Upstream Security, Inc., "Security researchers manage to control multiple vehicles from 3] various OEMs by exploiting an API based vulnerability in telematics service provider," November 2022. [Online]. Available: <https://upstream.auto/research/automotive-cybersecurity/?id=12360>.
- [3] L. Pan, X. Zheng, H. X. Chen, T. Luan, H. Bootwala and L. Batten, "Cyber security attacks 4] to modern vehicular systems," *J. Inf. Secur. Appl.*, vol. 36, pp. 90-100, October 2017.
- [3] S. Woo, H. J. Jo and D. H. Lee, "A practical wireless attack on the connected car and 5] security protocol for in-vehicle CAN," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 993-1006, April 2015.
- [3] M. J. Kang and J. W. Kang, "Intrusion detection system using deep neural network for in- 6] vehicle network security," *PLoS ONE*, vol. 11, no. 6, 2016.

- [3 O. Avatefipour, A. S. Al-Sumaiti, A. M. El-Sherbeeney, E. M. Awwad, A. Elmeligy, M. A. Mohamed and H. Malik, "An Intelligent Secured Framework for Cyberattack Detection in Electric Vehicles' CAN Bus Using Machine Learning," *IEEE Access*, vol. 7, 2019.
- [3 Q. Wang and S. Sawhney, "VeCure: A Practical Security Framework to Protect the CAN Bus of Vehicles," in *International Conference on the Internet of Things (IOT)*, Cambridge, MA, 2014.
- [3 K.-T. Cho and K. G. Shin, "Fingerprinting Electronic Control Units for Vehicle Intrusion Detection," in *Proceedings of the 25th USENIX Security Symposium (USENIX Security 16)*, 2016.
- [4 J. Petit, M. Feiri and F. Kargl, "Revisiting attacker model for smart vehicles," in *2014 IEEE 6th International Symposium on Wireless Vehicular Communications, WiVec 2014 Proceedings*, 2014.
- [4 J.-P. Monteuis, J. Petit, J. Zhang, H. Labiod, S. Mafrica and A. Serval, "Attacker Model for Connected and Automated Vehicles," in *ACM Computer Science in Cars Symposium (CSCS'18)*, Berlin, Germany, 2018.
- [4 M. Wolf and T. Gendrullis, "Design, Implementation, and Evaluation of a Vehicular Hardware Security Module," in *14th International Conference on Information Security and Cryptology*, Seoul, South Korea, 2011.
- [4 S. Lokman, T. Othman and M. Abu-Bakar, "Intrusion Detection System for Automotive Controller Area Network (CAN) Bus System: a Review," *EURASIP Journal on Wireless Communications and Networking*, vol. 184, 2019.
- [4 EVITA Project, "EVITA E-Safety Vehicle Intrusion Protected Applications," 01 December 2011. [Online]. Available: <https://www.evita-project.org/>. [Accessed 13 November 2018].
- [4 PRESERVE, "About the Project," June 2015. [Online]. Available: <https://preserve-project.eu/about>. [Accessed 12 October 2019].
- [4 SeVeCom, "Security on the Road," 2008. [Online]. Available: <https://www.sevecom.eu/>. [Accessed 13 October 2019].
- [4 Society of Automotive Engineers (SAE), "Cybersecurity Guidebook for Cyber-Physical Vehicle Systems J3061," 12 January 2012. [Online]. Available: <https://www.sae.org/standards/content/j3061/>. [Accessed 13 October 2019].
- [4 S. Bauer and P. Schartner, "Reducing Risk Potential by Evaluating Specialized Countermeasures for Electronic Control Units," in *17th escar Europe conference 2019*, Stuttgart, Germany, 2019.
- [4 Government Accountability Office (GAO), United States, "Vehicle Cybersecurity: DOT and Industry Have Efforts Under Way, but DOT Needs to Define Its Role in Responding to a Real-world Attack. GAO Report 16-350.," 2016. [Online]. Available: <https://www.gao.gov/assets/680/676064.pdf>. [Accessed 14 November 2018].
- [5 C. McCarty, K. Harnett and A. Carter, "A Summary of Cybersecurity Best Practices," 0 National Highway Traffic Safety Administration, Washington, DC, 2014.
- [5 C. McCarthy, K. Harnett and A. Carter, "Characterization of Potential Security Threats in Modern Automobiles: A Composite Modeling Approach.," October 2014. [Online]. Available: <https://rosap.ntl.bts.gov/view/dot/12119>. [Accessed 25 February 2020].

- [5 B. Sheehan, F. Murphy, M. Mullins and C. Ryan, "Connected and autonomous vehicles: A
2] cyber-risk classification framework," *Transportation Research Part A*, vol. 124, pp. 523-
536, 2019.
- [5 G. A. Francia and X. P. Francia, "Critical Infrastructure Protection and Security
3] Benchmarks," in *Encyclopedia of Information Science and Technology, 3rd Edition*,
Hershey, PA, IGI Global, 2015, pp. 4267-4278.
- [5 G. Francia, "Baseline Operational Security Metrics for Industrial Control Systems," in
4] *International Conference on Security and Management*, Las Vegas, NV, 2016.
- [5 G. A. Francia and S. Jarupathirun, "Security Metrics-Review and Research Directions," in
5] *Proceedings of the 2009 International Conference on Security and Management*, Las Vegas,
NV, 2009.
- [5 G. Conti, M. Ahamad and J. Stasko, "Attacking Information Visualization System Usability
6] Overloading and Deceiving the Human," in *SOUPS 2005*, Pittsburgh, PA, 2005.
- [5 H. Hochheiser and B. Schneiderman, "Using Interactive Visualizations of WWW Log Data
7] to Characterize Access Patterns and Inform Site Design," *Journal of the American Society
for Information Science and Technology*, vol. 52, no. 4, pp. 331-343, 2001.
- [5 Auto-ISAC, Inc., "Best Practices," 16 February 2023. [Online]. Available:
8] <https://automotiveisac.com/best-practices/>.
- [5 Upstream Security, Inc., "Recent spike in car thefts prompts South Korean OEMs to offer
9] security kits in the US," September 2022. [Online]. Available:
<https://upstream.auto/research/automotive-cybersecurity/?id=12080>.
- [6 G. A. Francia, III, "Vehicle Network Security Metrics," in *Advances in Cybersecurity
0] Management*, Cham, Switzerland, Springer Nature, 2021, pp. 55-73.
- [6 Fortinet, "What Is An Attack Surface?," 2023. [Online]. Available:
1] <https://www.fortinet.com/resources/cyberglossary/attack-surface>. [Accessed February 2023].
- [6 C. Maple, M. Bradbury, A. T. Le and K. Ghirardello, "A Connected and Autonomous
2] Vehicle Reference Architecture for Attack Surface Analysis," *Appl. Sci.*, vol. 9, no. 23,
2019.
- [6 K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B.
3] Kantor, D. Anderson, H. Shacham and S. Savage, "Experimental Security Analysis of a
Modern Automobile," in *2010 IEEE Symposium on Security and Privacy*,
Berkeley/Oakland, CA, 2010.
- [6 S. Checkoway, D. McCoy, B. Kantor, D. Anderson, N. Shacham, S. Savage, K. Koscher, A.
4] Czeskis, F. Roesner and T. Kohno, "Comprehensive experimental analyses of automotive
attack surfaces," in *20th USENIX Conference on Security (SEC'11)*, San Francisco, CA,
2011.
- [6 US Department of Transportation, "Cybersecurity Best Practices for the Safety of Modern
5] Vehicles," 9 September 2022. [Online]. Available:
<https://www.govinfo.gov/content/pkg/FR-2022-09-09/pdf/2022-19507.pdf>. [Accessed 18
February 2023].
- [6 Society of Automotive Engineers (SAE) International, "Hardware Protected Security for
6] Ground Vehicles," 10 February 2020. [Online]. Available:
https://www.sae.org/standards/content/j3101_202002/. [Accessed 12 November 2020].

- [6] British Standard Institution, "IATF 16949:2016 Automotive Quality Management," 2020.
- 7] [Online]. Available: <https://www.bsigroup.com/en-US/iatf-16949-automotive/introduction-to-iatf-16949/>. [Accessed 12 November 2020].
- [6] American National Standards Institute (ANSI), "ISO/IEC/IEEE 29119-1:2013," 2020.
- 8] [Online]. Available: https://webstore.ansi.org/Standards/ISO/ISOIECIEEEE291192013?gclid=CjwKCAiA17P9BRB2EiwAMvwNyKt4mT9KW0hN-taVxEzZBa7nN5sfZQzDV6HdWGRQddq5dVFT6Pv8LxoCQrEQAvD_BwE. [Accessed 12 November 2020].
- [6] S. Saydjari, "Is Risk a Good Security Metric?," in *Proceedings of the 2nd ACM Workshop*
- 9] *on Quality of Protection*, 2006.
- [7] S. Schechter, "Toward Econometric Models of Security Risk from Remote Attack," *IEEE*
- 0] *Security and Privacy*, pp. 40-44, January-February 2005.
- [7] P. Manadhata and J. Wing, "An Attack Surface Metric--CMU-CS-05-155," Carnegie Mellon
- 1] University, Pittsburgh, PA, 2005.
- [7] T. W. Moore, C. W. Probst, K. Rannenber and M. van Eeten, "Assessing ICT Security
- 2] Risks in Socio-Technical Systems," 13-18 November 2016. [Online]. Available: https://drops.dagstuhl.de/opus/volltexte/2017/7039/pdf/dagrep_v006_i011_p063_s16461.pdf.
- [7] D. Gollman, C. Herley, V. Koenig, W. Pieters and M. A. Sasse, "Socio-Technical Security
- 3] Metrics," *Dagstuhl Reports*, vol. 4, no. 12, pp. 1-28, 3 March 2015.
- [7] MITRE Corporation, "CVE-2022-37305," 01 August 2022. [Online]. Available:
- 4] <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-37305>. [Accessed February 2023].
- [7] G. A. Francia, III, D. Snider and B. Cyphers, "Basic Safety Message (BSM) Test Data
- 5] Generation for Vehicle Security Machine Learning Systems," in *Proc. of the 2023 International Conference on Security and Management (SAM'23)*, Las Vegas, NV, 2023.
- [7] R. W. van der Heijden, T. Lukaseder and F. Kargl, "VeReMi: A Dataset for Comparable
- 6] Evaluation of Misbehavior Detection in VANETs," in *Security and Privacy in Communication Networks*, New York, NY, Springer, 2018, pp. 318-337.
- [7] H. Song, J. Woo and H. K. Kim, "Car-Hacking Dataset," 2020. [Online]. Available:
- 7] <https://ocslab.hksecurity.net/Datasets/CAN-intrusion-dataset>. [Accessed 20 May 2023].
- [7] U.S. Department of Transportation (DOT), "Connected Vehicle Pilot (CVP) Open Data,"
- 8] [Online]. Available: <https://data.transportation.gov/stories/s/Connected-Vehicle-Pilot-Sandbox/hr8h-ufhq#cv-pilot-data-sandbox>. [Accessed 20 May 2023].
- [7] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan and T. Darrell,
- 9] "BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [8] Society of Automotive Engineers (SAE), "J2735 Surface Vehicle Standard V2X
- 0] Communications Message Set Dictionary," 2023. [Online]. Available: https://www.sae.org/standards/content/j2735_202309. [Accessed 15 October 2023].

- [8 SAE International, "On-Board System Requirements for V2V Safety Communications 1] J2945/1_202004," 30 April 2020. [Online]. Available: https://www.sae.org/standards/content/j2945/1_202004/. [Accessed 20 May 2023].
- [8 G. Francia III, "Technical Report UWF-TR-FDOT-002-01," University of West Florida 2] Center for Cybersecurity, Pensacola, FL , 2023.
- [8 G. Francia III, "Technical Report UWF-TR-FDOT-002-02," University of West Florida 3] Center for Cybersecurity, Pensacola, FL, 2023.
- [8 G. Francia III, "Technical Report UWF-TR-FDOT-001-01," University of West Florida 4] Center for Cybersecurity, Pensacola, FL, 2022.
- [8 G. Francia III, "Technical Report UWF-TR-FDOT-003-01," University of West Florida, 5] Pensacola, FL, 2023.
- [8 G. Francia III, "Technical Report UWF-TR-FDOT-005-01," University of West Florida 6] Center for Cybersecurity, Pensacola, FL, 2023.
- [8 G. Francia III, "Technical Report UWF-TR-FDOT-006-01," University of West Florida 7] Center for Cybersecurity, Pensacola, FL, 2022.
- [8 G. Francia III, "Technical Report UWF-TR-FDOT-006-02," University of West Florida 8] Center for Cybersecurity, Pensacola, FL, 2022.
- [8 G. Francia III, "Technical Report UWF-TR-FDOT-006-03," University of West Florida 9] Center for Cybersecurity, Pensacola, FL, 2022.
- [9 G. Francia III, "Technical Report UWF-TR-FDOT-007-01," University of West Florida 0] Center for Cybersecurity, Pensacola, FL, 2023.
- [9 G. Francia III, "Technical Report UWF-TR-FDOT-007-02," University of West Florida 1] Center for Cybersecurity, Pensacola, FL, 2023.
- [9 G. Francia III, "Technical Report UWF-TR-FDOT-007-03," University of West Florida 2] Center for Cybersecurity, Pensacola, FL, 2023.
- [9 G. Francia III, "Technical Report UWF-TR-FDOT-008-01," University of West Florida 3] Center for Cybersecurity, Pensacola, FL, 2023.
- [9 G. Francia III, "Technical Report UWF-TR-FDOT-008-02," University of West Florida 4] Center for Cybersecurity, Pensacola, FL, 2023.
- [9 G. Francia III, "Technical Report UWF-TR-FDOT-008-03," University of West Florida 5] Center for Cybersecurity, Pensacola, FL, 2023.
- [9 G. Francia III, "Technical Report UWF-TR-FDOT-009-01," University of West Florida, 6] Center for Cybersecurity, Pensacola, FL, 2024.
- [9 G. Francia III, "Technical Report UWF-TR-FDOT-009-02," University of West Florida 7] Center for Cybersecurity, Pensacola, FL, 2024.
- [9 G. Francia III, "Technical Report UWF-TR-FDOT-009-03," University of West Florida 8] Center for Cybersecurity, Pensacola, FL, 2024.
- [9 J. Patterson and D. King, "Vehicle Security Metrics Visualization System," 2023. [Online]. 9] Available: <https://github.com/UWF-CfC-FDOT/VSMVS>. [Accessed December 2023].
- [1 G. Francia III, "Technical Report UWF-TR-FDOT-003-02," University of West Florida 00] Center for Cybersecurity, Pensacola, FL, 2023.
-]]

- [1 G. Francia III, "Technical Report UWF-TR-FDOT-003-01," University of West Florida
01 Center for Cybersecurity, Pensacola, FL, 2023.
]
- [1 G. Francia III, "Technical Report UWF-TR-FDOT-010-01," University of West Florida
02 Center for Cybersecurity, Pensacola, FL, 2024.
]
- [1 G. A. Francia III, "Technical Report UWF-TR-FDOT-010-03 Continuous Improvement
03 Report--TR," University of West Florida, Pensacola, FL, 2024.
]
- [1 S. Payne, "A Guide to Security Metrics," SANS Institute, 19 June 2006. [Online]. Available:
04 <http://www.sans.org/readingroom/papers/5/55.pdf>.
]
- [1 K. Kark, P. Stamp, J. Penn, S. Bernhardt and A. Dill, "Defining An Effective Security
05 Metrics Program," 16 May 2007. [Online]. Available:
] <https://www.forrester.com/report/Defining+An+Effective+Security+Metrics+Program/-/E-RES42354#>. [Accessed February 2020].
- [1 SAE International, "CAN Specification 2.0: Protocol and Implementations," 01 August
06 1998. [Online]. Available: [https://www.sae.org/publications/technical-](https://www.sae.org/publications/technical-papers/content/921603/)
] [papers/content/921603/](https://www.sae.org/publications/technical-papers/content/921603/). [Accessed 13 October 2019].
- [1 Gemalto, "Securing Vehicle to Everything," 2018. [Online]. Available:
07 <https://www.gemalto.com/brochures-site/download-site/Documents/auto-V2X.pdf>.
] [Accessed 13 April 2020].
- [1 C. McCarthy, K. Harnett and A. Carter, "Characterization of potential security threats in
08 modern automobiles: A composite modeling approach," Washington, D.C., September,
] 2014.
- [1 National Institute of Standards and Technology, "CVE-2019-13582 Detail," 15 November
09 2019. [Online]. Available: <https://nvd.nist.gov/vuln/detail/CVE-2019-13582>. [Accessed 13
] February 2020].
- [1 MITRE Corporation, "CWE-787: Out-of-bounds Write," 20 August 2020. [Online].
10 Available: <http://cwe.mitre.org/data/definitions/787.html>.
]

Unit Test Overview

for

Vehicle Security Metrics Visualization System (VSMVS)

Technical Report UWF-TR-FDOT-003-01

Version 0.1 unapproved

Version 1.0 approved

Prepared by Davis King, Joshua Patterson

Approved by Dr. Guillermo A. Francia, III

The University of West Florida

Florida Department of Transportation

February 23, 2023

Table of Contents

1. Introduction	1
1.1 Purpose	1
1.2 Document Revisions Table	1
2. VSMVS Test Framework	1
2.1 Framework Description	1
3. VSMVS Unit Tests	1
3.1 TreeMapTests	1
3.1.1 calculateTotals_valid	1
3.2 CWSSTests	2
3.2.1 calculateBaseSubScore_valid	2
3.2.2 calculateAttackSubScore_valid	2
3.2.3 calculateEnvironmentalSubScore_valid	2
3.2.4 calculateTotalScore_valid	2
3.3 SMOSTests	2
3.3.1 calculateSumOfMetrics_valid	2
3.3.2 calculateASM_valid	3
3.4 SVMCVTests	3
3.4.1 calculateTotal_valid	3
3.4.2 calculateECUScore_valid	3
3.4.3 calculateCommunicationScore_valid	3
3.4.4 calculateComplexityRisk_valid	3
3.4.5 calculateIODataRisk_valid	4
3.4.6 calculateHistoryRisk_valid	4
3.5 VSBPAMTests	4
3.5.1 public void calculateTotalScores_valid	4

Introduction

Purpose

This document describes the unit test framework of the Vehicle Security Metrics Visualization System (VSMVS) as a subcomponent of the Connected Vehicle Security Metrics and Threat Intelligence Project. This document provides an overview of unit test framework and tests for both the VSMVS and the user interface.

Document Revisions Table

Revisor	Revision Date	Reason
King, Davis	February 23, 2023	Initial draft
Patterson, Joshua	February 23, 2023	Initial draft

VSMVS Test Framework

Framework Description

The VSMVS project is being developed in the language C# and the choice was made to use the standard MsTest framework for unit testing the page functions. MsTest is a native unit testing library that comes with Microsoft's Visual Studio.

VSMVS Unit Tests

TreeMapTests

The TreeMapTests.cs file contains unit test coverage of the /Pages/TreeMapModel.cshtml.cs class. Performs validation of the score calculation method for the Tree Map page.

calculateTotals_valid

Validates that the OnGetCalculateTotals method successfully calculates and returns the correct number.

Status: Passed

CWSSTests

The CWSSTests.cs file contains unit test coverage of the /Pages/CWSSModel.cshtml.cs class. Performs validation of the scoring calculations for the CWSS page.

calculateBaseSubScore_valid

Validates that the base subscore method successfully calculates the correct number and returns the result.

Status: Passed

calculateAttackSubScore_valid

Validates that the attack subscore method successfully calculates the correct number and returns the result.

Status: Passed

calculateEnvironmentalSubScore_valid

Validates that the environmental subscore method successfully calculates the correct number and returns the result.

Status: Passed

calculateTotalScore_valid

Validates that the total score method successfully calculates the correct number and returns the result.

Status: Passed

SMOSTests

The SMOSTests.cs file unit test coverage of the /Pages/SVMCVModel.cshtml.cs class. Performs validation of the score calculation methods for the SVMCV page.

calculateSumOfMetrics_valid

Verifies that the SumOfMetrics method successfully calculates and returns the resulting number.

Status: Passed

calculateASM_valid

Verifies that the calculateASM method successfully calculates and returns the resulting number.

Status: Passed

SVMCVTests

The SVMCVTests.cs file unit test coverage of the /Pages/SMOSModel.cshtml.cs class. Performs validation of the score calculation method for the SMOS page.

calculateTotal_valid

Verifies that the OnGetCalculateTotalRisk method successfully calculates and returns the resulting number.

Status: Passed

calculateECUScore_valid

Verifies that the OnGetCalculateECURisk method successfully calculates and returns the resulting number.

Status: Passed

calculateCommunicationScore_valid

Verifies that the OnGetCalculateCommunicationRisk method successfully calculates and returns the resulting number.

Status: Passed

calculateComplexityRisk_valid

Verifies that the OnGetCalculateComplexityRisk method successfully calculates and returns the resulting number.

Status: Passed

calculateIODataRisk_valid

Verifies that the OnGetCalculateIODataRisk method successfully calculates and returns the resulting number.

Status: Passed

calculateHistoryRisk_valid

Verifies that the OnGetCalculateHistoryRisk method successfully calculates and returns the resulting number.

Status: Passed

VSBPAMTests

VSBPAMTests.cs file unit test coverage of the /Pages/VSBPAMModel.cshtml.cs class. Performs validation of the total score calculation method for the VSPAM page.

public void calculateTotalScores_valid

Verifies that the total scores method successfully calculates and returns the resulting number.

Status: Passed

Appendix V. BSM Dataset Attributes

Table 4: BSM Coredata Characteristics

Data Item	Data Type	Description
Acceleration	System Defined (in units of 0.01 m/sec ²)	<p>acceleration_set_4_way :</p> <p>long accel: integer, (acceleration along the X-axis or the direction of travel; negative value indicates braking action)</p> <p>lat accel: integer, (acceleration along the Y-axis or the direction of travel; negative value indicates left turning action, positive indicates right turning)</p> <p>vert accel: one-byte signed integer, (-127 represents unavailable data)</p> <p>yaw: integer (vehicle rotation about the vertical axis and expressed in 0.01 degrees/second)</p>
Angle: Steering Wheel Angle (units of 1.5 degree)	Signed Integer (range: -189 to +189)	<p>0x01 = 00 = +1.5 degree</p> <p>0x81 = -126 = -189 degree and beyond</p> <p>0x7E = +126 = +189 degree and beyond</p> <p>0x7F = +127 to be used for unavailable</p>
Brake System Status: 2-octet information about the current brake system of the vehicle	System Defined	<p>brakeAppliedStatus; (4 bits total--one bit for each wheel, value 1 means active; Thus, 0000 means all Off, 0001 left front active, 0010 left rear active, 0100 right front active, 1000 right rear active)</p> <p>brakesUnavailableStatus: (5th bit; 1 means true)</p> <p>sparebit: 6th bit unused; set to 0</p> <p>traction: (7th and 8th bits) (Traction Control Status)</p> <p>00-unavailable)</p> <p>01-off</p> <p>10-on but not engaged</p> <p>11-engaged,</p> <p>abs: (9th and 10th bits) (Anti-lock Brake Status)</p> <p>00-unavailable; 01-off; 10-on but not engaged; 11-engaged.</p> <p>Stability Control Status, Brake Boost Applied, and Auxiliary/Emergency Brake Status omitted for brevity.</p>

Elevation (unit is 10 cm)	Integer	Elevations from 0 to 61439 (0x0000 to 0xEFFF) meters Elevations from -409.5 to -0.1 (0xF001 to 0xFFFF) meters Unknown value is encoded as 0xF000
Heading: Represents 0.0125 degrees from the North	2 octets of unsigned integer. Range 0 to 28800	Value of 28800 indicates unavailable.
Latitude: 32- bit value represents 1/10 microdegrees with reference to the horizontal datum	Integer (range -- 900000000 to 900000001	Provides a range of plus-minus 180 degrees 900000001 indicates unavailable
Longitude: 32-bit value represents 1/10 microdegrees with reference to the vertical datum	Integer (range -1800000000 to 1800000001	Provides a range of plus-minus 180 degrees 1800000001 indicates unavailable
Msg Count	Non-negative Integer	Message Count
Vehicle ID	Non-negative Integer	Vehicle Identifier
SecMark	Non-negative Integer	Units of milliseconds
Speed (in units of 0.02 m/sec. Range: 0 to 8191	Non-negative Integer (13 bits of the 2-byte Transmission+Speed)	Use 8191 to indicate unavailability.
Transmission	System Defined Occupies bits 14 to 16 of the 2-byte Transmission+Speed	000-Neutral 001-Park 010-Forward gear 011-Reverse gear 100, 101, and 110 are unused 111 unavailable

Vehicle Size (in cm)	System Defined 3 octets	Width : Non-negative Integer (10 unsigned bit with values [0,1023] (0 when unavailable) Length : Non-negative Integer (14 unsigned bit with values [0,16383] (0 when unavailable)
-------------------------	----------------------------	---

Vehicle Security Metrics Visualization System (VSMVS) User Manual

Technical Report UWF-TR-FDOT-003-02

Version 0.3
04/08/2023

Table of Contents

User Manual	149
1. Introduction	5
1.1 Overview	5
2. Getting Started	6
2.1 Set-up Considerations	15
2.2 System Organization & Navigation	15
2.3 Exiting the System	16
3. Using the System	17
3.1 Common Vulnerability Scoring System (CVSS) Calculator	17
3.1.1 Base Score Metrics	17
3.1.2 Temporal Score Metrics	17
3.1.3 Environmental Score Metrics	17
3.1.4 Visualize	17
3.2 Attack Potential of Threats on Vehicle Assets (Tree Map)	18
3.2.1 Threats on Assets	18
3.2.2 Factors for Analysis	19
3.2.3 Calculating Score	19
3.2.4 Visualize	19
3.2.5 Reset	20
3.3 Common Weakness Scoring System (CWSS) Calculator	20
3.3.1 Base Finding Score Metrics	21
3.3.2 Attack Surface Score Metrics	21
3.3.3 Environmental Score Metrics	21
3.3.4 Calculating Scores	21
3.3.5 Visualize	21
3.4 Security Vulnerability Metrics for Connected Vehicles (SVMCV)	21
3.4.1 ECU Coupling Risk	21
3.4.2 Complexity Risk	22
3.4.3 Communication Channel Risk	22
3.4.4 History of Security Issue Risk	22
3.4.5 I/O Data Risk	22
3.4.6 Calculating Scores	22
3.4.7 Visualize	24
3.4.8 Reset	24
3.5 Security Metrics for Operational Safety (SMOS)	24
3.5.1 Security Metrics	25
3.5.2 Calculating Score	25
3.5.3 Visualize	26
3.5.4 Reset	26
3.6 Vehicle Security Best Practice Assessment Metrics (VSBPAM)	27
3.6.1 Security Metrics	27
3.6.2 Calculating Score	28
4. Troubleshooting & Support	29
4.1 Error Messages	29

Table of Figures

Figure 1 - Home Page	2
Figure 2 - Common Vulnerability Scoring System Page	4
Figure 3 - Common Vulnerability Scoring System Page (Continued)	5
Figure 4 - Attack Potential of Threats on Vehicle Assets (TreeMap) Page	6
Figure 5 - Common Weakness Scoring System Page	7
Figure 6 - Common Weakness Scoring System Page (Continued)	8
Figure 7 - Security Vulnerability Metrics for Connected Vehicles Page	9
Figure 8 - Security Vulnerability Metrics for Connected Vehicles Page (Continued)	9
Figure 9 - Security Metrics for Operational Safety	10
Figure 10 - Vehicle Security Best Practices Assessment Metrics Page	11
Figure 11 - Common Vulnerability Scoring System Page	14
Figure 12 - Attack Potential of Threats on Vehicle Assets (TreeMap) Page	16
Figure 13 - ECU Coupling Risk Section	18
Figure 14 - Complexity Risk Section	18
Figure 15 - Communications Channel risk Section	19
Figure 16 - History of Security Issue Risk Section	19
Figure 17 - I/O Data Risk Section	20
Figure 18 - Calculate Total Risk Button	20
Figure 19 - Visualize Results Button	20
Figure 20 - Reset Inputs Button	20
Figure 21 - Safety Metrics for Operation Safety	22
Figure 22 - Calculate Total Risk Button	22
Figure 23 - Visualize Results Button	22
Figure 24 - Reset Inputs Button	23

Introduction

This User Manual (UM) provides the information necessary for the Florida Dept of Transportation (FDOT) to effectively use the Vehicle Security Metrics Visualization System (VSMVS). This document provides comprehensive instructions and guidelines for using the website, which includes web pages that pertain to various security metrics such as Common Vulnerability Scoring System (CVSS), Common Weakness Scoring System (CWSS), Attack Potential on Vehicle Assets, Security Vulnerability Metrics for Connected Vehicles (SVMCV), Security Metrics for Operational Safety (SMOS), and Vehicle Security Best Practices Assessment Metrics (VSBPAM). The Vehicle Security Metrics Visualization System website is identified by its name and is designed to provide users with a visual representation of the security metrics related to vehicle assets. The website has been developed by a team of software engineers with a focus on improving the security posture of vehicle assets. This User Manual has been developed to provide the intended audience, which includes users, administrators, and system analysts, with the necessary information to use the website effectively.

Overview

The Vehicle Security Metrics Visualization System (VSMVS) is a web application that allows users to interact with different UI to generate scores, graphs, or both. Depending on the page users are on, the web application will have a slightly different UI. Each UI is catered towards a specific functionality. The purpose of the VSMVS is to help users recognize trends and patterns that are not easily recognized using non-visual methods. The system will provide a visual depiction of security metrics that were developed in another undertaking by employing the benefits of visual perception.

Getting Started

The user will begin their experience on the homepage of the website. This page allows users to traverse through the web app comfortably and gives a brief overview of each application that the website offers. The home page consists of an introductory description of the website as well as six cards which contain descriptions and links to each of the main application pages on the website. Once a user clicks on the button within a container, they will be brought to that page where they can input data and visualize the various threats across various security metrics.

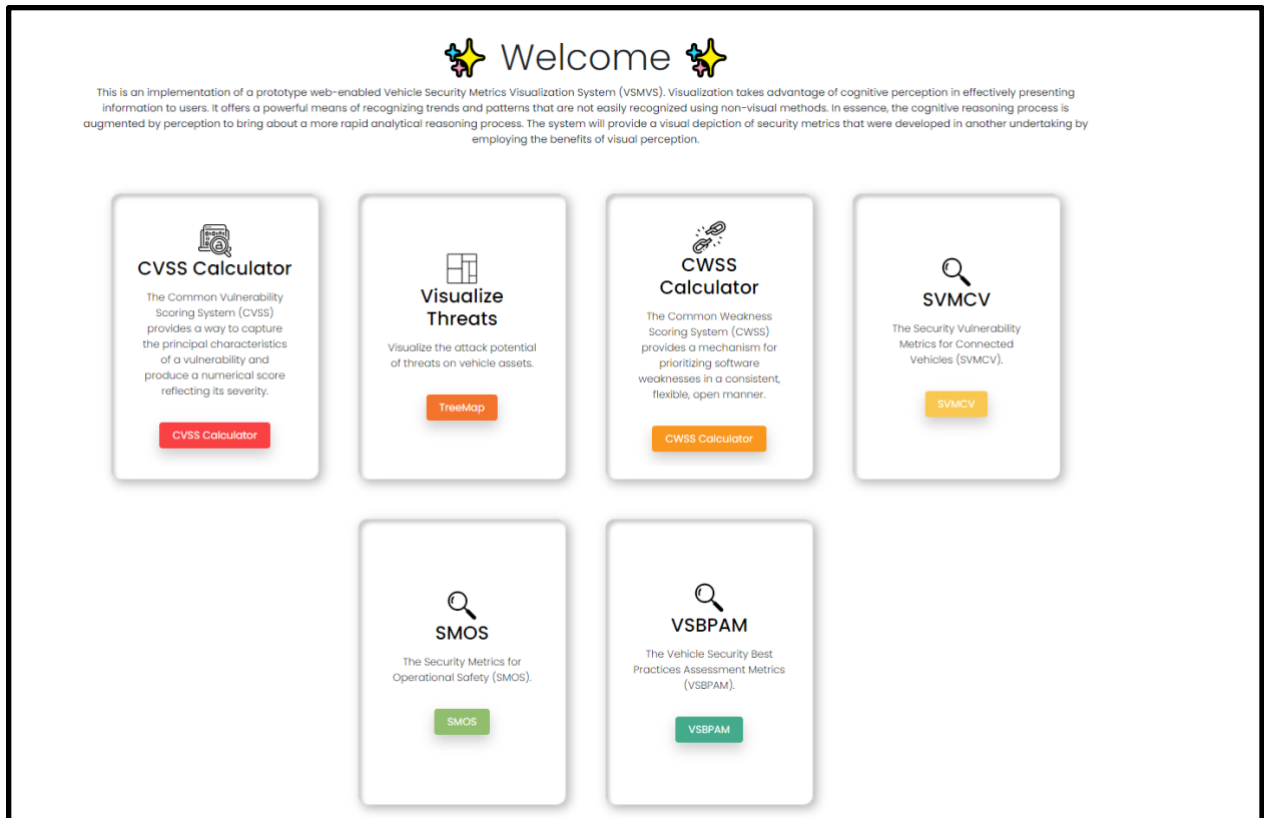


Figure 1 - Home Page

The first card takes the user to the CVSS Calculator page. This page implements all the functionality of a Common Vulnerability Scoring System calculator, which allows the user to generate a vector for the purpose of scoring vulnerability levels in various categories. Each metric has multiple buttons the user can choose to represent it. As the user selects an option for a metric the results card at the top of the page will update in real time with a CVSS vector representing the current choices selected by the user in each section.

Common Vulnerability Scoring System (CVSS)

Results

[Submit](#)

Base Score Metrics ⓘ

Exploitability Metrics ⓘ

Attack Vector (AV) ⓘ

[Network](#) [Adjacent Network](#) [Local](#) [Physical](#)

Attack Complexity (AC) ⓘ

[Low](#) [High](#)

Privileges Required (PR) ⓘ

[None](#) [Low](#) [High](#)

User Interaction (UI) ⓘ

[None](#) [Required](#)

Scope (S) ⓘ

[Unchanged](#) [Changed](#)

Impact Metrics ⓘ

Confidentiality Impact (C) ⓘ

[None](#) [Low](#) [High](#)

Integrity Impact (I) ⓘ

[None](#) [Low](#) [High](#)

Availability Impact (A) ⓘ

[None](#) [Low](#) [High](#)

Temporal Score Metrics ⓘ

Exploit Code Maturity (E) ⓘ

[Not Defined](#) [Unproven that exploit exists](#) [Proof of concept code](#) [Functional exploit exists](#) [High](#)

Remediation Level (RL) ⓘ

[Not Defined](#) [Official fix](#) [Temporary fix](#) [Workaround](#) [Unavailable](#)

Report Confidence (RC) ⓘ

[Not Defined](#) [Unknown](#) [Reasonable](#) [Confirmed](#)

Figure 2 - Common Vulnerability Scoring System Page

Environmental Score Metrics ⓘ

Exploitability Metrics ⓘ

Attack Vector (MAV) ⓘ

Attack Complexity (MAC) ⓘ

Privileges Required (MPR) ⓘ

User Interaction (MUI) ⓘ

Scope (MS) ⓘ

Impact Metrics ⓘ

Confidentiality Impact (MC) ⓘ

Integrity Impact (MI) ⓘ

Availability Impact (MA) ⓘ

Impact Subscore Modifiers ⓘ

Confidentiality Requirement (CR) ⓘ

Integrity Requirement (IR) ⓘ

Availability Requirement (AR) ⓘ

Source: <https://www.first.org/cvss/v3.1/specification-document>

Figure 3 - Common Vulnerability Scoring System Page (Continued)

The second card takes the user to the page called Attack Potential of Threats on Vehicle Assets, or the TreeMap page. This page will help users visualize the severity of different threats on vehicle assets. The page presents the user with a list of threats on vehicle assets, and a series of dropdown menus. The user can select values for each analysis factor on each threat. Clicking “Visualize” will cause the page to generate a TreeMap chart, displaying boxes for each threat selected, with distinct colors assigned based on the severity of the scores provided by the user.

Attack Potential of Threats on Vehicle Assets

Threats on Assets	Factors for Analysis				Total	
	Elapsed Time	Specialist Expertise	Knowledge of Target	Window of Opportunity		Equipment/Software
False Data from ECU	Immediate (20)	Laymen (20)	public knowledge (20)	unlimited (20)	specialized(10)	90
Blocking of CAN Bus	one day (15)	Laymen (20)	sensitive information (10)	moderate (10)	multi-specialized (1)	56
Malicious Software	one week (10)	proficient (10)	public knowledge (20)	easy (15)	highly specialized(5)	60
Denial of Telematics Service	one day (15)	novice (15)	critical information (5)	moderate (10)	multi-specialized (1)	46
Unauthorized Access	one month (5)	proficient (10)	sensitive information (10)	easy (15)	Select an option	40
Command Injection	one month (5)	expert (5)	sensitive information (10)	moderate (10)	standard (15)	45
Masquerading	one month (5)	expert (5)	critical information (5)	difficult (5)	multi-specialized (1)	21
Data Tampering	one month (5)	expert (5)	unknown (0)	difficult (5)	multi-specialized (1)	16

Visualize Reset

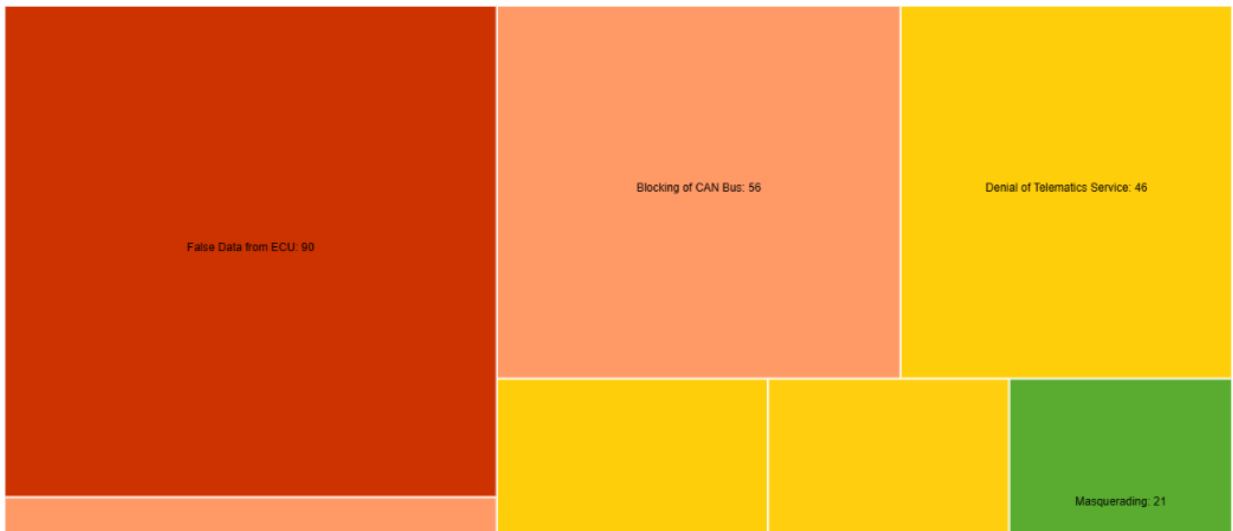


Figure 4 - Attack Potential of Threats on Vehicle Assets (TreeMap) Page

The third card takes the user to the page called Common Weakness Scoring System Calculator. This page implements the functionality of a Common Weakness Scoring System calculator, which generates a total score value, a vector representing that score, and the scores for each subcategory. The page also displays charts representing the data selected by the user in each category.

Common Weakness Scoring System (CWSS)

Results:

CWSS Vector: [Ti:C,1/AP:RU,0.7/AL:S,0.9/IC:L,0.9/FC:T,1/RP:RU,0.7/RL:E,1/AV:V,0.8/AS:W,0.9/IN:M,0.8/SC:R,0.5/BI:C,1/DI:H,1/EX:N,0/EC:I,0.5/P:C,0.8]

CWSS Score: 26.30

[Visualize](#)

Base Finding: 82.80

Technical Impact Critical	Acquired Privilege Regular User
Acquired Privilege Layer System	Internal Control Effectiveness Limited
Finding Confidence Proven True	

Attack Surface: 0.82

Required Privilege Regular User	Required Privilege Layer Enterprise Infrastructure
Access Vector Private Network	Authentication Strength Weak
Level of Interaction Moderate	Deployment Scope Rare

Environmental: 0.39

Business Impact Critical	Likelihood of Discovery High
Likelihood of Exploit None	External Control Effectiveness Indirect (Defense-in-Depth)
Prevalence Common	

Figure 5 - Common Weakness Scoring System Page

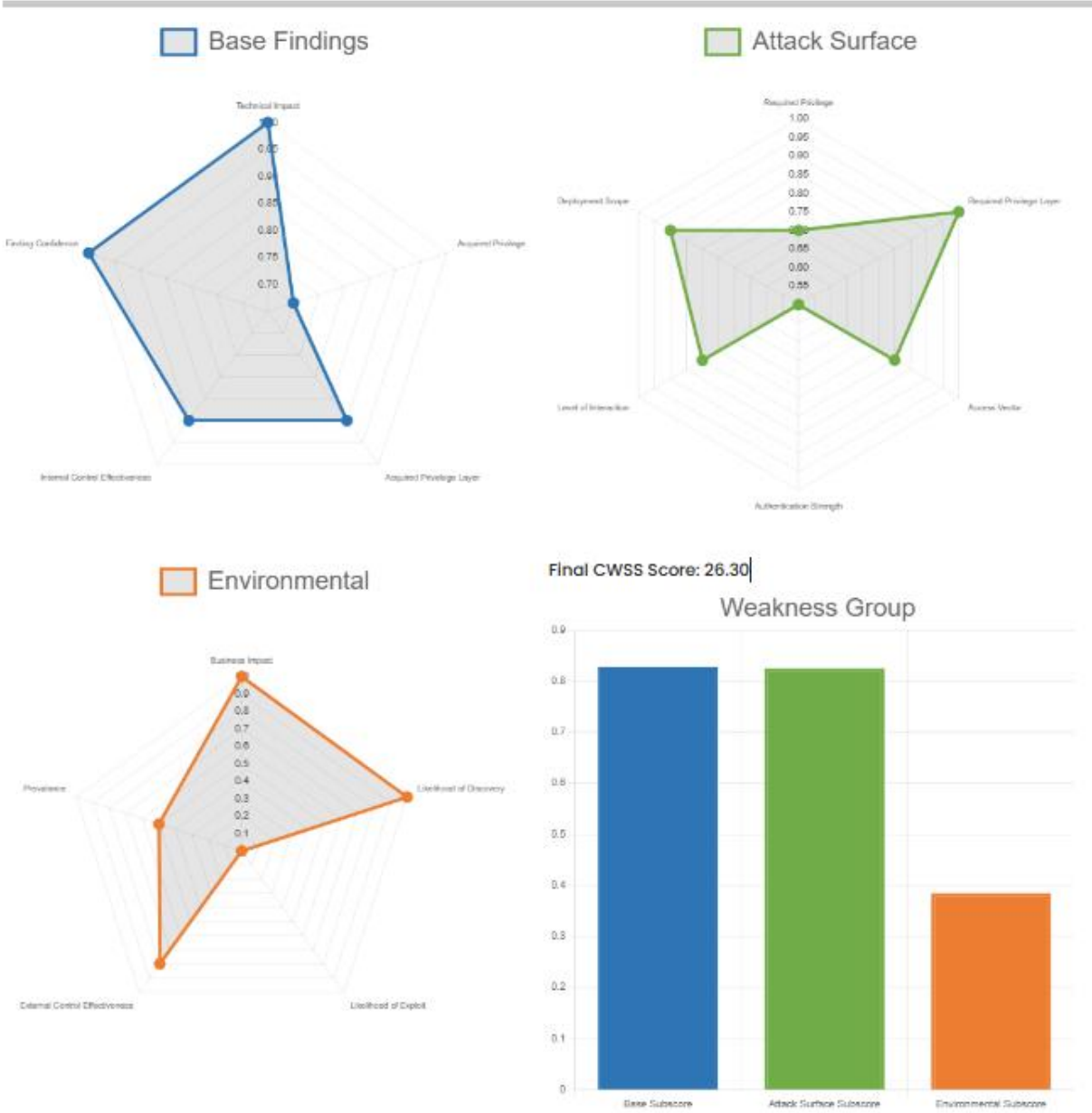


Figure 6 - Common Weakness Scoring System Page (Continued)

The fourth card takes the user to the page called Security Vulnerability Metrics for Connected Vehicles (SVMCV). This page provides a comprehensive analysis of the security vulnerabilities associated with connected vehicles.

Security Vulnerability Metrics for Connected Vehicles (SVMCV)

<p>ECU Coupling Risk Weight: 1</p> <p>Active ECU Links: <input type="text" value="3"/></p> <p>Total ECU Links: <input type="text" value="6"/></p> <p>Complexity Risk Weight: 3</p> <p>Source Lines of Code: <input type="text" value="60"/></p> <p>Number of Nestings: <input type="text" value="5"/></p> <p>Weight of Nestings: <input type="text" value="2"/></p>	<p>Communication Channel Risk Weight: 4</p> <p>Active V2V Links: <input type="text" value="3"/> Active V2I Links: <input type="text" value="4"/></p> <p>Total V2V Links: <input type="text" value="6"/> Total V2I Links: <input type="text" value="7"/></p> <p>V2V Weight: <input type="text" value="0.4"/> V2I Weight: <input type="text" value="0.2"/></p> <p>Active U2V Links: <input type="text" value="5"/> Active IV Links: <input type="text" value="2"/></p> <p>Total U2V Links: <input type="text" value="5"/> Total IV Links: <input type="text" value="8"/></p> <p>U2V Weight: <input type="text" value="0.6"/> IV Weight: <input type="text" value="0.4"/></p>
<p>History of Security Issue Risk Weight: 1</p> <p>Number of Years Since First Attack: <input type="text" value="2"/></p> <p>Number of First Attack: <input type="text" value="2"/></p> <p>Number of Second Attack: <input type="text" value="4"/></p> <p>Forgetting Factor: 0.5</p>	<p>I/O Data Risk Weight: 6</p> <p>Number of Fixed Input: <input type="text" value="4"/></p> <p>Number of Fluctuating Input: <input type="text" value="8"/> FI Weight: <input type="text" value="2"/></p> <p>Number of Insensitive Output: <input type="text" value="5"/></p> <p>Number of Sensitive Output: <input type="text" value="8"/> SO Weight: <input type="text" value="3"/></p>

7.438103

Figure 7 - Security Vulnerability Metrics for Connected Vehicles Page

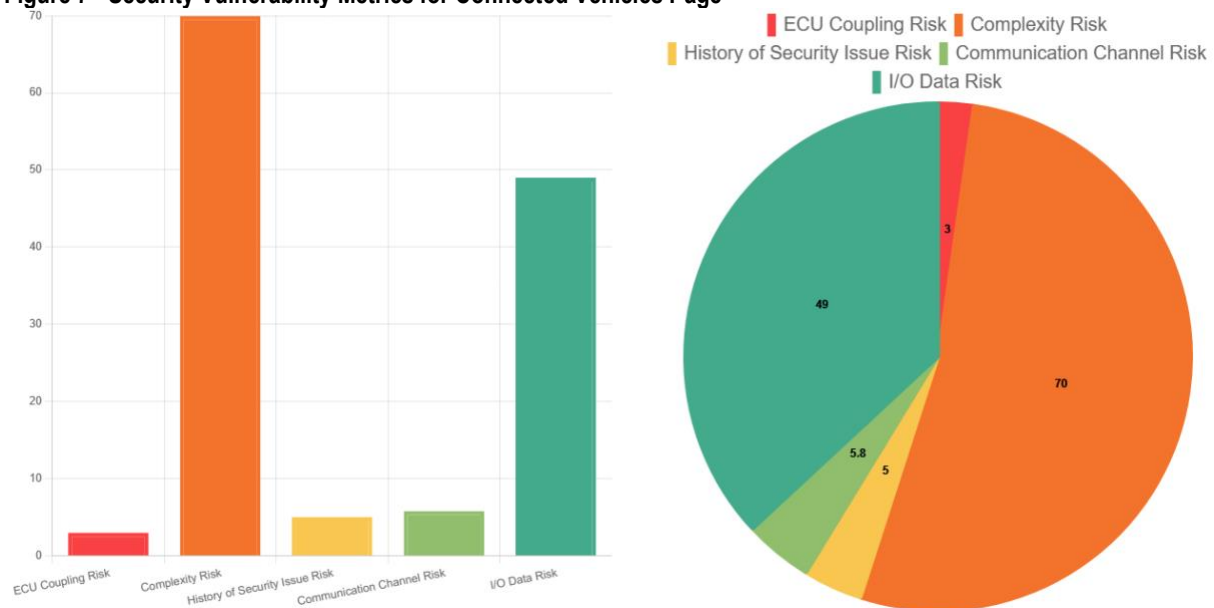


Figure 8 - Security Vulnerability Metrics for Connected Vehicles Page (Continued)

The fifth card takes the user to the page called Security Metrics for Operational Safety (SMOS). This page in the vehicle security metrics visualization system is designed to help users assess the operational safety risks of their vehicles. This page contains thirteen different metrics, with the first ten requiring users to enter two values for each metric: "Value" and "Criticality." The page also includes three buttons: "Calculate Total Risk," "Visualize Risk," and "Reset All Input."

Security Metrics for Operational Safety (SMOS)

Safety Envelope ⓘ Value <input type="text" value="0.2"/> Criticality <input type="text" value="0.3"/>	Behavioral ⓘ Value <input type="text" value="0.3"/> Criticality <input type="text" value="0.4"/>	Component ⓘ Value <input type="text" value="0.3"/> Criticality <input type="text" value="0.3"/>	Sensing ⓘ Value <input type="text" value="0.3"/> Criticality <input type="text" value="0.3"/>	Perception ⓘ Value <input type="text" value="0.2"/> Criticality <input type="text" value="0.4"/>
Planning ⓘ Value <input type="text" value="0.4"/> Criticality <input type="text" value="0.6"/>	Control ⓘ Value <input type="text" value="0.3"/> Criticality <input type="text" value="0.9"/>	Authentication ⓘ Value <input type="text" value="0.2"/> Criticality <input type="text" value="0.8"/>	Physical ⓘ Value <input type="text" value="0.4"/> Criticality <input type="text" value="0.7"/>	Communication ⓘ Value <input type="text" value="0.3"/> Criticality <input type="text" value="0.9"/>

Evaluation Factors

Reliability <input type="text" value="0.9"/>	Revelance <input type="text" value="0.9"/>	Extent <input type="text" value="0.9"/>
--	--	---

Calculate Total Risk 1.2101399
Visualize
Reset

Figure 9 - Security Metrics for Operational Safety

The sixth card takes the user to the page called Vehicle Security Best Practices Assessment Metrics (VSBPAM). The VSBPAM page is designed to help the user assess the security of a vehicle using a series of questions organized into four phases: Assessment, Design, Implementation, and Operation. Each phase contains several processes, and each process includes a series of yes-or-no questions. The responses to these questions are tallied up in real-time to provide the user with a visual representation of the security level of the vehicle.

Vehicle Security Best Practices Assessment Metrics (VSBPAM)				
Phase	Process	Checklist	Response	
			Yes	No
Assessment	Security Policy ⓘ	Are security policies established?	<input type="radio"/>	<input type="radio"/>
		Are security policies properly documented and widely disseminated within the organization?	<input type="radio"/>	<input type="radio"/>
		Are security policies strictly enforced?	<input type="radio"/>	<input type="radio"/>
		Are security policies periodically reviewed/updated?	<input type="radio"/>	<input type="radio"/>
	Data Security & Privacy ⓘ	Is collected/stored data protected/encrypted?	<input type="radio"/>	<input type="radio"/>
		Is transmitted data encrypted?	<input type="radio"/>	<input type="radio"/>
		Is there a control mechanism for sharing data?	<input type="radio"/>	<input type="radio"/>
		Does the site comply with data protection standards and regulations (e.g., ISO/IEC 27001 certification, GDPR)?	<input type="radio"/>	<input type="radio"/>
	Risk Assessment ⓘ	Do you conduct a periodic risk assessment of vehicle cybersecurity?	<input type="radio"/>	<input type="radio"/>
		Is there a developed and implemented organization-wide risk management strategy?	<input type="radio"/>	<input type="radio"/>
		Is there a Supply Chain Risk Management (SCRM) policy?	<input type="radio"/>	<input type="radio"/>
		Are security controls in place and periodically evaluated and/or enhanced?	<input type="radio"/>	<input type="radio"/>

Figure 10 - Vehicle Security Best Practices Assessment Metrics Page

Set-up Considerations

The Vehicle Security Metrics Visualizations System Website is a web-based application that can be accessed through a standard desktop computer, or a mobile device connected to the internet.

The system has been developed using modern web technologies and can be accessed through popular web browsers such as Google Chrome, Mozilla Firefox, and Microsoft Edge.

The input device for the system is a standard keyboard and mouse for desktop computers, or touch input for mobile devices. The output device is a computer screen or mobile device display. The system does not require any additional hardware or specialized equipment.

To set up the system, the user needs to ensure that they have a reliable internet connection and a modern web browser installed on their device. The system does not require any installation or configuration on the user's device.

The only set up consideration is ensuring that the device being used to access the system is up to date with the latest security patches and updates. Additionally, users should be aware of their own organizational security policies and practices and ensure they are adhering to them when using the system.

System Organization and Navigation

The Vehicle Security Metrics Visualization System website is organized into several web pages, each of which pertains to a different security metric. The following features are available:

- CVSS (Common Vulnerability Scoring System)
- CWSS (Common Weakness Scoring System)
- Attack Potential on Vehicle Assets
- SVMCV (Security Vulnerability Metrics for Connected Vehicles)
- SMOS (Security Metrics for Operational Safety)
- VSBPAM (Vehicle Security Best Practices and Attack Mitigation)

Each metric can be found on its own page that is listed in a navigation bar located at the top of each page. Additionally, the main menu on the home page features cards/containers with links to each of the metric pages.

There is no specific order in which users must navigate the website. However, we recommend that users start with the home page, which provides an overview of the system and access to all available metrics. From there, users can navigate to the specific metric pages to view the relevant data.

Each metric page is organized in a similar fashion, with a brief description of the metric and a visualization of the data. Users can interact with visualizations to filter and explore the data in more detail.

We encourage users to explore the different metrics and experiment with visualizations tools to gain a deeper understanding of vehicle security.

Exiting the System

When you are finished using the Vehicle Security Metrics Visualizations System Website, you should take the following actions to properly exit the system:

1. Close the browser: close the browser window or tab to ensure that no one else can access the website through your computer.

Using the System

The following sub-sections provide detailed, step-by-step instructions on how to use the various functions or features of the VSMVS.

Common Vulnerability Scoring System (CVSS) Calculator

The CVSS calculator implements the formula defined in the CVSS version 3.1 standard, generating scores based on the metric values you enter. Clicking the information icon for the metric group names, and metric names displays a summary of the information in the standard.

Base Score Metrics

The Base metric group represents the intrinsic characteristics of a vulnerability that are constant over time and across user environments. There are two sub-groups within the base metric group. The Exploitability sub-score, and the Impact sub-score. The Exploitability sub-score equation is derived from the Base Exploitability metrics, while the Impact sub-score equation is derived from the Base Impact metrics. The exploitability metrics contain Attack Vector, Attack Complexity, Privileges Required, User Interaction, and Scope, while the Impact metrics contain the Confidentiality Impact, Integrity Impact, and Availability Impact.

Temporal Score Metrics

The next group is the Temporal Score Metrics. Temporal metrics measure the current state of exploit techniques or code availability, the existence of any patches or workarounds, or the

confidence in the description of a vulnerability. The temporal metrics group contains three measured metrics: Exploit Code Maturity, Remediation Level, and Report Confidence.

Environmental Score Metrics

The final group of metrics is the Environmental Score Metrics. These metrics enable the analyst to customize the CVSS score depending on the importance of the affected IT asset to a user's organization, measured in terms of complementary/alternative security controls in place, Confidentiality, Integrity, and Availability. The metrics are the modified equivalent of Base metrics and are assigned values based on the component placement within organizational infrastructure. These metrics include the modified exploitability metrics, modified impact metrics, and the impact sub score modifiers.

Visualize

To generate a Common Vulnerability Scoring System vector, a user would select the choices that best apply for the vulnerability that they would like to score. Pressing the submit button will take the user to the NIST online CVSS calculator with the generated vector applied to the calculator. This will automatically generate graphs and score values which the user can view based on their choices for each metric. (NIST Online CVSS Calculator: <https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator>)

Common Vulnerability Scoring System (CVSS)

Results

AV:N/AC:H/PR:H/UI:R/S:C/C:L/I:L/A:L/E:U/RL:T/RC:C/MAV:A/MAC:L/MPR:L/MUI:N
/X/MC:X/MI:N/MA:L/CR:L/IR:X/AR:L

Submit

Base Score Metrics ⓘ

Exploitability Metrics ⓘ

Attack Vector (AV) ⓘ

Network Adjacent Network Local Physical

Attack Complexity (AC) ⓘ

Low High

Privileges Required (PR) ⓘ

None Low High

User Interaction (UI) ⓘ

None Required

Scope (S) ⓘ

Impact Metrics ⓘ

Confidentiality Impact (C) ⓘ

None Low High

Integrity Impact (I) ⓘ

None Low High

Availability Impact (A) ⓘ

None Low High

Figure 11 - Common Vulnerability Scoring System Page

Attack Potential of Threats on Vehicle Assets (Tree Map)

The Attack Potential of Threats on Vehicle Assets allows users to visualize various threats on vehicles. The scores for each threat are affected by several factors. Clicking the information icon for the threat names and factor names displays a summary of the information about each.

Threats on Assets

The following information describes each threat for which a score is generated in this page:

1. **False Data from ECU:** This type of attack occurs when a vehicle device or Electronic Control Unit (ECU), connected to the CAN bus, sends false information to other ECUs to induce them to behave abnormally.
2. **Blocking of CAN Bus:** The CAN protocol has an inherent vulnerability that can easily be exploited. The Arbitration ID field determines who has preference in using the bus. The node that has lowest Arbitration ID field value gets preference over the other nodes. An attack can be realized by the continuous transmission of CAN messages having very low Arbitration ID field values, which essentially blocks the CAN bus to other traffic with higher Arbitration ID.

3. **Malicious Software:** This is a common attack that is enabled by malicious software originating from the supply chain, the vehicle system patches, the telematics channel, or the On-Board Device (OBD) port.
4. **Denial of Telematics Service:** This is a special case of the denial-of-service attack in which the telematics communication channel is overwhelmed by excessive unwanted traffic to prevent the legitimate transfer of information to materialize.
5. **Unauthorized Access:** An unauthorized access may originate locally, such as from a cloned key fob, or remotely through an internetwork communication channel such as WiFi or 5G broadband.
6. **Command Injection:** In this type of attack, malicious commands are injected into the vehicle intranet to induce an abnormal behavior.
7. **Masquerading:** This attack potential is realized by impersonating an authorized system or user to take control of the vehicle.
8. **Data Tampering:** This attack potential can be realized by the widespread Man-In-The-Middle (MITM) attack tools. The time to accomplish such an attack can take place very quickly.

Factors for Analysis

The following information describes the affecting factors for each threat about which a score is generated in this page:

1. **Elapsed Time:** Time taken by an attacker to identify a potential vulnerability, to develop an attack method, and to sustain effort required to execute the attack.
2. **Specialist expertise:** Describes the level of sophistication of the attacker.
3. **Knowledge of the target:** Refers to the familiarity of the attacker on the target.
4. **Window of opportunity:** This refers to the duration of time in which the vulnerability is exploitable.
5. **IT hardware/software or other equipment:** This refers to the availability and the level of complexity of equipment/software needed to identify or exploit a vulnerability.

Calculating Score

To calculate a score for each threat the user must simply select an option for the factors which affect that threat. The total will update as the user selects the factors for each threat and be displayed on the page.

Visualize

To visualize the results in a graph, the user must first generate the scores for the threats by selecting the options for the factors affecting each threat. Once the scores have been generated the user should press the button labeled “Visualize” to generate a Tree Map chart which arranges each threat into boxes whose size and color are determined by the calculated score for that threat.

Reset

To reset the scores for each threat the user must simply press the reset button. This will clear all user selected choices and reset the scores as well as the graph.

Attack Potential of Threats on Vehicle Assets

Threats on Assets	Factors for Analysis					Total
	Elapsed Time	Specialist Expertise	Knowledge of Target	Window of Opportunity	Equipment/Software	
False Data from ECU	Immediate (20)	Laymen (20)	public knowledge (20)	unlimited (20)	specialized(10)	90
Blocking of CAN Bus	one day (15)	Laymen (20)	sensitive information (10)	moderate (10)	multi-specialized (1)	56
Malicious Software	one week (10)	proficient (10)	public knowledge (20)	easy (15)	highly specialized(5)	60
Denial of Telematics Service	one day (15)	novice (15)	critical information (5)	moderate (10)	multi-specialized (1)	46
Unauthorized Access	one month (5)	proficient (10)	sensitive information (10)	easy (15)	Select an option	40
Command Injection	one month (5)	expert (5)	sensitive information (10)	moderate (10)	standard (15)	45
Masquerading	one month (5)	expert (5)	critical information (5)	difficult (5)	multi-specialized (1)	21
Data Tampering	one month (5)	expert (5)	unknown (0)	difficult (5)	multi-specialized (1)	16

[Visualize](#) [Reset](#)

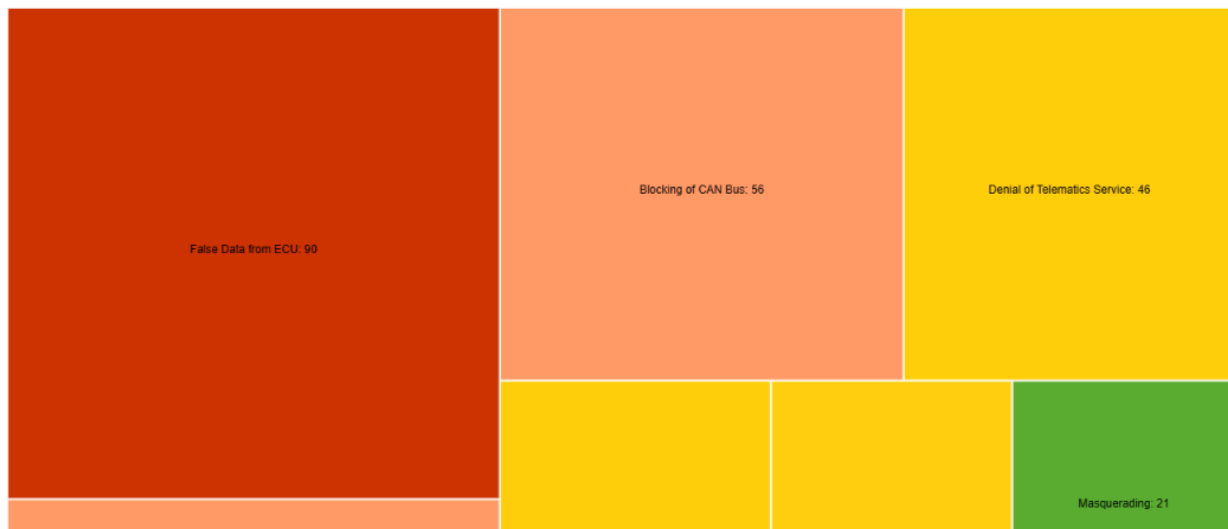


Figure 12 - Attack Potential of Threats on Vehicle Assets (TreeMap) Page

Common Weakness Scoring System (CWSS) Calculator

CWSS can be used in cases where there is little information at first, but the quality of information can improve over time. It is anticipated that in many use-cases, the CWSS score for an individual

weakness finding may change frequently, as more information is discovered. Different entities may determine separate factors at different points in time. The CWSS contains three categories of metrics: Base Finding Metrics, Attack Surface Metrics, and Environment Metrics. A CWSS 1.0 score can range between 0 and 100. The Base Finding Subscore supports values between 0 and 100. Both the Attack Surface Subscore and Environment Subscore support values between 0 and 1.

Base Finding Score Metrics

The Base Finding metric group consists of the following factors: Technical Impact (TI), Acquired Privilege (AP), Acquired Privilege Layer (AL), Internal Control Effectiveness (IC), and Finding Confidence (FC). The combination of values from Technical Impact, Acquired Privilege, and Acquired Privilege Layer gives the user some expressive power. For example, the user can characterize "High" Technical Impact with "Administrator" privilege at the "Application" layer.

Attack Surface Score Metrics

The Attack Surface metric group consists of the following factors: Required Privilege (RP), Required Privilege Layer (RL), Access Vector (AV), Authentication Strength (AS), Level of Interaction (IN), and Deployment Scope (SC).

Environmental Score Metrics

The Environmental metric group consists of the following factors: Business Impact (BI), Likelihood of Discovery (DI), Likelihood of Exploit (EX), External Control Effectiveness (EC), and Prevalence (P).

Calculating Scores

To calculate scores for each group the user must select a value for each metric within the groups. Once the user has selected an option from each metric the "Visualize" button can be clicked to calculate the scores for each metric. This will display the calculated score at the top of the page in the "Results" window, as well as generate a vector representing the choices for each metric.

Visualize

Once the user has selected an option from each metric the "Visualize" button can be clicked to calculate the scores for each metric. This will also populate the four graphs at the bottom of the page with the updated subscores. A radar chart is generated for each subscore and displays the

individual scores for the metrics within them. A bar chart is generated which compares the three metrics to each other.

Security Vulnerability Metrics for Connected Vehicles (SVMCV)

The purpose of this feature is to provide guidance to security engineers and testers using security vulnerability metrics that measure weak or vulnerable features in the software system of connected vehicles.

ECU Coupling Risk

The first metric on the SVMCV page is ECU Coupling Risk. This metric represents the potential risk of an attack that exploits vulnerabilities in the Electronic Control Units (ECUs) in a connected vehicle.

Complexity Risk

The next metric on the SVMCV page is Complexity Risk. This metric represents the potential risk of an attack that exploits the complexity of the software and hardware in a connected vehicle.

Communication Channel Risk

The third metric on the SVMCV page is Communications Channel Risk. This metric represents the potential risk of an attack that exploits vulnerabilities in the communication channels used by the vehicle.

History of Security Issue Risk

The fourth metric on the SVMCV page is the History of Security Issue Risk. This metric represents the potential risk of an attack that exploits known security issues in the vehicle.

I/O Data Risk

The fifth metric on the SVMCV page is I/O Data Risk. This metric represents the potential risk of an attack that exploits vulnerabilities in the Input/Output (I/O) data used by the vehicle.

Calculating Scores

To calculate the ECU Coupling Risk, enter the number of Active ECU Links along with the total ECU links and weight.

ECU Coupling Risk ⓘ		Weight
Active ECU Links	<input type="text" value="0"/>	<input type="text" value="1"/> ▼
Total ECU Links	<input type="text" value="0"/>	

Figure 13 - ECU Coupling Risk Section

To calculate the Complexity Risk, enter the number of lines of code in the vehicle's software, then enter the number of nestings and the weight of the nestings. Lastly enter the weight of the risk.

Complexity Risk ⓘ		Weight
Source Lines of Code	<input type="text" value="0"/>	<input type="text" value="1"/> ▼
Number of Nestings	<input type="text" value="0"/>	
Weight of Nestings	<input type="text" value="1"/> ▼	

Figure 14 - Complexity Risk Section

To calculate the Communications Channel Risk, enter the number of communication channels in the vehicle.

Communication Channel Risk ⓘ				Weight
Active V2V Links	<input type="text" value="0"/>	Active V2I Links	<input type="text" value="0"/>	<input type="text" value="1"/> ▼
Total V2V Links	<input type="text" value="0"/>	Total V2I Links	<input type="text" value="0"/>	
V2V Weight	<input type="text" value="0.1"/> ▼	V2I Weight	<input type="text" value="0.1"/> ▼	
Active U2V Links	<input type="text" value="0"/>	Active IV Links	<input type="text" value="0"/>	
Total U2V Links	<input type="text" value="0"/>	Total IV Links	<input type="text" value="0"/>	
U2V Weight	<input type="text" value="0.1"/> ▼	IV Weight	<input type="text" value="0.1"/> ▼	

Figure 15 - Communications Channel risk Section

To calculate the History of Security Issue Risk metric, you will need to enter values for the following three sub-metrics:

1. Number of years since first attack: This sub-metric represents the number of years since the first attack on the vehicle. Enter the number of years since the first attack in the input field provided.
2. Number of First Attack: This sub-metric represents the number of attacks that have been attempted on the vehicle in the past. Enter the number of attacks in the input field provided.
3. Number of Second Attack: This sub-metric represents the number of successful attacks that have been carried out on the vehicle in the past. Enter the number of successful attacks in the input field provided.

Once you have entered values for all three the Forgetting Factor will update accordingly.

History of Security Issue Risk ⓘ Weight

Number of Years Since First Attack	<input type="text" value="0"/>	
Number of First Attack	<input type="text" value="0"/>	
Number of Second Attack	<input type="text" value="0"/>	
Forgetting Factor	0.00	

Figure 16 - History of Security Issue Risk Section

To calculate the I/O Data Risk, enter the number of I/O data sources in the vehicle.

I/O Data Risk ⓘ Weight

Number of Fixed Input	<input type="text" value="0"/>	
Number of Fluctuating Input	<input type="text" value="0"/>	FI Weight <input type="text" value="1"/>
Number of Insensitive Output	<input type="text" value="0"/>	
Number of Sensitive Output	<input type="text" value="0"/>	SO Weight <input type="text" value="1"/>

Figure 17 - I/O Data Risk Section

Once you have entered values for each of the 5 metrics, you can click the Calculate Total Risk button to see the overall risk score for the vehicle.

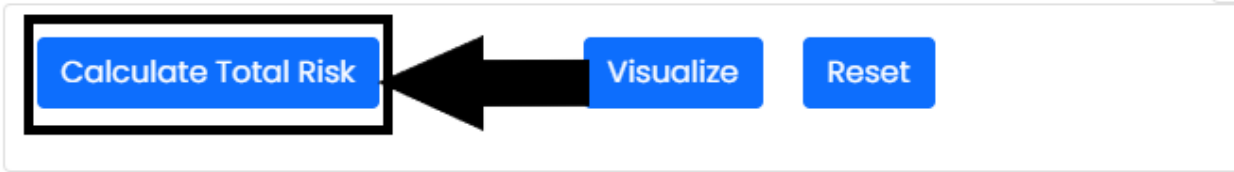


Figure 18 - Calculate Total Risk Button

Visualize

If you would like to see a visual representation of the risk score, click the Visualize Risk button. This will display a bar chart and a pie chart that provide an overview of each risk with their calculated score.



Figure 19 - Visualize Results Button

Reset

The Reset button will reset all user inputs to their default values.

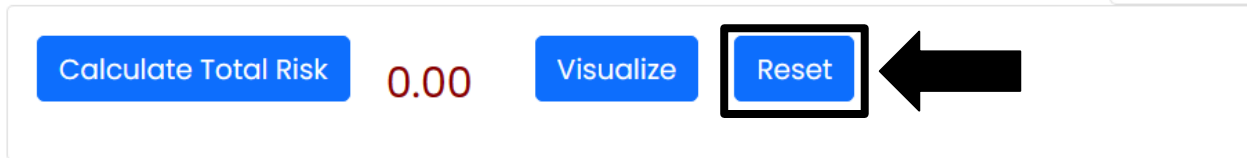


Figure 20 - Reset Inputs Button

Security Metrics for Operational Safety (SMOS)

The purpose of this feature is to provide an insight on the impact of cybersecurity to operational safety assessment (OSA).

Security Metrics

Within this page are the following operational safety metrics:

1. **Safety Envelope Metric:** This is an OSA measure of the connected vehicle's maintenance of safe boundary. An example is the vehicle's capability of maintaining the safe distance driving rule.
2. **Behavioral Metric:** This OSA measure indicates the improper behavior of the subject vehicle. Examples include speeding and sudden or hard braking.

3. **Component Metric:** This is an OSA measure of the performance of the vehicle components under normal operating condition. For example, an Electronic Control Unit (ECU) operating according to its specification.
4. **Sensing Metric:** This is an OSA measure of the accuracy of data collected by the CAV sensors. An example is the data collected by Roadside Units (RSUs). For non-autonomous vehicles, it is the measure of the quality of data transmitted or collected by vehicles participating in a vehicle-to-infrastructure (V2I) environment.
5. **Perception Metric:** This OSA measure pertains to the quality of interpretation of environment data collected by the CAV sensors. An example would be the automated recognition of traffic signs and signals.
6. **Planning Metric:** This OSA metric measures the ability of the CAV to devise a suitable trajectory through the CAV environment. An example is the quality of the CAV's planned trajectory in collision avoidance. For non-autonomous vehicles, it is the measure of the quality of maintaining control of the vehicle in the presence of an unexpected obstacle.
7. **Control Metric:** This OSA metric measures the ability of the CAV to execute the planned route. An example is the ability of the CAV to stay on the predetermined route. For non-autonomous vehicles, it is the measure of the quality of maintaining control of the vehicle navigation.
8. **Authentication Metric:** This OSA metric measures the quality of the authentication system deployed in the vehicle. This is extremely useful in modern vehicles that rely on communications such as those in V2V or V2I environment.
9. **Physical Access Metric:** This OSA metric measures the strength of physical access protection of vehicle controls. An example is the unsecured physical access to an OBD port which could compromise the vehicle's CAN bus.
10. **Communication Channel Metric:** This OSA metric pertains to the quality of the communication channel used by the vehicle.

Calculating Score

The user will see thirteen different metrics listed on the page, with the first ten requiring them to enter two values for each metric: "Value" and "Criticality." The user should enter their desired values in the appropriate input fields.

Security Metrics for Operational Safety (SMOS)				
Safety Envelope ⓘ Value <input type="text" value="0.2"/> Criticality <input type="text" value="0.3"/>	Behavioral ⓘ Value <input type="text" value="0.3"/> Criticality <input type="text" value="0.4"/>	Component ⓘ Value <input type="text" value="0.3"/> Criticality <input type="text" value="0.3"/>	Sensing ⓘ Value <input type="text" value="0.3"/> Criticality <input type="text" value="0.3"/>	Perception ⓘ Value <input type="text" value="0.2"/> Criticality <input type="text" value="0.4"/>
Planning ⓘ Value <input type="text" value="0.4"/> Criticality <input type="text" value="0.6"/>	Control ⓘ Value <input type="text" value="0.3"/> Criticality <input type="text" value="0.9"/>	Authentication ⓘ Value <input type="text" value="0.2"/> Criticality <input type="text" value="0.8"/>	Physical ⓘ Value <input type="text" value="0.4"/> Criticality <input type="text" value="0.7"/>	Communication ⓘ Value <input type="text" value="0.3"/> Criticality <input type="text" value="0.9"/>
Evaluation Factors				
Reliability <input type="text" value="0.9"/>	Revelance <input type="text" value="0.9"/>	Extent <input type="text" value="0.9"/>		
<input type="button" value="Calculate Total Risk"/> 1.2101399		<input type="button" value="Visualize"/>	<input type="button" value="Reset"/>	

Figure 21 - Safety Metrics for Operation Safety

After inputting values for each metric, the user can click the "Calculate Total Risk" button to calculate the total risk score. The system will use the values to generate a total risk score for the vehicle's operational safety. The user can view the total risk score on the page.

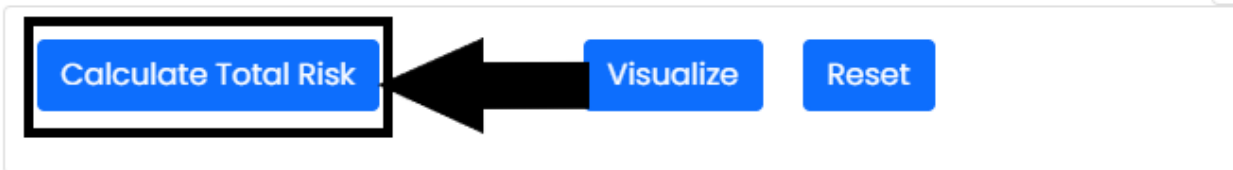


Figure 22 - Calculate Total Risk Button

Visualize

If the user wants to see a visual representation of the calculated risk score, they can click the "Visualize Risk" button. The system will display two charts showing the user's calculated scores for the security metrics.



Figure 23 - Visualize Results Button

Reset

If the user needs to start over or make changes to their input, they can click the "Reset All Input" button. The system will clear all values from the input fields, allowing the user to start over.

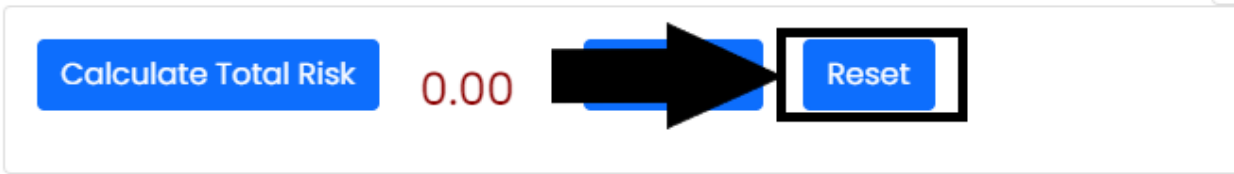


Figure 24 - Reset Inputs Button

Vehicle Security Best Practice Assessment Metrics (VSBPAM)

Security Metrics

The four phases and processes are described in the following:

Assessment Phase:

The assessment phase is designed to help you establish a baseline for your vehicle's security level.

This phase includes three processes:

1. **Security Policy:** This process assesses whether the organization has a security policy in place and whether it is reviewed and updated regularly.
2. **Data Security & Privacy:** This process assesses whether the organization has policies and procedures in place to protect sensitive data, and whether they are compliant with relevant privacy regulations.
3. **Risk Assessment:** This process assesses whether the organization has conducted a risk assessment to identify potential security threats and vulnerabilities, and whether appropriate controls have been implemented to mitigate those risks.

Design Phase:

The design phase is designed to help you establish a secure design for your vehicle. This phase includes two processes:

1. **System Protection and Prioritization:** This process assesses whether the organization has implemented appropriate security controls to protect the system from unauthorized access, and whether the organization has prioritized those controls based on risk.

2. **Security Architecture:** This process assesses whether the organization has designed a secure system architecture that meets the organization's security requirements and protects against known threats.

Implementation Phase:

The implementation phase is designed to help you implement the security measures identified in the design phase. This phase includes two processes:

1. **Remediation and Implementation:** This process assesses whether the organization has implemented appropriate remediation measures to address identified security vulnerabilities, and whether those measures have been effectively implemented.
2. **Security Test and Evaluation:** This process assesses whether the organization has conducted appropriate security testing to ensure that the system is secure and meets the organization's security requirements.

Operation Phase:

This phase focuses on maintaining and monitoring the vehicle security environment. This phase includes two processes:

1. **Awareness and Security Training:** This process assesses whether the organization has implemented appropriate security awareness and training programs for employees, and whether those programs are regularly updated and evaluated.
2. **Intrusion Detection and Response:** This process assesses whether the organization has implemented appropriate intrusion detection.

Calculating Score

To use the VSBPAM page, start by reading the list of questions that are displayed on each section. Each phase of the assessment is represented by a group of questions that are grouped together, and there are four questions for each process. Simply answer each question with either a "yes" or "no" by clicking on the corresponding radio button next to the question. As you progress through the questions, the system will automatically update the visual graph to show your progress in real time.




Phase	Process	Checklist	Response	
			Yes	No
Assessment	Security Policy ⓘ	Are security policies established? 	<input type="radio"/>	<input type="radio"/>
		Are security policies properly documented and widely disseminated within the organization?	<input type="radio"/>	<input type="radio"/>
		Are security policies strictly enforced? 	<input type="radio"/>	<input type="radio"/>
		Are security policies periodically reviewed/updated?	<input type="radio"/>	<input type="radio"/>
	Data Security & Privacy ⓘ	Is collected/stored data protected/encrypted? 	<input type="radio"/>	<input type="radio"/>
		Is transmitted data encrypted?	<input type="radio"/>	<input type="radio"/>
		Is there a control mechanism for sharing data?	<input type="radio"/>	<input type="radio"/>
	Risk Assessment ⓘ	Does the site comply with data protection standards and regulations (e.g., ISO/IEC 27001 certification, GDPR)?	<input type="radio"/>	<input type="radio"/>
		Do you conduct a periodic risk assessment of vehicle cybersecurity?	<input type="radio"/>	<input type="radio"/>
		Is there a developed and implemented organization-wide risk management strategy?	<input type="radio"/>	<input type="radio"/>
Is there a Supply Chain Risk Management (SCRM) policy?		<input type="radio"/>	<input type="radio"/>	
		Are security controls in place and periodically evaluated and/or enhanced?	<input type="radio"/>	<input type="radio"/>

Figure 25 - Vehicle Security Best Practice Assessment Metrics Page

Troubleshooting & Support

Error Messages

404 - Page not Found:

The page you are searching for does not exist on the website or could not be found. Try to check the spelling of the page in the URL or access the page from the links provided on the home page of the website.

500 - Internal Server Error:

This indicates that there is an error with the website's server. Try reloading the page, clearing your browser cache, deleting browser cookies, or coming back later.

Appendix A: Record of Changes

Table 2 - Record of Changes

Version Number	Date	Author/Owner	Description of Change
0.1	11/01/2022	VSMVS Development Team	Initial Draft
0.2	3/10/2023	VSMVS Development Team	Initial Release
0.3	4/08/2023	VSMVS Development Team	Latest Release

Appendix B: Referenced Documents

Table 4 - Referenced Documents

Document Name	Document Location and/or URL	Issuance Date
Auto-ISAC, Inc. Best Practices.	https://automotiveisac.com/best-practices/	2023, February 16
Common Weakness Scoring System (CWSS)	https://cwe.mitre.org/cwss/cwss_v1.0.1.html	2014, September 5
Common Criteria for Information Technology Security Evaluation.	https://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R5.pdf	2017, April
Common Vulnerability Scoring System version 3.1: Specification Document.	https://www.first.org/cvss/specification-document	2019, June
Best Practice for Metrics and Methods for Assessing Safety Performance of Automated Driving Systems (ADS).	SAE Industry Technologies Consortium.	2021, March

Document Name	Document Location and/or URL	Issuance Date
Evaluation of Operational Safety Assessment (OSA) Metrics for Automated Vehicles in Simulation.	SAE International.	2021, April 06

Connected Vehicle Dataset

Attributes, Collection, Processing, Generation and Mutation

Technical Report UWF-TR-FDOT-004-01
Version 4.0 approved

Prepared by Guillermo Francia, III
The University of West Florida
Florida Department of Transportation

June 1, 2023 (Initial)

June 8, 2023 (Revision 1)

July 15, 2023 (Revision 2)

September 2, 2023 (Revision 3)

November 14, 2023 (Revision 4)

Table of Contents

Introduction	182
Purpose	182
Related CAV Datasets	182
1.0 <i>Vehicle Reference Misbehavior Dataset (VeReMi)</i>	182
2.0 <i>Car Hacking Dataset</i>	182
3.0 <i>Connected Vehicle Pilot (CVP) Open Data</i>	183
4.0 <i>BDD100K</i>	183
Working Datasets	183
1.0 <i>Roadside Unit (RSU) Dataset</i>	183
1.1 Basic Safety Message (BSM)	183
1.2 Common Safety Request (CSR)	184
1.3 MAP Message.....	184
1.4 Personal Safety Message (PSM).....	184
1.5 Signal Phase and Timing (SPaT).....	184
1.6 Signal Request Message (SRM).....	185
1.7 Traveler Information Message (TIM)	185
2.0 <i>Synthetic Dataset</i>	185
3.0 <i>Malicious Dataset</i>	187
3.1 Brake System Anomaly	187
3.2 Transmission System Anomaly.....	188
3.3 Longitudinal Acceleration Anomaly	188
3.4 Hard Braking Anomaly.....	189
3.5 Speed Anomaly.....	189
References	189
Appendix I	145
Table 1. <i>BSM Coredata Characteristics</i>	191
Appendix II	383
Algorithm 1: <i>RSU Data extraction and Cleansing</i>	383
Appendix III	196
Algorithm 2: <i>Synthetic data generation</i>	196
Algorithm 3: <i>Malicious data generation and injection</i>	197

Introduction

The advancements in vehicle connectivity are quickly outpacing the developments in safety and security. The Connected and Automated Vehicles (CAV) infrastructure ushers an attack surface that is continuously evolving and expanding. Society can not tolerate another critical infrastructure wherein security is regarded as an afterthought. Security safeguards need to be designed, implemented, and evaluated to prevent or mitigate incidents that may jeopardized the safety of people and assets. To this end, the application of artificial intelligence using machine learning systems is proposed to investigate their feasibility in identifying malicious datasets that are purposely injected into the CAV infrastructure.

Purpose

This document contains the details of the datasets that will be used by the Machine Learning system for the purpose of distinguishing normal information from abnormal information and for classifying various classes of anomalous data.

Related CAV Datasets

The following concise literature review of CAV datasets provides a guidance on test data generation that can be used to study the feasibility of applying artificial intelligence in securing CAVs. These datasets are similarly cited in [75].

1.0 Vehicular Reference Misbehavior Dataset (VeReMi)

The VeReMi dataset [76] is a publicly extensible dataset for detecting misbehavior on Vehicular Ad-hoc Networks (VANETs). Its purpose is to facilitate the reproducibility and the evaluation of detection mechanisms using a reference dataset. Further, it provides a repository of newly discovered attack datasets.

2.0 Car Hacking Dataset

The car hacking datasets [77] include Denial of Service attack, fuzzy attack, spoofing the drive gear, and spoofing the RPM gauge. Datasets were constructed by logging Controller Area Network (CAN)

traffic via the On-Board Diagnostic (OBD)-II port from a real vehicle. For each dataset, 300 message intrusions were injected.

3.0 Connected Vehicle Pilot (CVP) Open Data

The CVP Open Data is a product of the U.S. Department of Transportation Intelligent Transportation Systems Joint Program Office (ITS JPO's) Connected Vehicle Pilot Deployment Program. Data were collected at each pilot site: the Wyoming DOT, the Tampa-Hillsborough Expressway Authority (THEA), and the New York City DOT. The purpose of the data collection is to facilitate independent evaluations of the use of connected vehicle technology on real roadways [78].

4.0 BDD100K

The BDD100K dataset is an open driving video dataset with 100K videos and 10 tasks to evaluate image recognition algorithms for autonomous driving [79]. The Berkeley Deep Drive (BDD) consortium is the curator of the dataset. It is a comprehensive dataset that contains, among others, Global Positioning System (GPS) information, weather conditions, road object information, road lanes, etc.

Working Datasets

To gain a better understanding of CAV datasets, we start by scrutinizing the datasets that were collected and shared by the research team in the FDOT pilot site at the University of Florida.

1.0 Roadside Unit (RSU) Dataset

The dataset, in SAE J2735 message format, were collected by Roadside Units (RSUs) in Gainesville, FL and can be availed through an AWS Simple Storage Service (S3) repository. The raw dataset, in compressed XML format, were converted to a readable comma separated value (csv) format in preparation for data cleansing. The data component types are described in the following.

1.1 Basic Safety Message (BSM)

The *Basic Safety Message (BSM)* is used to exchange safety data and consists of two parts:

- The mandatory part, also called BSMcoreData, is typically described in Abstract Syntax Notation One (ASN.1) [14] format, a formal notation to describe data transmitted by

telecommunication protocols. Instead of describing the BSMcoreData in ASN.1 notation, we present each item in detail using a tabular format as shown in Appendix I. The data characteristics presented on the table are excerpted from [15] .

- The optional part of the BSM includes the Vehicle Safety Extension consisting of event flags, path history, path prediction, and the Radio Technical Commission for Maritime Services (RTCM) package.

1.2 Common Safety Request (CSR)

The *Common Safety Request (CSR)* message is a unicast request sent by a vehicle requesting additional information from other vehicles required for the active safety applications. Vehicles responding to the request may add CSR data elements in their appropriate place in the BSM when broadcasting back to the requesting vehicle. Data elements may include heading, speed, and spatial position [80].

1.3 MAP Message

The *Map Data (MAP)* message contains geographic road information. Typically, the message is sent from the infrastructure to the vehicle. A message instance could be one that describes the geometric layout of one or more complex intersections, road segments, or high-speed curve outlines; each of which can have their own data structures within a single message [80].

1.4 Personal Safety Message (PSM)

The *Personal Safety Message (PSM)* is used to broadcast safety data on vulnerable road users (VRU) such as pedestrians, cyclists, or road workers. Data elements in this message include position, speed and heading of the VRU similar to data transmitted by vehicles., along with the VRU's path history and predicted path. Furthermore, crosswalk data will be transmitted when applicable [80].

1.5 Signal Phase and Timing (SPaT)

The *Signal Phase and Timing (SPAT)* message originates from the infrastructure and provides the current status of one or more signalized intersections. SPAT and MAP data can be used together to provide vehicle safety systems with the state of the signal phase, when the signal phase is expected to change, and geometry of an intersection including ingress and egress lanes where applicable [80].

1.6 Signal Request Message (SRM)

The *Signal Request Message (SRM)* is a message sent by V2X-equipped entities to the Roadside Units (RSUs) in a signalized intersection for priority signaling or preemption of signal request. Similar to the above-mentioned SPAT and MAP, the returned data defines a path through the intersection including ingress and egress lanes. Optional components of the SRM are the time of arrival and the expected duration of the service. Also, data for one-to-many intersections are supported. The SRM contains a RequestorDescription data frame that allows the requestor to identify itself in various ways and includes current speed, heading, and location [80].

1.7 Traveler Information Message (TIM)

The *Traveler Information Message (TIM)* provides advisory and roadside information to V2X devices originating from the infrastructure. The TIM is used to send information such as advisory and road sign messages to equipped devices. The International Traveler Information Systems (ITIS) encoding system is implemented to send standard message phrases, but the TIM allows for local place names. Due to dynamic conditions found on roadways, TIMs are activated at specified times and duration with a resolution of one minute. The geographic area in which TIMs are broadcast can be defined as a radius around a specific location or defined using the roadway geometry [80].

2.0 Synthetic Dataset

After gaining an understanding of the BSM dataset characteristics, we investigated the viability of generating synthetic datasets that depict a typical vehicle moving in a straight-line trajectory without regard to traffic condition. The motivation behind this initiative is three-fold:

- to derive CAV datasets that can be used to study the application of Machine Learning (ML) algorithms to identify and classify malicious messages;
- to be able to create additional malicious datasets that will test attack scenarios for each of the V2V safety applications described in [81]; and

- to be able to perform a comparative study on the speed of convergence of various ML training algorithms on disparate datasets.

The synthetic BSM CoreData generation proceeds using the following assumptions and constraints:

- Starting geolocation (Central Florida) with coordinates:
 - latitude: 28.890658
 - longitude: -82.097812
- Data collection time interval: 20 sec. Note that SAE J2945/1 Standard [81] requires vehicles to transmit 10 BSMs per second.
- Travel is on a straight line towards north
- Acceleration is in units of 0.0328 ft/sec²
- Acceleration (deceleration) is randomly applied every 10 minutes. The random value will range from -10 to 10 mph. This represents lateral acceleration (deceleration)
- Angle steering is 127 (unavailable)
- Brake system status: 0 during acceleration; 5 during deceleration (assuming front wheel brakes). The brake system status is adjusted to reflect the acceleration or deceleration condition
- Elevation: 61440 (unknown)
- Heading: 28800 (unavailable)
- Latitude is calculated using equations (1)-(5)
 - earth_radius = 6371 km (1)
 - current_lat= math.radians(initial_lat) (2)
 - # distance after 20 seconds of travel
 - dist_meters = speed_mph * 0.44704 * 20 (3)
 - # change in latitude
 - delta_lat = dist_meters / earth_radius (4)
 - # new latitude is calculated as
 - new_lat=math.degrees(current_lat+delta_lat) (5)
- Longitude is assumed constant at -82.097812
- Msg Count: 0 (unavailable)
- Vehicle ID: 0 (unavailable)
- SecMark : Calculated elapsed time in milliseconds

- Transmission: 2 (forward gear)
- Hard Braking: 0 (no), 1 (yes). Deceleration of 8 mph in 1 second (23.09 ft/sec² when applying the unit measure of 0.01 meters/sec²) while traveling at a speed of > 25 mph indicates hard braking.
- Vehicle size:
 - Length = 21 ft
 - Width = 7 ft
- Starting speed is 25 mph
- The brake status is adjusted to reflect the acceleration or deceleration condition
- The speed is calculated using equation (6)

$$V_1 = V_0 + a * t \tag{6}$$

where

V_1 is the current speed, miles/hr

V_0 is the initial speed, miles/hr

a is the acceleration, miles/hr²

t is the elapsed time, hr

- The data record is terminated with a status flag: 1 for normal and 0 for abnormal (malicious).

3.0. Malicious Dataset

After obtaining datasets from two sources: the RSU datasets for the Gainesville pilot site and the synthetically generated dataset, we implemented an application to inject malicious data records into each of the datasets. This process, in essence, produces two mutated datasets. The mutation process is described in the following.

Using the assumption for a typical acceleration of 0.577 ft/sec² (derived from 17.6 ft/sec² and applying the unit measure of 0.01 meters/sec²), malicious BSMcoreData test datasets are generated according to the following parameters. Note that for each data record, the status flag value is 1 to indicate an abnormal (malicious) data record.

3.1 Brake System Anomaly

3. To simulate an accelerating vehicle while brakes are engaged:

Randomly generate a BSMcoreData record with Brake System Status = 1111 (decimal value 15; brakes applied) and Longitudinal Acceleration = 5.364 m/sec^2 .

4. To simulate an accelerating vehicle in reverse while the brakes are engaged:

Randomly generate a BSMcoreData record with Brake System Status = 0000 (decimal value = 0; all brakes not engaged), Longitudinal Acceleration = 5.364 m/sec^2 and Transmission Status=011 (decimal value 3; reverse gear).

3.2 Transmission System Anomaly

4. To simulate a speeding vehicle with transmission system in neutral: randomly generate a BSM data record with speed = 15 mph and Transmission Status = 000 (decimal value 0).

5. To simulate a speeding vehicle with transmission system in park: randomly generate a BSM data record with speed = 25 mph and Transmission Status = 001 (decimal value 1).

6. To simulate a speeding vehicle with transmission system in reverse gear: randomly generate a BSM data record with speed = 55 mph and Transmission Status = 011 (decimal value 3).

3.3 Longitudinal Acceleration Anomaly

1. To simulate an accelerating vehicle with transmission gear in neutral: randomly generate a BSM data record with Longitudinal Acceleration = 5.364 m/sec^2 and Transmission Status=000.

2. To simulate a decelerating vehicle with the transmission gear in park: randomly generate a BSM data record with Longitudinal Acceleration = -5.364 m/sec^2 and Transmission Status=001.

3. To simulate an accelerating vehicle with the transmission gear in reverse: randomly generate a BSM data record with Longitudinal acceleration = 5.364 m/sec^2 and Transmission Status=011.

3.4 Hard Braking Anomaly

To simulate malicious hard braking with the transmission in forward gear: randomly generate a BSM data record with Longitudinal Deceleration = -25.59 ft/sec^2 (2.5 times the value of deceleration rate considered as hard braking) and Transmission Status=010 (decimal value 2)

3.5 Speed Anomaly

To simulate malicious speed value with transmission in forward gear: randomly generate a BSM data record with speed 3 times the speed value of the previous speed reading and Transmission Status=010 (decimal value 2)

References

- [1] Forum of Incident Response and Security Teams (FIRST), "Common Vulnerability Scoring System version 3.1: Specification Document," June 2019. [Online]. Available: <https://www.first.org/cvss/specification-document>. [Accessed 13 February 2020].
- [2] G. A. Francia, "Connected Vehicle Security," in *15th International Conference on Cyber Warfare and Security (ICWS 2020)*, Norfolk, VA, 2020.
- [3] NIST, "CSV-2019-13582 Detail," 15 November 2019. [Online]. Available: <https://nvd.nist.gov/vuln/detail/CVE-2019-13582>. [Accessed 20 February 2023].
- [4] Common Vulnerabilities and Exposure, "CVE-2018-9322," 31 May 2018. [Online]. Available: <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-9322>. [Accessed 13 February 2020].
- [5] Common Criteria Portal, "Common Criteria for Information Technology Security Evaluation," April 2017. [Online]. Available: <https://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R5.pdf>. [Accessed 24 February 2020].
- [6] MITRE Corporation, "Common Weakness Scoring System (CWSS)," 2 April 2018. [Online]. Available: https://cwe.mitre.org/cwss/cwss_v1.0.1.html.
- [7] Automated Vehicle Safety Consortium, "Best Practice for Metrics and Methods for Assessing Safety Performance of Automated Driving Systems (ADS)," SAE Industry Technologies Consortium, March 2021.
- [8] M. Elli, J. Wishart, S. Como, S. Dhakshinamoorthy and J. Weast, "Evaluation of Operational Safety Assessment (OSA) Metrics for Automated Vehicles in Simulation," SAE, 2021.

[9] SAE, "Operational Safety Metrics for Verification and Validation (V&V) of Automated Driving Systems (ADS) J3237," SAE International, September 2020.

Appendix I

Table 1. BSM Coredata Characteristics

Data Item	Data Type	Description
Acceleration	System Defined (in units of 0.01 m/sec ²)	acceleration_set_4_way : long accel: integer, (acceleration along the X-axis or the direction of travel; negative value indicates braking action) lat accel: integer, (acceleration along the Y-axis or the direction of travel; negative value indicates left turning action, positive indicates right turning) vert accel: one-byte signed integer, (-127 represents unavailable data) yaw: integer (vehicle rotation about the vertical axis and expressed in 0.01 degrees/second)
Angle: Steering Wheel Angle (units of 1.5 degree)	Signed Integer (range: -189 to +189)	0x01 = 00 = +1.5 degree 0x81 = -126 = -189 degree and beyond 0x7E = +126 = +189 degree and beyond 0x7F = +127 to be used for unavailable
Brake System Status: 2-octet information about the current brake system of the vehicle	System Defined	brakeAppliedStatus; (4 bits total--one bit for each wheel, value 1 means active; Thus, 0000 means all Off, 0001 left front active, 0010 left rear active, 0100 right front active, 1000 right rear active) brakesUnavailableStatus: (5 th bit; 1 means true) sparebit: 6 th bit unused; set to 0 traction: (7 th and 8 th bits) (Traction Control Status) 00-unavailable) 01-off 10-on but not engaged 11-engaged, abs: (9 th and 10 th bits) (Anti-lock Brake Status) 00-unavailable; 01-off; 10-on but not engaged; 11-engaged. Stability Control Status, Brake Boost Applied, and Auxiliary/Emergency Brake Status omitted for brevity.

Elevation (unit is 10 cm)	Integer	Elevations from 0 to 61439 (0x0000 to 0xEFFF) meters Elevations from -409.5 to -0.1 (0xF001 to 0xFFFF) meters Unknown value is encoded as 0xF000
Heading: Represents 0.0125 degrees from the North	2 octets of unsigned integer. Range 0 to 28800	Value of 28800 indicates unavailable.
Latitude: 32- bit value represents 1/10 microdegrees with reference to the horizontal datum	Integer (range -- 900000000 to 900000001	Provides a range of plus-minus 180 degrees 900000001 indicates unavailable
Longitude: 32-bit value represents 1/10 microdegrees with reference to the vertical datum	Integer (range -1800000000 to 1800000001	Provides a range of plus-minus 180 degrees 1800000001 indicates unavailable
Msg Count	Non-negative Integer	Message Count
Vehicle ID	Non-negative Integer	Vehicle Identifier
SecMark	Non-negative Integer	Units of milliseconds
Speed (in units of 0.02 m/sec. Range: 0 to 8191	Non-negative Integer (13 bits of the 2-byte Transmission+Speed)	Use 8191 to indicate unavailability.
Transmission	System Defined Occupies bits 14 to 16 of the 2-byte Transmission+Speed	000-Neutral 001-Park 010-Forward gear 011-Reverse gear 100, 101, and 110 are unused 111 unavailable

Vehicle Size (in cm)	System Defined 3 octets	Width : Non-negative Integer (10 unsigned bit with values [0,1023] (0 when unavailable) Length : Non-negative Integer (14 unsigned bit with values [0,16383] (0 when unavailable)
-------------------------	----------------------------	---

Appendix II

Algorithm 1: RSU Data extraction and Cleansing

```
Function to Load BSM Data (load_BSMs):
    # Open the compressed XML file containing the BSM data
    with gzip.open(FPATH, 'rt') as fz:
        # Read and parse each line into an XML tree
        trees = [et.fromstring(l) for l in fz]

    # Print the tag of the first XML element
    print(trees[0].tag)

    # Filter out specific BSM messages
    BSMs_raw = [tree for tree in trees if tree.tag ==
                "MessageFrame"
                if tree.findall('messageId')[0].text == '20']

    # Define the desired BSM data attributes
    PAYLOAD_INFO_KEYS = ['id', 'secMark', 'lat', 'long',
                        'speed', 'heading', 'angle']

    # Make a list to store the BSM dictionaries
    BSMs_dicts_list = []

    # Extract the data from each BSM message
    for tree in BSMs_raw:
        message_dict = {}
        for k in PAYLOAD_INFO_KEYS:
            # Find the text for each key from the XML tree
            message_dict[k] = [ch.text for ch in \
                tree.findall(f'./value//BasicSafetyMessage//core
                Data/{k}')][0]

        # Process and update timestamp and other fields in #
        message_dict
        message_dict['timestamp'] =
            processSecMark(int(message_dict['secMark']),
                datetime.datetime.now(datetime.timezone.utc).strftime("%
                Y-%m-%d_%H:%M:%S"))
        message_dict['lat'] = int(message_dict['lat'])/10 ** 7
        message_dict['long'] = int(message_dict['long'])/10 ** 7
        message_dict['speed_mph'] =
            processSpeedMPH(int(message_dict['speed']))
        # ... (Other hardcoded values)
```

```
# Append the cleaned dictionary to the list
BSMs_dicts_list.append({k: v for k, v in
                        message_dict.items()
                        if k not in ['id', 'speed', 'lat_long']})
# Return the list of processed BSM dictionaries
return BSMs_dicts_list

# Load the BSMs from the specified file path
BSMs_dicts_list = load_BSMs(FPATH)
```

Appendix III

Algorithm 2: Synthetic data generation

```
# Constants
earth_radius = 6371e3 # Earth radius in meters

# Initial position coordinates
initial_latitude = 28.890658
initial_longitude = -82.097812

# Initial time and end time
start_time = datetime.strptime('08:00 AM', '%I:%M %p')
end_time = start_time + timedelta(hours=10) # 10 hours drive

# Initial speed and acceleration
speed_mph = 25
acceleration = 0

# Generate BSM dataset
bsm_dataset = []

while start_time < end_time:
    # Calculate new latitude based on the distance traveled
    # Convert mph to m/s and multiply by 20 seconds
    distance_meters = speed_mph * 0.44704 * 20
    delta_latitude = distance_meters / earth_radius
    new_latitude = initial_latitude +
                    math.degrees(delta_latitude)

    # Update position coordinates
    initial_latitude = new_latitude

    # Change speed and acceleration randomly every 10 minutes
    if start_time.minute % 10 == 0 and start_time.second == 0:
        # Randomly choose to accelerate or decelerate by a value
        # between 1 and 10 mph
        acceleration = random.uniform(-10, 10)
        speed_mph += acceleration
        # Ensure speed is within 0 and 90 mph range
        speed_mph = max(0, min(speed_mph, 90))

    # Determine braking status based on acceleration
    brake_system_status = 5 if acceleration < 0
                          else 0
    hard_braking = 1 if acceleration < 0 and speed_mph > 25
                  else 0
```

```

# Create BSM record with the current data
bsm_record = { # Python code dictionary structure
    'Timestamp': start_time.strftime('%Y-%m-%d %H:%M:%S'),
    'Latitude': initial_latitude,
    'Longitude': initial_longitude,
    # Other fields
}

# Append the record to the dataset
bsm_dataset.append(bsm_record)

# Increment time by 20 seconds
start_time += timedelta(seconds=20)

# The synthetic data is now created and able to be put in CSV

```

Algorithm 3: Malicious data generation and injection

```

AnomaliesDict = [
    {'Brake Status': 15, 'Acceleration': 536.4}, #Anomaly A.1
    {'Brake Status': 0, 'Acceleration': 536.4, 'transmission
        status': 3}, #Anomaly A.2
    {'speed_mph': 15, 'transmission': 0}, #Anomaly B.1
    {'speed_mph': 25, 'transmission': 1}, #Anomaly B.2
    {'speed_mph': 55, 'transmission': 3}, #Anomaly B.3
    {'Acceleration': 536.4, 'transmission': 0}, #Anomaly C.1
    {'Acceleration': -536.4, 'transmission': 1}, #Anomaly C.2
    {'Acceleration': 536.4, 'transmission': 3}, #Anomaly C.3
    {'Acceleration': -780, 'transmission': 2}, # Anomaly D.1
]

Iterate through all BSM records:
    Select every 2nd, 5th or 10th record:

        anomalySelected = Randomly select from AnomaliesDict[]

        for each parameter in anomalySelected:
            if parameter == Brake Status:
                message_dict[Brake Status] =
                    anomalySelected[Brake Status]

            if parameter == Acceleration:
                message_dict[accelSet long] =
                    anomalySelected[Acceleration]

            if parameter == transmission:

```

```
        message_dict[transmission] =  
            anomalySelected[transmission]  
  
if parameter == speed_mph:  
    message_dict[speed_mph] =  
        anomalySelected[speed_mph]  
  
set normalFlag value based on anomalySelected
```


Software Requirements Specification

for

Vehicle Security Machine Learning (VSML) System

Technical Report UWF-TR-FDOT-005-01

Version 0.7 approved

Prepared by Guillermo Francia, III

The University of West Florida

Florida Department of Transportation Contract: BED34 977-01

April 1, 2023 (Initial)

April 8, 2023 (Revision 1)

April 17, 2023 (Revision 2)

April 27, 2023 (Revision 3)

May 9, 2023 (Revision 4)

June 6, 2023 (Revision 5)

Table of Contents

1. Introduction	202
1.1 Purpose	202
1.2 Document Conventions	203
1.3 Related Documents	203
1.4 Document Revisions Table	203
1.5 Business Processes	204
2.0 Overview of the Product	205
2.1 Vehicle Security Machine Learning (VSML) System	205
2.2 Vehicle Security Machine Learning System Workflow	205
2.2.1 Data Ingestion	205
2.2.2 Data Cleansing	206
2.2.3 Data Mutation	206
2.2.4 Data Validation and Visualization	207
2.2.5 Training Data Selection	207
2.2.6 Model Training	207
2.2.7 Model Tuning	207
2.2.8 Deployment	207
2.2.9 Monitoring	208
2.3 Hosting	208
2.3 User Classes and Characteristics	208
2.4 Operating Environment	208
2.5 Design and Implementation Constraints	209
2.6 Assumptions and Dependencies	209
3.0 Interface Requirements	209
IR-1: Jupyter Notebook	209
IR-2: Data Visualization	210
IR-3: Performance Metrics	210
IR-4: Data Input	210
4.0 Functional Requirements	210
FR-1: Login	210
FR-2: Data Repository	210
FR-2.1. Data Entry	210
FR-2.2. Data Cleansing and Transformation	211
FR-2.3. Malicious Data Injection	211
FR-3: Machine Learning Models	211
FR-3.1: Model Training and Validation	211
FR-3.2: Model Testing	211

<i>FR-4: System Deployment</i>	211
5.0 Non-Functional Requirements	212
<i>NF-1: Portability</i>	212
<i>NF-2: Usability</i>	212
<i>NF-3: Speed</i>	212
6.0 Quality Attributes	212
7.0 Source Code Repository and Version Control Requirement	212
<i>SC-1: Source Code Repository and Control</i>	212
8.0 Requirements Table	213
9.0 Performance Indicators	214
<i>9.1 Definitions</i>	214
<i>9.2 Recall</i>	214
<i>9.3 Precision</i>	215
<i>9.4 Accuracy</i>	215
<i>9.5 F-Measure</i>	215
10.0 REFERENCES	215
Appendix I	217
Appendix II	218
Appendix III	221
Appendix IV	222
Appendix V	223
Appendix VI	224
Appendix VII	227

1. Introduction

The innovations in the interconnectivity of vehicles enable both expediency and insecurity. Surely, the convenience of gathering real-time information on traffic and weather conditions, the vehicle maintenance status, and the prevailing condition of the transport system at a macro level for infrastructure planning purposes is a boon to society. However, this newly found conveniences present unintended consequences. Specifically, the advancements on automation and connectivity are outpacing the developments in security and safety. Starting at the lowest level, numerous vulnerabilities have been identified in the internal communication network of vehicles. These insecurities become even more pronounced with the advancement of external communication systems such as those found in connected vehicles.

Most of the existing research on machine learning for vehicle security has focused on detecting anomalies and cyberattacks in the Controller Area Network (CAN) bus, which serves as a protocol for in-vehicle network communication in electric vehicles, using various machine learning methods. In this research and application development effort, we focus on applying Machine Learning to connected vehicle data.

This undertaking entails the design, implementation, and testing of a prototype Machine Learning system for vehicle security and performance monitoring. It utilizes the Amazon Web Services (AWS) SageMaker Studio for Machine Learning (ML) development and deployment. Data ingestion, cleansing, normalization, and data mutation (malicious data injection) of the Connected Vehicle Roadside Unit (RSU) will also be enabled. The dataset, in SAE J2735 message format, were collected by Roadside Units (RSUs) in Gainesville, FL. The data will be preprocessed and normalized for Machine Learning. In addition, synthetic anomalous data will be randomly injected to simulate cyber threats and attacks.

This is a prototype system to demonstrate the capability of a Machine Learning system in enhancing Vehicle Security.

1.1 Purpose

This document describes the software requirements for Version 1.0 of the Vehicle Security Machine Learning (VSML) System. The VSML is a subcomponent of the Connected Vehicle Security Metrics and Threat Intelligence Project. This document provides an overview of the users and context of the VSML and covers all of its functional, non-functional, and data requirements.

1.2 Document Conventions

This document is based on the IEEE 830 Standards and the Florida Department of Transportation Requirements Standards. Specific conventions used in this document are listed below:

- Priorities are indicated for each feature as well as in the Requirements Table. A green highlighting indicates must have features, while a yellow highlight represents a should have feature.
- Requirements follow the form of <TAG>-#.#.# where a tag indicates a category of requirements. And the # represents the ID of the requirement in a hierarchical fashion.

1.3 Related Documents

The following reference documents were used in the creation of this document:

- IEEE 830 Standards on Software Requirement Specifications
- UWF Scope of Service Document for the Connected Vehicle Security Metrics and Threat Intelligence Project

1.4 Document Revisions Table

Revisor	Revision Date	Reason
Guillermo Francia, III	April 1, 2023	Added the Machine Learning (ML) System workflow
Elizabeth Uebele	April 3, 2023	Added
Guillermo Francia, III	April 8, 2023	Added details to the ML System workflow. Added the section on Performance Indicators
Elizabeth Uebele	April 12, 2023	Added Functional Requirements Added Portability as Nonfunctional Requirement

Guillermo Francia, III	April 12, 2023	Nonfunctional Requirements Hosting, Constraints Assumptions and Dependencies Quality Attributes Source Code Repository
Elizabeth Uebele	April 17, 2023	Inserted Appendix X Added Table of Contents
Guillermo Francia, III	April 17, 2023	Added User Classes and Characteristics Added the Operating Environment information
Guillermo Francia, III	May 3, 2023	Added Interface Requirements Added Appendices Added preliminary ML runs on Matlab™
Elizabeth Uebele	May 9, 2023	Updated Requirements Table Proofread and Edited
Guillermo Francia, III	June 6, 2023	Added Appendix VII : SecMark conversion to Timestamp

1.5 Business Processes

N/A

2.0 Overview of the Product

2.1 Vehicle Security Machine Learning (VSML) System

The VSML system will utilize the Amazon Web Services (AWS) SageMaker Studio for ML development and deployment. The Machine Learning System workflow is depicted below.

2.2 Vehicle Security Machine Learning System Workflow

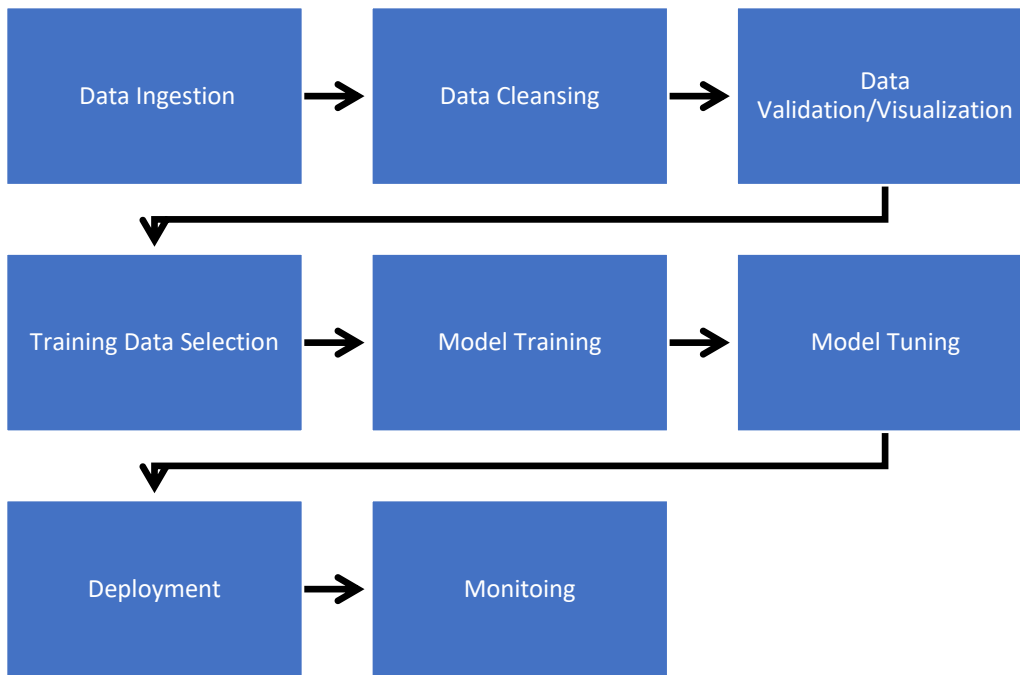


Figure 1. Vehicle Security Machine Learning System Workflow

2.2.1 Data Ingestion

The dataset, in SAE J2735 message format, were collected by Roadside Units (RSUs) in Gainesville, FL and can be availed through an AWS Simple Storage Service (S3) repository. The raw dataset in compressed XML format will be converted to a readable comma separated value (csv) format in preparation for data cleansing.

The details of the data preparation algorithm are shown on [Appendix III](#).

The dataset, in SAE J2735 message format, include the following: BasicSafetyMessage (BSM), PersonalSafetyMessage (PSM), SignalPhaseAndTiming (SPAT), SignalRequestMessage (SRM), SignalStatusMessage (SSM), and TravelerInformationMessage (TIM).

Other possible data sources are Sample MAP and SPaT datasets from the California Connected Vehicle Testbed which can be found in the URL:

<https://www.caconnectedvehicletestbed.org/datasample>. Additional datasets can be found in [Appendix V](#).

2.2.2 Data Cleansing

After data ingestion, the result is trimmed and cleansed. Data attributes that are of interest are the following:

- i. Latitude
- ii. Longitude
- iii. Heading
- iv. Angle
- v. Timestamp¹
- vi. Vehicle_ID
- vii. Speed_mph
- viii. Transmission Status
- ix. Longitudinal Acceleration
- x. Elevation
- xi. Vehicle Length
- xii. Vehicle Width
- xiii. Brake System Status

2.2.3 Data Mutation

The preprocessed data will be augmented with malicious data that will affect the normal operation of a vehicle. These synthetic anomalous data will be randomly injected to simulate cyber threats and attacks on the RSU data.

Also, in this stage, the data is analyzed, statistics are collected, missing values are checked, quantiles are calculated, and data correlations are identified. These tasks will be initiated with the

¹ The BSM CoreData SecMark is converted into TimeStamp in YYYY-MM-DD HH:MM:SS.sss format. See Appendix VII for the conversion algorithm implemented as a Python code snippet.

SageMaker Studio and a hosted Jupyter notebook environment. Data analysis will be done using an open-source Python data analysis and manipulation tool, Panda (<https://pandas.pydata.org>).

2.2.4 Data Validation and Visualization

In order to be able to build a quality model, quality data must be provided as input. Unit tests on datasets must be run to ensure data quality expectations. This data validation process will be accomplished using AWS Deequ (<https://aws.amazon.com/blogs/big-data/test-data-quality-at-scale-with-deequ/0>) and AWS Glue Data Quality (<https://aws.amazon.com/glue/features/data-quality/>).

Data visualization could provide additional insight into the datasets. Dashboard-style data visualization will be implemented using AWS QuickSight (<https://aws.amazon.com/quicksight/>).

2.2.5 Training Data Selection

The training data selection will be accomplished using SageMaker Autopilot

(https://aws.amazon.com/sagemaker/autopilot/?nc2=type_a&sagemaker-data-wrangler-whats-new.sort-by=item.additionalFields.postDateTime&sagemaker-data-wrangler-whats-new.sort-order=desc). The S3 data repository will be provided to the SageMaker Autopilot for model training and validation. The SageMaker Autopilot generates code to transparently execute a set of model pipelines running on different algorithms.

2.2.6 Model Training

Model training with SageMaker Autopilot entails selecting an algorithm to train with training feature set and verifying that model code and algorithm are suitable in classifying the dataset as normal or malicious.

2.2.7 Model Tuning

The ML model tuning involves hyper-parameter tuning and evaluation of model performance against the validation feature set. This is an iterative process of incrementing the dataset or hyper-parameter tweaking until the desired results are achieved on the test feature set.

2.2.8 Deployment

The VSML model will be deployed via SageMaker Endpoints. The model will provide an online and real-time prediction of the type of input data: normal or malicious.

2.2.9 Monitoring

The deployed VSML system will be continuously monitored for performance degradation after its deployment and uninterrupted utilization.

This VSML workflow is partially implemented using Machine Learning Systems on Matlab™ to illustrate the viability of a more extensive study. The results of the Matlab™ implementations are shown on [Appendix VI](#).

2.3 Hosting

The system will be hosted inside an AWS SageMaker and Jupyter instance reachable at <https://XX.XX.XX.XX>. This IP address will be changed with a more user-friendly name once the domain name is decided.

2.3 User Classes and Characteristics

User Class	Characteristics
Data Engineer	This user is responsible for data collection, analysis, assessment, and ingestion for the Machine Learning (ML) System
Data Scientist	This user is responsible for exploring data, designing ML algorithms, building ML models, wrangling data trends for the ML system, and validating the ML system.
Ordinary User	This user will be able to input a BSM data instance and to receive a prediction on whether it is normal or malicious
ML Engineer	This user is responsible for deploying, monitoring, and fine-tuning the VSML System.

2.4 Operating Environment

The VSML operating environment is defined by the following:

OE-1: The VSML shall run within an Amazon Web Services (AWS) SageMaker.

OE-2: The VSML shall utilize AWS Simple Storage Service (S3) for its data repository.

OE-3: The VSML shall utilize the Jupyter Notebook development environment for interactive development.

OE-4: The VTME shall utilize the Python language as a primary development language.

OE-5: The VSML shall utilize the SageMaker Autopilot for Machine Learning training, testing, validation, and deployment.

2.5 Design and Implementation Constraints

DIC-1: The VSML shall be developed using AWS Sagemaker's Autopilot and Jupyter Notebook

DIC-2: The VSML shall be developed using the Python programming language.

DIC-3: The VSML will be constrained by the limitations of the RSU data source.

DIC-4: The VSML will be designed and implemented as a working prototype capable of future expansion.

DIC-5: The initial iterations of the VSML will be limited to the capability of the AWS SageMaker Autopilot application.

2.6 Assumptions and Dependencies

Assumptions and dependencies for the VSML implementation include the following:

ASS-1: The VSML assumes the availability of the RSU datasets in an AWS S3 bucket repository.

ASS-2: The VSML assumes the availability of Machine Learning (ML) algorithm implementations in SageMaker.

ASS-3: The VSML assumes the ability of sharing the Jupyter notebooks created by SageMaker .

3.0 Interface Requirements

The VSML System will require interfacing with the user. These interface requirements are described in the following.

IR-1: Jupyter Notebook

Description and Priority

The user interface for VSML will be provided with Jupyter notebooks that are derived from Sagemaker Autopilot. The notebooks provide an interactive computational environment for developing Python based Machine Learning applications.

Priority: Must Have

IR-2: Data Visualization

Description and Priority

The VSML System will provide a dataset visualization capability to enable visual analytics.

Priority: Must Have

IR-3: Performance Metrics

Description and Priority

The VSML System will provide an interface for displaying the system performance metrics defined in [section 9](#). These include, but are not limited to, Confusion Matrices, Receiver Operating Characteristics (ROC), and a summary table of results.

Priority: Must Have

IR-4: Data Input

Description and Priority

The VSML System will provide an interface for data input to the AWS S3 bucket.

Priority: Must Have

4.0 Functional Requirements

FR-1: Login

Description and Priority

The VSML System will provide the user the ability to log in to the system given the proper credentials.

Priority: Must Have

FR-2: Data Repository

Description and Priority

The VSML System will provide a repository of datasets on an AWS S3 bucket.

Priority: Must Have

FR-2.1. Data Entry

The VSML System will allow for the entry of vehicle-related data taken from various sources. The specific data and formatting can be found under [section 2.2.1](#).

FR-2.2. Data Cleansing and Transformation

The VSML System will provide for the cleansing of data out of non-essential attributes. The attributes of interest are listed under [section 2.2.2](#). The VSML will also enable data transformation, i.e., the conversion of data to the required format. The data can then be analyzed and used by the system.

FR-2.3. Malicious Data Injection

The VSML System will allow for the injection of malicious data for training, validation and testing purposes. The data mutation specifications are shown in [Appendix IV](#).

FR-3: Machine Learning Models

The VSML System will use machine learning to distinguish between normal data and malicious data. The VSML will utilize various machine learning models to determine the best fitting model for the vehicle dataset.

Priority: Must Have

FR-3.1: Model Training and Validation

The VSML System will use the cleansed and transformed normal data and augmented with malicious data, the aggregated dataset, to train and validate the machine learning system to distinguish normal data from malicious data. This will be done through Sagemaker Autopilot, as described in sections 2.2.4-2.2.6. The performance metrics defined in [Section 9.0](#) will be used to validate the trained Machine Learning models.

FR-3.2: Model Testing

The VSML System will test the validated Machine Learning models utilizing the performance metrics defined in Section 9.0. Ten percent (10%) of the aggregated dataset will be used for this purpose.

FR-4: System Deployment

The VSML System will be deployed for real-time utilization on vehicle BSM CoreData.

Priority: Must Have

5.0 Non-Functional Requirements

Non-functional requirements for the VSML are system attributes that are desired but not required.

The following are the non-functional requirements for the VSML:

NF-1: Portability

The VSML system will be usable across multiple devices, with user being able login from any device with the proper credentials. Additionally, it will be usable with multiple datasets. SageMaker Autopilot runs on Python code that works well with multiple datasets; it only requires minor tweaking to change between datasets.

NF-2: Usability

The development team will attempt to satisfy system usability features such as navigation, performance quality, and intuitiveness of interfaces.

NF-3: Speed

The development team will attempt to enhance the VSML system responsiveness to user interactions.

6.0 Quality Attributes

The VSML is a proof of concept and not meant for a production release. As such, traditional quality attributes such as availability, security, robustness, etc. are not as relevant.

7.0 Source Code Repository and Version Control Requirement

The development team shall facilitate a source code repository and version control for the project.

SC-1: Source Code Repository and Control

The development team shall maintain a source code repository and version control on GitHub. The GitHub project URL is <https://github.com/UWF-CfC-FDOT/VSML>

8.0 Requirements Table

Requirement ID	Type	Name	Description	Priority
B1	Business Rule	No business constraint	There is no business rule associated with this system	Out of Scope
D1	Design	Design and Implementation	The project team shall provide the system design document	Must Have
DA1	Data	Data Specification	The project team shall provide the ML data specification	Must Have
E1.1	Feature	Visual Display	The system shall provide a visualization of requested metrics	Must Have
E1.2	Feature	Code Repository	The project team shall maintain a Git repository for all source code	Must Have
F1	Functional	Login	The system shall provide an access and authentication mechanism	Must Have
F2.1	Functional	Data Entry	The system shall provide a mechanism for entering vehicle-related data.	Must Have
F2.2	Functional	Data cleansing	The system shall provide a mechanism for data cleansing	Must Have
F2.3	Functional	Data mutation	The project team shall provide a mechanism for creating and injecting mutated data	Must Have
F3.1	Functional	Model Training	The system will train the machine learning model to distinguish data.	Must Have
F3.2	Functional	Model Testing	The system shall test the machine learning model with defined performance metrics.	Must Have
F4	Functional	Data ingestion	The project team shall provide a mechanism for data ingestion	Must Have
I1	Interface	User-System interaction	The system shall provide an interface between the user and the ML system	Must Have
I2	Interface	User-ML output Interaction	The system shall provide an interface between the user and the ML output	Must Have
IR1	Interface	Jupyter Notebook	The project team shall provide Jupyter Notebooks from Sagemaker Autopilot	Must Have
IR2	Interface	Data Visualization	The system shall provide visual data analytics	Must Have
IR3	Interface	Performance Metrics	The system shall provide an interface for performance metrics.	Must Have

IR4	Interface	Data Input	The system shall provide an interface for inputting data.	Must Have
NR1	Non-functional	Portability	The project team will attempt to make the system usable across multiple devices and datasets.	Could Have
NR2	Non-functional	Usability and Performance	The project team will attempt to perform system usability and performance assessment	Could Have
NR3	Non-functional	Speed	The project team will attempt to make the system quickly responsive.	Could Have
P1	Policy and Regulations	Policy	The system is not constrained by any State of Federal policy or regulation	Out of Scope
R1	Report	Process Document	The PI shall document the process and submit a report on system development	Must Have
S1	Scope	System Scope	Defines the scope of the system	Must Have
T1	Test	Test Cases	The PI shall provide a documentation of all conducted ML system testing activities	Must Have
TR1	Training	User Training	The system is a prototype and can be demonstrated in an online meeting	Out of scope

9.0 Performance Indicators

9.1 Definitions

The following terms are used to describe the ML performance indicators.

True Positives (TP). These are cases which the system correctly predicted that it belongs to the class.

False Positives (FP). These are cases which the system predicted that it belongs to the class but, in fact, it does not. These are also known as Type I errors.

True Negatives (TN). These are cases which the system correctly predicted that it does not belong to the class.

False Negatives (FN). These are cases which the system predicted that it does not belong to the class but, in fact, it does. These are also known as Type II errors.

9.2 Recall

The proportion of actual positives that are correctly classified. It is formally defined as

$$Recall = \frac{TP}{TP + FN}$$

9.3 Precision

The proportion of positive predictions as truly positive. It is formally defined as

$$Precision = \frac{TP}{TP + FP}$$

Note that with high recall and low precision, there are few data samples that are classified as false negative while, at the same time, there are more data samples classified as false positive. With low recall and high precision, there are more data samples that classified as false negative and, at the same time, there are less data samples that are classified as false positive.

9.4 Accuracy

The proportion of positive and negative predictions that are correctly classified. It is a measure of the ratio of the correctly classified data to the entire data population. It is formally defined as

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

9.5 F-Measure

The harmonic mean of precision and recall. It is formally defined as

$$F - measure = \frac{2 * Recall * Precision}{Recall + Precision}$$

The F-measure is a representative of the Recall and Precision measures and uses the harmonic mean instead of the arithmetic mean. This implies that the F-measure is biased to the lower value of either the Precision or Recall.

10.0 REFERENCES

Wyoming Department of Transportation. (2019, Updated daily). Wyoming CV Pilot Basic Safety Message One Day Sample. [Dataset]. Provided by ITS DataHub through Data.transportation.gov. Accessed 2022-07-01 from <http://doi.org/10.21949/1504479>"

SAE J2735 Surface Vehicle Standard V2X Communications Message Set Dictionary. July, 2020. Website: <https://my.sae.org/servlets/collectionstore/downloadMyItem.do>

F. Maggi, "A Vulnerability in Modern Automotive Standards and How We Exploited It," Trend Micro, 2017.

C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2007.

G. A. Francia, "Connected Vehicle Security," in 15th International Conference on Cyber Warfare and Security (ICCWS 2020), Norfolk, VA, 2020.

S. Kumar, SinghK., S. Kumar, O. Kaiwartya, Y. Cao and H. Zhao, "Delimitated Anti Jammer Scheme for Internet of Vehicle: Machine Learning Based Security Approach," *IEEE Access*, vol. 7, pp. 113311-113323, 2019.

G. De La Torre, P. Rad and K. R. Choo, "Driverless Vehicle Security: Challenges and Future Research Opportunities," *Future Generation Computer Systems*, 2017.

M. L. Han, B. I. Kwak and H. K. Kim, "Anomaly intrusion detection method for vehicular networks based on survival analysis," *Vehicular Communications*, vol. 14, pp. 52-63, 2018.

Appendix I

Connected Vehicle (CV) Data²

A publicly available CV data can be found in the [ITS DataHub](#). It includes Basic Safety Messages (BSM), Traveler Information Messages (TIM) and Signal Phase and Timing (SPaT) messages which can be transmitted via dedicated short-range communications (DSRC). A brief description of each is provided in [US DoT website](#) and recapped in the following:

Basic Safety Messages (BSM)

Basic Safety Messages (BSM) are packets of data that contain information about vehicle position, heading, speed, and other information relating to a vehicle's state and predicted path. These data are received by other vehicles via Vehicle-to-Vehicle (V2V) or Vehicle-to-Infrastructure (V2I) or Vehicle-to-Roadside (V2R) communications through Roadside Units (RSUs) to help determine immediate threats and alert drivers as necessary. This study on the application of machine learning for connected vehicle security will focus on this type of messages.

Traveler Information Messages (TIM)

The Traveler Information Messages (TIM) provide location-based travel advisory information to provide drivers with situational awareness related to traffic information, traffic incidents, major events, evacuations and more. These messages utilize Vehicle-to-Infrastructure (V2I) or Vehicle-to-Roadside(V2R) communications to send messages between vehicles and roadside units (RSUs).

Signal Phase and Timing (SPaT)

The Signal Phase and Timing (SPaT) messages provide information about the current state of all lanes and signal phases at an intersection and other pieces of information. SPaT messages are exchanged between traffic controllers at intersections and vehicles via V2I communications utilizing RSUs to support driver/vehicle decision-making.

² Source: <https://data.transportation.gov/stories/s/Connected-Vehicle-Pilot-Sandbox/hr8h-ufhq#about-the-data>

Appendix II

BSM CoreData

The BSM CoreData that are utilized in this Machine Learning project are adapted from the *DSRC Implementation Guide for Users of SAE J2735 message sets over DSRC*³. These core data consist of the following:

Acceleration:

Data Type: System Defined (in units of 0.01 m/sec²)

acceleration_set_4_way :

long accel: integer, (acceleration along the X-axis or the direction of travel; negative value indicates braking action)

lat accel: integer, (acceleration along the Y-axis or the direction of travel; negative value indicates left turning action, positive indicates right-turning)

vert accel: one-byte signed integer, (-127 represents unavailable data)

yaw: integer (vehicle rotation about the vertical axis and expressed in 0.01 degrees/second)

Angle: Steering Wheel Angle (units of 1.5 degree)

Data Type: signed Integer (range: -189 to +189)

0x01 = 00 = +1.5 degree

0x81 = -126 = -189 degree and beyond

0x7E = +126 = +189 degree and beyond

0x7F = +127 to be used for unavailable

Brake System Status: 2-octet information about the current brake system of the vehicle

Data Type: System Defined

brakeAppliedStatus; (4 bits total--one bit for each wheel, value 1 means active; Thus, 0000 means all Off, 0001 left front active, 0010 left rear active, 0100 right front active, 1000 right rear active)

brakesUnavailableStatus: (5th bit; 1 means true)

sparebit: 6th bit unused; set to 0

traction: (7th and 8th bits) (Traction Control Status)

00-unavailable)

01-off

10-on but not engaged

11-engaged,

abs: (9th and 10th bits) (Anti-lock Brake Status)

³ SAE International DSRC Implementation Guide. A guide to users of SAE J2735 message Sets over DSRC. 2008.

00-unavailable
 01-off
 10-on but not engaged
 11-engaged
 scs: (11th and 12th bit) (Stability Control Status)
 00-unavailable)
 01-off
 10-on or active
 11-unused,
 brakeBoost: (13th and 14th bits) (Brake Boost Applied)
 00-unavailable
 01-off
 10-on but not engaged
 11-engaged
 auxBrakes: (15th and 16th bit) (Auxiliary/Emergency Brake Status)
 00-unavailable
 01-off
 10-on or engaged
 11-unused

Elevation (unit is 10 cm)

Data Type: Integer
 Elevations from 0 to 6143.9 (0x0000 to 0xEFFF) meters
 Elevations from -409.5 to -0.1 (0xF001 to 0xFFFF) meters
 Unknown value is encoded as 0xF000

Heading (Orientation of the front of the vehicle). Represents 0.0125 degrees from the North.

Data Type: 2 Octets of unsigned integer. Range 0..28800
 Value of 28800 indicates unavailable.

Latitude (32 bit value representing 1/10th integer microdegrees with reference to the horizontal datum)

Data Type: Integer (range -900000000 to 900000001).
 Provides a range of plus-minus 180 degrees
 900000001 indicates unavailable.

Longitude (32 bit value representing 1/10th integer microdegrees with reference to the horizontal datum)

Data Type: Integer (range -1800000000 to 1800000001) Provides a range of plus-minus 180 degrees
1800000001 indicates unavailable.

Msg Count

Data Type: Non-negative Integer

Msg ID

Data Type: Non-negative Integer

SecMark

Data Type: Non-negative Integer

Speed (Vehicle Speed in units of 0.02 m/s with a range 0...8191 using bits 1 to 13)

Data Type: Non-negative Integer (13 bits of the 2-byte Transmission+Speed)
Use the value 8191 to indicate unavailability.

Transmission (Current state of transmission; occupies bits 14 to 16 of the 2-byte Transmission+Speed)

Data Type: System Defined
000 Neutral
001 Park
010 Forward gear
011 Reverse gear
100, 101, and 110 are unused
111 unavailable

Vehicle ID

Data Type: Non-negative Integer

Vehicle Size (in centimeters)

Data Type: (3 octets)
Width : Non-negative Integer (10 unsigned bit with values 0..1023) (0 when unavailable)
Length : Non-negative Integer (14 unsigned bit with values 0..16383) (0 when unavailable)

Note: Integer types are 2 octets.

Appendix III

Data Preparation Algorithm

1. Unzip five compressed Gainesville BSM data files.
2. For each uncompressed Gainesville BSM data file
 - a. Extract the available BSM CoreData
 - b. Cleanse the BSM CoreData by removing duplicate information such as id, speed, lat_long
 - c. Convert the BSM CoreData SecMark into Timestamp using the Python code snippet found in Appendix VII. Augment the dataset with additional CoreData attributes to resemble the following:
 - i. Latitude (G)
 - ii. Longitude (G)
 - iii. Heading (G)
 - iv. Angle (G)
 - v. TimeStamp (G)
 - vi. Veh_ID (G)
 - vii. Speed_mph (G)
 - viii. Transmission Status (default=010 (decimal value 2) for forward gear)
 - ix. Longitudinal Acceleration (S) (default = 0.577)
 - x. Elevation (S) (default=1005 ft)
 - xi. Vehicle Length (S) (default=21 ft; Ford F-150)
 - xii. Vehicle Width (S) default=7 ft; Ford F-150)
 - xiii. Brake System Status (S) (default= decimal value 0)
 - xiv. Normal Flag (value==0)
3. Combine the five datasets. Name this file *Normal_BSM.csv*.
4. Create the malicious dataset (use Normal Flag value==1). Name this file *Malicious_BSM.csv*. See the **Malicious BSM Data Generation Techniques** below.
5. Randomly select records from the *Malicious_BSM.csv* file and inject them into the *Normal_BSM.csv* file, Save this file as *Combined+Normal_Malicious_BSM.csv*.
 - a. Randomly generate an integer, N, between 1 to 20. Use this value as the incremental value from the current position.
 - b. The record on that position will be edited with the malicious BSM CoreData values found in the current *Malicious_BSM.csv* record.
 - c. Repeat steps (a) and (b) until the end of the *Normal_BSM.csv* file is reached.

Keys: **G**-Gainesville data **S**-Synthetic data

Appendix IV

Malicious BSM Data Generation Techniques

1. Speed Anomaly Class

- a. Perform an analysis on the Normal dataset to calculate: min, max, average, and outliers
- b. Randomly generate malicious BSM data by
 1. Using speed data outliers
 2. Using speed values that are increments of 5%, 10%, or 15% of the maximum or decrements of 5%, 10%, or 15% of the minimum
 3. Using maximum speed value with Brake Status “On” for all 4 wheels, i.e. BrakeStatus==15 or 1111

2. Transmission Anomaly Class

- a. Randomly generate a BSM data record with speed==15 mph and Transmission Status==000 (moving with transmission in neutral position)
- b. Randomly generate a BSM data record with speed==25 mph and Transmission Status==001 (1) (moving with transmission in park)
- c. Randomly generate a BSM data record with speed==55 mph and Transmission Status==011 (3) (speeding with reverse gear)

3. Longitudinal Acceleration Anomaly Class

Using the typical acceleration of $17.6 \text{ ft/sec}^2 \times 1 \text{ m}/3.28 \text{ ft} \times [\text{units of } 0.01 \text{ m/sec}^2] = 0.577 \text{ ft/sec}^2$

- a. Randomly generate a BSM data record with lateral acceleration == 0.577 ft/sec^2 and Transmission Status==000 (neutral)
- b. Randomly generate a BSM data record with lateral acceleration == 0.577 ft/sec^2 and Transmission Status==001 (park)
- c. Randomly generate a BSM data record with lateral acceleration == 0.577 ft/sec^2 and Transmission Status==011 (3) (reverse gear)

4. Brake System Anomaly

- a. Randomly generate a BSM data record with brake system status == 1111 (decimal value =15 ; brakes applied) and lateral acceleration == 5.577 ft/sec^2 (Vehicle is speeding while brakes are engaged)
- b. Randomly generate a BSM data record with brake system status == 0000 (decimal value =16; all brakes not engaged), lateral acceleration == 5.577 ft/sec^2 and Transmission Status==011 (3) (reverse gear) (Vehicle is speeding in reverse with all brakes not engaged)

Appendix V

Other Data Sources

Connected Vehicle Datasets

- Tampa CV Testbed: <https://catalog.data.gov/nl/dataset?tags=roadside-equipment-rse>
- California CV Testbed: <https://www.caconnectedvehicletestbed.org/datasample>
- US DOT ITS Connected Vehicle Pilot Sandbox (NY, Tampa, and Wyoming Datasets): <http://usdot-its-cvpilot-publicdata.s3.amazonaws.com/index.html>
- Wyoming CV Pilot Basic Safety Message One Day Sample <https://www.opendatanetwork.com/dataset/data.transportation.gov/9k4m-a3jc>.

Connected Vehicle Data Description

<https://data.transportation.gov/stories/s/Connected-Vehicle-Pilot-Sandbox/hr8h-ufhq>

Tampa-Hilssborough Expressway Authority (THEA) Pilot data. Use the API to extract the data. It has 3 sets of data: BSM, TIM and SPaT.

Here is the BSM data and API description: <https://data.transportation.gov/Automobiles/Tampa-CV-Pilot-Basic-Safety-Message-BSM-Sample/nm7w-nvbm>

Here is for the TIM data: <https://data.transportation.gov/Automobiles/Tampa-CV-Pilot-Traveler-Information-Message-TIM-Sa/in46-gmir>

And the SPaT data: <https://data.transportation.gov/Automobiles/Tampa-CV-Pilot-Signal-Phasing-and-Timing-SPaT-Samp/xn7c-yu2n>

Appendix VI

Preliminary Results of Applied Machine Learning on Vehicle BSM CoreData Anomaly using Matlab™

Machine Learning Models

The Machine Learning (ML) Prototype utilizes six Supervise Learning models to be able to compare as to which model fits the classification problem best. The models are briefly described in the following.

1. Neural Network

The fully connected, feedforward neural network is used for supervised machine learning to classify the dataset into malicious or normal BSM CoreData. It has 2 fully connected layers with size 12 on the first layer and size 10 on the second layer. It uses the non-linear Rectified Linear Unit (ReLU) activation function.

2. Decision Tree

We used decision tree or classification tree in supervised learning to predict the responses to the given dataset. The model essentially creates a tree model in which the decisions start at the root node and descend to the leaf node that contains the predicted response, 0 for normal BSM data and 1 for malicious BSM data.

3. Optimizable Ensemble

A classification ensemble is a predictive model composed of a weighted combination of multiple classification models. The optimizable ensemble model utilizes the Bayesian optimization for supervised learning. The model is predicated on the idea that the combination of multiple classification models increases predictive performance.

4. K-Nearest Neighbor (KNN)

The KNN or k-nearest neighbor algorithm is a supervised learning classifier which uses proximity to make classifications or predictions predetermined responses. The model parameters used in our KNN ML system are 10 for number of neighbors and Euclidean for distance metric calculation.

5. Logistic Regression

Logistic regression models the probability of the response as a function of the predictor values. Our logical regression model uses regularization to reduce the complexity of the prediction function by imposing a penalty on the coefficients of features to overcome overfitting.

6. Support Vector Machine (SVM)

The SVM algorithm for supervised learning finds a hyperplane to best separate data points of one class from those of another. The best hyperplane is that with the largest margin, i.e., the maximum width of the hyperplane that has no interior points.

The Basic Safety Message Dataset

Basic Safety Messages (BSM) are packets of data that contain information about vehicle position, heading, speed, and other information relating to a vehicle's state and predicted path. These data are received by other vehicles via Vehicle-to-Vehicle (V2V) or Vehicle-to-Infrastructure (V2I) or Vehicle-to-Roadside (V2R) communications through Roadside Units (RSUs) to help determine immediate threats and alert drivers as necessary. This study on the application of machine learning for connected vehicle security will focus on this type of messages.

The dataset features include the following:

- b. Latitude
- c. Longitude
- d. Heading
- e. Angle
- f. TimeStamp⁴
- g. Veh_ID
- h. Speed_mph
- i. Transmission Status
- j. Longitudinal Acceleration
- k. Elevation
- l. Vehicle Length
- m. Vehicle Width
- n. Brake System Status
- o. Normal Flag (value==0 for normal; 1 for malicious)

Malicious Data Generation

The malicious BSM data generation consists of the following steps:

1. Create the Transmission Anomaly Class.
 - A. Randomly generate an anomalous BSM data record using one of the following:
 - a. Set speed==15 mph and Transmission Status==000 (moving with transmission in neutral position)
 - b. Set speed==25 mph and Transmission Status==001 (1) (moving with transmission in park)
 - c. Set speed==55 mph and Transmission Status==011 (3) (speeding with reverse gear)
 - B. Save the generated BSM data record in a file titled *Malicious_BSM.csv*.
 - C. Repeat steps A and B until 500 malicious BSM data records are created.
 - D. Save and close the *Malicious_BSM.csv* file.

⁴ Refer to Appendix VII for the conversion of BSM CoreData to Timestamp.

Malicious BSM Data Injection

Malicious BSM data injection consists of the following steps:

- A. Select a record from the *Malicious_BSM.csv* file and inject them into the *Normal_BSM.csv* file, Save this file as *Combined+Normal_Malicious_BSM.csv*.
- B. Randomly generate an integer, N, between 1 to 20. Use this value as the incremental value from the current position on the *Normal_BSM.csv* .
- C. The record on that position will be edited with the malicious BSM data values found in the current *Malicious_BSM.csv* record.
- D. Repeat steps (A), (B) and (C) until either the end of the *Normal_BSM.csv* file or the *Malicious_BSM.csv* file is reached.
- E. Save the *Normal_BSM.csv* file as *Combined_Normal_Malicious.csv*.

The *Combined_Normal_Malicious.csv* file will be used for the Machine Learning Classification system.

Machine Learning System for BSM CoreData

The dataset attributes and the summary of the ML prototype runs on Matlab™ are shown in the following tables.

Table 1. Dataset Attributes

Validation Observations	Testing Observations	Number of Predictors	Response Classes
2283	253	12	2

Table 2. Summary of ML Validation and Testing

Machine Learning Model	Validation Accuracy, %	Test Accuracy, %
Neural Network	100.0	100.0
Decision Tree	99.7	99.6
Optimizable Ensemble	99.9	99.6
K-Nearest Neighbor (KNN)	99.2	99.2
Logistic Regression	80.0	79.8
Support Vector Machine (SVM)	80.0	79.8

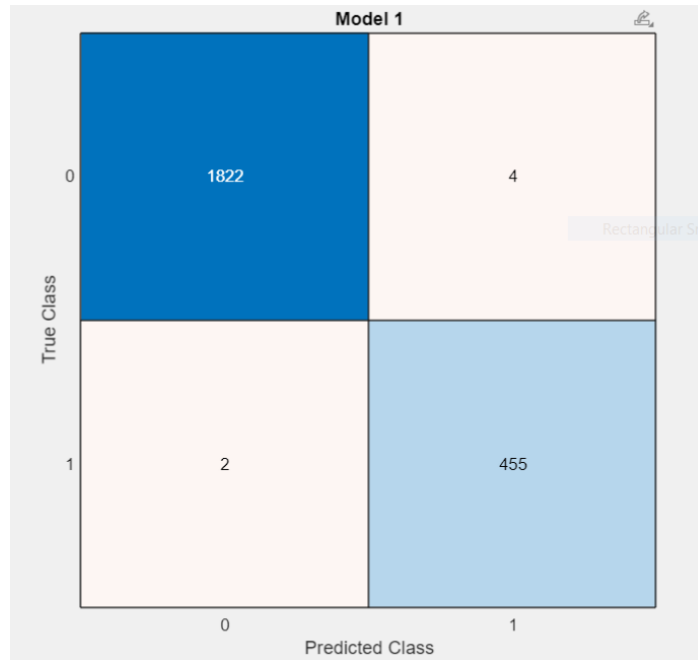


Figure 1. Validation Confusion Matrix of the Fine Tree Model

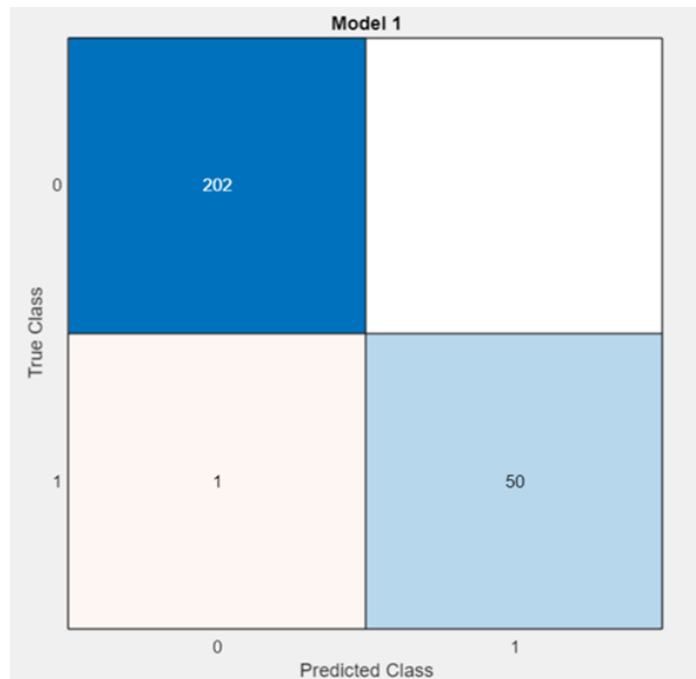


Figure 2. Test Confusion Matrix of the Fine Tree Model

BSM CoreData SecMark data conversion to Timestamp

The following Python code snippet was shared by the Gainesville team. Its purpose is to convert a BSM CoreData SecMark data into Timestamp in YY-MM-DD HH:MM:SS.sss format.

```
def processSecMark(secMark, time_now):
    def getSecMarkTime(seconds):
        def getUTCminute():
            import datetime
            date_time_str = time_now
            utcnow=datetime.datetime.strptime(date_time_str,'%Y-%m-%d_%H:%M:%S')

            utcstr=utcnow.strftime("%Y-%m-%d %H:%M:00")
            utcmin=datetime.datetime.strptime(utcstr, "%Y-%m-%d %H:%M:00")
            return utcmin

        utcminute = getUTCminute()
        delta = datetime.timedelta(milliseconds=seconds)
        utctime = utcminute + delta

        def convertUTCtoLocal(utc):
            from_zone = tz.tzutc()
            to_zone = tz.tzlocal()
            utc = utc.replace(tzinfo=from_zone)
            localtime = utc.astimezone(to_zone)
            return localtime

        localtime = convertUTCtoLocal(utctime)
        return localtime

    ms = secMark
    if (ms == None):
        #basetime = datetime.datetime.now()
        raise Exception("Invalid timestamp")
    else:
        basetime = getSecMarkTime(ms)

    return basetime
```

Software Requirements Specification

for

Vehicle Threat Modeling Engine (VTME)

Technical Report UWF-TR-FDOT-006-01

Version 0.8 approved

Prepared by Guillermo Francia, III

The University of West Florida

Florida Department of Transportation

September 1, 2022

November 4, 2022 (Revision 1)

November 11, 2022 (Revision 2)

November 28, 2022 (Revision 3)

December 3, 2022 (Revision 4)

December 7, 2022 (Revision 5)

Table of Contents

1. Introduction	232
1.1 Purpose	232
1.2 Document Conventions	203
1.3 References.....	203
1.4 Document Revisions Table	203
2. Overview of Product.....	205
2.1 VTME.....	205
2.2 User Classes and Characteristics.....	234
2.3 Operating Environment.....	234
2.4 Design and Implementation Constraints	235
2.5 Assumptions and Dependencies	235
3. Interface Requirements.....	236
3.1 The CVE Data Viewer Interface	236
4. Functional Requirements.....	237
4.1 FR-1: Obtaining Threat Information	237
4.2 FR-2: Mapping Threat Data To Kill Chain.....	238
5. Test Requirements.....	90
5.1 T-1: Unit Tests	90
5.2 T-2: Integration Tests.....	90
5.3 T-3: Test Report.....	90
6. Non-Functional Requirements	90
6.1 NF-1: Portability.....	90
6.2 NF-2: Usability.....	91
6.3 NF-3: Speed	91
7. Quality Attributes.....	239
8. Source Code Repository and Version Control Requirement	91
8.1 SC-1: Source Code Repository and Control.....	91
Appendix A: Requirements Table	240
Appendix B: Requirements Traceability Matrix	243

Appendix C: Glossary..... 245
Appendix D: Lockheed Martin’s Cyber Kill Chain..... 246
Appendix E: OSInt--Cyber Kill Chain Mapping..... 247

Introduction

Purpose

This document describes the software requirements for Version 1.0 of the Vehicle Threat Modeling Engine (VTME) as a subcomponent of the Connected Vehicle Security Metrics and Threat Intelligence Project. This document provides an overview of the users and context of the VTME and covers all functional, non-functional, and data requirements of the VTME.

Document Conventions

This document is based on the IEEE 830 Standards and the Florida Department of Transportation Requirements Standards. Specific conventions used in this document are listed below:

- Priorities are indicated for each feature as well as in the Requirements Table. A green highlighting indicates must have features, while a yellow highlight represents a should have feature.
- Requirements follow the form of <TAG>-#.#.# where a tag indicates a category of requirements. And the # represents the ID of the requirement in a hierarchical fashion.

References

The following references were used in the creation of this document:

- IEEE 830 Standards on Software Requirement Specifications
- UWF Scope of Service Document for the Connected Vehicle Security Metrics and Threat Intelligence Project

Document Revisions Table

Revisor	Revision Date	Reason
Guillermo Francia, III	November 4, 2022	Updated the mapping of threat data with Kill Chain stages
Guillermo Francia, III	November 11, 2022	Updated the Dictionary and Glossary of Terms sections
Guillermo Francia, III	November 28, 2022	Updated the Requirements Table
Guillermo Francia, III	December 3, 2022	Added the Test Requirements
Guillermo Francia, III	December 7, 2022	Added the GitHub information

Overview of Product

VTME

The VTME will use a variety of Open-Source Intelligence (OSINT) data sources for collating realistic vehicle threat intelligence to support threat models. Each cyber threat model will be built based on the stages identified in the Lockheed Martin's Cyber Kill Chain (see Appendix D). Specific Tactics, Techniques and Procedures (TTPs), endemic to connected vehicle systems, will be initially populated from MITRE's Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) matrix.

For each stage of the Kill Chain, the VTME creates a Threat Model using the information gathered from the MITRE ATT&CK Framework, the Common Vulnerability Enumeration (CVE) and the Common Weakness Enumeration (CWE) sources. For instance, for the first stage, Reconnaissance, the system will collect the threat information, the vulnerability, and mitigation associated with that threat and upload it in the threat database. The other stages are weaponization, delivery, exploitation, installation, command and control, and actions on objectives. Note that some of those may not have any information available. Also, that information will be labeled according to which vehicle manufacturer it applies (e.g. Tesla, Honda, Kia, Ford, etc.).

A snapshot of the Project System Architecture is depicted in Figure 2.1.

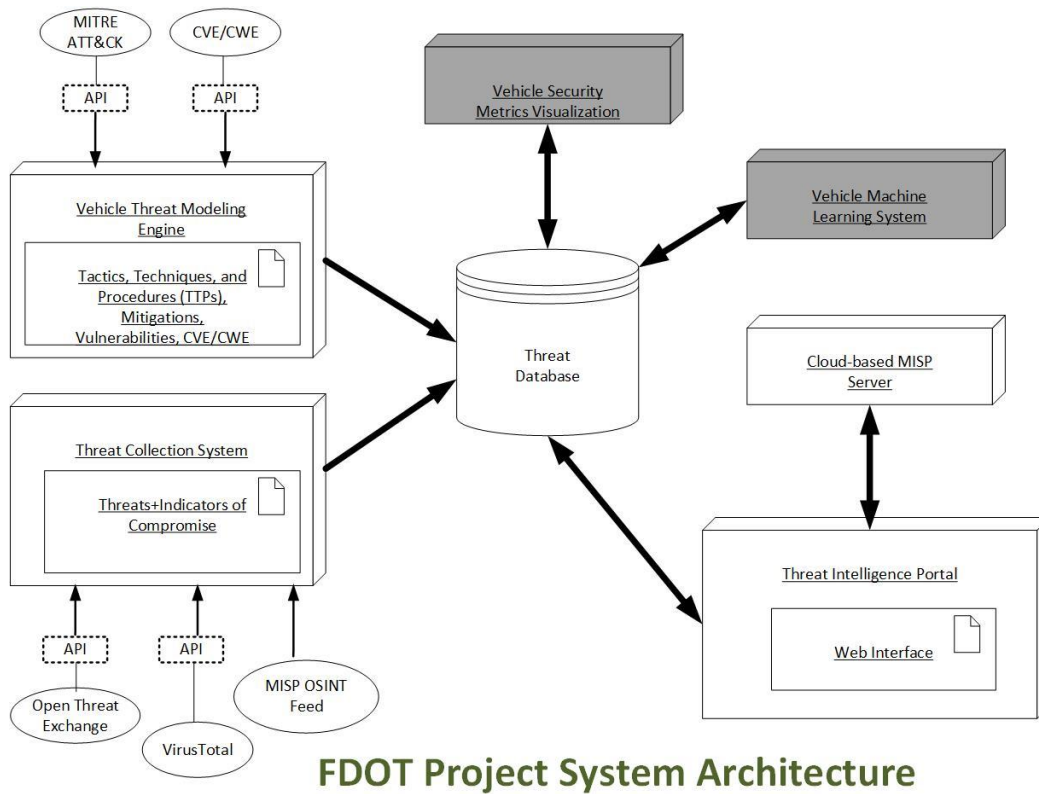


Figure 2.1 The Project System Architecture

User Classes and Characteristics

User Class	Characteristics
Ordinary User	This user will be able to query the Vehicle Threat Database System (VTDBS)
Threat Model Builder	This user will utilize the VTME to build threat models based on the Cyber Kill Chain stages.
Threat Collector	This user will utilize the Vehicle Threat Collection System (VTCS) to retrieve additional information for the models.
Database Administrator	The user responsible for administering and maintaining the VTDBS

Operating Environment

The VTME operating environment is defined by the following:

OE-1: The VTME shall run within an Amazon Elastic Compute Cloud (EC2) Web Service utilizing a Windows Server environment.

OE-2: The VTME shall run as a .NET application on an Internet Information Services (IIS) on a Windows Server within the AWS EC2 instance.

OE-3: The AWS EC2 instance shall be configured with type t2.xlarge having 4 vCPU and 16 GB of memory.

OE-4: The VTME shall interact with a Vehicle Threat Database System (VTDBS) backend. The VTDBS will be designed and implemented as a major deliverable of the project.

OE-5: The VTDBS shall be configured using MS SQL Server 2018.

Design and Implementation Constraints

The VTME design and implementation are constrained by the following:

DIC-1: The VTME shall be developed using Microsoft Visual Studio 2022 or Visual Studio Code

DIC-2: The VTME shall be developed using the C# programming language

DIC-3: The VTME shall be developed using .NET Core

DIC-4: The VTME will be constrained by the limitations of the data source APIs

DIC-5: The VTME will be designed and implemented as a working prototype capable of future expansion

DIC-6: The initial iterations of the VTME will be limited to CVE and CWE data sources

DIC-7: Subsequent iterations of the VTME will include other OSINT such the Open Threat Exchange (OTX), VirusTotal, etc.

Assumptions and Dependencies

Assumptions and dependencies for the VTME implementation include the following:

ASS-1: The VTME assumes the availability of information conduits tapped by APIs from the CVE, CWE, MITRE ATT&CK Framework and OSINT sources.

DEP-1: The VTME shall be dependent on information coming from the MITRE ATT&CK Framework and OSINT sources.

DEP-2: The VTME shall reference a pre-built mapping table of the MITRE ATT&CK Framework to the Cyber Kill Chain.

Interface Requirements

The VTME will require minimal input from the user, however, will need to provide the following information for debugging and tracing purposes:

INT-1: The CVE Data Viewer. The VTME shall provide a Graphical User Interface to be able to periodically collect information from threats using the MITRE ATT&CK and OSINT APIs. This user interface is fully described below.

INT-2: The VTME shall log any interactions with external APIs in a format containing the time, message type, sent request, response code, and response.

INT-3: The VTME shall log all threat database transactions in a format containing the time, query type, query contents, and response.

INT-4: The VTME shall log errors or exceptions in a format containing the time of the event and a stack trace.

The CVE Data Viewer Interface

The CVE Data Viewer User Interface will be used to collect and display data from vehicle manufacturers. Figure 3.1 depicts a prototype of the interface.

The screenshot displays the 'CVE Data Viewer' interface. At the top, there is a title 'CVE Data Viewer'. Below the title, there is a 'Manufacturer' section with a dropdown menu showing 'Honda' and a 'Search' button. Underneath, there is a table with four columns: 'CVE ID', 'CVE Time Stamp', 'CWE ID', and 'Vulnerability Status'. The table contains one row of data: 'CVE-2015-2943', '2022-09-14T20:22Z', 'CWE-295', and 'True'. Below the table, there is a 'Reference URL' section with a text input field containing 'http://jvndb.jvn.jp/en/contents/2015/JVND-2015-000065.html'. Underneath that is a 'Description' section with a text area containing 'Honda Moto LINC 1.6.1 does not verify SSL certificates.' At the bottom of the interface, there are three buttons: 'Back', 'Next', and 'Reset'.

Figure 3.1 CVE Data Viewer Interface

Functional Requirements

There are two major features that the VTME will need to deliver. VTME is a proof of concept and as such, limitations in its implementation will be identified.

FR-1: Obtaining Threat Information

4.1.1 Description and Priority

The VTME needs to collect information from both the MITRE ATT&CK Framework and OSInt sources. There is not currently a mapping of the OSInt sources to the ATT&CK Framework. For the purpose of this proof of concept, a subset of OSInt will be manually mapped on to the Framework and used as the basis for the purposes of this project.

MITRE has developed a mapping for mapping OSInt sources to the framework, but this work is early. Furthermore, while a more advanced approach such as language modeling or machine learning may be applicable, it is outside the scope of the VTME.

Priority: Must Have

4.1.2 Related User Classes

All Users

4.1.3 Functional Requirements

FR-1.1: The VTME shall facilitate access to the MITRE ATT&CK Framework and OSInt APIs to gather threat data

FR-1.2: A CVE Data Viewer shall provide for the querying and displaying of the CVE and CWE repository. The CVE Data Viewer will have the following features:

- Preloaded list of vehicle manufacturers
- A search capability for CVEs for the selected manufacturer
- A display capability of each CVE record found for the selected manufacturer including the CVE ID, CVE Timestamp, CWE ID, Vulnerability Status, the Reference URL for the CVE, and the CVE description.

- The capability to navigate through all the CVE records.

The user interface prototype is depicted in Figure 3.1.

FR-1.3: The VTME shall provide an internal predefined table associating the MITRE® ATT&CK Framework threat data to the Cyber Kill Chain.

FR-2: Mapping Threat Data To Kill Chain

4.2.1 Description and Priority

The VTME will utilize an ontology to map an OSInt and Tactic within the MITRE® ATT&CK framework to stages in the Cyber Kill Chain.

Priority: Must Have

4.2.2 Related User Classes

All Users

4.2.3 Functional Requirements

FR-2.1: The VTME shall map an attack from an OSInt to stages in the Cyber Kill Chain by using an ontology map

FR-2.2: The VTME shall provide a database for the Attack-Kill Chain mapping

Test Requirements

The VTME requires testing and validation of the main application functionalities.

T-1: Unit Tests

Unit system testing shall be conducted for all functional system components. Unit tests shall be integrated and documented in the source code. The GitHub repository is found at this URL:

<https://github.com/UWF-CfC-FDOT/VTMECS>.

T-2: Integration Tests

System integration testing is not within the scope of the VTME system.

T-3: Test Report

An associated documentation of all system testing activities shall be provided. See the attached Unit Test Overview document.

Non-Functional Requirements

Non-functional requirements for the VTME are system attributes that are desired but not required.

The following are the non-functional requirements for the VTME:

NF-1: Portability

The development team will attempt to make the web enabled VTME system portable across multiple computing form factors.

NF-2: Usability

The development team will attempt to satisfy system usability features such as navigation, performance quality, and intuitiveness of interfaces.

NF-3: Speed

The development team will attempt to enhance the VTME system responsiveness to user interactions and database transactions.

Quality Attributes

The VTME is a proof of concept and not meant for a production release. As such, traditional quality attributes such as availability, security, robustness, etc. are not as relevant.

Source Code Repository and Version Control Requirement

The development team shall facilitate a source code repository and version control for the project.

SC-1: Source Code Repository and Control

The development team shall maintain a source code repository and version control on GitHub. The GitHub project URL is <https://github.com/UWF-CfC-FDOT/VTMECS>.

A: Requirements Table

Requirement ID	Requirement Type	Requirement Name	Requirement Description	Priority
S1	Scope	System Scope	Defines the scope of the system	Must have
F1.1	Functional	Gather MITRE ATT&CK threat data	The VTME shall facilitate access to the MITRE ATT&CK Framework and OSInt APIs to gather threat data	Must have
F1.2	Functional	CVE Data Viewer	A CVE Data Viewer shall facilitate the querying and displaying of the CVE and CWE data	Must have
F1.3	Functional	Threat-Kill Chain Table	The VTME will provide an internal predefined table associating the MITRE® ATT&CK Framework threat data to the Cyber Kill Chain	Must have
F2.1	Functional	Threat-Kill Chain Mapping	The VTME will map an attack from an OSInt to stages in the Cyber Kill Chain by using an ontology map	Must have

F2.2	Functional	Threat DBMS	The VTME shall provide a database management system for the MITRE ATT&CK--Kill Chain mapping	Must have
INT -1	Interface	CVE Data Viewer GUI	The VTME shall provide a Graphical User Interface to be able to periodically collect information from threats using the MITRE ATT&CK and OSINT APIs.	Must have
INT-2	Interface	API Interaction Logger	The VTME shall log any interactions with external APIs in a format containing the time, message type, sent request, response code, and response	Must have
INT-3	Interface	DBMS Transaction logger	The VTME shall log all threat database transactions in a format containing the time, query type, query contents, and response	Must have
INT-4	Interface	Error/Exception Logger	The VTME shall log errors or exceptions in a format containing the time of the event and a stack trace	Must have

T-1	Test	Unit Test	Unit system testing shall be conducted for all functional system components	Must have
T-2	Test	Integration Test	System integration testing is not within the scope of the VTME system	Out of scope
T-3	Test	Test Report	An associated documentation of all system test activities shall be provided	Must have
NF-1	Non-functional	Portability	The development team will attempt to make the web enabled VTME system portable across multiple computing form factors	Could have
NF-2	Non-functional	Usability	The development team will attempt to satisfy system usability features such as navigation, performance quality, and intuitiveness of interfaces	Could have
NF-3	Non-functional	Speed	The development team will attempt to enhance the VTME system responsiveness to user interactions and database transactions	Should have
SC-1	Source Code	Source Code Control	The development team shall maintain a source code repository and version control on GitHub	Should have

Appendix B: Requirements Traceability Matrix

Requirement ID	Requirement Description	Test Case	Status
S1	Defines the scope of the system	N/A	N/A
F1.1	The VTME shall facilitate access to the MITRE ATT&CK Framework and OSInt APIs to gather threat data	CVEToAttackTest	Passed
F1.2	A CVE Data Viewer shall facilitate the querying and displaying of the CVE and CWE data	CVEToAttackTest	Passed
F1.3	The VTME will provide an internal predefined table associating the MITRE® ATT&CK Framework threat data to the Cyber Kill Chain	N/A	Completed
F2.1	The VTME will map an attack from an OSInt to stages in the Cyber Kill Chain by using an ontology map	CVEToAttackTest	Passed
F2.2	The VTME shall provide a database management system for the MITRE ATT&CK--Kill Chain mapping	N/A	Completed
INT -1	The VTME shall provide a Graphical User Interface to be able to periodically collect information from threats using the MITRE ATT&CK and OSINT APIs.	CVEViewerTest	Passed
INT-2	The VTME shall log any interactions with external APIs in a format containing the time, message type, sent request, response code, and response	API_LogTest	Passed

INT-3	The VTME shall log all threat database transactions in a format containing the time, query type, query contents, and response	DB_Transaction_LogTest	Passed
INT-4	The VTME shall log errors or exceptions in a format containing the time of the event and a stack trace	Error_LogTest	Passed
T-1	Unit system testing shall be conducted for all functional system components	Multiple Test Cases	Passed
T-2	System integration testing is not within the scope of the VTME system	N/A	N/A
T-3	An associated documentation of all system test activities shall be provided	N/A	Completed
NF-1	The development team will attempt to make the web enabled VTME system portable across multiple computing form factors	N/A	N/A
NF-2	The development team will attempt to satisfy system usability features such as navigation, performance quality, and intuitiveness of interfaces	N/A	N/A
NF-3	The development team will attempt to enhance the VTME system responsiveness to user interactions and database transactions.	N/A	N/A
SC-1	The development team shall maintain a source code repository and version control on GitHub	N/A	N/A

Appendix C: Glossary

Term	Description
API	Application Program Interface
ATT&CK	Adversarial Tactics, Techniques, and Common Knowledge
ATT&CK Framework	A knowledge base of adversary tactics and techniques based on real-world observations.
AWS	Amazon Web Services
CVE	Common Vulnerabilities and Exposures
CWE	Common Weakness Enumeration
Cyber Kill Chain	A model developed by Lockheed Martin® used for the identification and prevention of cyber intrusions.
EC2	Elastic Compute Cloud
GUI	Graphical User Interface
IIS	Internet Information Services
IoC	Indicators of Compromise
.NET	A cross-platform, open-source developer platform created by Microsoft
OSINT	Open-Source Intelligence
OTX	Open Threat Exchange
Tactics, Techniques and Procedures (TTPs)	Activities and methods used by an adversary to carry out a cyber attack
VTCS	Vehicle Threat Collection System
VTDBS	Vehicle Threat Database System
VTME	Vehicle Threat Modeling Engine

Appendix D: Lockheed Martin's Cyber Kill Chain



Figure D.1 Lockheed Martin's Cyber Kill Chain (Source: <https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html>)

Appendix E: OSInt--Cyber Kill Chain Mapping

Based on our proposed system architecture, the Vehicle Threat Modeling Engine (VTME) utilizes the MITRE ATT&CK framework to characterize the impact of a vulnerability as described in the CVEs. ATT&CK tactics and techniques provide an ingenious way to describe the attack vectors and empower defenders and threat hunters to better strategize and model threats by incorporating vulnerabilities in the threat model themselves. In the process, defenders get better equipped in implementing controls if they can have CVEs with ATT&CK tactic and technique references.

As a first step in our methodology, we map ATT&CK tactics and techniques with existing CVEs for connected vehicles. We start with a manual process for this ontology mapping. Our next step is to look up existing vulnerabilities that have been defined in the literature and have associated CVE reference numbers. For each of these vulnerabilities, we researched the attack vector and map it to the relevant stage in the ATT&CK tactics. We delved deeper into the techniques associated with the tactic and mapped the relevant techniques with the CVE. For example, CVE-2022-23126 uses techniques associated with exploiting default credentials (sub-technique T1078.001) for obtaining valid accounts (technique T1078) and using it for further exploitation. As an added step, we mapped the vulnerability with the Cyber Kill Chain stages. The mapping is depicted in the attached document, [Ontology_CVE-Kill_Chain_Mapping.pdf](#).

Appendix X. Technical Report UWF-TR-FDOT-006-02

Threat	Vehicle Type and Model	ATT&CK Tactic	ATT&CK Technique		
				Reconnaissance	Weaponization
CVE-2022-23126 TeslaMate before 1.25.1	Tesla	Initial Access	Valid Accounts (T1078) -> Default Accounts (T1078.001)	Internet connectivity of Tesla vehicle	
CVE-2020-29440 Tesla Model X	Tesla Model X	Credential Access Impact	Steal Application Access Token (T1528) Firmware Corruption (T1495)	Key fob signal sniffing	Cloning of key fob
CVE-2022-37305 Honda Remote Keyless Entry (RKE)	Honda models through 2018	Credential Access Inhibit Response Function	Input Capture (T1056) Block Command Message (T0803)	Remote keyless entry (RKE) signal sniffing	Cloning of RKE key
CVE-2022-27254 CVE-2021-46145 Honda Remote Keyless Entry Replay Attack	Honda Civic 2018	Credential Access	Input Capture (T1056)	Remote keyless system signal sniffing	Cloning of remote key
CVE-2022-37418 Kia, Nissan, Hyundai RemoteKeyless Entry (RKE)	Kia, Nissan, Hyundai models through 2017	Credential Access Inhibit Response Function	Input Capture (T1056) Block Command Message (T0803)	Remote Keyless Entry (RKE) signal sniffing	Cloning of RKE key
CVE-2020-8539 Kia Motors Head Unit with Software version: SOP.003.30.18.0703, SOP.005.7.181019, and SOP.007.1.191209 may allow an	Kia	Defense Evasion	Indirect Command Execution (T1202)	Identify Kia head units with software versions: SOP.003.30.18.0703, SOP.005.7.181019, and	Remote control of head unit

attacker to inject unauthorized commands					
<p>CVE-2018-9322</p> <p>The Head Unit HU_NBT (aka Infotainment) component on BMWi Series, BMW X Series, BMW 3 Series, BMW 5 Series, and BMW 7 Series vehicles produced in 2012 through 2018 allows local attacks involving the USB or OBD-II interface. An attacker can bypass the code-signing protection mechanism for firmware updates, and consequently obtain a root shell.</p>	<p>BMW Series, BMW X Series, BMW 3 Series, BMW 5 Series, and BMW 7 Series vehicles produced in 2012 through 2018</p>	<p>Defense Evasion</p> <p>Credential Access</p> <p>Impact</p>	<p>Indirect Command Execution (T1202)</p>	<p>Firmware Corruption (T1495)</p>	<p>Obtain root shell access</p>
<p>CVE-2017-9647</p> <p>A Stack-Based Buffer Overflow issue was discovered on BMW, Ford, Infiniti, and Nissan.</p> <p>An attacker with a physical connection to the TCU may exploit a buffer overflow condition that exists</p>	<p>Defense Evasion</p>	<p>Indirect Command Execution (T1202)</p>	<p>Identify vehicles with the Continental AG Infineon S-Gold2 (PMB8876) chipset</p>	<p>Remote command execution of arbitrary code on the telematics control module (TCU)</p>	

in the processing of AT commands. This may allow arbitrary code execution on the baseband radio processor of the TCU.					
---	--	--	--	--	--

Kill Chain Phase				
Delivery	Exploitation	Installation	Command & Control	Actions on Objectives
Intrusion into Grafana login system via Internet	Exploitation of Docker configuration		Keyless entry and remote control of vehicle	Partial control of Tesla vehicle
	Exploitation using spoofed key fob			Complete control of Tesla vehicle
	Exploitation using cloned RKE key		Adversary retains control indefinitely unlocking the vehicle after sniffing and cloning 5 consecutive RF signals	Complete control of Honda vehicle
	Exploitation using replay attack		Adversary gains control of unlocking the vehicle	Complete control of the Honda vehicle
	Exploitation using cloned RKE key		Adversary retains the ability to unlock indefinitely	Complete control of the Kia, Nissan, and Hyundai vehicles
Physical access to head unit	Exploitation using micomd executable daemon	Adversary can install malware on the head unit as a third-party application	Inject unauthorized commands into the head unit	Partial control of the Kia vehicle

Physical access to head unit	Exploitation using USB or OBD-II interface	Adversary can bypass code-signing protection mechanism for firmware updates, and obtain a root shell	Unrestricted root shell access to the head unit	Partial control of the BMW vehicle
Physical access to the vehicle	Exploitation using Stack-Based Buffer Overflow	Physical connection to the telematics control module (TCU)	Adversary can execute arbitrary code remotely	Partial control of vehicle

Unit Test Overview for Vehicle Threat Modeling Engine (VTME)

Technical Report UWF-TR-FDOT-006-03

Version 0.1 unapproved

Version 1.0 approved

Prepared by Doug Woodall

Revised by Dr. Guillermo Francia III

**The University of West Florida
Florida Department of Transportation**

December 1, 2022

Revisions: December 4, 2022

Table of Contents

1. Introduction	254
1.1 Purpose	254
2. VTME Test Framework	254
2.1 Framework Description	254
3. VTME Unit Tests	254
3.1 CveToAttackTest	254
3.2 CweTest	255
3.3 DateTimeHelperTest.....	255
4. User Interface Test Framework	256
5. User Interface Unit Tests	256
5.1 CVEViewerTest.....	256
5.2 ManufacturerNotSelectedWhenSubmitted	257
6. Logging Tests	257
6.1 API_LogTest	257
6.2 DB_Transaction_LogTest	257
6.3 Error_LogTest	257

1. Introduction

3. Purpose

This document describes the unit test framework of the Vehicle Threat Modeling Engine (VTME) as a subcomponent of the Connected Vehicle Security Metrics and Threat Intelligence Project. This document provides an overview of unit test framework and tests for the both VTME and the user interface.

4. Document Revisions Table

Revisor	Revision Date	Reason
Woodall, Douglas	December 1, 2022	Initial draft
Guillermo Francia, III	December 4, 2022	Miscellaneous revisions

2. VTME Test Framework

2.1 Framework Description

The VTME project is developed in C# language and the choice was made to use the standard MsTest framework for unit testing the engine. MsTest is a native unit testing library that comes with Visual Studio from Microsoft.

3. VTME Unit Tests

3.1 CveToAttackTest

The CveToAttackTest.cs file contains unit test coverage of the /Library/CveToAttack.cs class. Performs validation of the mapping process from CWE to ATTACK information stored in /Data/CveAttackMap. Validates properties of the CveToAttack object.

3.1.1 CheckValidMap_NotNull

Validates that the CveToAttack object is successfully created from the CveAttackMap JSON file.

Status: Passed

3.1.2 CheckValidMap_VehicleTypeAndModel_NotNullOrEmpty

Validates that the VehicleTypeAndModel property exists as a real value in every CVE mapping entry in the CveAttackMap JSON file. Must be present.

Status: Passed

3.1.3 CheckValidMap_Tactics_GreaterThanZero

Validates that at minimum at least one tactic entry must be present in each CVE mapping entry in the CveAttackMap JSON file. Must be present.

Status: Passed

3.1.4 CheckValidMap_Techniques_GreaterThanZero

Validates that at minimum at least one technique entry must be present in each tactic per each CVE mapping entry in the CveAttackMap JSON file. Must be present.

Status: Passed

3.1.5 CheckValidMap_KillChainPhases_AtLeastOneValid

Validates that at least one kill chain phase has been populated with information in every CVE mapping entry in the CveAttackMap JSON file. Must be present.

Status: Passed

3.2 CweTest

The CweTest.cs file contains unit test coverage of the /Library/Cwe.cs class. This class contains the object that represents a CWE for a given record and includes as parameters the CWE ID and URL for mitre.

3.2.1 ValidCwe_ValidateId

Validates that a provided CWE key at initialization will be properly stored as a property that is publicly accessible.

Status: Passed

3.2.2 ValidCwe_ValidateUrl

Validates that a provided CWE key at initialization will be properly converted into a URL of the convention <https://cwe.mitre.org/data/definitions/{key}.html> and stored as a property that is publicly accessible.

Status: Passed

3.3 DateTimeHelperTest

The DateTimeHelperTest.cs file contains unit test coverage of the /Library/DateTimeHelper.cs class. This class contains the helper methods for validating that the provided start and stop query times for the CVE search are valid according to the NVD API requirements.

3.3.1 CheckDateFormatValid

Verifies that provided times conform to ISO-8061 datetime format.

Status: Passed

3.3.2 CheckDateFormat_HalfInvalid

Verifies that validity checks fail if only one of the time inputs (start or stop) are properly ISO-8061 formatted.

Status: Passed

3.3.3 CheckDateFormat_RangeTooLarge

Verifies that validity checks fail if the calculated date range is greater than 120 days (an NVD API requirements).

Status: Passed

3.3.4 CheckDateFormat_EndBeforeStart

Verifies that validity checks fail if the provided end date is prior in time to the provided start date.

Status: Passed

3.3.5 CheckDateFormat_EmptyString

Verifies that validity checks fail if there is no provided start or stop date.

Status: Passed

4. User Interface Test Framework

The VTIP test interface project is developed in C# language and BUnit framework was chosen for unit testing the interface.

5. User Interface Unit Tests

5.1 CVEViewerTest

Verifies the page has rendered correctly.

Status: Passed

5.2 ManufacturerNotSelectedWhenSubmitted

Verifies that one of the listed manufacturers has to be selected when submit the form.

Status: Passed

6. Logging Tests

6.1 API_LogTest

Verifies the API transactions are logged

Status: Passed

6.2 DB_Transaction_LogTest

Verifies the DB transactions are logged

Status: Passed

6.3 Error_LogTest

Verifies the web error events are logged

Status: Passed

Software Requirements Specification

for

Vehicle Threat Collection System (VTCS)

Technical Report UWF-TR-FDOT-007-01

Version 0.4 approved

Prepared by Daniel Miller

Approved by Dr. Guillermo Francia, III

The University of West Florida

Florida Department of Transportation

April 3, 2023 (Initial)

June 1, 2023 (Revision 0.1)

June 9, 2023 (Revision 0.2)

July 3, 2023 (Revision 0.3)

July 18, 2023 (Revision 0.4)

Table of Contents

1. Introduction	261
1.1 Purpose	261
1.2 Document Conventions	261
1.3 References	261
1.4 Document Revisions Table	261
2. Overview of Product	263
2.1 VTCS	263
2.2 User Classes and Characteristics	264
2.3 Operating Environment	265
2.4 Design and Implementation Constraints	265
2.5 Assumptions and Dependencies	266
3. Interface Requirements	266
3.1 The VTCS Query Interface	266
3.2 The VTCS Record Viewer Interface	267
4. Functional Requirements	268
4.1 FR-1: Obtain Threat Information	268
4.2 FR-2: Store Threat Information	270
5. Test Requirements	271
5.1 T-1: Unit Tests	271
5.2 T-2: Integration Tests	271
5.3 T-3: Test Report	271
6. Non-Functional Requirements	271
6.1 NF-1: Portability	271
6.2 NF-2: Usability	271
6.3 NF-3: Speed	271
7. Quality Attributes	271
8. Source Code Repository and Version Control Requirement	271
8.1 SC-1: Source Code Repository and Control	272
9. Appendix A: Requirements Table	273
10. Appendix B: Requirements Traceability Matrix	276
	259

Introduction

Purpose

This document describes the software requirements for Version 1.0 of the Vehicle Threat Collection System (VTCS) as a subcomponent of the Connected Vehicle Security Metrics and Threat Intelligence Project. This document provides an overview of the users and context of the VTCS and covers all functional, non-functional, interface, and data requirements of the VTCS.

Document Conventions

This document is based on the IEEE 830-1998 Standards and the Florida Department of Transportation Requirements Standards. Specific conventions used in this document are listed below:

- Priorities are indicated for each feature as well as in the Requirements Table. A green highlighting indicates must have features, while a yellow highlight represents a should have feature.
- Requirements follow the form of <TAG>-#.#.# where a tag indicates a category of requirements. And the # represents the ID of the requirement in a hierarchical fashion.

References

The following references were used in the creation of this document:

- IEEE 830-1998 Standards on Software Requirement Specifications
- UWF Scope of Service Document for the Connected Vehicle Security Metrics and Threat Intelligence Project

Document Revisions Table

Revisor	Revision Date	Reason
Guillermo Francia, III	July 18, 2023	Revised Section 8.1 for consistency with Appendix A.
Guillermo Francia, III	July 3, 2023	Revised section 2.4 to reflect the specific elements developed by DIC-1 and DIC-2. Revised section 2.5 to refer to CVE and CWE instead off

		<p>OTX and VirusTotal as the data sources for the initial iteration.</p> <p>Revised SC-1 as a “should have.”</p> <p>Revised Appendix B to include applicable test cases for each functional requirement.</p>
Daniel Miller	June 9, 2023	Added detailed descriptions to UI function controls sections 3.1 and 3.2
Daniel Miller	June 8, 2023	proof-read/minor edits to document before submission
Daniel Miller	June 7, 2023	Updated the Requirements Tables
Daniel Miller	June 7, 2023	Updated Section 3.1 and added 3.2 to more accurately reflect VTCS GUI requirements to include screenshots of the GUI.
Daniel Miller	May 25, 2023	Updated Section 4 to more accurately reflect VTCS requirements
Daniel Miller	May 20, 2023	Updated Section 2 to more accurately reflect VTCS requirements
Daniel Miller	May 14, 2023	Updated Section 4 to more accurately reflect VTCS requirements

Daniel Miller	April 28, 2023	Updated Sections 4-Appendix of VTCS SRS document by removing nonrelated information and inserting draft explanations of what is required for each respective section.
Daniel Miller	April 14, 2023	Updated Sections 1-3 of VTCS SRS document by removing nonrelated information and inserting draft explanations of what is required for each respective section.
Guillermo Francia, III	November 4, 2022	Updated the mapping of threat data with Kill Chain stages
Guillermo Francia, III	November 11, 2022	Updated the Dictionary and Glossary of Terms sections
Guillermo Francia, III	November 28, 2022	Updated the Requirements Table
Guillermo Francia, III	December 3, 2022	Added the Test Requirements
Guillermo Francia, III	December 7, 2022	Added the GitHub information

Overview of Product

VTCS

The VTCS is an automated system that collects threat intelligence feeds from various sources and stores that data to the Vehicle Threat Database System (VTDBS) for ingestion and processing by other subcomponents of the Project System. Threat Intelligence data will come from publicly available Open-Source Intelligence (OSINT) sites such as Open Threat Exchange (OTX) and VirusTotal. Configurable, automated queries to these sources will generate tailored threat intelligence

feeds and provide any associated Common Vulnerabilities and Exposures (CVEs), Common Weaknesses Enumerations (CWEs) and any other relevant Indicators of Compromise (IoCs).

A user will provide one or more vehicle’s year, make, and/or model as search parameters for the VTCS. The system will then generate individual records containing IOCs associated with cyber-attacks and/or known system vulnerabilities correlated to specific makes, models, and versions of vulnerable devices in connected vehicles.

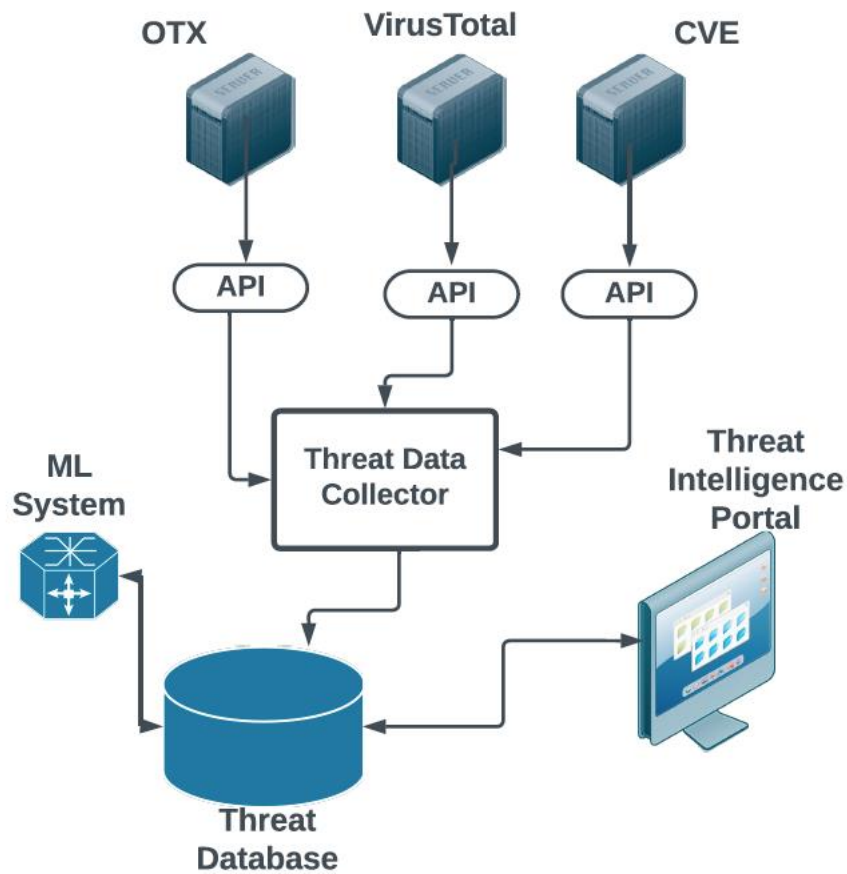


Figure 2.1 The Project System Architecture

User Classes and Characteristics

User Class	Characteristics
Ordinary User	This user will be able to query the VTDBS

Threat Model Builder	This user will utilize the VTME to build threat models based on the Cyber Kill Chain stages.
Threat Collector	This user will utilize the VTCS to retrieve additional information for the models.
Database Administrator	The user responsible for administering and maintaining the VTDBS

Operating Environment

The VTCS operating environment is defined by the following:

OE-1: The VTCS shall run within an Amazon Elastic Compute Cloud (EC2) Web Service utilizing a Windows Server environment.

OE-2: The VTCS shall run as a .NET application on an Internet Information Services (IIS) on a Windows Server within the AWS EC2 instance.

OE-3: The AWS EC2 instance shall be configured with type t2.xlarge having 4 vCPU and 16 GB of memory.

OE-4: The VTCS shall interact with the VTDBS backend. The VTDBS will be designed and implemented as a major deliverable of the project.

OE-5: The VTDBS shall be configured using MS SQL Server 2018.

Design and Implementation Constraints

The VTCS design and implementation are constrained by the following:

DIC-1: The Web GUI of the VTCS shall be developed using Microsoft Visual Studio 2022 or Visual Studio Code.

DIC-2: The backend functional implementations of the VTCS shall be developed using the C# programming language.

DIC-3: The VTCS will be constrained by the limitations of the data source APIs.

DIC-4: The VTCS will be designed and implemented as a working prototype capable of future expansion.

DIC-5: The initial iterations of the VTCS will be limited to CVE and CWE data sources.

DIC-6: Subsequent iterations of the VTCS will include other OSINT such as Open Threat Exchange (OTX), VirusTotal, etc.

Assumptions and Dependencies

Assumptions and dependencies for the VTCS implementation include the following:

ASS-1: The VTCS assumes the availability of information conduits tapped by APIs from the OTX and VirusTotal OSINT sources.

DEP-1: The VTCS shall be dependent on information coming from the CVE and CWE data sources. Subsequent iterations shall be dependent on OTX and VirusTotal OSINT sources.

DEP-2: The VTCS shall reference the VTME's pre-built mapping table of the MITRE ATT&CK Framework to the Cyber Kill Chain to configure how data is parsed to the VTDBS.

Interface Requirements

The VTCS will require vehicle search parameters as input from the user in order to generate records for the VTDBS. Additionally, the system will require input from the user as verification before a record is saved to the VTDBS.

INT-1: The VTCS shall provide a Graphical User Interface for users to submit vehicle search parameters. This user interface is fully described in section 3.1.

INT-2: The VTCS shall provide a Graphical User Interface for users to manage query results and update the VTDBS. This user interface is fully described in section 3.2.

INT-3: The VTCS shall log any interactions with external APIs in a format containing the time, message type, sent request, response code, and response.

INT-4: The VTCS shall log all threat database transactions in a format containing the time, query type, query contents, and response.

INT-5: The VTCS shall log errors or exceptions in a format containing the time of the event and a stack trace.

The VTCS Query Interface

The Query Interface will be used to provide the VTCS with vehicle search parameters. Figure 3.1 depicts a prototype of the interface where the user can submit one or more vehicles for processing. Using the *Threat Source* drop-down selection box, the user is provided an option to select a specific

threat intelligence source such as OTX, VirusTotal, or ALL in an attempt to query all sources. The user can also filter the information to collect by specifying the *Year*, *Make*, and *Model* of the vehicle through drop-down selection boxes. The *Add* button will take the options selected by the user in the *Year*, *Make*, and *Model* selections and place the vehicle into the Query Interfaces *Vehicle Selection* table at the bottom of the page. Once a vehicle is added to the selection table the *Year*, *Make*, and *Model* drop-down selection boxes are reset to their default option and the user can add another vehicle. When the user is satisfied with the data source and vehicle(s) selected the *Search* button will initiate the VTCS to run the specified user query using the API of the selected open threat intelligence source. Clicking the *Search* button will also bring the user to the *Record Viewer Interface* (Fig. 3.2).

Vehicle Threat Collection System (VTCS)

Query Page

Threat Source

OTX

Vehicle Selection

Year Make Model

2021 Toyota RAV4
2019 Ford Explorer

Add

Search

Figure 3.1 CVE Data Viewer Interface Prototype

The VTCS Record Viewer Interface

The Record Viewer Interface will be used to display and manage the records generated by the VTCS search parameters. Figure 3.2 depicts a prototype of the interface where the user can review the details of individual records, save records to the VTDBS, and navigate to other records generated by the query. When first populated, the Record Viewer will focus on the first record of the search results.

The Record Viewer Interface is broken into two main sections; a vehicle threat record details on the top-half and a set of vehicle threat records on the bottom-half.

The top-half of the interface contains individual record details which display the title of the record, a detailed summary of the record, a table of meta-data information concerning the record, and the *Save Selected Record* and *Save All Records* buttons. Metadata information for the table include date reported, date last updated, list of associated CVE's, list of Associated CWE's, list of data-sources retrieved from, list of IoC's, and the record's CVSS score and severity level. The *Save Selected Record* and *Save All Records* buttons, when selected, will submit appropriate SQL queries to the VTDBS to update the database with the respective record(s). The *Save Selected Record* button will have the VTCS submit the selected record to the VTDBS and the *Save All Records* will submit all records to the VTDBS. Once the records are saved to the VTDBS a success dialog will appear for the user. In the event the VTCS is unsuccessful in saving records to the VTDBS an error dialog will appear with a transaction ID for further research by database administrators. This transaction ID will be the date-time-group of the errored SQL query submitted by the VTCS to the VTDBS,

The bottom-half of the Record Viewer is a table that allows users to navigate to all other records generated by the query. The *Generated Records* table displays the record title, data source hits, CVE hits, CWE hits, number of IoCs and the record's CVSS score and severity level. The table will list records ten at a time and the *Record Title* will be a selectable option for the user in order to focus on a new record on the top-half of the interface. The user will be able to navigate through the lists of records ten items at a time using the *Prev* and *Next* buttons.

Functional Requirements

Functional requirements for the VTCS are fundamental actions that the system must execute in order to be considered operational. The VTCS is a proof of concept and as such, limitations in its implementation will be identified.

FR-1: Obtain Threat Information

4.1.1 Description and Priority

The VTCS is required to take user input to collect and aggregate threat intelligence information from both the OTX and VirusTotal OSINT sources as records to be saved to the VTDBS.

Priority: Must Have

4.1.2 Related User Classes

Threat Collector, Database Administrator

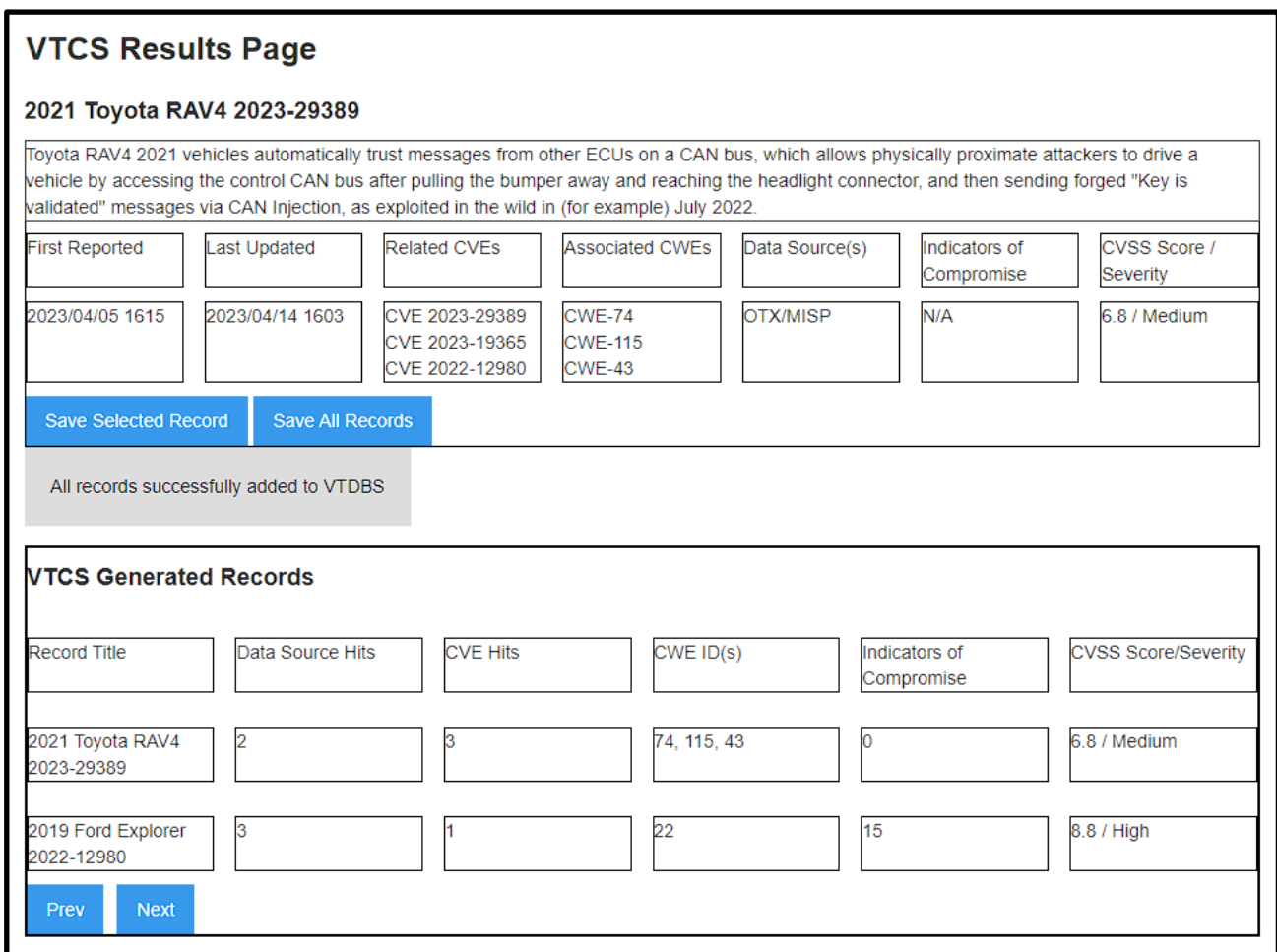


Figure 3.2 VTCS Record Viewer Interface Prototype

4.1.3 Functional Requirements

FR-1.1: The VTCS shall facilitate access to the OSINT APIs to gather threat data.

FR-1.2: The VTCS user interface shall provide for the querying and displaying of the CVE and CWE repository. The VTCS Record Viewer will have the following features:

- Preloaded list of vehicle manufacturers
- A search capability for CVEs for the selected manufacturer
- A display capability of each CVE record found for the selected manufacturer including the CVE ID, CVE Timestamp, CWE ID, Vulnerability Status, the Reference URL for the CVE, and the CVE description.
- The capability to navigate through all the CVE records.

The user interface prototype for data retrieval from the open-source repository is depicted in Figure 3.1.

FR-2: Store Threat Information

4.2.1 Description and Priority

The VTCS is required to process the information collected from both the OTX and VirusTotal OSINT sources and store them in the VTDBS.

Priority: Must Have

4.2.2 Related User Classes

Threat Collector, Database Administrator

4.2.3. Functional Requirements

FR-2.1: The VTCS shall facilitate the interfacing by the website with the VTDBS.

FR-2.2: The VTCS shall facilitate data record review and selection before committing it for storage.

The user interface prototype that will interface with the VTDBS is depicted in Figure 3.2 .

Test Requirements

The VTCS requires testing and validation of the main application functionalities.

T-1: Unit Tests

Unit system testing shall be conducted for all functional system components. Unit tests shall be integrated and documented in the source code. The GitHub repository is found at this URL:
<https://github.com/UWF-CfC-FDOT/VTMECS>.

T-2: Integration Tests

System integration testing is not within the scope of the VTCS system.

T-3: Test Report

Documentation of all system testing activities shall be provided. See the attached Unit Test Overview document.

Non-Functional Requirements

Non-functional requirements for the VTCS are system attributes that are desired but not required.

The following are the non-functional requirements for the VTCS:

NF-1: Portability

The development team will attempt to make the web enabled VTCS system portable across multiple computing form factors.

NF-2: Usability

The development team will attempt to satisfy system usability features such as navigation, performance quality, and intuitiveness of interfaces.

NF-3: Speed

The development team will attempt to enhance the VTCS system responsiveness to user interactions and database transactions.

Quality Attributes

The VTCS is a proof of concept and not meant for a production release. As such, traditional quality attributes such as availability, security, robustness, etc. are not as relevant.

Source Code Repository and Version Control Requirement

The development team shall facilitate a source code repository and version control for the project.

SC-1: Source Code Repository and Control

The development team shall maintain a source code repository and version control on GitHub.
The GitHub project URL is <https://github.com/UWF-CfC-FDOT/VTMECS>.

Appendix A: Requirements Table

Requirement ID	Requirement Type	Requirement Name	Requirement Description	Priority
S1	Scope	VTCS System Scope	The VTCS is required to take user input to collect and aggregate threat intelligence information from both the OTX and VirusTotal OSINT sources as records to be saved to the VTDBS.	Must have
FR-1	Functional	Obtain OSINT threat data	The VTCS shall facilitate the collection of threat data from open-source repository through to the OSINT APIs.	Must have
FR-2	Functional	Store Threat Information	The VTCS shall facilitate the storage of collected threat information into the VTDBS.	Must have
INT -1	Interface	VTCS Query Interface	The VTCS shall provide a Graphical User Interface for users to submit vehicle search parameters to generate records for review.	Must have
INT-2	Interface	VTCS Record Viewer	The VTCS shall provide a Graphical User Interface for users to manage query results and update the VTDBS	Must have

INT-3	Interface	API Interaction Logger	The VTCS shall log any interactions with external APIs in a format containing the time, message type, sent request, response code, and response	Must have
INT-4	Interface	DBMS Transaction logger	The VTCS shall log all threat database transactions in a format containing the time, query type, query contents, and response	Must have
INT-5	Interface	Error/Exception Logger	The VTCS shall log errors or exceptions in a format containing the time of the event and a stack trace	Must have
T-1	Test	Unit Test	Unit system testing shall be conducted for all functional system components	Must have
T-2	Test	Integration Test	System integration testing is not within the scope of the VTCS system	Out of scope
T-3	Test	Test Report	An associated documentation of all system test activities shall be provided	Must have
NF-1	Non-functional	Portability	The development team will attempt to make the web enabled VTCS system portable across multiple computing form factors	Could have

NF-2	Non-functional	Usability	The development team will attempt to satisfy system usability features such as navigation, performance quality, and intuitiveness of interfaces	Could have
NF-3	Non-functional	Speed	The development team will attempt to enhance the VTCS system responsiveness to user interactions and database transactions	Should have
SC-1	Source Code	Source Code Control	The development team shall maintain a source code repository and version control on GitHub	Should have

Appendix B: Requirements Traceability Matrix

Requirement ID	Requirement Description	Test Case⁵	Status
S1	Defines the scope of the system	N/A	N/A
FR-1	The VTCS shall facilitate access to the OSINT APIs to gather threat data.	Not Started Test cases: 3.1-3.2	Not Started
FR-2	The VTCS shall facilitate the storage and retrieval of collected threat information into the VTDBS.	Not Started Test cases: 6.1-6.2	Not Started
INT -1	The VTCS shall provide a Graphical User Interface for users to submit vehicle search parameters to generate records for review.	Not Started Test cases: 5.1.1-5.1.4	Not Started
INT-2	The VTCS shall provide a Graphical User Interface for users to manage query results and update the VTDBS	Not Started Test cases: 5.1.5-5.1.6	Not Started
INT-3	The VTCS shall log any interactions with external APIs in a format containing the time, message type, sent request, response code, and response	Not Started Test case: 3.2.4	Not Started
INT-4	The VTCS shall log all threat database transactions in a format containing the time, query type, query contents, and response	Not Started Test case: 6.3	Not Started

⁵ Test cases are fully defined in a document titled "VTCS TestPlan Overview.docx"

INT-5	The VTCS shall log errors or exceptions in a format containing the time of the event and a stack trace	Not Started Test cases: 6.3	Not Started
T-1	Unit system testing shall be conducted for all functional system components	Multiple Test Cases Test cases: 3-6	Not Started
T-2	System integration testing is not within the scope of the VTCS.	N/A	N/A
T-3	An associated documentation of all system test activities shall be provided	N/A	Not Started
NF-1	The development team will attempt to make the web enabled VTCS portable across multiple computing form factors	N/A	N/A
NF-2	The development team will attempt to satisfy system usability features such as navigation, performance quality, and intuitiveness of interfaces	N/A	N/A
NF-3	The development team will attempt to enhance the VTCS system responsiveness to user interactions and database transactions.	N/A	N/A
SC-1	The development team shall maintain a source code repository and version control on GitHub	N/A	N/A

Appendix C: Glossary

Term	Description
API	Application Program Interface
ATT&CK	Adversarial Tactics, Techniques, and Common Knowledge
ATT&CK Framework	A knowledge base of adversary tactics and techniques based on real-world observations.
AWS	Amazon Web Services
CVE	Common Vulnerabilities and Exposures
CWE	Common Weakness Enumeration
Cyber Kill Chain	A model developed by Lockheed Martin® used for the identification and prevention of cyber intrusions.
EC2	Elastic Compute Cloud
GUI	Graphical User Interface
IIS	Internet Information Services
IoC	Indicators of Compromise
.NET	A cross-platform, open-source developer platform created by Microsoft
OSINT	Open-Source Intelligence
OTX	Open Threat Exchange
VTCS	Vehicle Threat Collection System
VTDBS	Vehicle Threat Database System
VTME	Vehicle Threat Modeling Engine

Unit Test Plan Overview

for

Vehicle Threat Collection System (VTCS)

Technical Report UWF-TR-FDOT-007-02

Version 0.3 approved

Prepared by Dr. Guillermo Francia, III

**The University of West Florida
Florida Department of Transportation**

**June 3, 2023
July 3, 2023
July 18, 2023**

Table of Contents

1. Introduction	281
1.1 Purpose	281
1.2 Document Revisions Table	281
2. VTCS Test Framework.....	281
2.1 Framework Description.....	281
3. VTCS Unit Tests	281
3.1 APICallTest	281
3.1.1 CheckDomainAPI_NotNull	281
3.1.2 CheckIPAddressAPI_NotNull	281
3.1.3 CheckUrlAPI_NotNull	281
3.2 CveTest.....	282
3.2.1 ValidCve_ValidateId	282
3.2.2 CheckFetchKeywordCveWithDateAPI_Not Null	282
3.2.3 CheckAPICall_Not Null	282
3.2.4 CheckLogTransactions_Record	282
4. User Interface Test Framework.....	282
4.1 Framework Description.....	282
5. User Interface Unit Tests	282
5.1 UI_Checks.....	282
5.1.1 QueryPageRendersCorrectly.....	282
5.1.2 MakeNotSuppliedWhenSubmitted	282
5.1.3 ThreatSourceNotSelected	282
5.1.4 ResultsPageRendersCorrectly	282
5.1.5 CheckSaveSelectedRecord	283
5.1.6 CheckSaveAllRecords	283
5.2 UI_NavigationCheck.....	283
5.2.1 CheckPrevNavigation.....	283
5.2.2 CheckNextNavigation	283
5.3 Check_UI_logs.....	283
6. Vehicle Threat Database System Unit Tests	283
6.1 VerifyInformationStorage	283
6.2 VerifyInformationRetrieval	283
6.3 VerifyVTDBS_TransactionLogs	283

1. Introduction

1.1 Purpose

This document describes the unit test framework of the Vehicle Threat Collection System (VTCS) as a subcomponent of the Connected Vehicle Security Metrics and Threat Intelligence Project. This document provides an overview of unit test framework and tests for both the VTCS and the user interface.

1.2 Document Revisions Table

Revisor	Revision Date	Reason
Guillermo Francia, III	December 1, 2022	Initial draft
Guillermo Francia, III	June 1, 2023	Added test cases
Guillermo Francia, III	July 3, 2023	Added UI interaction and database transaction tests
Guillermo Francia, III	July 18, 2023	Noted a separate supplementary file, the RTVM document, to accompany this document.

2. VTCS Test Framework

2.1 Framework Description

The VTCS project is developed in C# language and the choice was made to use the standard MsTest framework for unit testing the engine. MsTest is a native unit testing library that comes with Visual Studio from Microsoft.

A Requirements Traceability Verification Matrix (RTVM) is provided in a separate document.

3. VTCS Unit Tests

3.1 APICallTest

The APICall module will contain unit test coverage of the /External_API/VirusTotal/ classes. It will perform a validation of the API call processes to the VirusTotal repository.

3.1.1 CheckDomainAPI_NotNull

Validates that a domainModel object is successfully returned by the DomainAPI class APICall function.

3.1.2 CheckIPAddressAPI_NotNull

Validates that an ipAddressModel object is successfully returned by the DomainAPI class APICall function.

3.1.3 CheckUrlAPI_NotNull

Validates that a urlModel object is successfully returned by the DomainAPI class APICall function.

3.2 CveTest

The CveTest module will contain unit test coverage of the /External_API/Cve/KeywordSearchAPI class. This class contains the object that represents a CVE for a given record.

3.2.1 ValidCve_ValidateId

Validates that a provided CVE key at initialization will be properly stored as a property that is publicly accessible.

3.2.2 CheckFetchKeywordCveWithDateAPI_Not Null

Validates that a CveQuery object is successfully returned by the FetchKeywordCveWithDateAPI function.

3.2.3 CheckAPICall_Not Null

Validates that a queryModel object is successfully returned by the APICall function.

3.2.4 CheckLogTransactions_Record

Validates that the APICall transaction is properly logged .

4. User Interface Test Framework

4.1 Framework Description

The User Interface Test Framework is developed in C# language and BUnit framework was chosen for unit testing the interface.

5. User Interface Unit Tests

5.1 UI_Checks

5.1.1 QueryPageRendersCorrectly

Verifies that the Query Web page renders correctly.

5.1.2 MakeNotSuppliedWhenSubmitted

Verifies that the Make is supplied when form is submitted. Note that Year and Model are optional.

5.1.3 ThreatSourceNotSelected

Verifies that the Threat source is selected when form is submitted.

5.1.4 ResultsPageRendersCorrectly

Verifies that the Results Web page renders correctly.

5.1.5 CheckSaveSelectedRecord

Verifies that the *Save Selected Record* function successfully adds the record into the VTDBS.

5.1.6 CheckSaveAllRecords

Verifies that the *Save All Records* function successfully adds all the current records into the VTDBS.

5.2 UI_NavigationCheck

5.2.1 CheckPrevNavigation

Verifies that the *Prev* navigation function moves correctly to the previous record among the set of VTCS generated records. A wrap-around feature must also be correctly implemented.

5.2.2 CheckNextNavigation

Verifies that the *Next* navigation function moves correctly to the next record among the set of VTCS generated records. A wrap-around feature must also be correctly implemented.

5.3 Check_UI_logs

Verifies that User Interface transactions are logged.

6. Vehicle Threat Database System Unit Tests

6.1 VerifyInformationStorage

Verifies that the threat information is properly stored in the VTDBS.

6.2 VerifyInformationRetrieval

Verifies that the threat information is properly retrieved from the VTDBS.

6.3 VerifyVTDBS_TransactionLogs

Verifies that all VTDBS transactions are logged.

Appendix XIV. Technical Report UWF-TR-FDOT-007-03

Project Name:
Project Description:

Vehicle Threat Collection System (VTCS)
The Vehicle Threat Collection System (VTCS) project is a subcomponent of the Connected Vehicle Security Metrics and Threat Intelligence Project.

Project Manager Name:
Agency/Firm:

Dr. Guillermo Francia, III
UWF-FDOT

User Need ID	User Need Summary	Requirement ID	Detailed Requirement Summary	Document Section	DR Source Document	Verification Test Case ID	Compliance (Y/N/Partial/NA)	Notes/Comments/Date	Reviewer Initials
GAF001	Specifies the scope of the system	S1	Define the scope of the system	2.1	VTCS Requirements	NA	NA	Non Testable	DH
GAF001	Obtain OSINT threat data	FR-1	The VTCS shall facilitate the collection of threat data from open-source repository through to the OSINT APIs.	4.1	VTCS Requirements	Test cases: 3.1-3.2	Partially	Not started	DH
GAF001	Store Threat Information	FR-2	The VTCS shall facilitate the storage of collected threat information into the VTDBS.	4.2	VTCS Requirements	Test cases: 6.1-6.2	Partially	Not started	DH
GAF001	VTCS Query Interface	INT-1	The VTCS shall provide a Graphical User Interface for users to submit vehicle search parameters to generate records for review.	3.1	VTCS Requirements	Test cases: 5.1.1-5.1.4	Partially	Not started	DH
GAF001	VTCS Record Viewer	INT-2	The VTCS shall provide a Graphical User Interface for users to manage query results and update the VTDBS	3.2	VTCS Requirements	Test cases: 5.1.5-5.1.6	Partially	Not started	DH

GAF001	API Integration Logger	INT-3	The VTCS shall provide a Graphical User Interface for users to manage query results and update the VTDBS	3.0	VTCS Requirements	Test case: 3.2.4	Partially	Not started	DH
--------	---------------------------	-------	--	-----	-------------------	------------------	-----------	-------------	----

Project Name:
Project Description:

Vehicle Threat Collection System (VTCS)
The Vehicle Threat Collection System (VTCS) project is a subcomponent of the Connected Vehicle Security Metrics and Threat Intelligence Project.

Project Manager Name:
Agency/Firm:

Dr. Guillermo Francia, III
UWF-FDOT

User Need ID	User Need Summary	Requirement ID	Detailed Requirement Summary	Document Section	DR Source Document	Verification Test Case ID	Compliance (Y/N/Partial/NA)	Notes/Comments/Date	Reviewer Initials
GAF001	DBMS Transaction Logger	INT-4	The VTCS shall log all threat database transactions in a format containing the time, query type, query contents, and response.	3.0	VTCS Requirements	Test case: 6.3	Partially	Not started	DH
GAF001	Error/Exception Logger	INT-5	The VTCS shall log errors or exceptions in a format containing the time of the event and a stack trace.	3.0	VTCS Requirements	Test case: 6.3	Partially	Not started	DH
GAF001	Unit Test	T-1	Unit system testing shall be conducted for all functional system components	5.1	VTCS Requirements	Test cases: 3-6	Partially	Not started	DH
GAF001	Integration Test	T-2	System integration testing is not within the scope of the VTCS system	5.2	VTCS Requirements	NA	Partially	Non Testable	DH
GAF001	Test Report	T-3	An associated documentation of all system test activities shall be provided	5.3	VTCS Requirements	NA	Partially	Not started	DH

GAF001	Portability	NF-1	The development team will attempt to make the web-enabled VTCS system portable across multiple computing form factors	6.1	VTCS Requirements	NA	Partially	Non Testable	DH
--------	-------------	------	---	-----	-------------------	----	-----------	--------------	----

Project Name:
Project Description:

Vehicle Threat Collection System (VTCS)
The Vehicle Threat Collection System (VTCS) project is a subcomponent of the Connected Vehicle Security Metrics and Threat Intelligence Project.

Project Manager Name:
Agency/Firm:

Dr. Guillermo Francia, III
UWF-FDOT

User Need ID	User Need Summary	Requirement ID	Detailed Requirement Summary	Document Section	DR Source Document	Verification Test Case ID	Compliance (Y/N/Partial/NA)	Notes/Comments/Date	Reviewer Initials
GAF001	Usability	NF-2	The development team will attempt to satisfy system usability features such as navigation, performance quality, and intuitiveness of interfaces	6.2	VTCS Requirements	NA	Partially	Non Testable	DH
GAF001	Speed	NF-3	The development team will attempt to enhance the VTCS system responsiveness to user interactions and database transactions	6.3	VTCS Requirements	NA	Partially	Non Testable	DH
GAF001	Source Code Control	SC-1	The development team shall maintain a source code repository and version control on GitHub	8.1	VTCS Requirements	NA	Partially	Non Testable	DH

Software Requirements Specification

for

Vehicle Threat Database System (VTDS)

Technical Report UWF-TR-FDOT-008-01

Version 3.0 Approved

Prepared by David Huson

Approved by Dr. Guillermo Francia, III

The University of West Florida

Florida Department of Transportation

July 21, 2023 (Initial)

Table of Contents

1. Introduction	1
1.1 Purpose	1
1.2 Document Conventions	1
1.3 References	1
1.4 Document Revisions Table	1
2. Overview of Product	2
2.1 VTDS	2
2.2 User Classes and Characteristics	3
2.3 Operating Environment	3
2.4 Design and Implementation Constraints	3
2.5 Assumptions and Dependencies	5
3. Interface Requirements	5
3.1 INT-1: VTDS Application Programming Interface (API)	5
4. Functional Requirements	6
4.1 FR-1: Data Ingestion	6
4.1.1 Description and Priority	6
4.1.2 Related User Classes	6
4.2 FR-2: Data Retrieval	6
4.2.1 Description and Priority	6
4.2.2 Related User Classes	6
4.2.3 Functional Requirements	7
4.3 FR-3: Data Integrity	7
4.3.1 Description and Priority	7
4.3.2 Related User Classes	7
4.3.3 Functional Requirements	7
4.4 FR-4: User Authentication	8
4.4.1 Description and Priority	8
4.4.2 Related User Classes	8
4.4.3 Functional Requirements	8
4.5 FR-5: Transaction Logs	9
4.5.1 Description and Priority	9
4.5.2 Related User Classes	9
4.5.3 Functional Requirements	9
4.6 FR-6: Backup and Recovery	9
4.6.1 Description and Priority	9
4.6.2 Related User Classes	10
4.6.3 Functional Requirements	10

5. Test Requirements	10
5.1 T-1: Data Integrity Tests	10
5.2 T-2: Security Tests	11
5.3 T-3: Performance Tests	11
5.4 T-4: Recovery Tests	11
5.5 T-5: Transaction Log Tests	11
6. Non-Functional Requirements	11
6.1 NF-1: Performance	12
7. Quality Attributes	12
8. Source Code Repository and Version Control Requirement	12
8.1 SC-1: Source Code Repository and Control	12
9. Appendix A: Requirements Table	13
10. Appendix B: Requirements Traceability Matrix	17
11. Appendix C: Glossary	20

1. Introduction

1.1 Purpose

This document describes the software requirements for Version 1.0 of the Vehicle Threat Database System (VTDS) as a subcomponent of the Connected Vehicle Security Metrics and Threat Intelligence Project. This document provides an overview of the users and context of the VTDS and covers all functional, non-functional, interface, and data requirements of the VTDS.

1.2 Document Conventions

This document is based on the IEEE 830-1998 Standards and the Florida Department of Transportation Requirements Standards. Specific conventions used in this document are listed below:

- Priorities are indicated for each feature as well as in the Requirements Table. A green highlighting indicates must have features, while a yellow highlight represents a should have feature.
- Requirements follow the form of <TAG>-#.#.# where a tag indicates a category of requirements. And the # represents the ID of the requirement in a hierarchical fashion.

1.3 References

The following references were used in the creation of this document:

- IEEE 830-1998 Standards on Software Requirement Specifications
- UWF Scope of Service Document for the Connected Vehicle Security Metrics and Threat Intelligence Project
- VTDS_Documentation_20230503

1.4 Document Revisions Table

Revisor	Revision Date	Reason
David Huson	07/24/2023	Initial Draft
David Huson	08/25/2023	Add SQL injection vulnerability test requirement
David Huson	08/30/2023	Add Id constraints and new
David Huson	09/17/2023	Update Source Control URL
David Huson	09/22/2023	Add API interface Requirement
Dr. Guillermo Francia, III	09/29/2023	Added functional requirements

David Huson	10/4/2023	Added detail to new functional requirements and interface requirements
David Huson	11/13/2023	Synchronize with Test Plan Document
Dr. Guillermo Francia, III	1/10/2024	Fixed inconsistencies with the Requirements Traceability Matrix

2. Overview of Product

2.1 VTDS

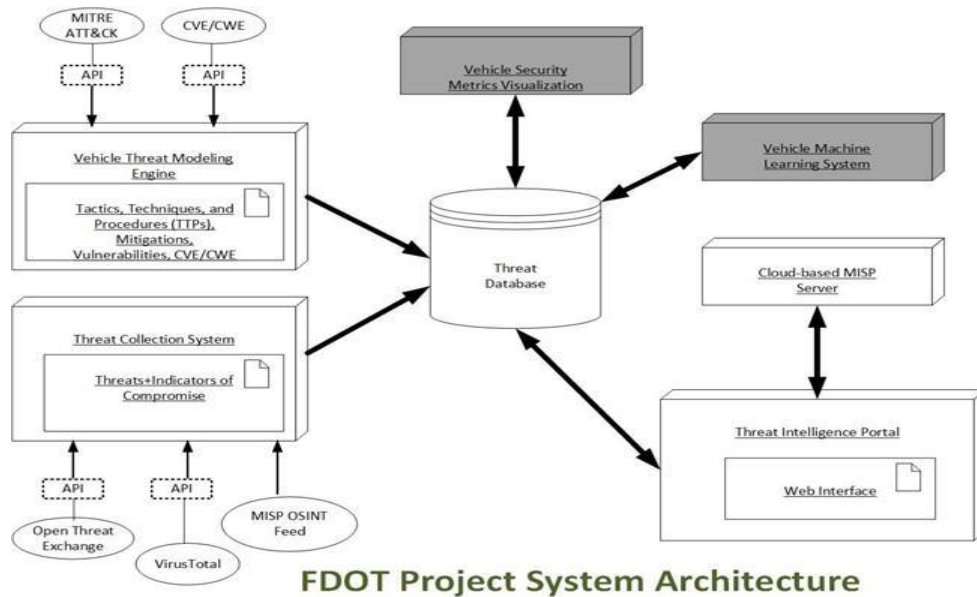


Figure 1.1 - Detailed Project System Architecture Diagram

The VTDS is a database server which is designed to support the Vehicle Threat Modeling Engine (VTME), Vehicle Threat Collection System (VTCS), Vehicle Security Metrics Visualization, Vehicle Machine Learning System, and Threat Intelligence Portal. It will do so by facilitating Create Read Update Delete (CRUD) operations to the various subcomponents of the Project System formerly mentioned.

2.2 User Classes and Characteristics

User Class	Characteristics
VTCS User	This user will be able to query the VTDS via the VTCS Query Interface

Threat Intelligence Portal User	This user will be able to query vehicle threat information gathered by the VTME
Threat Model Builder	This user will utilize the VTDS to store individual threat information to include data from the following sources MITRE ATT&CK and CVE/CWE.
Threat Collector	This user will utilize the VTDS to store information that is retrieved from the following data sources Open Threat Exchange, VirusTotal, and MISP OSINT Feed.
Threat Model Visualization User	This user will be able to store and query CVE data on the VTDS
Threat Model Visualization Admin	This user will be able to send data to the VTDS for storage
VSMLS Engineer	This user will be able to submit data to the BSM data to be ingested and stored by the VTDS
Database Administrator	The user responsible for administering and maintaining the VTDS

2.3 Operating Environment

The VTCS operating environment is defined by the following:

OE-1: The VTDS shall run within an Amazon Elastic Compute Cloud (EC2) Web Service utilizing a Windows Server 2019 Operating System.

OE-2: The AWS EC2 instance shall be configured with type m5.xlarge having 4 vCPU and 16 GB of memory.

OE-3 The VTDS shall be configured using MS SQL Server 2019.

2.4 Design and Implementation Constraints

The VTDS design and implementation are constrained by the following:

DIC-1: The VTDS shall be developed, tested, and maintained using the [SQL Server Management Studio](#) and/or AWS.

DIC-2: The VTDS test suite shall be developed using Microsoft Visual Studio 2022

DIC-3: The VTDS test suite shall be developed using the C# programming language.

DIC-4: The VTDS design shall be constrained by the Entity Relationship (ER) diagram depicted in Figure 1.2 below

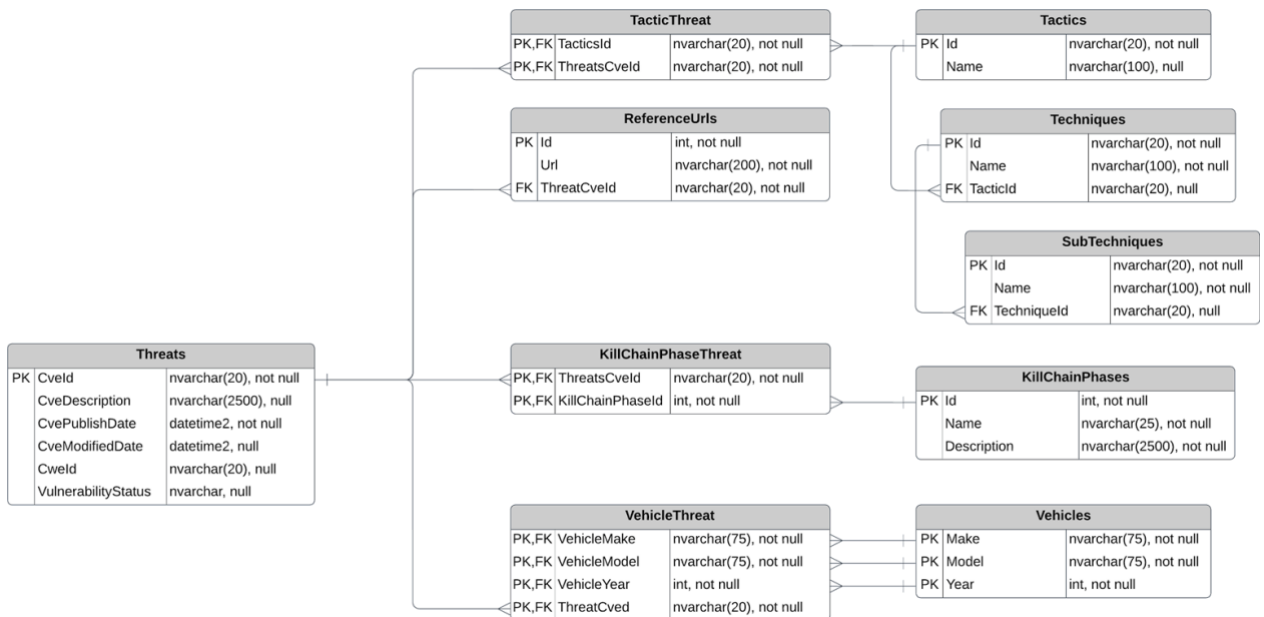


Figure 1.2 - Entity Relationship Diagram with Primary Key, Foreign Key, and Not Null

DIC-5: The VTDS will place the following check constraints on primary key fields:

1. The Threats table **CveId** field will follow the standard format documented [here](#).
 - a. Example: **CVE-0000-0000**
2. The Threats table **CweId** field will adhere to the following format: **CWE prefix + arbitrary digits (up to 10)**
 - a. Example: **CWE-0**
3. The Tactics table Id field will adhere to the following format: **TA prefix + arbitrary digits (up to 10)**
 - a. Example: **TA-0**
4. The Techniques Id field will adhere to the following format: **TECH prefix + arbitrary digits (up to 10)**
 - a. Example: **TECH-0**
5. The SubTechniques Id field will adhere to the following format **SUBT prefix + arbitrary digits (up to 10)**
 - a. Example: **SUBT-0**

DIC-6: The VTDS Shall normalize all data before ingesting said data into the Vehicle Threat Database.

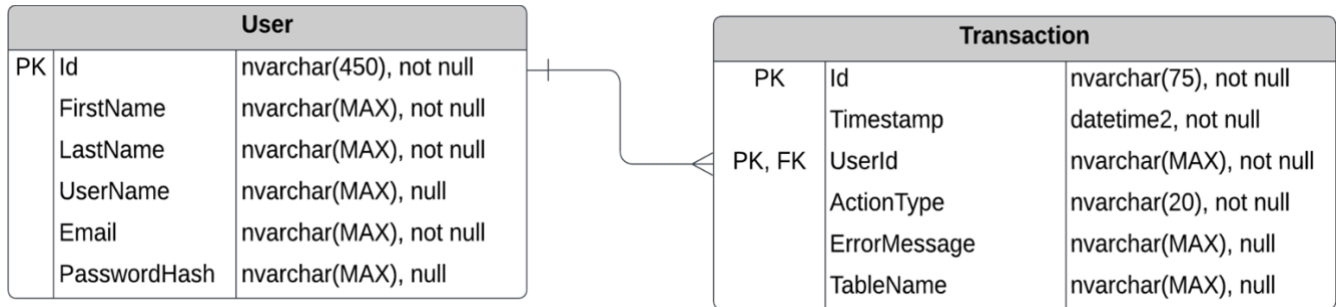


Fig 1.3 – ER diagram for User Authentication Table and Transaction Log Table

DIC-7: The VTDS shall require a table to store user information when creating new users.

DIC-8: The VTDS shall require a table to store API transactions made by users.

2.5 Assumptions and Dependencies

Assumptions and dependencies for the VTDS implementation include the following:

DEP-1: The VTDS shall be dependent on data coming from the VTME, VTCS, VSMVS, and VSMLS.

3. Interface Requirements

INT-1 - The VTDS shall provide an API to be used by the MISP, VTIP, and other components of the system to send and retrieve data from the Vehicle Threat Database via a web API endpoint.

INT-2: The query interface, offered by the VTCS, shall be provided to users for submitting vehicle search parameters. This Graphical User Interface is fully described in section 3.1 of the VTCS Requirements document. It is implemented as part of the VTCS.

INT-3: The VTDS shall provide an API route to facilitate the creation of new users in the system when accessed through the User Registration Interface provided by the VTCS.

INT-4: The VTDS shall provide an API route for user session management via the User Login Interface provided by the VTCS

INT-5: The VTDS shall provide a GUI for Database Administrators to use to backup or recover the VTDS via the appropriate VTDS API routes

INT-6: The VTDS shall provide a GUI to facilitate the review of database transaction logs.

3.1 INT-1: VTDS Application Programming Interface (API)

The VTDS will provide an API endpoint which can be used by all components of the System to interact with and retrieve data from the database. The requests made to these API routes must be

sent in the request body in JSON format and must adhere to the database schema outlined in the VTDS ER diagram in section 2.4 of this document. This API will be provided via an endpoint which is available as a subdirectory of the VTCS base URL hosted on IIS. The API requires requests be made as JSON objects in the request body with the appropriate method (i.e. POST, GET, PUT, DELETE). These endpoints will only be accessible from an authenticated and authorized client. Examples of the JSON objects for each table will be provided on the GitHub repository in the technical documentation.

4. Functional Requirements

Functional requirements for the VTDS are fundamental actions that the system must execute or facilitate to be considered operational.

4.1 FR-1: Data Ingestion

4.1.1 Description and Priority

The VTDS is required to receive data from several data sources (VTME, VTCS, VSMVS, VTIP, and VMLS), and save the data to the database in the appropriate tables.

Priority: Must Have

4.1.2 Related User Classes

Threat Collector, Threat Model Builder, VSMLS Engineer, VTMVS Admin, VTCS User, Database Administrator

4.2 FR-2: Data Retrieval

4.2.1 Description and Priority

The VTDS is required to return the appropriate information queried by the VSMVS, VTIP, and VSMLS.

Priority: Must Have

4.2.2 Related User Classes

VTIP User, VSMVS User, VSMLS Engineer, Database Administrator.

4.2.3 Functional Requirements

FR-2.3.1: API Routes

The VTDS API must provide the appropriate routes to retrieve data from the corresponding tables in the Vehicle Threat Database.

4.3 FR-3: Data Integrity

4.3.1 Description and Priority

The VTDS is required to store data in a manner which maintains the integrity of the data. Data Integrity will be defined using three metrics outlined below.

Priority: Must Have

4.3.2 Related User Classes
Database Administrator

4.3.3 Functional Requirements

4.3.3.1 FR-3.1: Data Consistency

The VTDS must ensure that all data entered into the system conforms to the defined data type and format.

4.3.3.2 FR-3.2: Data Completeness

The VTDS must ensure that all expected fields are present and filled with the required information in the database

4.3.3.3 FR-4.3: Data Accuracy

The VTDS must ensure that the data entered in the database is consistent with the source data

4.4 FR-4: User Authentication

4.4.1 Description and Priority

The VTDS will only allow requests which originate from an authenticated source and will validate users sessions on a per request basis.

Priority: Should Have

4.4.2 Related User Classes

Database Administrator, Threat Intelligence Portal User, VTCS user, Threat Collector

4.4.3 Functional Requirements

4.4.3.1 FR-4.1: Login API Route

- The VTDS API shall provide an API route to facilitate user authentication
- The route must take in user credentials and validate them with those stored in the Vehicle Threat Database.

4.4.3.2 FR-4.2: Registration API Route

- The API must provide a means of creating a new user record in the database (INT-4)

4.4.3.3 FR-4.3: Logout API Route

- The API must provide a route for logging a user out of their session

4.4.3.4 FR-4.5: Identity Table

- The VTDS will require a new table to store user credentials (username, password hash, and role”

4.5 FR-5: Transaction Logs

4.5.1 Description and Priority

The VTDS shall keep a record of all transactions (requests) made to the API. And allow related user classes to view these logs via an additional interface.

Priority: Should Have

4.5.2 Related User Classes

Database Administrator

4.5.3 Functional Requirements

4.5.3.1 FR-5.1: VTDS Transaction Log Table

- The VTDS shall record all transactions made via the API in a separate Logs table.
- The Logs table shall record all relevant information such as:
 - transaction type (POST, GET, etc.)
 - transaction timestamp
 - affected tables
 - initiating user
 - status code (success or failure)
 - if transaction fails, it will also record any error messages.

4.5.3.2 FR-5.2: VTDS Transaction Log Authentication

- The VTDS shall prohibit any user other than those with administrative authentication role from accessing either the endpoint or the interface (see INT-7)

4.6 FR-6: Backup and Recovery

4.6.1 Description and Priority

The VTDS shall provide a means of initiating a full database backup or recovery from a backup via a request to either the /backup or /recovery route made by a user with elevated privileges (i.e. administrator role).

Priority: Should Have

4.6.2 Related User Classes

Database Administrator

4.6.3 Functional Requirements

4.6.3.1 FR-6.1: VTDS Backup API Route

- The VTDS shall implement an API route for initiating a full database backup

4.6.3.2 FR-6.2: VTDS Recovery API Route

- The VTDS shall implement an API for initiating a database restoration from a previous backup

4.6.3.3 FR-6.4: VTDS Backup and Recovery Route Authorization

- The VTDS shall prohibit all users other than those with the administrative authorization role to access the backup/recovery routes or the interface (see INT-6.)

5. Test Requirements

The VTDS requires testing and validation of the database functionalities and requirements. All test cases will be integrated and documented in the source code, which can be found on the [VTMECS GitHub Repository](#).

5.1 T-1: Data Integrity Tests

Data integrity testing aims to ensure the accuracy and completeness of the data stored in the database. These tests include the three following subtests:

1. Data consistency test
2. Data completeness test
3. Data accuracy test

5.2 T-2: Security Tests

This test aims to ensure the security of the database. The test includes the following sub-tests:

1. User authentication test
2. User authorization test
3. SQL injection vulnerability test
4. User registration tests

5.3 T-3: Performance Tests

This test aims to evaluate the performance of the database. The test includes the following sub-test:

1. Query execution time test

5.4 T-4: Recovery Tests

This test aims to ensure that the database can be recovered in the event of a failure. The test includes the following sub-tests:

1. Backup and recovery test

5.5 T-5: Transaction Log Tests

This test aims to ensure that the VTDS properly logs all transactions made via the VTDS API

1. Successful Transaction Tests
2. Unsuccessful Transaction Tests

6. Non-Functional Requirements

Non-functional requirements for the VTCS are system attributes that are desired but not required.

The following are the non-functional requirements for the VTCS:

6.1 NF-1: Performance

The development team will attempt to enhance the VTDS system responsiveness to query operations by reducing the query execution time.

7. Quality Attributes

The VTDS is a proof of concept and not meant for a production release. As such, traditional quality attributes such as availability, security, robustness, etc. are not as relevant.

8. Source Code Repository and Version Control Requirement

The development team shall facilitate a source code repository and version control for the project.

8.1 SC-1: Source Code Repository and Control

The development team shall maintain a source code repository and version control on [GitHub](#).

9. Appendix A: Requirements Table

Requirement ID	Requirement Type	Requirement Name	Requirement Description	Priority
S1	Scope	VTDS System Scope	The VTDS is required to facilitate CRUD operations from the VTCS, VMLS, VSMS, VTME, and VTIP.	Must have
FR-1	Functional	Data Ingestion	The VTDS is required to receive data from several data sources (VTME, VTCS, VSMV, VTIP, and VMLS), and save the data to the database	Must Have
FR-2	Functional	Data Retrieval	The VTDS is required to return the appropriate information queried by the VSMVS, VTIP, and VSMLSS.	Must Have
FR-3	Functional	Data Integrity	The VTDS is required to store data in a manner which maintains the integrity of the data. Data Integrity will be defined using three metrics outlined below.	Must Have
FR-4	Functional	User Authentication	The VTDS will only allow requests which originate from an authenticated source. This requires an additional database table of users with their email and password hash to be used by the VTCS login page as well as a table of sessions which relate a user to a session token which is to be sent with requests to the VTDS routes as a means of	Should Have

			authenticating the user on a per request basis.	
FR-5	Functional	Transaction Logs	The VTDS shall keep a record of all transactions (requests) made to the API. And allow related user classes to view these logs via an additional interface.	Should Have
FR-6	Functional	Backup and recovery	The VTDS shall provide a means of initiating a full database backup or recovery from a backup via a request to either the /backup or /recovery route made by a user with elevated privileges (i.e. administrator).	Should Have
T-1	Test	Data Integrity Tests	Data integrity testing aims to ensure the accuracy and completeness of the data stored in the database. These tests include the three following subtests:	Must Have
T-2	Test	Security Tests	This test aims to ensure the security of the database.	Should have
T-3	Test	Performance Tests	This test aims to evaluate the performance of the database.	Could Have
T-4	Test	Recovery Tests	This test aims to ensure that the database can be recovered in the event of a failure.	Must Have
T-5	Test	Transaction Log Tests	This test case aims to ensure that the VTDS properly logs all transactions made via the VTDS API	Should Have

NF-1	Non-functional	Performance	The development team will attempt to enhance the VTDS system responsiveness to query operations by reducing the time each query takes to complete.	Could Have
INT-1	Interface	VTDS Application Programming Interface (API)	The VTDS shall provide an API to be used by the MISIP, VTIP, and other components of the system to send and retrieve data from the Vehicle Threat Database.	Must Have
INT-2	Interface	VTCS Query Interface	The VTDS shall provide an API route to facilitate the creation of new records in the Vehicle Threats Database via the VTCS Record Viewer Interface (Described in detail in Section 3.2 of the VTCS Requirements Document).	Must Have
INT-3	Interface	VTDS Registration Interface	The VTDS shall provide an API route to facilitate the creation of new users in the system when accessed through the User Registration Interface provided by the VTCS.	Should Have
INT-4	Interface	VTDS User Login Interface	The VTDS shall provide an API route for user session management via the User Login Interface provided by the VTCS	Should have
INT-5	Interface	VTDS Backup and Recovery Interface	The VTDS shall provide a GUI for Database Administrators to use to backup or recover the VTDS via the appropriate VTDS API routes	Should have
INT-6	Interface	VTDS Transaction Log Interface	The VTDS shall provide a GUI to facilitate the review of database transaction logs.	Should have

SC-1	Source Code	Source Code Control	The development team shall maintain a source code repository and version control on GitHub	Should have
------	-------------	---------------------	--	-------------

10. Appendix B: Requirements Traceability Matrix

Requirement ID	Requirement Description	Test Case ⁶	Status
S1	Specifies the scope of the system	N/A	N/A
FR-1	The VTDS is required to receive data from several data sources (VTME, VTCS, VSMV, VTIP, and VMLS), and save the data to the database	Complete Test Cases: 3.1-3.3	Complete
FR-2	The VTDS is required to return the appropriate information queried by the VSMVS, VTIP, and VSMLSS.	Complete Test Cases: 3.2-3.3	Complete
FR-3	The VTDS is required to store data in a manner which maintains the integrity of the data. Data Integrity will be defined using three metrics outlined below.	Complete Test Cases: 3.1-3.3	Complete
FR-4	The VTDS will only allow requests which originate from an authenticated source.	Partially Test Cases: 4.1-4.4	The login and registration routes are completed. Must complete the token validation system for creating new admin users for this to be fully complete.
FR-5	The VTDS shall keep a record of all transactions (requests) made to the API. And allow related user classes to view these logs via an additional interface.	Partially Test Case: 7	The methods for logging transactions to the VTDS are complete. Waiting for the user interface to complete.

⁶ Test cases are fully defined in a document titled "VTDS_TestPlan Overview.docx"

FR-6	The VTDS shall provide a means of initiating a full database backup or recovery from a backup via a request to either the /backup or /recovery route made by a user with elevated privileges (i.e. administrator).	Partially Test Case: 6.1	The methods for completing a VTDS Backup and Recovery are complete. This requirement is waiting on an Interface for it to be considered fully completed.
T-1	Data integrity testing aims to ensure the accuracy and completeness of the data stored in the database.	Complete Test Cases: 3.1-3.3	Complete
T-2	This test aims to ensure the security of the database.	Complete Test Cases: 4.1 - 4.4	Complete
T-3	This test aims to evaluate the performance of the database.	Partial Test Cases: 5.1	The tests are partially completed. To be completed by the development team for the production version of the system.
T-4	This test aims to ensure that the database can be recovered in the event of a failure.	Complete Test Case: 6.1	Complete
T-5	This test case aims to ensure that the VTDS properly logs all transactions made via the VTDS API	Not Started Test Case: 7	To be completed after the User Interfaces are ready.
NF-1	The development team will attempt to enhance the VTDS system responsiveness to query operations by reducing the time each query takes to complete.	Partial Test Case: 5.1	The tests are partially completed. To be completed by the development team for the production version of the system.
INT-1	The VTDS shall provide an API to be used by the MISP, VTIP, and other components of the system to	Complete	Complete

	send and retrieve data from the Vehicle Threat Database.	Test Cases: 3.1-3.3	
INT-2	The query interface, offered by the VTCS, shall be provided to users for submitting vehicle search parameters. This Graphical User Interface is fully described in section 3.1 of the VTCS Requirements document. It is implemented as part of the VTCS.	Complete	Complete
INT-3	The VTDS shall provide an API route to facilitate the creation of new users in the system when accessed through the User Registration Interface provided by the VTCS.	Partial Test Case: 4.3	The test is partially complete, waiting on the token validation system for full completion.
INT-4	The VTDS shall provide an API route for user session management via the User Login Interface provided by the VTCS and the User Session Authentication Middleware.	Partial Test Cases: 4.1-4.2.1	Working on a method of redirecting back to the original endpoint requested after successful login or registration.
INT-5	The VTDS shall provide a GUI for Database Administrators to use to backup or recover the VTDS via the appropriate VTDS API routes	Not Started Test Case: 6.1	To be completed by the developers of the production version.
INT-6	The VTDS shall provide a GUI to facilitate the review of database transaction logs.	Partial Test Case: 7	The methods which this interface relies on are complete. Need to complete the remaining User Interfaces for full completion.
SC-1	The development team shall maintain a source code repository and version control on GitHub	Complete	Considered complete with the source code made available on Github

11. Appendix C: Glossary

Term	Description
API	Application Programming Interface
ATT&CK	Adversarial Tactics, Techniques, and Common Knowledge
ATT&CK Framework	A knowledge base of adversary tactics and techniques based on real-world observations.
AWS	Amazon Web Services
CRUD	Create Read Update Delete
CVE	Common Vulnerabilities and Exposures
CWE	Common Weakness Enumeration
EC2	Elastic Compute Cloud
JSON	JavaScript Object Notation
OSINT	Open-Source Intelligence
OTX	Open Threat Exchange
SQL	Structured Query Language
VSMLS	Vehicle Security Machine Learning
VSMVS	Vehicle Security Model Visualization System
VTCS	Vehicle Threat Collection System
VTDS	Vehicle Threat Database System
VTME	Vehicle Threat Modeling Engine

Appendix XVI. Technical Report UWF-TR-FDOT-008-02

Project Name:				<i>Vehicle Threat Collection System (VTDS)</i>					
Project Description:				<i>The Vehicle Threat</i>					
Project Manager Name:				<i>Dr. Guillermo Francia, III</i>					
Agency/Firm:				<i>UWF-FDOT</i>					
User Need ID	User Need Summary	Requirement ID	Detailed Requirement Summary	Document Section	DR Source Document	Verification Test Case ID	Compliance (Y/N/Partial/NA)	Notes/Comments /Date	Reviewer Initials
GAF001	VTDS System Scope	S1	Specifies the scope of the system	2.1	VTDS Requirements	NA	Yes	Specified in the Requirements document	DH
GAF001	Data Ingestion	FR-1	The VTDS is required to receive data from several data sources (VTME, VTCS, VSMV, VTIP, and VMLS), and save the data to the database	4.1	VTDS Requirements	Test Cases: 3.1-3.3	Yes	Completed 08/29/2023	DH
GAF001	Data Retrieval	FR-2	The VTDS is required to return the appropriate information queried by the VSMVS, VTIP, and VSMLSS.	4.2	VTDS Requirements	Test Cases: 3.2-3.3	Yes	Completed 08/29/2023	DH
GAF001	Data Integrity	FR-3	The VTDS is required to store data in a manner which maintains the	4.3	VTDS Requirements	Test Cases: 3.1-3.3	Yes	Completed 08/28/23	DH

GAF001	User Authentication	FR-4	integrity of the data. The VTDS will only allow requests which originate from an authenticated source.	4.4	VTDS Requirements	Test Case: 4.1-4.4	Partially	The login and registration routes are completed. Must complete the token validation system for creating new admin users for this to be fully complete.	DH
GAF001	Transaction Logs	FR-5	The VTDS shall keep a record of all transactions (requests) made to the api. And allow related user classes to view these logs via an additional interface.	4.5	VTDS Requirements	Test Case: 7	Partially	The methods for logging transactions to the VTDS are complete. Waiting for the user interface to complete.	DH
GAF001	Backup and Recovery	FR-6	The VTDS shall provide a means of initiating a full database backup or recovery from a backup via a request to either the /backup or /recovery route	4.6	VTDS Requirements	Test Case: 6.1	Partially	The methods for completing a VTDS Backup and Recovery are complete. This requirement is waiting on an Interface for it to be considered fully completed.	DH

GAF001	Data Integrity Tests	T-1	made by an user with elevated privileges (i.e. administrator). Data integrity testing aims to ensure the accuracy and completeness of the data stored in the database.	5.1	VTDS Requirements	Test Cases: 3.1-3.3	Yes	Completed 08/28/23	DH
GAF001	Security Tests	T-2	This test aims to ensure the security of the database.	5.2	VTDS Requirements	Test Cases: 4.1 - 4.4	Yes	Completed 08/22/23	DH
GAF001	Performance Tests	T-3	This test aims to evaluate the performance of the database.	5.3	VTDS Requirements	Test Cases: 5.1	Partially	The tests are partially completed. To be completed by the development team for the	DH

Unit Test Plan Overview

for

Vehicle Threat Database System (VTDS)

Technical Report UWF-TR-FDOT-008-03

Version 0.1 Approved

Prepared by David Huson

**The University of West Florida
Florida Department of Transportation**

Table of Contents

1. Introduction	1
1.1 Purpose	1
1.2 Document Revisions Table	1
2. VTDS Test Framework	1
2.1 Framework Description	1
3. Data Integrity Tests	1
3.1 Data Consistency test	2
3.1.1 TestStringFieldInputValidation	2
3.1.2 TestDateTimeFieldInputValidation	2
3.2 Data Completeness Test	2
3.2.1 VerifyAllRequiredFieldsPresent	2
3.2.2 VerifyReturnAllNotNullFields	2
3.3 Data Accuracy Test	2
3.3.1 VerifyNewRecordMatchSource	2
4. Data Security Tests	2
4.1 User Authentication Test	2
4.1.1 VerifyUserHasAccess_Valid	2
4.1.2 VerifyUserHasAccess_Invalid	3
4.2 User Authorization Test	3
4.2.1 Access Tests	3
4.2.2 Access Permissions Modification Test	3
4.3 User Registration Tests	3
4.3.1 VerifyUserRegistration_WithToken	3
4.3.2 VerifyUserRegistraion_NoToken	3
4.4 SQL Injection Vulnerability Test	4
5. Performance Test	4
5.1 Query Execution Time Test	4
5.1.1 CheckQueryExecutionTime	4
6. Recovery Test	4
6.1 Backup and Restore Test	4
6.1.1 VerifyDataBackupProcedure	4
6.1.2 VerifyRestoreDataFromBackup	4
7. Transaction Log Test	5
7.1 Successful Transaction Tests	5
7.1.1 VerifyTransactionLog_Valid_POST	5
7.1.2 VerifyTransactionLog_Valid_GET	5
7.1.3 VerifyTransactionLog_Valid_PUT	5
7.1.4 VerifyTransactionLog_Valid_DELETE	5
7.2 Unsuccessful Transaction Tests	5

7.2.1 VerifyTransactionLog_Invalid_POST	5
7.2.2 VerifyTransactionLog_Invalid_GET	5
7.2.3 VerifyTransactionLog_Invalid_PUT	5
7.2.4 VerifyTransactionLog_Invalid_DELETE	5

1. Introduction

1.1 Purpose

This document describes the unit test framework of the Vehicle Threat Collection System (VTCS) as a subcomponent of the Connected Vehicle Security Metrics and Threat Intelligence Project. This document provides an overview of the unit test framework and tests for both the VTCS and the user interface.

1.2 Document Revisions Table

Revisor	Revision Date	Reason
David Huson	Jul 24, 2023	Initial draft
David Huson	Aug 25, 2023	Add SQL Injection Vulnerability test case
David Huson	Oct 9, 2023	Add Transaction Log and User Registration Tests
Elizabeth Uebele	Oct 12, 2023	Suggest document edits
David Huson	Oct 12, 2023	Accept proofreading edits
David Huson	Nov 7, 2023	Minor edits based on proof reading
David Huson	Nov 13, 2023	Sync with the rest of the deliverables

2. VTDS Test Framework

2.1 Framework Description

The VTDS project is developed in C# language and the choice was made to use the standard MsTest framework for the unit testing engine. MSTest is a native unit testing library that comes with Visual Studio from Microsoft. The SQL injection vulnerability tests will be carried out using an open source tool – SQLmap – which is designed to automate SQL injection attacks.

3. Data Integrity Tests

The Data Integrity Tests aim to ensure the consistency, accuracy, and completeness of the data stored in the database.

3.1 Data Consistency test

This test scenario involves checking if all the data entered in the database conforms to the defined data type and format.

3.1.1 TestStringFieldInputValidation

Verify that fields accepting string values (nvarchar) only accept strings as their input.

3.1.2 TestDateTimeFieldInputValidation

Verify that fields accepting dates (datetime2) only accept dates and times.

3.2 Data Completeness Test

This test scenario involves checking if all the expected fields are present and filled with the required information in the database. These tests will be repeated for all tables in the database.

3.2.1 VerifyAllRequiredFieldsPresent

Verify that required fields are present and filled with information.

3.2.2 VerifyReturnAllNotNullFields

Verify that optional fields are absent if not filled with information.

3.3 Data Accuracy Test

This test scenario involves comparing the data entered in the database with the source data and verifying its accuracy. These tests will be repeated for all tables in the database.

3.3.1 VerifyNewRecordMatchSource

Compare data entered in the database with the source data to ensure the accuracy of the data.

4. Data Security Tests

4.1 User Authentication Test

This test scenario involves verifying that the database requires user authentication and that only authorized users can access the database.

4.1.1 VerifyUserHasAccess_Valid

Verify that a user can access the database when providing the correct username and password.

4.1.2 VerifyUserHasAccess_Invalid

Verify that the system prohibits access to the database if the user fails to provide valid credentials.

4.2 User Authorization Test

This test scenario involves verifying that the database grants the appropriate level of access to the authenticated users.

4.2.1 Access Tests

Verify that only authorized users have access to the database API endpoints.

4.2.1.1 TestUnauthorizedUserAccess

Attempt to access the SQL database and the API with a user account that is not authorized to access it.

4.2.1.2 TestAuthorizedUserAccess

Attempt to access the SQL database and the API with a user account that is authorized to access it.

4.2.2 Access Permissions Modification Test

Verify that access permissions can be modified based on user roles and responsibilities.

4.2.2.1 TestAuthorizedUserCanCreateRole_Valid

Create an Application Role on the server with the authorized user.

4.2.2.2 TestAuthorizedUserCanCreateRole_Invalid

Attempt to create an Application Role on the server with the unauthorized user.

4.3 User Registration Tests

This test aims to ensure that a new user can create an account to gain basic access to the API

4.3.1 VerifyUserRegistration_WithToken

Verify a new user with a valid registration token can create an account with elevated privileges

4.3.2 VerifyUserRegistration_NoToken

Verify a new user can create an account with basic privileges

4.4 SQL Injection Vulnerability Test

This test scenario aims to ensure the VTMECS database controller is not susceptible to SQL injection attacks.

4.4.1 SQLmap injection test

This test will use the SQL map tool to automatically test the VTDS API for SQL injection vulnerabilities.

5. Performance Test

This test aims to evaluate the performance of the database.

5.1 Query Execution Time Test

This test scenario involves measuring the time to execute a series of complex SQL queries and ensuring that it meets the expected performance benchmarks.

5.1.1 CheckQueryExecutionTime

Run a complex query and measure the time taken to return the results.

6. Recovery Test

This test aims to ensure that the database can be recovered in the event of a failure.

6.1 Backup and Restore Test

This test scenario involves verifying that the database can be backed up and restored in the event of a failure.

6.1.1 VerifyDataBackupProcedure

Backup the database and verify that the backup file is created and saved in the appropriate location.

6.1.2 VerifyRestoreDataFromBackup

Restore the database from the backup file and verify that the data is restored correctly.

7. Transaction Log Test

This test case aims to ensure that the VTDS properly logs all transactions made via the VTDS API

7.1 Successful Transaction Tests

This test scenario aims to ensure all relevant information is logged for a successful transaction

7.1.1 VerifyTransactionLog_Valid_POST

Verify that the VTDS properly logs all relevant information about a successful POST transaction

7.1.2 VerifyTransactionLog_Valid_GET

Verify that the VTDS properly logs all relevant information about a successful GET transaction

7.1.3 VerifyTransactionLog_Valid_PUT

Verify that the VTDS properly logs all relevant information about a successful PUT transaction

7.1.4 VerifyTransactionLog_Valid_DELETE

Verify that the VTDS properly logs all relevant information about a successful DELETE transaction

7.2 Unsuccessful Transaction Tests

This test scenario aims to ensure all relevant information is logged for an unsuccessful transaction

7.3 VerifyTransactionLog_Invalid_POST

Verify that the VTDS properly logs all relevant information about an unsuccessful POST transaction

7.3.1 VerifyTransactionLog_Invalid_GET

Verify that the VTDS properly logs all relevant information about an unsuccessful GET transaction

7.3.2 VerifyTransactionLog_Invalid_PUT

Verify that the VTDS properly logs all relevant information about an unsuccessful PUT transaction

7.3.3 VerifyTransactionLog_Invalid_DELETE

Verify that the VTDS properly logs all relevant information about a failed DELETE transaction

Software Requirements Specification

for

Vehicle Threat Intelligence Portal (VTIP)

Technical Report UWF-TR-FDOT-009-01

Contract Number: BED34 Task Order: 977-01

Version 2.0

Prepared by:

David Huson, Daniel Miller, and Elizabeth Uebele

The University of West Florida

Florida Department of Transportation

January 30, 2024

Table of Contents

1. Introduction	322
1.1 Purpose	322
1.2 Document Conventions	322
1.3 References	322
1.4 Document Revisions Table	322
2. Overview of Product	324
2.1 Vehicle Threat Information Portal	324
2.2 User Classes and Characteristics	325
2.3 Operating Environment	325
2.4 Design and Implementation Constraints	326
2.5 Assumptions and Dependencies	326
3 Interface Requirements	326
3.1 The VTIP Threat Record Interface	327
4. Functional Requirements	330
4.1 FR-1: Query Collected Threats From VTDS	330
4.1.1 Description and Priority	330
4.1.2 Related User Classes	330
4.1.3 Functional Requirements	330
4.2 FR-2: Display Collected Threats From VTDS	330
4.2.1 Description and Priority	330
4.2.2 Related User Classes	330
4.2.3 Functional Requirements	330
4.3 FR-3: Update Threat Record Information in VTDS	331
4.3.1 Description and Priority	331
4.3.2 Related User Classes	331
4.3.3 Functional Requirements	331
4.4 FR-4: MISP Server Access	332
4.4.1 Description and Priority	332
4.4.2 Related User Classes	332
4.4.3 Functional Requirements	332
4.5 FR-5: Export Threat Record	332
4.5.1 Description and Priority	332
4.5.2 Related User Classes	332
4.5.3 Functional Requirements	332
4.6 FR-6: Explore Threat Records	333
4.6.1 Description and Priority	333
4.6.2 Related User Classes	333
4.6.3 Functional Requirements	333

5. Test Requirements	333
5.1 <i>T-1: Data Accuracy Tests</i>	333
5.2 <i>T-2: Data Security Tests</i>	333
5.3 <i>T-3: Performance Tests</i>	334
5.4 <i>T-4: Data Export Tests</i>	334
6. Non-Functional Requirements	334
6.1 <i>NF-1: Portability</i>	334
6.2 <i>NF-2: Usability</i>	334
6.3 <i>NF-3: Speed</i>	334
6.4 <i>NF-4: Role-based Record Modification</i>	334
7. Quality Attributes	335
8. Source Code Repository and Version Control Requirement	335
9. Appendix A: Requirements Table	336
10. Appendix B: Requirements Traceability Matrix	340
11. Appendix C: Glossary	344

1 Introduction

1.1 Purpose

This document describes the software requirements for Version 1.0 of the Vehicle Threat Intelligence Portal (VTIP) as a subcomponent of the Connected Vehicle Security Metrics and Threat Intelligence Project (Project System). This document provides an overview of the users and context of the VTIP and covers all functional, non-functional, and data requirements of the VTIP.

1.2 Document Conventions

This document is based on the IEEE 830 Standards and the Florida Department of Transportation Requirements Standards. Specific conventions used in this document are listed below:

- Priorities are indicated for each feature as well as in the Requirements Table. A green highlighting indicates must have features, while a yellow highlight represents a should have feature.
- Testing statuses are indicated for each feature as well as in the Requirements Table. A green highlighting indicates the test has passed, a yellow highlight represents a test has not been implemented yet, and a red highlighting represents the test has failed.
- Requirements follow the form of <TAG>-#.#.# where a tag indicates a category of requirements. And the # represents the id of the requirement in a hierarchical fashion.

1.3 References

The following references were used in the creation of this document:

- IEEE 830 Standards on Software Requirement Specifications
- UWF Scope of Service Document for the Connected Vehicle Security Metrics and Threat Intelligence Project

1.4 Document Revisions Table

Revisor	Revision Date	Reason
Elizabeth Uebele	January 30, 2024	Proofread and edited
Daniel Miller	January 30, 2024	Added Prototype UI graphics to section 3
David Hudson	January 29, 2024	Updated test table information
Elizabeth Uebele	January 12, 2024	Proofread and edited

Daniel Miller	January 12, 2024	Reviewed entire document for formatting corrections as part of Final draft turn-in
Daniel Miller	January 3, 2024	Section 3: reworded and finalized all subsections except GUI figures
David Huson	December 29, 2023	Section 9: add all testing fields Section 10: add all testing fields
David Huson	December 29, 2023	Section 5: updated and finalized all sub sections
Daniel Miller	December 23, 2023	Section 9: updated all fields Section 10: updated all fields
Daniel Miller	December 15, 2023	Section 4: reworded and finalized all subsections
Daniel Miller	December 8, 2023	Section 4: updated areas marked incomplete. Appendix A: updated areas marked incomplete. Appendix B: updated areas marked incomplete.
Daniel Miller	November 24, 2023	Appendix B: added appendix B and all sections completed with at least baseline info. Highlighted sections indicate incomplete.
Daniel Miller	October 27, 2023	Appendix A: added appendix A and all sections completed with at least baseline info. Highlighted sections indicate incomplete.
Daniel Miller	October 13, 2023	Section 4: updated with all approved FRs and all sections completed with at least baseline info. Highlighted sections indicate incomplete.
Daniel Miller	September 24, 2023	Section 4: Updated additional Functional Requirements. still needs lots of work. waiting on approval of initial FR/NFRs
Daniel Miller	September 22, 2023	Section 2.3: Updated/Added additional Operational Environment elements.

Elizabeth Uebele	September 21, 2023	Wrote basic outline of the document
------------------	--------------------	-------------------------------------

2 Overview of Product

2.1 Vehicle Threat Information Portal

The VTIP functions as a subcomponent within the Project System, serving as the front-end Graphical User Interface (GUI) for secure user access to essential features of the Project system. It enables users to query and modify threat records stored in the Vehicle Threat Database System (VTDS). Additionally, the VTIP acts as the primary interface for users to access the Malware Information Sharing Platform (MISP) Server subcomponent of the Project System for threat intelligence information sharing.

Key functionalities of the VTIP include the ability to query stored threat records based on date range, vehicle information, CVE ID, and/or keyword. The system presents threat records to users in a comprehensible format, displaying all available threat record information, such as CVE ID, publish/modification date, CWE information, and threat description. The VTIP also provides role-based record modifications, enabling the correction or addition of data to a threat record based on user derived information. This ensures the accuracy and completeness of threat intelligence information within the VTDS.

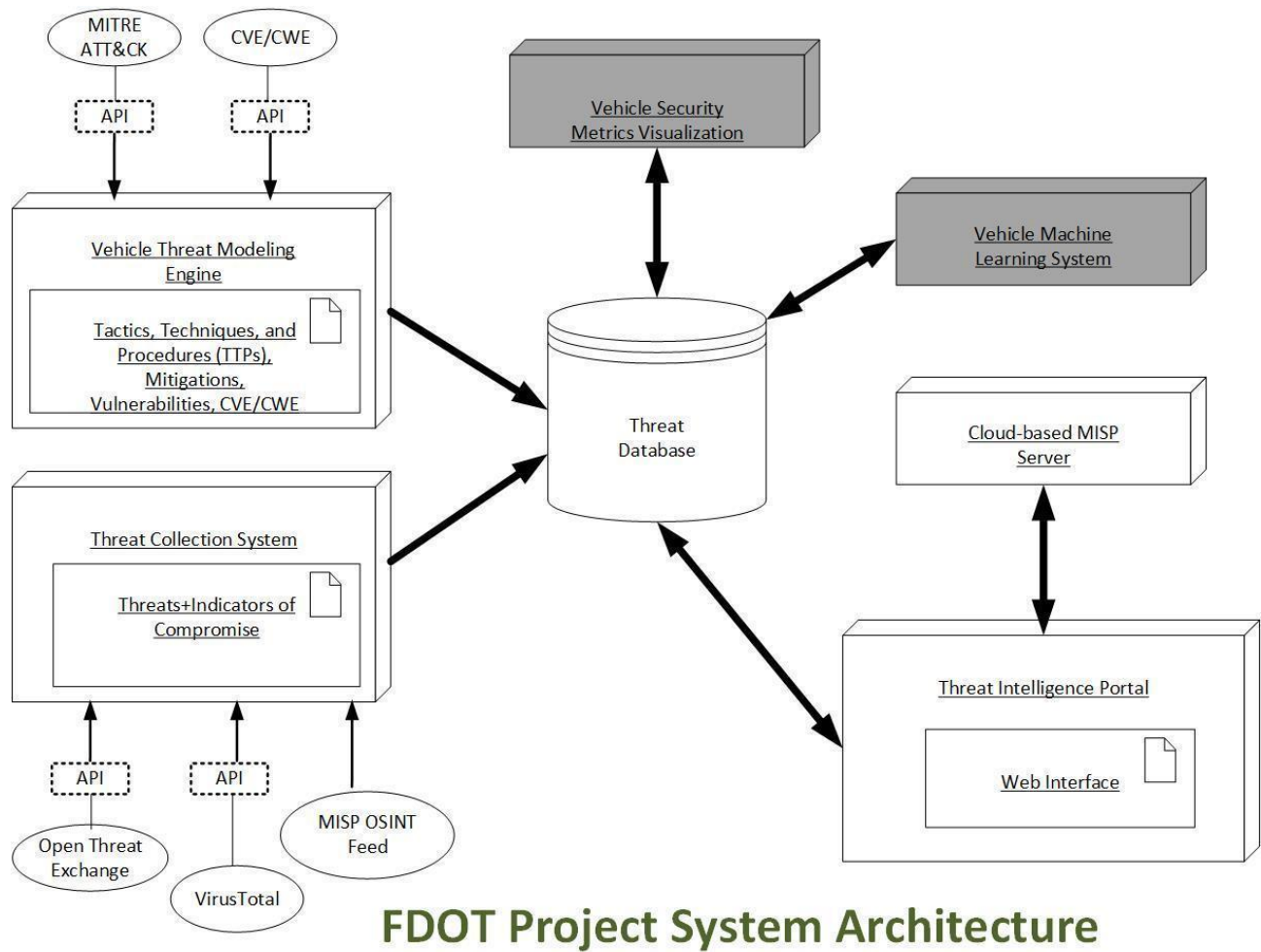


Figure 2.1 FDOT Project System Architecture

2.2 User Classes and Characteristics

User Class	Characteristics
Threat Analyst	This user will utilize the VTIP to search for CVEs to use for analysis.
Threat Model Builder	This user will utilize the VTME to build threat models based on the Cyber Kill Chain stages.
Threat Collector	This user will utilize the VTCS to retrieve additional information for the models.
Database Administrator	Administers the Threat DB

2.3 Operating Environment

The VTIP operating environment is defined by the following:

OE-1: The VTIP shall run within an Amazon Elastic Compute Cloud (EC2) Web Service utilizing a Windows Server environment.

OE-2: The VTIP shall run as a .NET application on an Internet Information Services (IIS) on a Windows Server within the AWS EC2 instance.

OE-3: The VTIP AWS EC2 instance shall be configured with type t2.xlarge having 4 vCPU and 16 GB of memory.

OE-4: The VTIP shall interact with the VTDS backend. The VTDS will be designed and implemented as a major deliverable of the project.

OE-5: The VTIP shall interact with the MISP backend. The MISP will be designed and implemented as a major deliverable of the project.

2.4 Design and Implementation Constraints

DIC-1: The VTIP will be developed using Microsoft Visual Studio 2022.

DIC-2: The VTIP will be developed using the C# programming language.

DIC-3: The VTIP will be developed using .NET Core.

DIC-4: The VTIP will be constrained by the information available from the VTDS.

DIC-5: The VTIP will be designed and implemented as a working prototype capable of future expansion.

2.5 Assumptions and Dependencies

ASS-1: The VTIP assumes availability of information from the VTDBS.

ASS-2: The VTIP assumes availability of information from the MISP Server.

DEP-1: The VTIP will be dependent on information from the VTDS.

DEP-2: The VTIP will be dependent on information from the MISP Server.

3 Interface Requirements

INT-1: The VTIP will provide a Graphical User Interface for querying threat records in the VTDS.

INT-2: The VTIP will provide a Graphical User Interface for displaying queried threat records.

INT-3: The VTIP will provide a Graphical User Interface for modifying queried threat records.

INT-4: The VTIP will provide a Graphical User Interface for importing and exporting threat record information to and from the MISP server.

INT-5: The VTIP will provide a Graphical User Interface for exporting threat records into PDF format.

INT-6: The VTIP will provide a Graphical User Interface for providing users with hyperlinked threat record source information.

3.1 The VTIP Threat Record Interface

The VTIP Threat Record Interface will be used to display the records generated by querying the VTDS. The upper portion of the GUI will display a threat record search functionality where users may query the VTIP by date range, vehicle information, CVE ID, and/or keyword. The center and lower portions of the GUI will display the threat record information to include CVE ID, Threat record publish information, CWE information, threat record description, CVSS information, vehicle information, and Indicators of Compromise (IOC).

VTIP Interface Prototype: Search and Display Threat Record

CVE ID: Date Start: Date End: Keyword: [Search](#)

Threat Record Information

CVE ID: CVE-0001 dig	Publish Info: 09/16/2017	CWE Information: 74.88
CVSS Information: Base Score: 6.8 MEDIUM Attack Vector (AV): Physical	Impact Score: 5.9 Attack Complexity (AC): Low	Exploitability Score: 0.9 Privileges Required (PR): None

Threat Description:
Toyota RAV4 2021 vehicles automatically trust messages from other ECUs on a CAN bus, which allows physically proximate attackers to drive a vehicle by accessing the control CAN bus after pulling the bumper away and reaching the headlight connector, and then sending forged "Key is validated" messages via CAN injection, as exploited in the wild in (for example) July 2022.

Vehicle Information:
[Add](#)

IOCs:
[... Edit](#)

[Update Record](#) [Access MISP](#) [Export Record](#)

VTIP Records

- CVE-0001
- CVE-0002
- CVE-0003
- CVE-0004
- CVE-0005

[Previous](#) [Next](#)

Figure 3.1 VTIP Interface Prototype

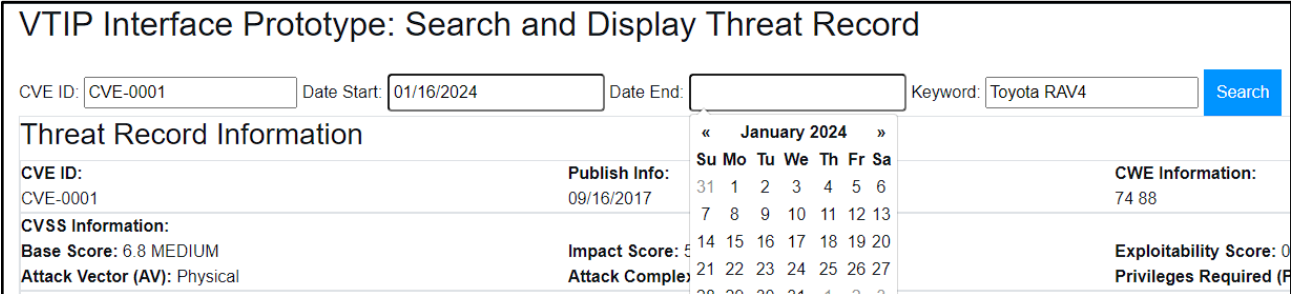


Figure 3.2 VTIP Interface Prototype: Search Threat Record

The VTIP Threat Record Interface will provide an edit data mechanism allowing users to make changes to threat record data in the VTDS based on analyst derived information. This includes the ability to add/remove/edit data such as vehicle information or IOCs.

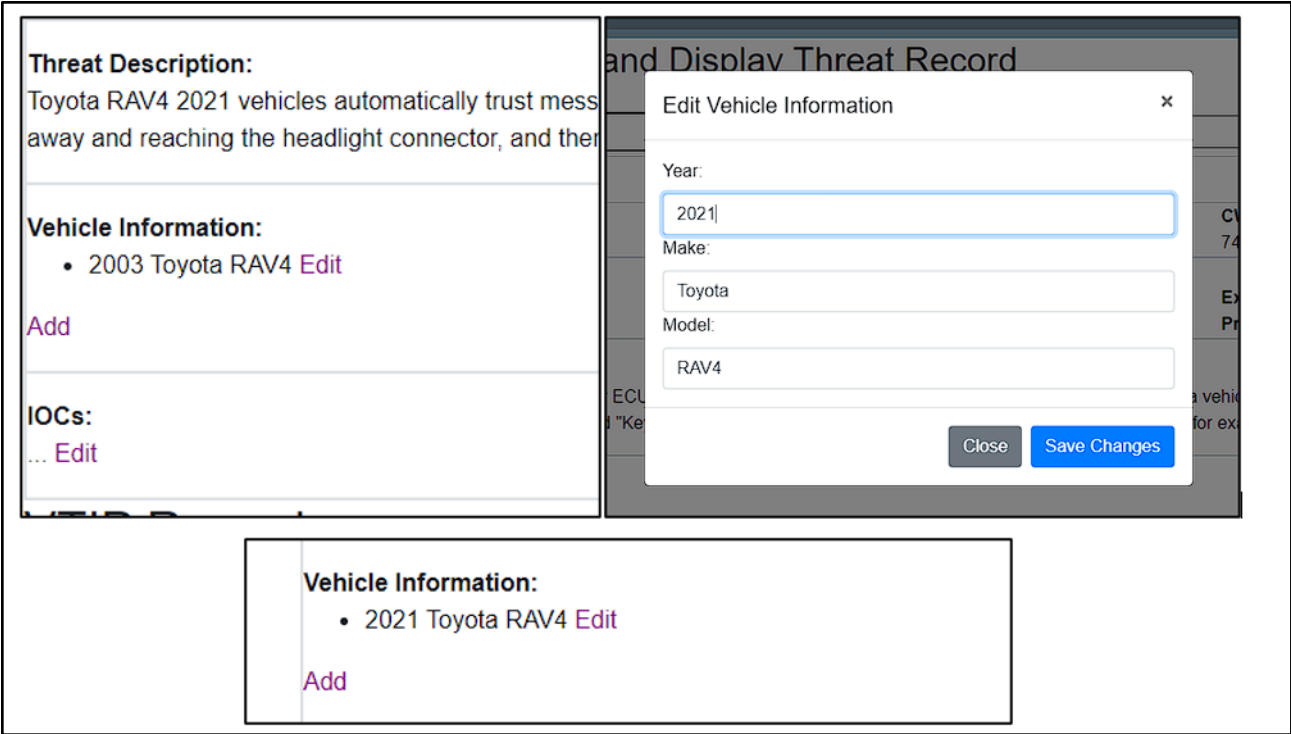


Figure 3.3 VTIP Interface Prototype: Modify Threat Record

The VTIP Threat Record Interface will provide MISP server access to users through an "Access MISP" button allowing them to move to the MISP server interface regarding the displayed threat record.



Figure 3.4 VTIP Interface Prototype: MISP Server Access

The VTIP Threat Record Interface will provide a mechanism to allow users to export one or more selected threat records into a downloadable PDF format.

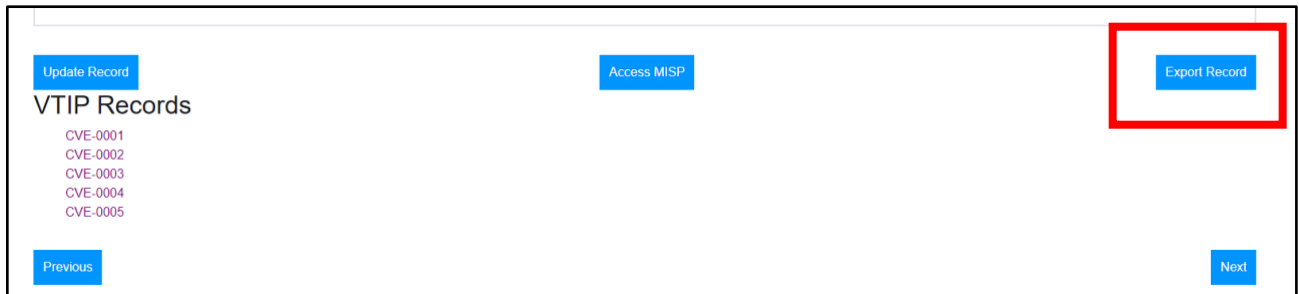


Figure 3.5 VTIP Interface Prototype: Generate Threat Record Report

The VTIP Threat Record Interface will provide users with a “dig-deeper” mechanism that will display a pop-up window containing the hyperlinked addresses of a threat record's source information. The user will be able to select a threat records source from the pop-up window which will open the hyperlink address in a different web browser tab.

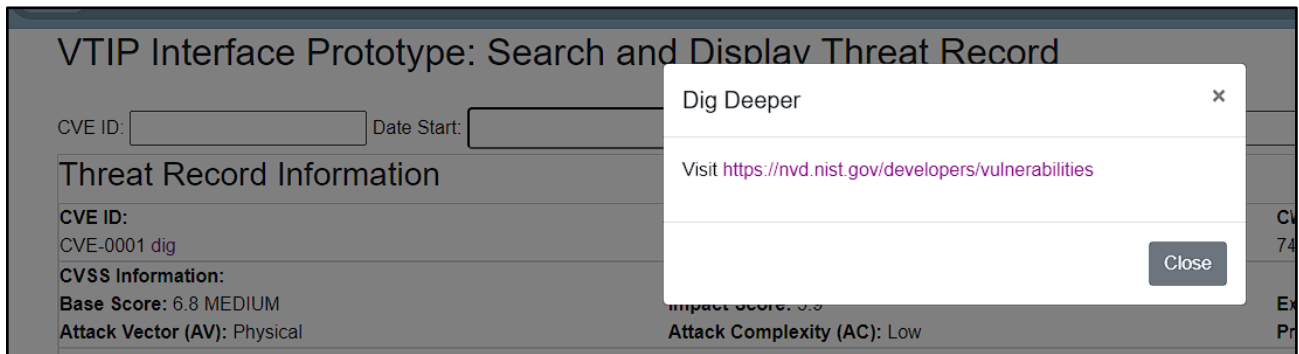


Figure 3.6 VTIP Interface Prototype: Dig Deeper

4. Functional Requirements

Functional requirements for the VTIP are fundamental actions that the system must execute to be considered operational. The VTIP is a proof of concept and as such, limitations in its implementation will be identified.

4.1 FR-1: Query Collected Threats From VTDS

4.1.1 Description and Priority

The VTIP will allow a user to query stored threat records based on date range, vehicle information, CVE ID, and/or keyword.

Priority: Must Have

4.1.2 Related User Classes

Threat Analyst

4.1.3 Functional Requirements

FR-1.1: The VTIP will query threat records from the VTDS.

FR-1.2: The VTIP will query the VTDS using the following attributes:

- Threat activity date range
- Vehicle Information (year, make, model)
- CVE record ID
- Keyword search within the description of the threat record

The user interface prototype is depicted in Figure 3.1.

4.2 FR-2: Display Collected Threats From VTDS

4.1.1 Description and Priority

The VTIP will display threat records to users in a comprehensible format, displaying all available threat record information such as CVE ID, publish/modification date, CWE information, and threat description.

Priority: Must Have

4.1.2 Related User Classes

Threat Analyst

4.1.3 Functional Requirements

FR-2.1: The VTIP will display queried Threat records contained in the VTDS.

FR-2.2: The VTIP Threat Record Viewer must display the following threat attributes:

- CVE ID
- Threat record publish and/or most recent modification date
- CWE information related to the threat record
- Threat description
- CVSS information
- Vehicle information
- IOCs

The user interface prototype is depicted in Figure 3.1.

4.3. FR-3: Update Threat Record Information in VTDS

4.3.1 Description and Priority

The VTIP will allow users to make modifications to threat records in the VTDS based on analyst derived information. This includes the ability to add/remove/edit data such as vehicle information, CWE information, or IOCs.

Priority: Must Have

4.3.2 Related User Classes

Threat Analyst

4.3.3 Functional Requirements

FR-3.1: The VTIP will allow a user to make updates to existing threat records based on analyst derived information.

FR-3.2: The VTIP will allow the following attributes to be edited:

- Vehicle information (year, make, model)
- CWE information
- IOCs

The user interface prototype is depicted in Figure 3.2.

4.4 FR-4: MISP Server Access

4.4.1 Description and Priority

The VTIP will serve as the main access point for analysts to import and export threat record information from the VTDS to the MISP server information sharing platform.

Priority: Must Have

4.4.2 Related User Classes

Threat Analyst

4.4.3 Functional Requirements

FR-4.1: The VTIP will allow users to update threat records based on MISP server data.

FR-4.2: The VTIP will allow users to upload threat records to the MISP server.

The user interface prototype is depicted in Figure 3.3.

4.5 FR-5: Export Threat Record

4.5.1 Description and Priority

The VTIP will provide users with the ability to export selected threat records into comprehensible PDF documents.

Priority: Must Have

4.5.2 Related User Classes

Threat Analyst

4.5.3 Functional Requirements

FR-5.1: The VTIP will provide export records functionality to allow users to download records in PDF format.

FR-5.2: The VTIP will format exported threat records in easy-to-understand, human readable format.

The user interface prototype is depicted in Figure 3.4.

4.6 FR-6: Explore Threat Records

4.6.1 Description and Priority

The VTIP will provide users with all source information associated with a threat record. This provides threat analysts with hyperlink addresses to sources reporting the threat information within the record.

Priority: Must Have

4.6.2 Related User Classes

Threat Analyst

4.6.3 Functional Requirements

FR-6.1: The VTIP will provide users with hyperlink addresses to the source information provided for the threat record.

FR-6.2: The VTIP will open a new web browser tab to the hyperlinked address when the analyst utilizes this functionality.

The user interface prototype is depicted in Figure 3.5.

5 Test Requirements

The VTIP requires testing and validation of the main application functionalities.

5.1 T-1: Data Accuracy Tests

This test case aims to ensure that the data retrieved from all sources is accurate and maintains accuracy after modifications by threat analysts. It includes the following test scenarios:

- VTDS Data Retrieval Tests
- MISP Data Retrieval Tests
- VTDS Data Modification Tests
- MISP Data Modification Tests

5.2 T-2: Data Security Tests

This test case aims to verify the security of the VTIP against SQL injection attacks. It includes the following test scenario:

- SQL Injection Vulnerability Tests

5.3 T-3: Performance Tests

This test case aims to test the performance of the VTIP. It includes the following test scenarios:

- TestServerResponseTimes_VTDS
 - Tests the server response times for the required VTDS API requests
- TestServerResponseTimes_MISP
 - Tests the server response times for the required MISP API requests

5.4 T-4: Data Export Tests

This test case will test the data exporting feature of the VTIP. It includes the following subtest:

Test Data Export to PDF

6 Non-Functional Requirements

Non-functional requirements for the VTIP are system attributes that are desired but not required.

The following are the non-functional requirements for the VTIP:

6.1 NF-1: Portability

The development team will attempt to make the web enabled VTIP system portable across multiple computing form factors.

6.2 NF-2: Usability

The development team will attempt to satisfy system usability features such as navigation, performance quality, and intuitiveness of interfaces.

6.3 NF-3: Speed

The development team will attempt to enhance the VTIP system responsiveness to user interactions and database transactions.

6.4 NF-4: Role-based Record Modification

The VTIP will implement a role-based record modification schema that will only allow authorized users to make modifications to a threat record stored in the VTDS as outlined in *FR-3*.

7 Quality Attributes

The VTIP is a proof of concept and not meant for a production release. As such, traditional quality attributes such as availability, security, robustness, etc. are not as relevant.

1. Source Code Repository and Version Control Requirement

The development team shall facilitate a source code repository and version control for the project.

8 SC-1: Source Code Repository and Control

The development team shall maintain a source code repository and version control on GitHub.

The GitHub project URL is <https://github.com/UWF-CfC-FDOT/VTIP>.

9 Appendix A: Requirements Table

Requirement ID	Requirement Type	Requirement Name	Requirement Description	Priority
S1	Scope	VTIP System Scope	The VTIP is required to allow a user to query, display, and modify threat records from the VTDS and allow for the query/display of threat record MSIP server data.	Must have
FR-1	Functional	Query Collected Threats From VTDS	The VTIP is required to allow a user to query threat records from the VTDS.	Must have
FR-2	Functional	Display Collected Threats From VTDS	The VTIP is required to display threat record information queried from the VTDS.	Must have
FR-3	Functional	Update Threat Record Information in VTDS	The VTIP is required to allow a user to modify threat record information.	Must have
FR-4	Functional	MISP Server Access	The VTIP is required to import and export threat record information from the VTDS to the MISP Server	Must have
FR-5	Functional	Export Threat Record	The VTIP is required to export selected threat records into PDF format.	Must have
FR-6	Functional	Explore Threat Records	The VTIP is required to allow a user to research a threat record further by providing all available threat record data sources.	Must have

INT-1	Interface	Query Collected Threats From VTDS	The VTIP is required to provide a Graphical User Interface for querying threat records in the VTDS.	Must have
INT-2	Interface	Display Collected Threats From VTDS	The VTIP is required to provide a Graphical User Interface for displaying queried threat records.	Must have
INT-3	Interface	Update Threat Record Information in VTDS	The VTIP is required to provide a Graphical User Interface for modifying queried threat records.	Must have
INT-4	Interface	MISP Server Access	The VTIP is required to provide a Graphical User Interface for importing and exporting threat record information to and from the MISP server.	Must have
INT-5	Interface	Export Threat Record	The VTIP is required to provide a Graphical User Interface for exporting threat records into PDF format.	Must have
INT-6	Interface	Explore Threat Records	The VTIP is required to provide a Graphical User Interface for providing users with hyperlinked threat record source information.	Must have

T-1	Test	Data Accuracy Tests	This test case aims to ensure that the data retrieved from all sources is accurate and maintains accuracy after modifications by threat analysts.	Must have
T-2	Test	Data Security Tests	This test case aims to verify the security of the VTIP against SQL injection attacks.	Must have
T-3	Test	Performance Test	This test scenario involves testing the response times for different MISP API calls of varying complexity.	Must have
T-4	Test	Data Export Tests	This test case will test the data exporting feature of the VTIP	Must have
NF-1	Non-functional	Portability	The development team will attempt to make the web enabled VTIP portable across multiple computing form factors	Could have
NF-2	Non-functional	Usability	The development team will attempt to satisfy system usability features such as navigation, performance quality, and intuitiveness of interfaces	Could have

NF-3	Non-functional	Speed	The development team will attempt to enhance the VTIP system responsiveness to user interactions and database transactions.	Could have
NF-4	Non-functional	Role-based Record Modification	The VTIP will implement a role-based record modification schema allowing authorized users to make threat record modifications as outlined in FR-3.	Could have
SC-1	Source Code	Source Code Control	The development team shall maintain a source code repository and version control on GitHub	Should have

10 Appendix B: Requirements Traceability Matrix

Requirement ID	Requirement Description	Test Case	Status
S1	The VTIP is required to allow a user to query, display, and modify threat records from the VTDS and allow for the query/display of threat record MSIP server data.	N/A	N/A
FR-1	The VTIP is required to allow a user to query threat records from the VTDS.	Test Case: 3.1, 7	Not Started
FR-2	The VTIP is required to display threat record information queried from the VTDS.	Test Case: 3.1, 7	Not Started
FR-3	The VTIP is required to allow a user to modify threat record information.	Test Case: 3.3, 7	Not Started
FR-4	The VTIP is required to import and export threat record information from the VTDS to the MISP Server	Test Case: 3.2, 7	Not Started
FR-5	The VTIP is required to export selected threat records into PDF format.	Test Case: 6	Not Started

FR-6	The VTIP is required to allow a user to research a threat record further by providing all available threat record data sources.	Test Case: 3.1, 7	Not Started
INT-1	The VTIP is required to provide a Graphical User Interface for querying threat records in the VTDS.	Test Case: 3.1, 7	N/A
INT-2	The VTIP is required to provide a Graphical User Interface for displaying queried threat records.	Test Case: 3.1, 7	N/A
INT-3	The VTIP is required to provide a Graphical User Interface for modifying queried threat records.	Test Case: 3.3, 7	N/A
INT-4	The VTIP is required to provide a Graphical User Interface for importing and exporting threat record information to and from the MISP server.	Test Case 3.2, 7	N/A
INT-5	The VTIP is required to provide a Graphical User Interface for exporting threat records into PDF format.	Test Case: 6	N/A

INT-6	The VTIP is required to provide a Graphical User Interface for providing users with hyperlinked threat record source information.	Test Case: 3.1, 7	Not Started
T-1	This test case aims to ensure that the data retrieved from all sources is accurate and maintains accuracy after modifications by threat analysts.	Test Case: 3	Not Started
T-2	This test case aims to verify the security of the VTIP against SQL injection attacks.	Test Case: 4	Not Started
T-3	This test scenario involves testing the response times for different MISP API calls of varying complexity.	Test Case: 5	Not Started
T-4	This test case will test the data exporting feature of the VTIP	Test Case: 6	Not Started
NF-1	The development team will attempt to make the web enabled VTIP portable across multiple computing form factors	N/A	N/A

NF-2	The development team will attempt to satisfy system usability features such as navigation, performance quality, and intuitiveness of interfaces	N/A	N/A
NF-3	The development team will attempt to enhance the VTIP system responsiveness to user interactions and database transactions.	N/A	N/A
NF-4	The development team will attempt to implement role-based modification of existing threat records in the VTDS	N/A	N/A
SC-1	The development team shall maintain a source code repository and version control on GitHub	Github Access	No

11 Appendix C: Glossary

Term	Description
API	Application Program Interface
ATT&CK	Adversarial Tactics, Techniques, and Common Knowledge
ATT&CK Framework	A knowledge base of adversary tactics and techniques based on real-world observations.
AWS	Amazon Web Services
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
CWE	Common Weakness Enumeration
Cyber Kill Chain	A model developed by Lockheed Martin® used for the identification and prevention of cyber intrusions.
EC2	Elastic Compute Cloud
GUI	Graphical User Interface
IIS	Internet Information Services
IoC	Indicators of Compromise
MISP	Malware Information Sharing Platform
.NET	A cross-platform, open-source developer platform created by Microsoft
OSINT	Open-Source Intelligence
OTX	Open Threat Exchange
VTCS	Vehicle Threat Collection System
VTDBS	Vehicle Threat Database System
VTME	Vehicle Threat Modeling Engine

Appendix XIX. Technical Report UWF-TR-FDOT-009-02

Vehicle Threat Information Portal (VTIP)- Requirements Traceability Matrix		Contract Number: BED34 Task Order: 977-01					
<i>The Vehicle Threat Information Portal (VTIP) is a subcomponent of the Connected Vehicle Security Metrics and Threat Intelligence Project.</i>							
<i>Dr. Guillermo Francia, III</i>							
<i>UWF-FDOT</i>							
User Need Summary	Requirement ID	Detailed Requirement Summary	Document Section	DR Source Document	Verification Test Case ID	Compliance (Y/N/Partial/NA)	Notes/Comments/Date
VTIP System Scope	S1	The VTIP is required to allow a user to query, display, and modify threat records from the VTDS and allow for the query/display of threat record MSIP server data.	2.1	VTIP Requirements	NA	NA	Non Testable
Query Collected Threats From VTDS	FR-1	The VTIP is required to allow a user to query threat records from the VTDS.	4.1	VTIP Requirements	Test Case: 3.1, 7	No	These test cases will be completed after the implementation
Display Collected Threats From VTDS	FR-2	The VTIP is required to display threat record information queried from the VTDS.	4.2	VTIP Requirements	Test Case: 3.1, 7	No	These test cases will be completed after the implementation

Update Threat Record Information in VTDS	FR-3	The VTIP is required to allow a user to modify threat record information.	4.3	VTIP Requirements	Test Case: 3.3, 7	No	These test cases will be completed after the implementation
Display MISP Server Data from Threat Record	FR-4	The VTIP is required to display MISP server data from threat records.	4.4	VTIP Requirements	Test Case: 3.2, 7	No	These test cases will be completed after the implementation
Export Threat Record	FR-5	The VTIP is required to export selected threat records into PDF format.	4.5	VTIP Requirements	Test Case: 6	No	These test cases will be completed after the implementation
Explore Threat Records	FR-6	The VTIP is required to allow a user to research a threat record further by providing all available threat record data sources.	4.6	VTIP Requirements	Test Case: 3.1, 7	No	These test cases will be completed after the implementation
Data Accuracy Tests	T-1	This test case aims to ensure that the data retrieved from all sources is accurate and maintains accuracy after modifications by threat analysts.	5.1	VTIP Requirements	Test Case: 3	No	These test cases will be completed after the implementation
Data Security Tests	T-2	This test case aims to verify the security of the VTIP against SQL injection attacks.	5.2	VTIP Requirements	Test Case: 4	No	These test cases will be completed after the implementation
Performance Test	T-3	This test scenario involves testing the response times for different MISP API calls of varying complexity.	5.3	VTIP Requirements	Test Case: 5	No	These test cases will be completed after the implementation
Data Export Tests	T-4	This test case will test the data exporting feature of the VTIP	5.4	VTIP Requirements	Test Case: 6	No	These test cases will be completed after the implementation

Portability	NF-1	The development team will attempt to make the web enabled VTIP portable across multiple computing form factors	6.1	VTIP Requirements	N/A	NA	No tests will be written for this at this time. This is beyond the scope of this project and will require additional support from the VTDS to implement and test.
Usability	NF-2	The development team will attempt to satisfy system usability features such as navigation, performance quality, and intuitiveness of interfaces	6.2	VTIP Requirements	N/A	NA	No tests will be written for this at this time. This is beyond the scope of this project and will require additional support from the VTDS to implement and test.
Speed	NF-3	The development team will attempt to enhance the VTIP system responsiveness to user interactions and database transactions.	6.3	VTIP Requirements	N/A	NA	No tests will be written for this at this time. This is beyond the scope of this project and will require additional support from the VTDS to implement and test.
Role-based Record Modification	NF-4	The development team will attempt to implement role-based modification of existing threat records in the VTDS	6.4	VTIP Requirements	NA	NA	No tests will be written for this at this time. This is beyond the scope of this project and will require additional support from the VTDS to implement and test
Query Collected Threats From VTDS	INT-1	The VTIP is required to provide a Graphical User Interface for querying threat records in the VTDS.	3.1	VTIP Requirements	Test Case: 3.1, 7	No	
Display Collected Threats From VTDS	INT-2	The VTIP is required to provide a Graphical User	3.1	VTIP Requirements	Test Case: 3.1, 7	No	

		Interface for displaying queried threat records.					
Update Threat Record Information in VTDS	INT-3	The VTIP is required to provide a Graphical User Interface for modifying queried threat records.	3.1	VTIP Requirements	Test Case: 3.3, 7	No	
MISP Server Access	INT-4	The VTIP is required to provide a Graphical User Interface for importing and exporting threat record information to and from the MISP server.	3.1	VTIP Requirements	Test Case 3.2, 7	No	
Export Threat Record	INT-5	The VTIP is required to provide a Graphical User Interface for exporting threat records into PDF format.	3.1	VTIP Requirements	Test Case: 6	No	
Explore Threat Records	INT-6	The VTIP is required to provide a Graphical User Interface for providing users with hyperlinked threat record source information.	3.1	VTIP Requirements	Test Case: 3.1, 7	No	Non-Testable
Source Code Control	SC-1	The development team shall maintain a source code repository and version control on GitHub	8	VTIP Requirements	Github Access	No	To be completed when the source code is made available on Github

Unit Test Plan Overview

for

Vehicle Threat Intelligence Portal (VTIP)

Technical Report UWF-TR-FDOT-009-03

Contract Number: BED34 Task Order: 977-01

Version 2.0 Approved

Prepared by David Huson

The University of West Florida

Florida Department of Transportation

January 18, 2024

Table of Contents

1. Introduction	352
1.1 Purpose	352
1.2 Document Revisions Table	352
2. VTIP Test Framework	352
2.1 Framework Description	352
3. Data Accuracy Tests	352
3.1 VTIP Data Retrieval Tests	352
3.1.1 VerifyThreatDataExists_Valid	352
3.1.2 VerifyThreatDataExists_Invalid	352
3.2 MISP Data Retrieval Tests	352
3.2.1 VerifyAllRequiredFieldsPresent	352
3.2.2 VerifyThreatDataExists_Valid	353
3.2.3 VerifyThreatDataExists_Invalid	353
3.3 VTIP Data Modification Tests	353
3.3.1 VerifyDataChangeOccurred	353
3.4 MISP Data Modification Tests	353
3.4.1 VerifyDataChangeOccurred	353
4. Data Security Tests	353
4.1 SQL Injection Vulnerability Test	353
5. Performance Tests	353
5.1 TestServerResponseTimes_VTDS	353
5.1.1 GET_SingleThreat	353
5.1.2 GET_MultiThreat	354
5.1.3 POST_Threat	354
5.1.4 PUT_Threat	354
5.2 TestServerResponseTimes_MISP	354
5.2.1 GET_SingleThreat	354
5.2.2 GET_MultiThreat	354
5.2.3 POST_Threat	354
5.2.4 PUT_Threat	354
6. Data Export Tests	354
6.1 TestDataExportToPDF	354
7. UI tests	354
7.1 Submit Button Tests	354
7.1.1 Submit search of existing data	354
7.1.2 Submit search of non-existent data	354
7.2 Back Button Tests	355
7.2.1 BackPressedWhileOnFirstRecord	355
7.2.2 BackPressedWhileNotOnFirstRecord	355

7.3	<i>Next Button Tests</i>	355
7.3.1	NextPressedWhileOnLastRecord	355
7.3.2	NextPressedWhileNotOnLastRecord	355
7.4	<i>Reset Button Tests</i>	355
7.4.1	ResetPressed_NoModifications	355
7.4.2	ResetPressed_ModificationsMade	355
7.5	<i>Data Field Modification Tests</i>	355
7.5.1	DateTime validation test - VALID case	355
7.5.2	datetiDateTimeme validation test - INVALID case	355
7.5.3	CWE_ID validation test - VALID case	355
7.5.4	CWE_ID validation test - INVALID case	355

1 Introduction

1.1 Purpose

This document describes the unit test framework of the Vehicle Threat Information Portal (VTIP) as a subcomponent of the Connected Vehicle Security Metrics and Threat Intelligence Project. This document provides an overview of the unit test framework and tests for both the VTIP and the Vehicle Threat Database System (VTDS).

1.2 Document Revisions Table

Revisor	Revision Date	Reason
Chase Lamkin	Jan 18, 2024	Describe UI tests
David Huson	Jan 15, 2024	Add UI tests
David Huson	Dec 28, 2023	Initial draft

2 VTIP Test Framework

2.1 Framework Description

The VTIP project is developed in C# language and the choice was made to use the standard MsTest framework for the unit testing engine. MSTest is a native unit testing library that comes with Visual Studio from Microsoft. The SQL injection vulnerability tests will be carried out using an open-source tool – SQLmap – which is designed to automate SQL injection attacks. Any tests which require a database connection should instead use a mock database so we can control the data and ensure accuracy without relying on a database connection.

3 Data Accuracy Tests

This test scenario aims to ensure that the data retrieved from all sources is accurate and maintains accuracy after modifications by threat analysts.

3.1 VTIP Data Retrieval Tests

This test case will test the VTIP's ability to retrieve data from the VTDS.

This test will verify the data retrieved from the VTDS contains at a minimum all not-null fields.

3.1.1 VerifyThreatDataExists_Valid

This test verifies that a request for an existing threat record returns the appropriate data.

3.1.2 VerifyThreatDataExists_Invalid

This test verifies that a request for a non-existent threat record returns the appropriate response.

3.2 MISP Data Retrieval Tests

This test scenario will test the VTIP's ability to retrieve data from the Malware Information Sharing Platform (MISP).

3.2.1 VerifyAllRequiredFieldsPresent

This test will verify the data retrieved from the MISP server contains at a minimum all not-null fields.

3.2.2 VerifyThreatDataExists_Valid

This test verifies that a request for an existing threat record returns the appropriate data.

3.2.3 VerifyThreatDataExists_Invalid

This test verifies that a request for a non-existent threat record returns the appropriate response.

3.3 VTIP Data Modification Tests

This test scenario verifies that the data stored in the VTDS can be modified.

3.3.1 VerifyDataChangeOccurred

This test verifies that the data stored in the VTDS can be modified and modifications will be reflected in subsequent queries.

3.4 MISP Data Modification Tests

This test scenario verifies that the data stored on the MISP Server can be modified.

3.4.1 VerifyDataChangeOccurred

This test verifies that the data stored on the MISP Server can be modified and modifications will be reflected in subsequent queries.

4 Data Security Tests

This test case aims to verify the security of the VTIP against unauthorized access and SQL injection attacks.

4.1 SQL Injection Vulnerability Test

This test scenario aims to ensure the VTIP is not susceptible to SQL injection attacks.

4.1.1 SQLmap injection test

This test will use the SQL map tool to automatically test the VTIP for SQL injection vulnerabilities.

5 Performance Tests

This test case aims to test the performance of the VTIP.

5.1 TestServerResponseTimes_VTDS

This test scenario involves testing the response times for different VTDS API calls of varying complexity.

5.1.1 GET_SingleThreat

This test will check the response time for getting a single threat record by ID from the VTDS.

5.1.2 GET_MultiThreat

This test will check the response time for getting multiple threat records from the VTDS.

5.1.3 POST_Threat

This test will check the response time for creating a single threat record in the VTDS.

5.1.4 PUT_Threat

This test will check the response time for updating a single threat record in the VTDS.

5.2 TestServerResponseTimes_MISP

This test scenario involves testing the response times for different MISP API calls of varying complexity.

5.2.1 GET_SingleThreat

This test will check the response time for getting a single record by CVE_ID from the MISP Server.

5.2.2 GET_MultiThreat

This test will check the response time for getting multiple threat records from the MISP Server.

5.2.3 POST_Threat

This test will check the response time for creating a single record in the MISP Server.

5.2.4 PUT_Threat

This test will check the response time for updating a single record in the MISP Server.

6 Data Export Tests

This test case will test the data exporting feature of the VTIP.

6.1 TestDataExportToPDF

This test will test to ensure that the export feature of the VTIP produces a PDF file with the desired data.

7 UI tests

7.1 Submit Button Tests

This test will test the responsiveness of the submit button and ensure that the form is properly submitted to the server.

7.1.1 Submit search of existing data

This test will test the retrieval of data from the API and ensure (client-side) filtering is functioning as intended.

7.1.2 Submit search of non-existent data

This test will test the retrieval of non-existent data from the API and ensure the (client-side) application handles no retrievals properly.

7.2 Back Button Tests

7.2.1 BackPressedWhileOnFirstRecord

This test will test that pressing the back button from the first record will properly route the user to the page of all searched records.

7.2.2 BackPressedWhileNotOnFirstRecord

This test will test that pressing the back button from nth record will properly route the user to the page of the previous record (via backUrl).

7.3 Next Button Tests

7.3.1 NextPressedWhileOnLastRecord

This test will test that pressing the next button from the last record will route the user back to the list of all records.

7.3.2 NextPressedWhileNotOnLastRecord

This test will test that pressing the next button from the nth record will properly route the user to the nth + 1 record.

7.4 Reset Button Tests

These tests will test the reset functionality of record forms.

7.4.1 ResetPressed_NoModifications

This test will test that pressing the reset button with no modification made will keep the original data.

7.4.2 ResetPressed_ModificationsMade

This test will test that pressing the reset button with modifications made will reset the data back to its original form.

7.5 Data Field Modification Tests

These tests will test the (client-side) form validation of submitted data.

7.5.1 DateTime validation test - VALID case

This test will test that the datetime fields will pass if they are the proper format for a datetime object.

7.5.2 DateTime validation test - INVALID case

This test will test that the datetime fields will throw a (client-side) error message if they do not match the proper datetime object format.

7.5.3 CWE_ID validation test - VALID case

This test will test that the CWE_ID field will pass if it matches the proper format (UUID)

7.5.4 CWE_ID validation test - INVALID case

This test will test that the `CWE_ID` field will throw a (client-side) error message if it does not match the proper format (CWE ID).

Software Requirements Specification, Traceability Matrix, and Test Plans

for

Vehicle Security System Integration

Technical Report UWF-TR-FDOT-010-01

Contract Number: BED34 Task Order: 977-01

Version 0.7 approved

**Prepared by Elizabeth Uebele
The University of West Florida
Florida Department of Transportation**

February 19, 2024

Table of Contents

1. Introduction	360
1.1 Purpose	360
1.2 Document Conventions	360
1.3 References	360
1.4 Document Revisions Table	360
2. Overview of Product	360
2.1 VSSI	360
2.2 User Classes and Characteristics	360
2.3 Operating Environment	361
2.4 Design and Implementation Constraints	361
2.5 Assumptions and Dependencies	361
3. Interface Requirements	361
3.1 The Landing Page	361
4. Functional Requirements	362
4.1 FR-1: Navigating to Subsystems	362
4.1.1 Description and Priority	362
4.1.2 Related User Classes	362
4.1.3 Functional Requirements	362
5. Test Requirements	363
5.1 T-1: Manual Tests	363
5.2 T-2: Integration Tests	363
5.3 T-3: Test Report	363
6. Non-Functional Requirements	363
6.1 NF-1: Portability	363
6.2 NF-2: Usability	363
6.3 NF-3: Speed	363
7. Quality Attributes	363
8. VSSI Requirements Traceability Matrix	363
9. The Unit Test Plan Overview	365
9.1 VSSI Test Framework	365
9.1.1 Framework Description	365
9.1.2 Running the Tests	365
9.1.2.1 Prerequisites	365
9.1.2.2 Test Execution	365

9.2 <i>Webpage Navigation Test</i>	366
9.2.1 VSSI-VTCS Navigation Test.....	366
TestVTCSNavigation.....	366
9.2.2 VSSI-VTIP Navigation Test.....	366
TestVTIPNavigation.....	366
9.2.3 VSSI-MISP Navigation Test	366
TestMISPNavigation	366
9.2.4 VSSI-VSMVS Navigation Test	366
TestVSMVSNavigation	366
9.2.5 VSSI Navigation Interface Test	366
TestVSSINavigation.....	366
Appendix A: Requirements Table	367
Appendix B: Requirements Traceability Matrix	369
Appendix C: Glossary	370

1. Introduction

1.1 Purpose

This document describes the software requirements for Version 1.0 of the Vehicle Security System Integration (VSSI) as a subcomponent of the Connected Vehicle Security Metrics and Threat Intelligence Project. This document provides an overview of the users and context of the VSSI and covers all functional, non-functional, and interface requirements.

1.2 Document Conventions

This document is based on the IEEE 830-1998 Standards and the Florida Department of Transportation Requirements Standards. Specific conventions used in this document are listed below:

- Priorities are indicated for each feature as well as in the Requirements Table. A green highlighting indicates must have features, while a yellow highlight represents a should have feature.
- Requirements follow the form of <TAG>-#.#.# where a tag indicates a category of requirements. And the # represents the ID of the requirement in a hierarchical fashion.

1.3 References

The following references were used in the creation of this document:

- IEEE 830-1998 Standards on Software Requirement Specifications
- UWF Scope of Service Document for the Connected Vehicle Security Metrics and Threat Intelligence Project

1.4 Document Revisions Table

Revisor	Revision Date	Reason
Elizabeth Uebele	February 19, 2024	Created initial draft of document
Dr. Guillermo Francia, III	March 1, 2024	Editorial and formatting changes
Dr. Guillermo Francia, III	April 1, 2024	Editorials in response to review comments

2. Overview of Product

2.1 VSSI System Description and Relevance

The VSSI is a subsystem that allows the user to navigate to different parts of the greater system. The VSSI provides different interface cards for the Vehicle Threat Intelligence Portal (VTIP), the Vehicle Threat Collection System (VTCS), the Vehicle Security Metrics Visualization System (VSMVS), and the Malware Information Sharing Platform (MISP). This interface provides the integration of the various subsystems using a common entry point and collectively binds them to function with a common purpose, i.e. supporting the security of CAVs.

2.2 User Classes and Characteristics

User Class	Characteristics
------------	-----------------

Ordinary User	This user will be able to navigate the VSSI
---------------	---

2.3 Operating Environment

The VSSI operating environment is defined by the following:

OE-1: The VSSI will run as an HTML page.

OE-2: The VSSI will run as a part of the VTDS AWS EC2 instance.

OE-3: The VSSI will interact with the VTIP, VTCS, VSMVS, and MISP server.

2.4 Design and Implementation Constraints

The VSSI design and implementation are constrained by the following:

DIC-1: The VSSI will be developed using the Hypertext Markup Language (HTML).

2.5 Assumptions and Dependencies

Assumptions and dependencies for the VSSI implementation include the following:

ASS-1: The VSSI assumes the AWS instances for VTIP, VTCS, VSMVS, and MISP are running when trying to navigate to them.

DEP-1: The VSSI is dependent on links to the VTIP, VTCS, VSMVS, and MISP working properly.

3. Interface Requirements

The VSSI will enable the user to select which page they wish to navigate to.

INT-1: The VSSI will provide a Graphical User Interface for the user to navigate to different pages of the system.

3.1 The Landing Page

The landing page, as depicted in Figure 3.1, will contain interface cards for each subsystem in the system, including the VTIP, VTCS, VSMVS, and MISP. Each card will contain the name of the subsystem, an associated image, and a button to navigate to the associated subsystem.

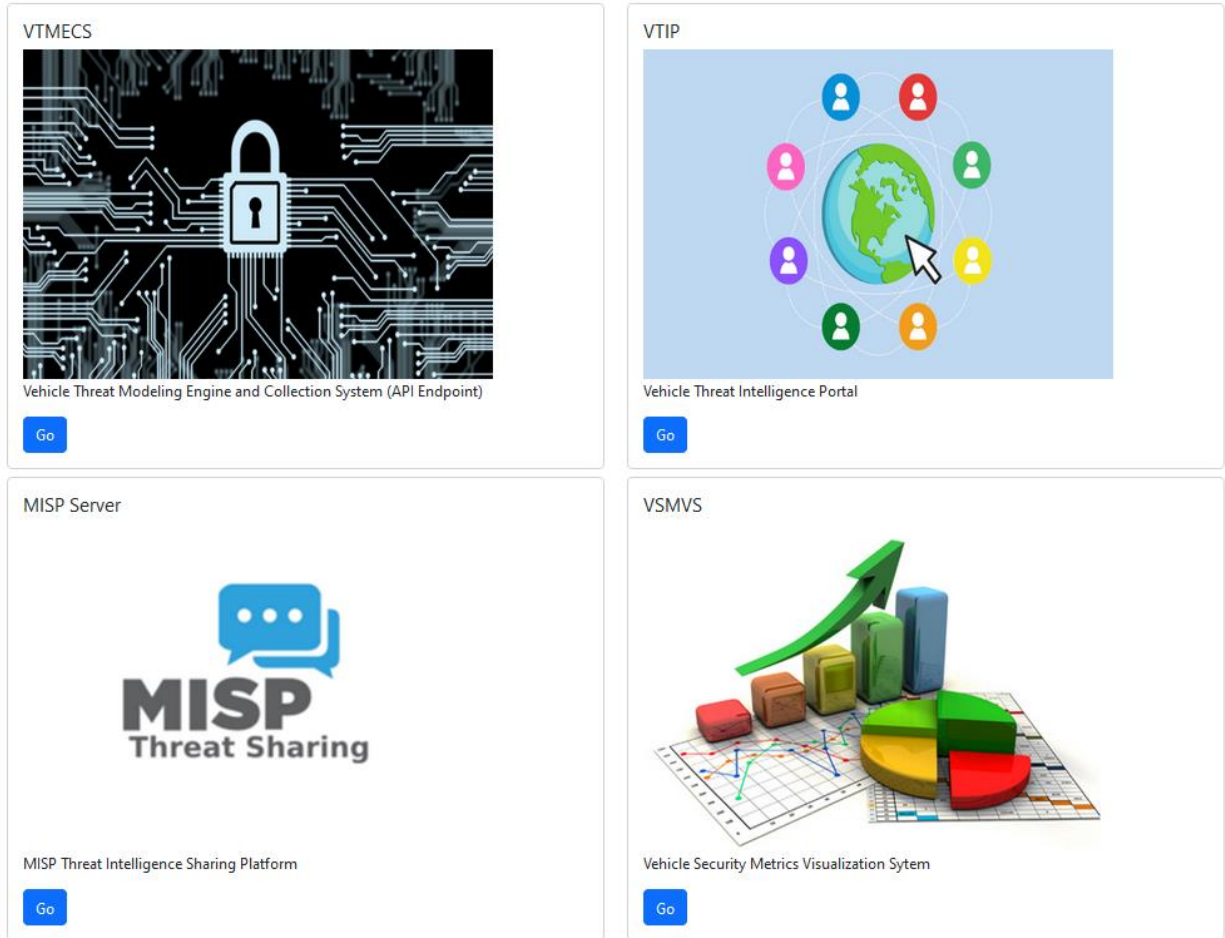


Figure 3.1 Landing Page Interface Prototype

4. Functional Requirements

Functional requirements for the VSSI are fundamental actions that the system must execute to be considered operational. The VSSI is a proof of concept and as such, limitations in its implementation will be identified.

4.1 FR-1: Navigating to Subsystems

4.1.1 Description and Priority

The VSSI is required to allow the user to navigate to the VTIP, VTCS, VSMVS, and MISP.

Priority: Must Have

4.1.2 Related User Classes

All Users

4.1.3 Functional Requirements

FR-1.1: The VSSI will be able to navigate to the VTIP subsystem.

- FR-1.2: The VSSI will be able to navigate to the VTCS subsystem.
- FR-1.3: The VSSI will be able to navigate to the VSMVS subsystem.
- FR-1.4: The VSSI will be able to navigate to the MISIP subsystem.

5. Test Requirements

The VSSI requires testing and validation of the main application functionalities.

5.1 T-1: Manual Tests

Manual testing will be performed to ensure the VSSI navigates to the appropriate pages.

5.2 T-2: Integration Tests

System integration testing is not within the scope of the VSSI system.

5.3 T-3: Test Report

Documentation of all system testing activities shall be provided. See the attached Test Overview document.

6. Non-Functional Requirements

Non-functional requirements for the VSSI are system attributes that are desired but not required. The following are the non-functional requirements for the VSSI:

6.1 NF-1: Portability

The development team will attempt to make the web enabled VSSI system portable across multiple computing form factors.

6.2 NF-2: Usability

The development team will attempt to satisfy system usability features such as navigation, performance quality, and intuitiveness of interfaces.

6.3 NF-3: Speed

The development team will attempt to enhance the VSSI speed in navigating to the different pages in the system.

7. Quality Attributes

The VSSI is a proof of concept and not meant for a production release. As such, traditional quality attributes such as availability, security, robustness, etc. are not as relevant.

8. VSSI Requirements Traceability Matrix

The VSSI Traceability Matrix provides a summary of all system requirements that are mapped to their corresponding Test Case IDs and testing statuses. The Traceability Matrix is depicted on the table below.

REQUIREMENTS TRACEABILITY VERIFICATION MATRIX

Project Name:	Vehicle Security System Integration (VSSI)				
Project Description:	The Vehicle Security System Integration (VSSI)				
Project Manager Name:	Dr. Guillermo Francia, III				
Agency/Firm:	UWF-FDOT				
User Need ID	User Need Summary	Requirement ID	Detailed Requirement Summary	Requirements Document Section	DR Source Document
GAF001	VSSI System Scope	S1	Specifies the scope of the system	2.1	VSSI Requirements
GAF001	VSSI-VTIP Navigation	FR-1.1	The VSSI must navigate to the VTIP subsystem	4.1	VSSI Requirements
GAF001	VSSI-VTCS Navigation	FR-1	The VSSI must navigate to the VTCS subsystem	4.1	VSSI Requirements
GAF001	VSSI-VSMVS Navigation	FR-1	The VSSI must navigate to the VSMVS subsystem	4.1	VSSI Requirements
GAF001	VSSI-MISP Navigation	FR-1	The VSSI must navigate to the MISP subsystem	4.1	VSSI Requirements
GAF001	VSSI Navigation Interface	INT-1	The VSSI shall provide a Graphical User Interface for subsystem navigation	3	VSSI Requirements

REQUIREMENTS TRACEABILITY VERIFICATION MATRIX						
User Need ID	User Need Summary	Verification Test Case ID	Compliance (Y/N/Partial/NA)	Notes/Comments/Date	Reviewer Initials	FDOT Initials
GAF001	VSSI System Scope	NA	NA	Non Testable	EU	
GAF001	VSSI-VTIP Navigation	Test Cases: 3.1.1	Yes	Completed 02/20/24	EU	
GAF001	VSSI-VTCS Navigation	Test Cases: 3.2.1	Yes	Completed 02/20/24	EU	
GAF001	VSSI-VSMVS Navigation	Test Cases: 3.3.1	Yes	Completed 02/20/24	EU	
GAF001	VSSI-MISP Navigation	Test Cases: 3.4.1	Yes	Completed 02/20/24	EU	
GAF001	VSSI Navigation Interface	Test Cases: 3.5.1	Yes	Completed 02/20/24	EU	

9. The Unit Test Plan Overview

This section describes the unit test framework of the Vehicle Security System Integrated Landing Page (VSSI) as a subcomponent of the Connected Vehicle Security Metrics and Threat Intelligence Project.

VSSI Test Framework

9.1.1 Framework Description

The VSSI was developed using HTML. Because no convenient automated test framework was found to test a single HTML document, all tests were run manually.

9.1.2 Running the Tests

You may use whichever browser you prefer to run the VSSI and follow along with the manual tests.

9.1.2.1 Prerequisites

The test suite requires the following before running any of the tests: The AWS instances for VTIP, VTCS, VSMVS, and MISP are currently in operation.

9.1.2.2 Test Execution

Because the tests are manual, to execute the tests, follow along with the descriptions of each test, executing each action that is described.

9.2 Webpage Navigation Test

The Webpage Navigation Tests aim to ensure that each interface card provides the proper navigation to the associated webpage.

9.2.1 VSSI-VTCS Navigation Test

This test scenario involves checking if the VTCS card navigates to the VTCS properly.

TestVTCSNavigation

Verify that the VTCS interface card navigates to the VTCS. Find the VTCS card, find the button labeled “Go,” and click on that button. On success, it will lead to the VTCS page.

9.2.2 VSSI-VTIP Navigation Test

This test scenario involves checking if the VTIP card navigates to the VTIP properly.

TestVTIPNavigation

Verify that the VTIP card navigates the VTIP. Find the VTIP card, find the button labeled “Go,” and click on said button. On a success, it will lead to the VTIP page.

9.2.3 VSSI-MISP Navigation Test

This test scenario involves checking if the MISP card navigates to the MISP properly.

TestMISPNavigation

Verify that the MISP card navigates the MISP. Find the MISP card, find the button labeled “Go,” and click on said button. On a success, it will lead to the MISP page.

9.2.4 VSSI-VSMVS Navigation Test

This test scenario involves checking if the VSMVS card navigates to the VSMVS properly.

TestVSMVSNavigation

Verify that the VSMVS card navigates the VSMVS. Find the VSMVS card, find the button labeled “Go,” and click on said button. On a success, it will lead to the VSMVS page.

9.2.5 VSSI Navigation Interface Test

This test scenario involves checking if the VSMVS card navigates to the VSMVS properly.

TestVSSINavigation

Verify that the VSSI landing page is displayed when the proper URL is invoked.

Appendix A: Requirements Table

Requirement ID	Requirement Type	Requirement Name	Requirement Description	Priority
S1	Scope	VSSI System Scope	The VSSI is required to provide the user a way to navigate to different pages in the system	Must have
FR1.1	Functional	Navigate to VTIP	The VSSI will facilitate navigation to the VTIP.	Must have
FR1.2	Functional	Navigate to VTCS	The VSSI will facilitate navigation to the VTCS.	Must have
FR1.3	Functional	Navigate to VSMVS	The VSSI will facilitate navigation to the VSMVS.	Must have
FR1.4	Functional	Navigate to MISP	The VSSI will facilitate navigation to the MISP server.	Must have
INT -1	Interface	VSSI Navigation Interface	The VSSI shall provide a Graphical User Interface for users to navigate to the different pages in the system.	Must have
T-1	Test	Manual Test	Manual testing system testing shall be conducted for all functional system components	Must have
T-2	Test	Integration Test	System integration testing is not within the scope of the VTCS system	Out of scope
T-3	Test	Test Report	An associated documentation of all system	Must have

			test activities shall be provided	
NF-1	Non-functional	Portability	The development team will attempt to make the web enabled VSSI system portable across multiple computing form factors	Could have
NF-2	Non-functional	Usability	The development team will attempt to satisfy system usability features such as navigation, performance quality, and intuitiveness of interfaces	Could have
NF-3	Non-functional	Speed	The development team will attempt to enhance the VSSI system ability to navigate to different pages quickly	Should have

Appendix B: Requirements Traceability Matrix

Requirement ID	Requirement Description	Test Case	Status
S1	Defines the scope of the system	N/A	N/A
FR1.1	The VSSI shall facilitate navigation to the VTIP page.	TestVTIPNavigation	Passed
FR1.2	The VSSI shall facilitate navigation to the VTMECS page.	TestVTCSNavigation	Passed
FR1.3	The VSSI shall facilitate navigation to the VSMVS page.	TestVSMVSNavigation	Passed
FR1.4	The VSSI shall facilitate navigation to the MISP page.	TestMISPNavigation	Passed
INT -1	The VSSI will provide an interface to navigate to different pages in the system.	N/A	N/A
T-1	Manual system testing shall be conducted for all functional system components	Multiple Test Cases	Passed
T-2	System integration testing is not within the scope of the VTCS.	N/A	N/A
T-3	An associated documentation of all system test activities shall be provided	N/A	Completed
NF-1	The development team will attempt to make the web enabled VSSI portable across multiple computing form factors	N/A	N/A
NF-2	The development team will attempt to satisfy system usability features such as navigation, performance	N/A	N/A

	quality, and intuitiveness of interfaces		
NF-3	The development team will attempt to enhance the VSSI speed in navigating to the different pages in the system.	N/A	N/A

Appendix C: Glossary

Term	Description
AWS	Amazon Web Services
EC2	Elastic Compute Cloud
HTML	Hypertext Markup Language
MISP	Malware Information Sharing Platform
VTCS	Vehicle Threat Collection System
VTDBS	Vehicle Threat Database System
VTME	Vehicle Threat Modeling Engine
VSMVS	Vehicle Security Metrics Visualization System

Vehicle Security System Integration (VSSI)

User Manual

Technical Report UWF-TR-FDOT-010-02

Contract Number: BED34 Task Order: 977-01

**Version 0.1
04/01/2024**

Table of Contents

Introduction	373
Starting the System	374
VSSI Interface	375
VTIP Interface.....	376
VTIP-MISP Interface	378
VSMVS Interface	382

Table of Figures

Figure 1. The VSSI Landing Webpage.....	375
Figure 2. VTIP Landing Webpage.....	376
Figure 3. A Sample VTIP Output	377
Figure 4. VTIP-MISP Interface for STIX Upload	378
Figure 5. VTIP-MISP Search Output.....	379
Figure 6. Formatted MISP Event View	380
Figure 7. Raw JSON MISP View	381
Figure 8. The VSMVS Landing Webpage.....	382

Introduction

This document was created to provide user's guide to the Vehicle Security System Integration (VSSI) system. The VSSI is a subsystem that allows the user to navigate to different parts of the greater system. The VSSI illustrates interface cards to enable access to the following subsystems: the Vehicle Threat Intelligence Portal (VTIP), the Vehicle Threat Collection System (VTCS), the Vehicle Security Metrics Visualization System (VSMVS), and the Malware Information Sharing Platform (MISP). This interface provides the integration of the various subsystems using a common entry point and collectively binds them to function with a common purpose, i.e. supporting the security of CAVs. The descriptions of the Vehicle Threat Collection System (VTCS) and the Vehicle Security Metrics Visualization System (VSMVS) are included in related documents.

The VTIP functions as a subsystem of the Vehicle Threat System (VTS), serving as the front-end Graphical User Interface (GUI) for secure user access to essential features of the VTS. It enables users to query and modify threat records stored in the Vehicle Threat Database System (VTDS). Additionally, the VTIP acts as the primary interface for users to access the Malware Information Sharing Platform (MISP) Server subcomponent of the Project System for threat intelligence information sharing.

The VTCS is an automated system that collects threat intelligence feeds from various sources and stores that data to the Vehicle Threat Database System (VTDS) for ingestion and processing by other subcomponents of the Project System. Threat Intelligence data will come from publicly available Open-Source Intelligence (OSINT) sites such as Open Threat Exchange (OTX) and VirusTotal. Configurable, automated queries to these sources will generate tailored threat intelligence feeds and provide any associated Common Vulnerabilities and Exposures (CVEs), Common Weaknesses Enumerations (CWEs) and any other relevant Indicators of Compromise (IoCs).

The Malware Information Sharing Platform (MISP) is a threat sharing platform which can both upload and download threats. It utilizes its own database and Application Program Interface

(API) to interact with the web and store threat information. For our purposes MISP is used to associate global threats vehicle-related threat stored in the VTDS.

The Vehicle Security Metrics Visualization System (VSMVS) is a web application that allows users to interact with different UI to generate scores, graphs, or both. Depending on the page users are on, the web application will have a slightly different User Interface (UI). Each UI is catered towards a specific functionality. The purpose of the VSMVS is to help users recognize trends and patterns that are not easily recognized using non-visual methods. The system will provide a visual depiction of security metrics that were developed in another undertaking by employing the benefits of visual perception.

Starting the System

To start the system, you must run the VTDS Amazon Web Services (AWS) Elastic Compute Cloud (EC2) instance. Log in to the AWS Management Console. Navigate to the EC2 instances. Right click the VTDS instance and click “Start Instance.” The start of the VTDS EC2 instance enables the VSSI. The landing page of the VSSI is displayed when the web page address or the Uniform Resource Locator (URL) address is entered on a web browser. It should be noted that preliminary activities must be completed to ensure that the VSSI runs smoothly. These activities include running the MISP Server and the VTCS, VTIP, and VSMVS EC2 instances.

VSSI Interface

The VSSI interface allows the user to navigate to different parts of the system. Each landing webpage of a subsystem has an associated card in the VSSI. Each card contains the name of the webpage, an image, and a button to navigate to the webpage. Clicking the “Go” button adjacent to a card image will take the user to the corresponding subsystem. Figure 1 illustrates the VSSI landing webpage.

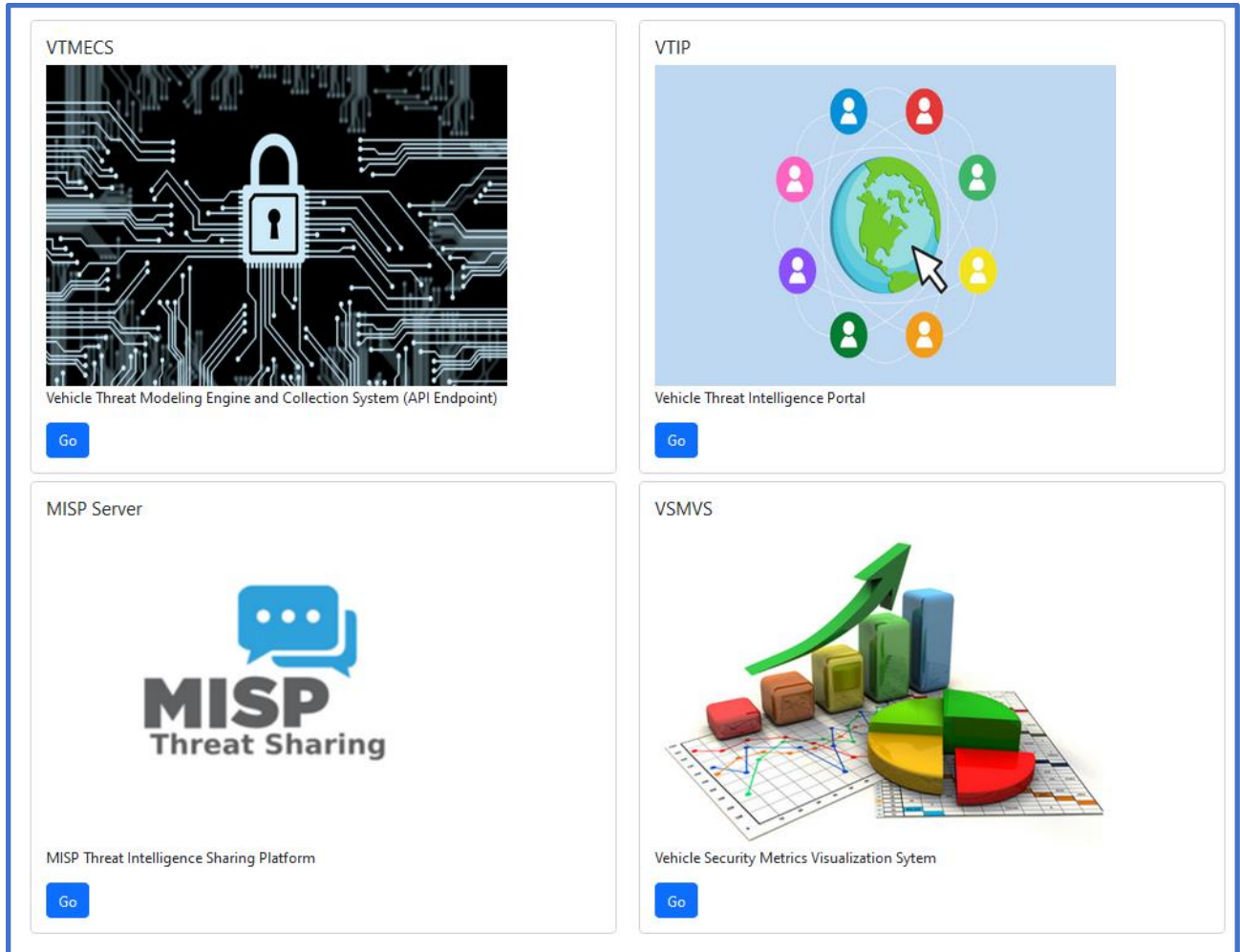


Figure 22. The VSSI Landing Webpage

VTIP Interface

The VTIP interface allows the user to query for CVEs based on CVE ID, keyword, or the vehicle's make, year, and model. It also allows for searching for CVEs within a specified date range. The VTIP Landing Webpage is shown in Figure 2. Upon entering the VTIP, the query fields and information will be blank.

The screenshot shows the Threat Intelligence Portal (VTIP) landing webpage. The page has a blue header with the title "Threat Intelligence Portal" and a "Toggle Search Menu" button. Below the header, there is a search section with a "Query by:" dropdown menu set to "CVE ID", a text input field, a "Date Range" dropdown menu set to "Without date range", and "Submit" and "Reset" buttons. Below the search section, there is a grid of input fields for CVE details: CVE ID, CVE Publish/Modified DTG (1/1/0001 12:00:00 AM), CWE ID, Vulnerability Status, CVSS Data (Base Score: 0), Confidentiality Impact Score, Integrity Impact Score, Availability Impact Score, CVSS Version, Vector String, Attack Vector, Attack Complexity, Privilege Level Required, User Interaction, Scope, and Base Severity. Below the grid, there is a large "Description" text area. At the bottom, there are sections for "Associated Vehicles", "CVE Results 0", and navigation buttons for "Back", "Page 1 of 0", and "Next".

Figure 23. VTIP Landing Webpage

To query for a CVE, select the method by which you are querying (ID, keyword, vehicle). Enter the value for which you are querying, select the date range if applicable, and click the submit button. The VTIP will show results for the first CVE found, displaying other CVEs that match the query along the bottom of the page. It will display the ID, the last date modified, the associated Common Weakness Enumeration (CWE), the vulnerability status, the Common Vulnerability Scoring System (CVSS) information, and the description. Additionally, it will show all the vehicles associated with the CVE in the case of multiple vehicles being affected. A sample output is depicted on the interface shown in Figure 3.

Threat Intelligence Portal

[Toggle Search Menu](#)

Query by:

Vehicle:
 2012:
 BMW:
 3 Series:

[Submit](#)

Date Range: [Reset](#)

CVE ID	CVE Publish/Modified DTG	CWE ID	Vulnerability Status
CVE-2018-9311	6/29/2018 6:03:20 PM	CWE-693	Vulnerability Exists

CVSS Data

Base Score	Confidentiality Impact Score	Integrity Impact Score	Availability Impact Score
9.8	HIGH	HIGH	HIGH
CVSS Version	Vector String	Attack Vector	Attack Complexity
3.0	CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:1	NETWORK	LOW
Privilege Level Required	User Interaction	Scope	Base Severity
NONE	NONE	UNCHANGED	CRITICAL

Description

The Telematics Control Unit (aka Telematic Communication Box or TCB), when present on BMW vehicles produced in 2012 through 2018, allows a remote attack via a cellular network.

Associated Vehicles

2012 BMW 3 Series	
2018 BMW 3 Series	
2012 BMW 5 Series	
2018 BMW 5 Series	
2012 BMW 7 Series	

CVE Results 7

CVE-2018-9311 CVE-2018-9312 CVE-2018-9313 CVE-2018-9314 CVE-2018-9318 CVE-2018-9320
 CVE-2018-9322

Page 1 of 1

[Back](#)
[Next](#)

Figure 24. A Sample VTIP Output

VTIP-MISP Interface

The VTIP-MISP Interface allows users to interact directly with the MISP through the VTIP for both uploading and downloading MISP events. In the option for searching MISP Events, users can add the options controller, eventinfo, value, limit, org, uuid, date_to, and date_from. For uploading MISP events, it utilizes the Structured Threat Information eXpression (STIX) 2.0 format, which must be copied and pasted into the textbox provided by the “Upload STIX data” button as shown in Figure 4.

Uploading events will notify the user if an error occurs. When uploading information to MISP it will convert the STIX 2.0 object to a MISP event and store it as well as send that object back to the user. This will allow the user to click on the new MISP Event and view it as if it was searched for. The conversion is not lossless but keeps vital information and allows for ease of access due to the high utilization of STIX format in cybersecurity.

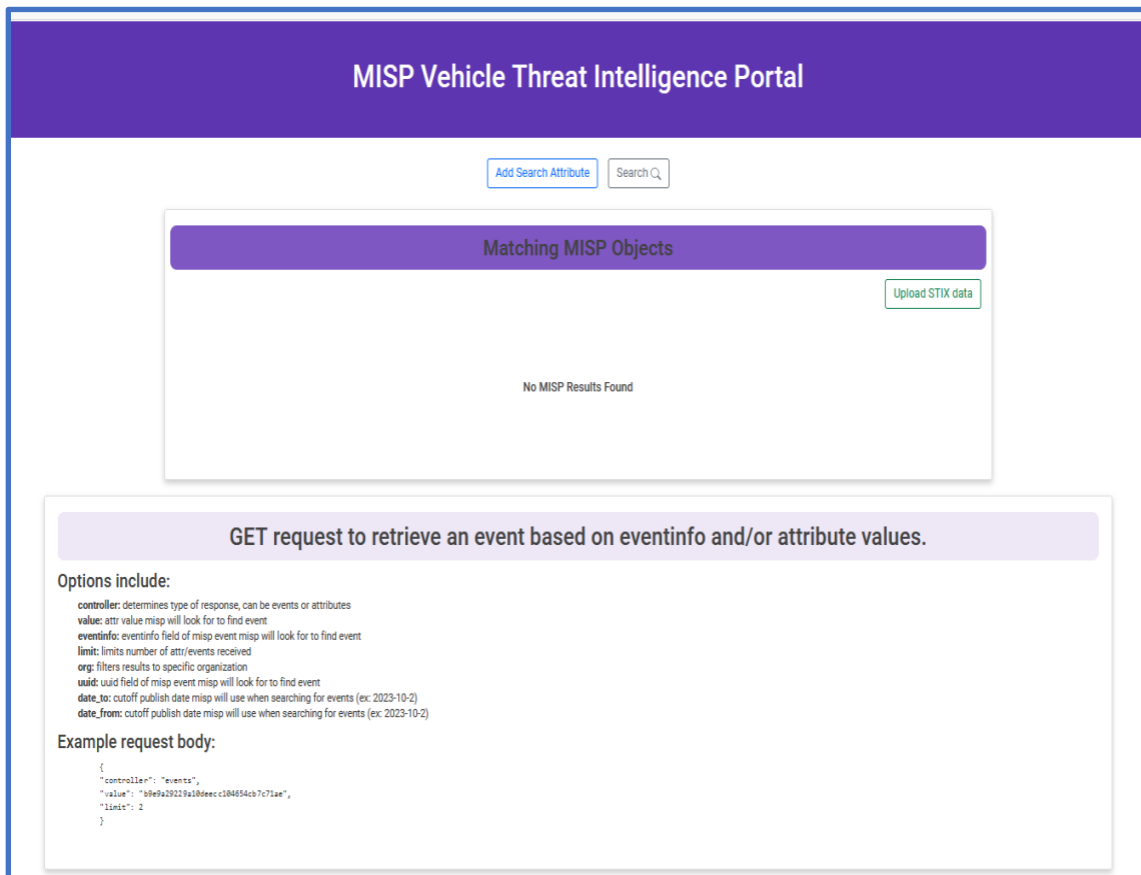


Figure 25. VTIP-MISP Interface for STIX Upload

When searching for events, users will mostly utilize the `value`, `eventinfo`, `org`, and `uuid` options. *Value* allows for users to search for MISP events based on whether they contain an attribute with a specific value. *Eventinfo* allows users to search for events based on a specific value in their description. *Org* allows users to search for all MISP events from a certain organization. *Uuid* allows users to search for MISP events based on the uuid of the event. The *controller*, *limit*, *date_to*, and *date_from* are all filtering options. The *controller* allows the user to specify whether they want to view events or attributes while *limit* allows users to specify the limit on the number of returned objects. By default, *controller* uses an event. A sample search interaction is shown in Figure 5.

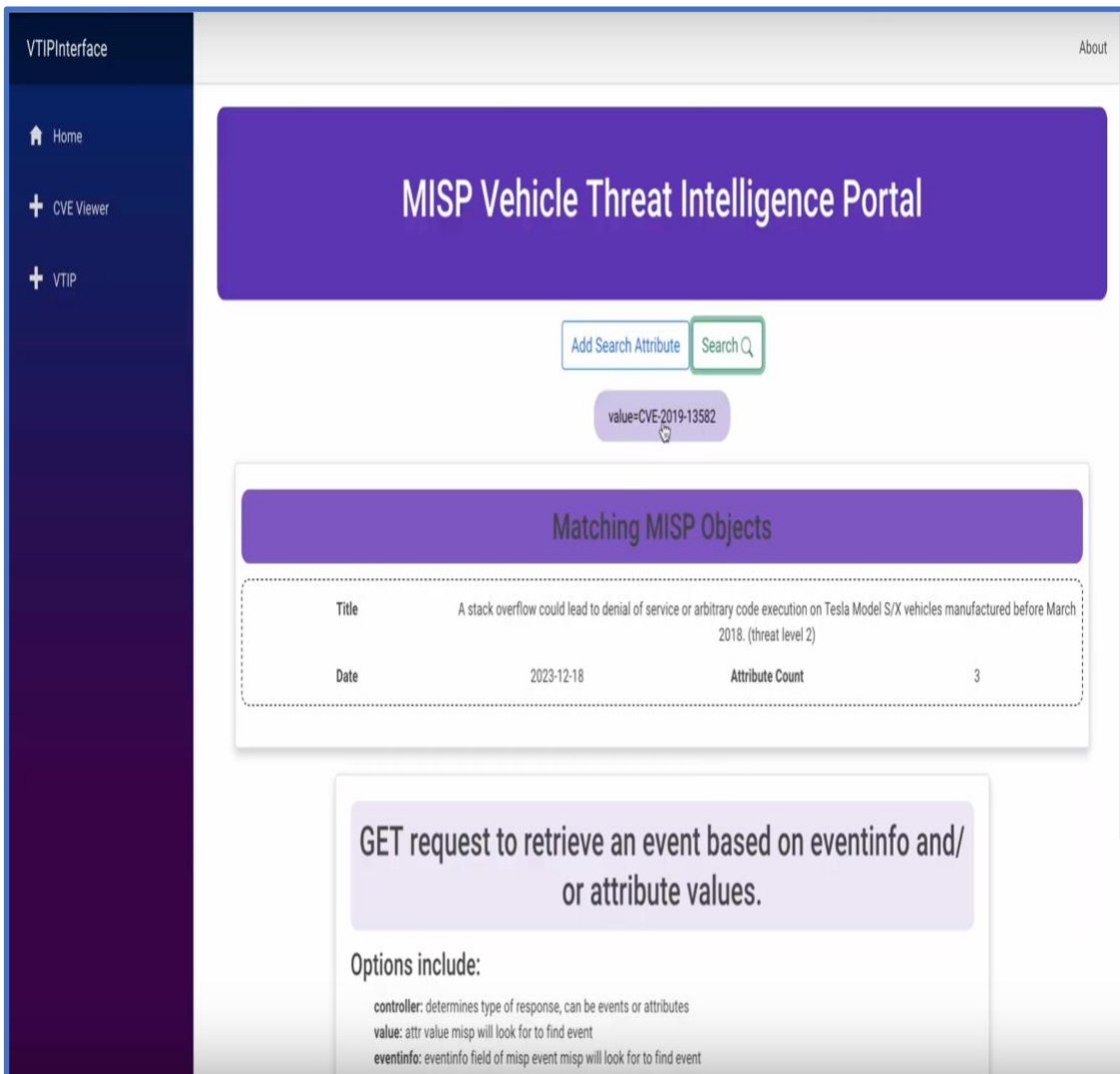


Figure 26. VTIP-MISP Search Output

Once a user has selected a MISP event, whether it be through search or through uploading a STIX object the details page is displayed. This details page offers a formatted view option and a raw JSON view option. The formatted view will display the event info in a card at the top, with important information included in each row. Subsequent attributes for the event will be included in the cards below (one card for each attribute) with important information for each attribute included in each card. A sample MISP Event View output is shown in Figure 6. This view is the suggested view, however, selecting the raw JSON view button at the top of the page will take the user to a page with the raw JSON as shown in Figure 7. This JSON is formatted properly for easy readability and is useful for finding information that might be deemed irrelevant by the formatted view.

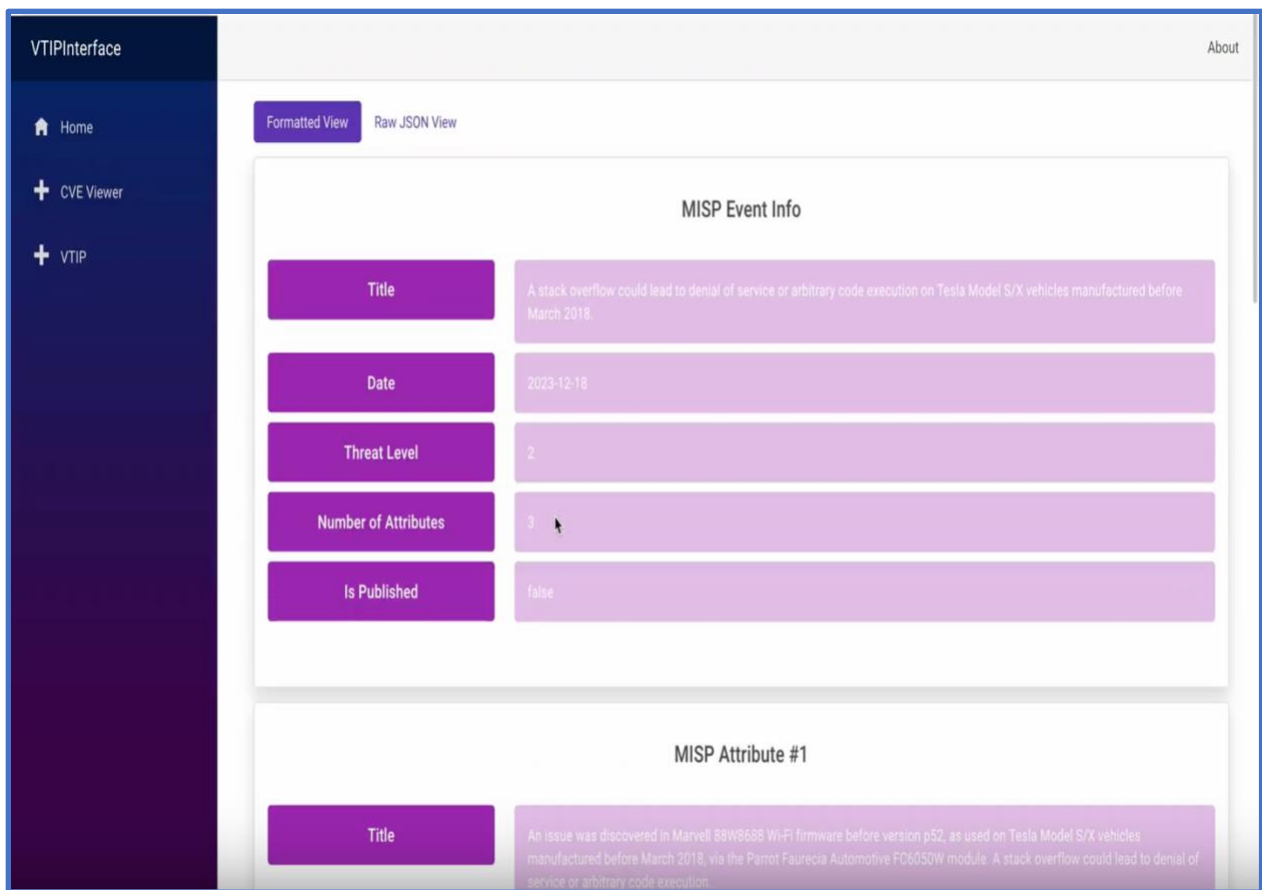


Figure 27. Formatted MISP Event View

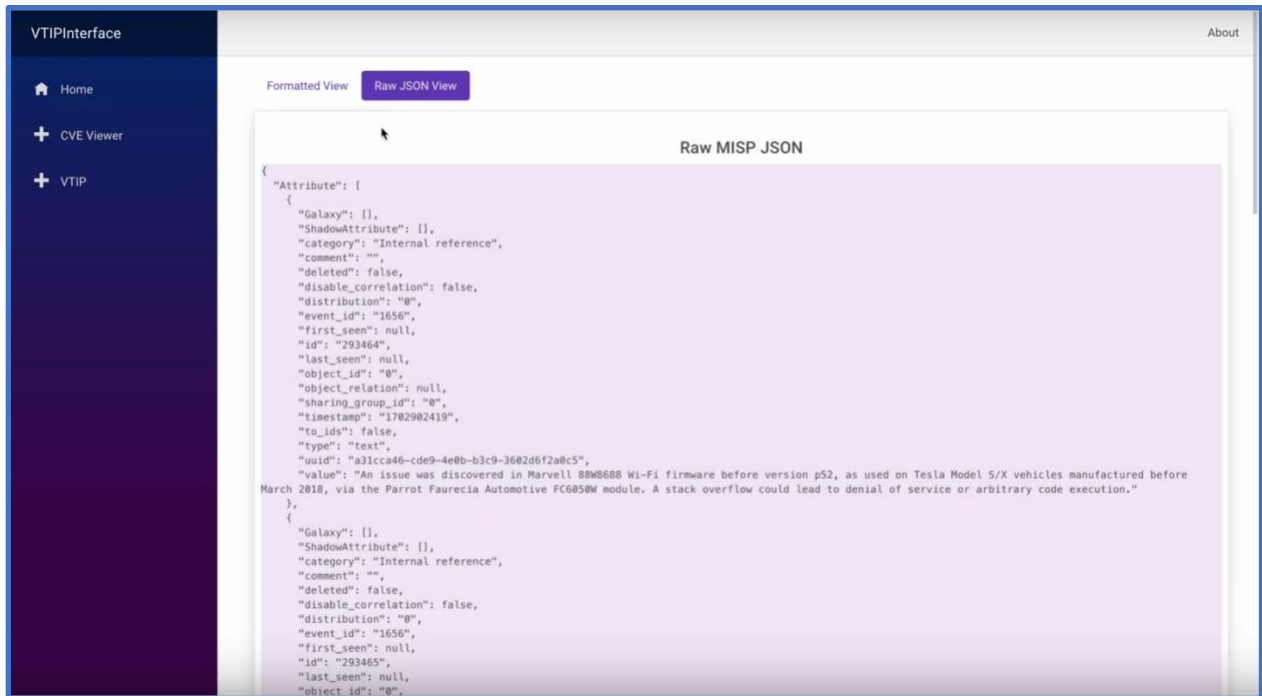


Figure 28. Raw JSON MISP View

VSMVS Interface

The VSMVS landing page consists of an introductory description of the website as well as six cards containing descriptions and links to each of the main application pages on the website. The landing webpage is depicted in Figure 8. Once a user clicks on the button within a container, they will be brought to that page where they can input data and visualize the various threats across various security metrics.

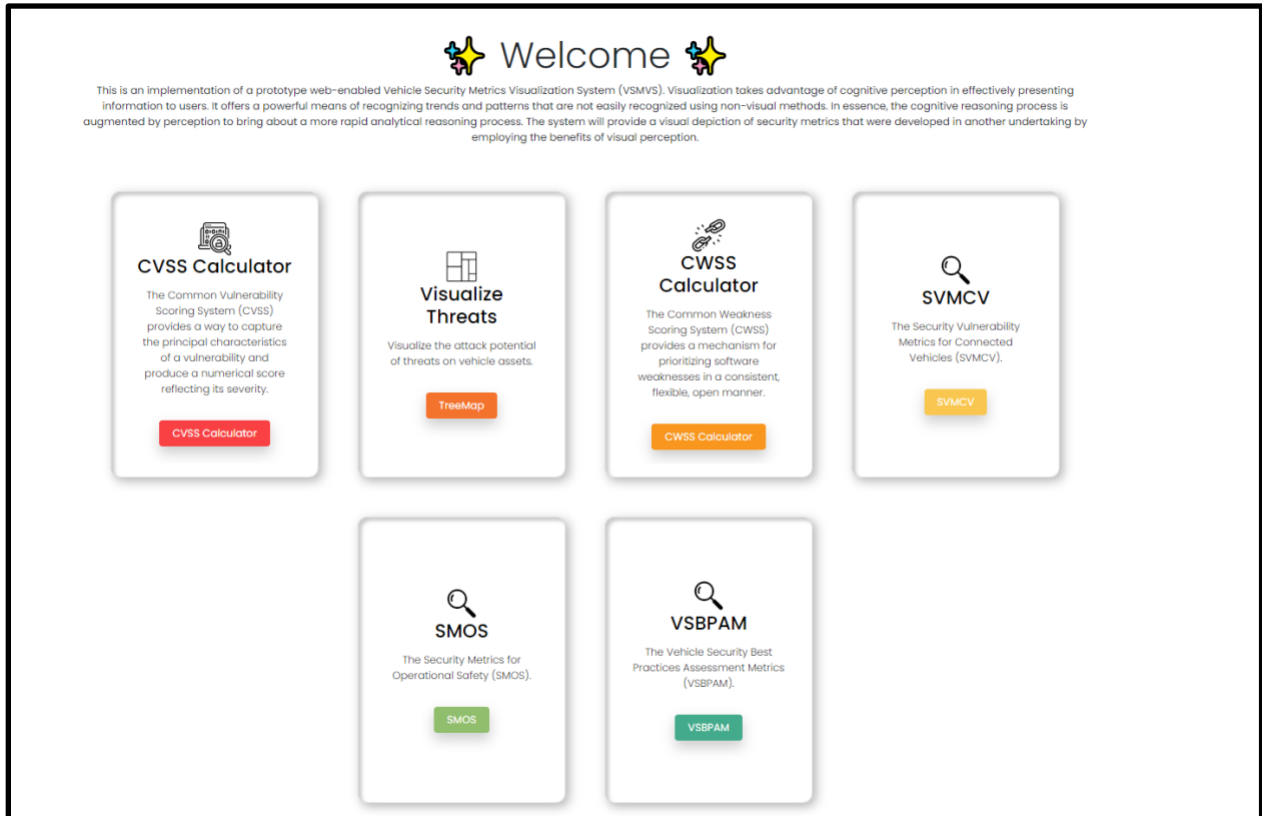


Figure 29. The VSMVS Landing Webpage

Appendix XXIII. Data Processing Algorithm

Algorithm 1: RSU Data extraction and Cleansing

```
Function to Load BSM Data (load_BSMs):
    # Open the compressed XML file containing the BSM data
    with gzip.open(FPATH, 'rt') as fz:
        # Read and parse each line into an XML tree
        trees = [et.fromstring(l) for l in fz]

    # Print the tag of the first XML element
    print(trees[0].tag)

    # Filter out specific BSM messages
    BSMs_raw = [tree for tree in trees if tree.tag ==
                "MessageFrame"
                if tree.findall('messageId')[0].text == '20']

    # Define the desired BSM data attributes
    PAYLOAD_INFO_KEYS = ['id', 'secMark', 'lat', 'long',
                        'speed', 'heading', 'angle']

    # Make a list to store the BSM dictionaries
    BSMs_dicts_list = []

    # Extract the data from each BSM message
    for tree in BSMs_raw:
        message_dict = {}
        for k in PAYLOAD_INFO_KEYS:
            # Find the text for each key from the XML tree
            message_dict[k] = [ch.text for ch in \
                tree.findall(f'./value//BasicSafetyMessage//core
                Data/{k}')][0]

        # Process and update timestamp and other fields in #
        message_dict
        message_dict['timestamp'] =
            processSecMark(int(message_dict['secMark']),
                datetime.datetime.now(datetime.timezone.utc).strftime(
                    "%Y-%m-%d_%H:%M:%S"))
        message_dict['lat'] = int(message_dict['lat'])/10 ** 7
        message_dict['long'] = int(message_dict['long'])/10 ** 7
        message_dict['speed_mph'] =
            processSpeedMPH(int(message_dict['speed']))
```

```
# ... (Other hardcoded values)

# Append the cleaned dictionary to the list
BSMs_dicts_list.append({k: v for k, v in
                        message_dict.items()
                        if k not in ['id', 'speed', 'lat_long']})
# Return the list of processed BSM dictionaries
return BSMs_dicts_list

# Load the BSMs from the specified file path
BSMs_dicts_list = load_BSMs(FPATH)
```

Appendix XXIV. Continuous Integration/Continuous Deployment Pipeline

For the FDOT project, the UWF project team utilizes GitHub actions to deploy code to AWS automatically. This appendix covers the overall architecture of the Continuous Integration/Continuous Deployment (CI/CD) pipelines and how to set up and configure the pipelines.

Currently, we have CI/CD pipelines configured for the following GitHub repositories:

- VTMECS (<https://github.com/UWF-CfC-FDOT/VTMECS>)
- VSMS (<https://github.com/UWF-CfC-FDOT/VSMS>)

The third GitHub repository

<https://github.com/UWF-CfC-FDOT/VMLFramework> does not require a CI/CD pipeline nor an AWS EC2 instance.

Pipeline Steps

Note: This workflow is illustrated in Figure 22.

1. Code is committed to the repository hosted by GitHub. This triggers the GitHub Actions workflow.
2. Checkout: The repository is checked out to the self-hosted runner to prepare for building.
3. Build: The application is built, and the compiled artifacts are generated.
4. Test: Automated tests by xUnit.net are run on the compiled artifacts to ensure that the application is functional.
5. Publish: The compiled artifacts are published to the EC2 instance.
6. Deploy: The published artifacts are deployed to an IIS site configured in the Windows Server EC2 instance.

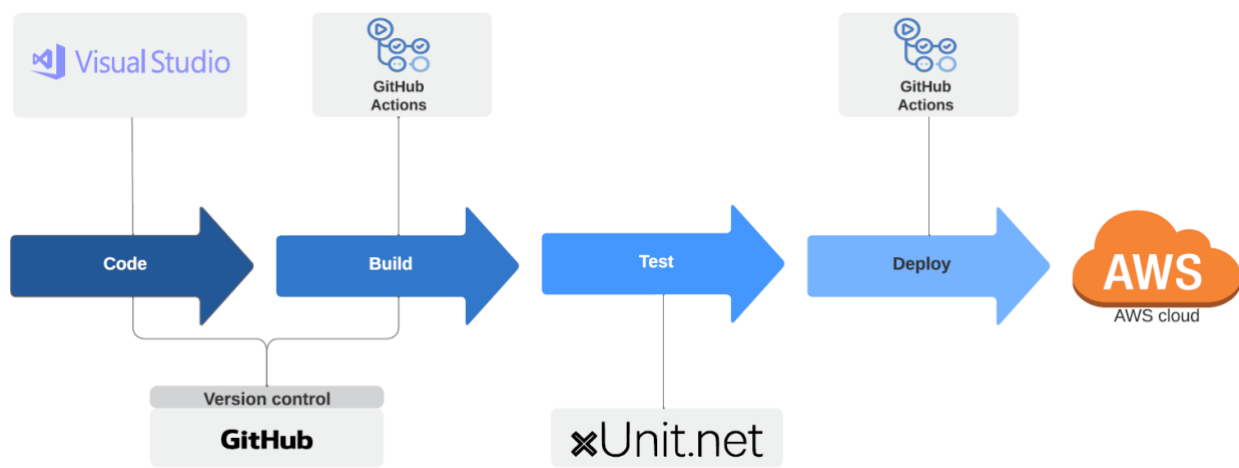


Figure 30: The Workflow

Self-Hosted Runner

The self-hosted runners can be configured within GitHub by navigating to a repository and visiting the Settings page for that repository. Once there, expand the Actions option, and click Runners.

On the Runners page, you will find the self-hosted runners associated with the repository you are viewing. If no self-hosted runners (as shown in Figure 23) are configured for this repository, this is the page where you can set them up.

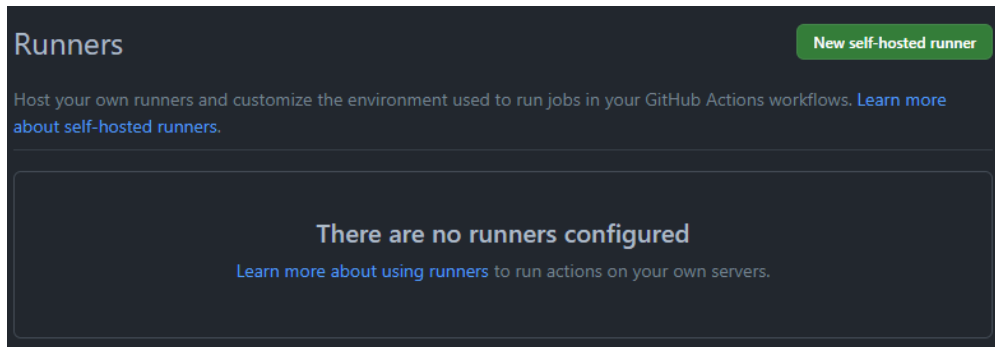


Figure 31: Runner Setup

GitHub Actions Runners

The self-hosted runners are currently installed on the following EC2 instances:

GitHub Project	EC2 Instance	Runner Directory	URL
VTMECS	VTDS (i-0a1b63cdf91bb0294)	C:\actions-runner	http://ec2-3-209-218-208.compute-1.amazonaws.com/VTMECS
VSMS	VSMS (i-05b8e38db7a5812be)	C:\vsms-runner	http://ec2-184-73-161-243.compute-1.amazonaws.com/

Setting Up the GitHub Actions Self-hosted Runner

This section will walk through the steps to configure the GitHub Actions self-hosted runners for our repositories.

What you will need:

- Access to the GitHub organization [UWF-CfC-FDOT](#).

- Access to the target EC2 instance running Windows Server.

Setup Procedure

1. Navigate to the repository where you would like to set up a self-hosted runner.
2. Once there, navigate to the Settings page > Actions > Runners.

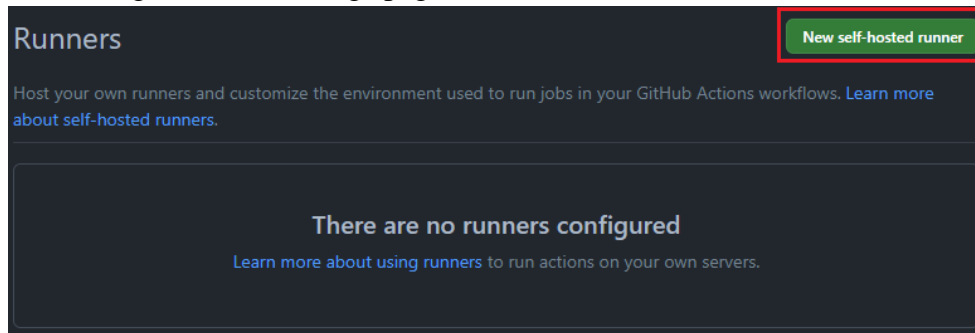


Figure 32: Creating New Self-hosted Runner

3. Click the *New self-hosted runner* button as shown in Figure 24.
4. Once you click the New self-hosted runner button will take you to a page where you can configure the runner's image and architecture. For all the self-hosted runners, we've used Windows x64.
5. Login to the EC2 instance where we will be installing the self-hosted runner.
6. Once you are logged into the Windows Server, open PowerShell and change directory (cd) into the root directory of the C drive. See Figure 25. *** If reinstalling a self-hosted runner, delete the existing actions-runner folder and start over.**

```

Download
We recommend configuring the runner under "\actions-runner". This will help avoid issues related to service identity folder permissions and long path restrictions on Windows.

# Create a folder under the drive root
$ mkdir actions-runner; cd actions-runner

# Download the latest runner package
$ Invoke-WebRequest -Uri https://github.com/actions/runner/releases/download/v2.302.1/actions-runner-win-x64-2.302.1.zip -OutFile actions-runner-win-x64-2.302.1.zip

# Optional: Validate the hash
$ if((Get-FileHash -Path actions-runner-win-x64-2.302.1.zip -Algorithm SHA256).Hash.ToUpper() -ne '18d678f79845d0f1270f755'.ToUpper()){ throw 'Computed checksum did not match' }

# Extract the installer
$ Add-Type -AssemblyName System.IO.Compression.FileSystem ;
[System.IO.Compression.ZipFile]::ExtractToDirectory("$PWD/actions-runner-win-x64-2.302.1.zip", "$PWD")

Configure

# Create the runner and start the configuration experience
$ ./config.cmd --url https://github.com/UWF-CfC-FDOT/VTMECS --token AIPOQ7AI

# Run it!
$ ./run.cmd

```

Figure 33: Windows PowerShell

7. Next, follow the instructions provided in GitHub for installing the self-hosted runner.
8. After you complete the PowerShell steps your self-hosted runner should appear in GitHub. See Figure 26.

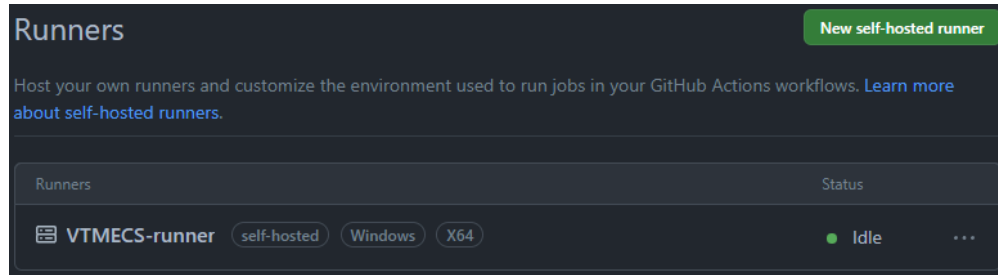


Figure 34: New Self-hosted Runner

9. The last step is configuring the GitHub Actions Runner service in the EC2 Windows instance.
 - a. Open the Services program by clicking the Windows icon and searching for "Services."
 - b. Find the GitHub Actions Runner service in the list and right-click the service and click Properties.
 - c. Navigate to the Log On tab and change the log on setting to Local System account. See Figure 27.

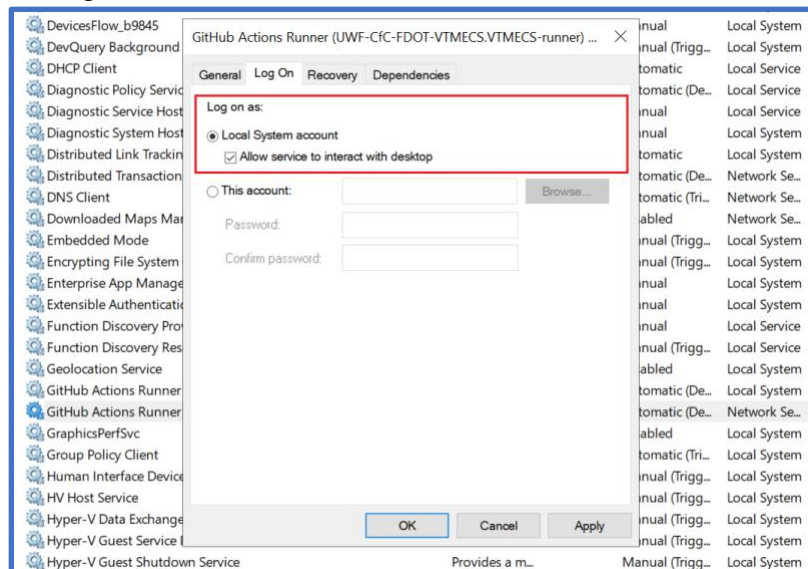


Figure 35: Log-on Configuration

Configuring the SQL Server

If the application that you plan to deploy using GitHub Actions requires a SQL Server database, then you will need to follow the steps below to configure the SQL Server credentials. See Figure 28.

1. Remote into the EC2 instance and open SQL Server Management Studio (SSMS).
2. Right-click the SQL Server and go to Properties.

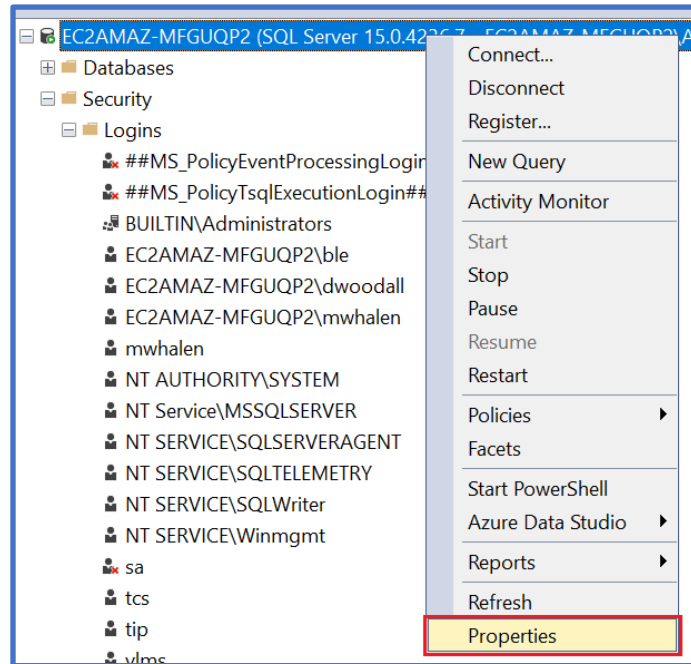


Figure 36: Configuring the SQL Server

3. Navigate to Security and set **Server Authentication** to **SQL Server and Windows Authentication**. See Figure 29.
4. Next, we will create and configure a new SQL Server user. Expand the Security folder and right-click Logins and click New Login. See Figures 30 and 31.

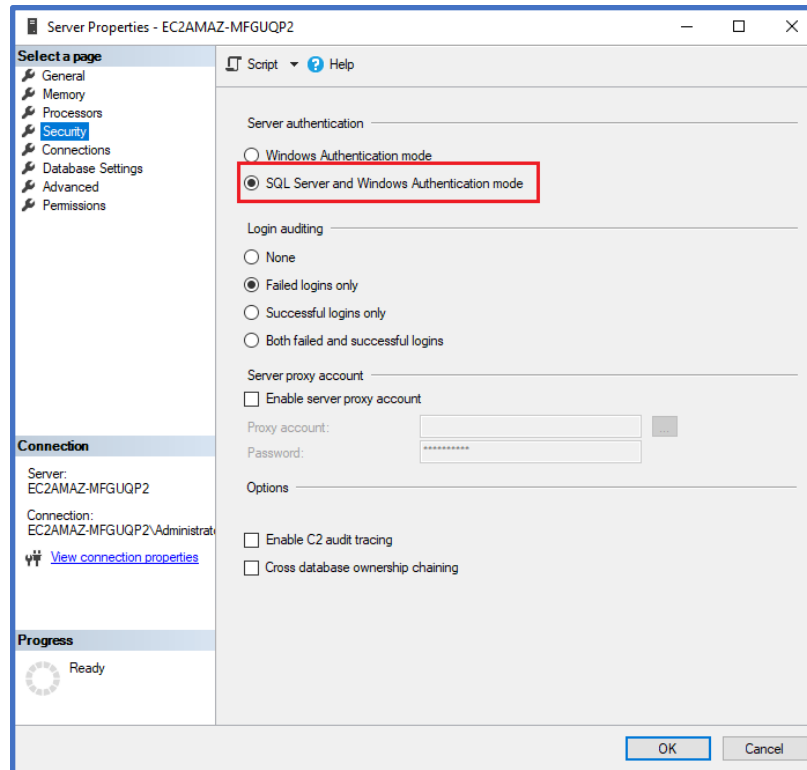


Figure 37: Configuring the SQL Server Authentication

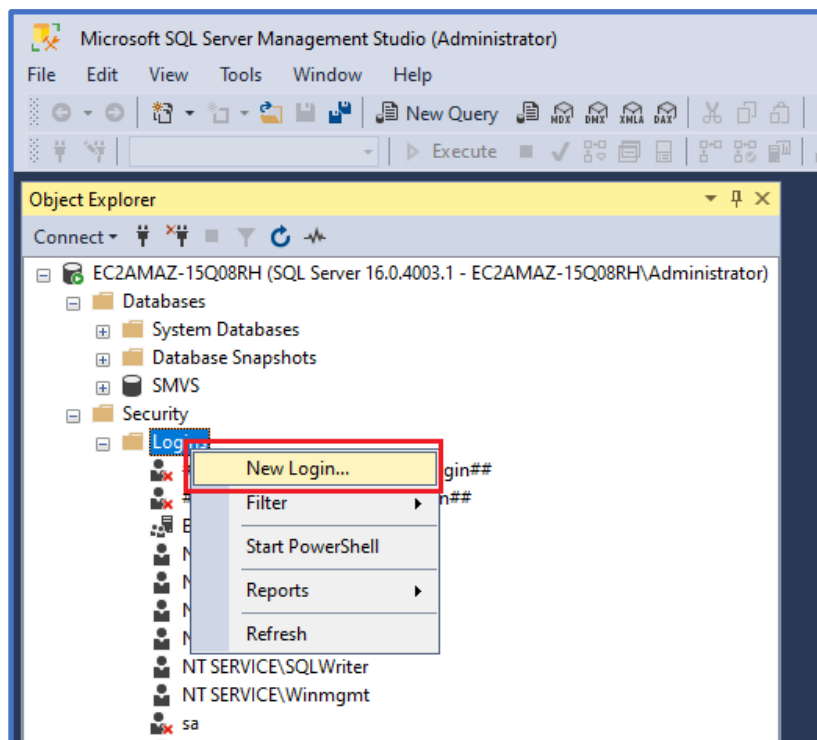


Figure 38: Creating a New Login Account

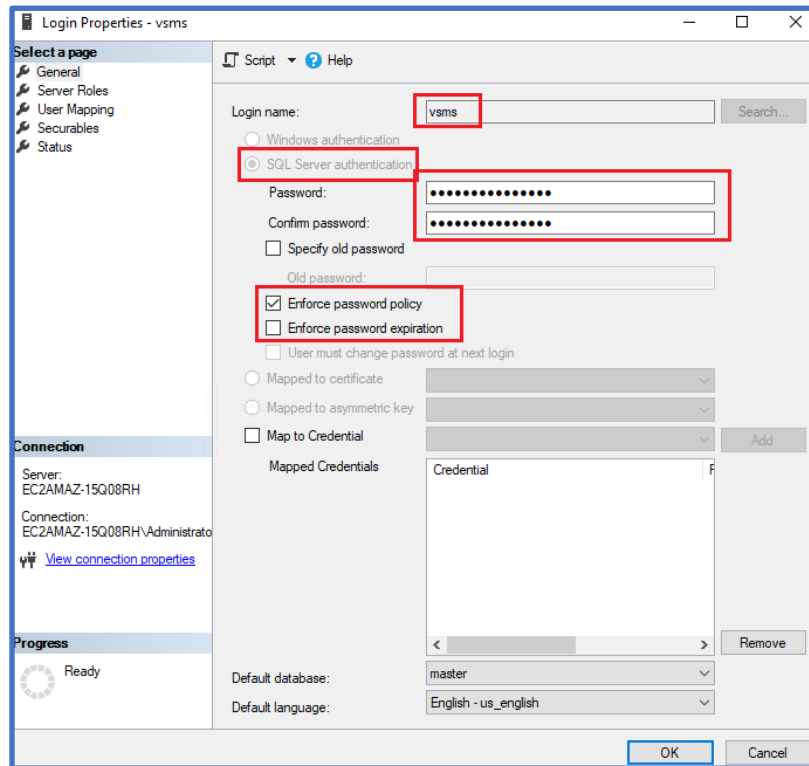


Figure 39: Setting Authentication Properties

5. Next, fill out the fields and the **Server Roles** tab. See Figure 32.

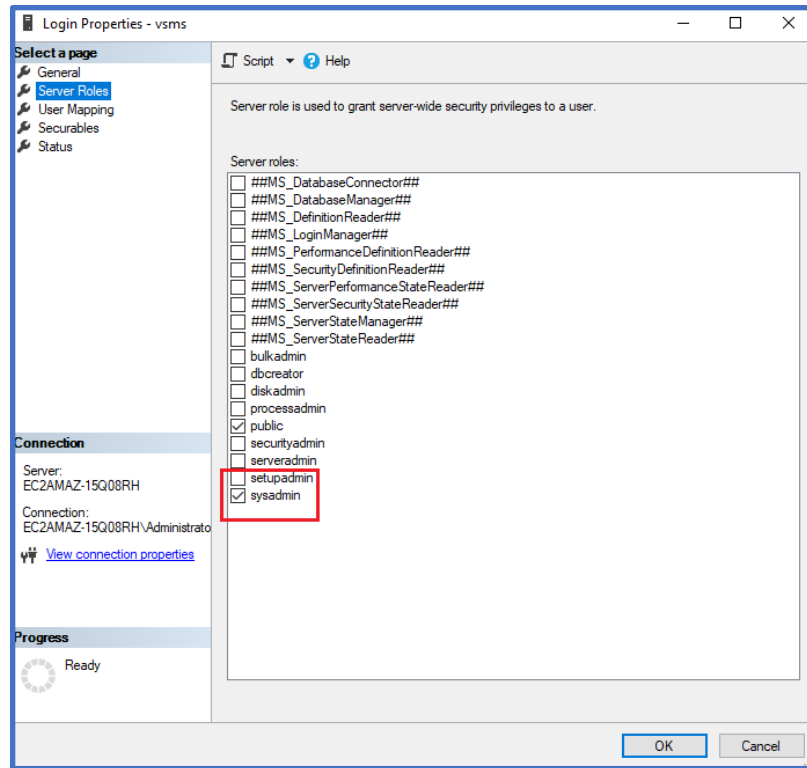


Figure 40: Setting Server Roles

Running the Pipeline

Currently, the CI/CD pipelines are set up so that they must be manually triggered to deploy code to AWS. The reason the pipelines are set up this way is that the EC2 instances are not always online when the pipeline is run, which will result in an error.

The pipeline can be run manually using the steps as shown in Figure 33.

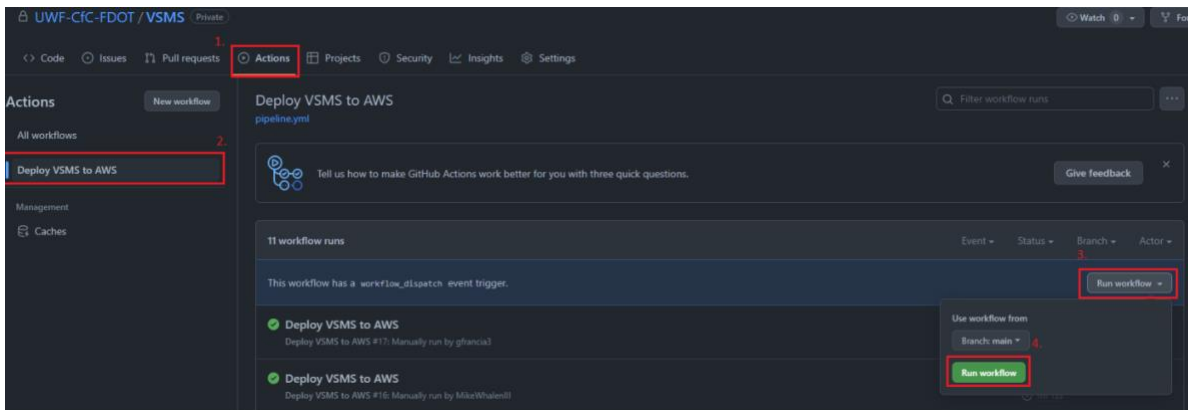


Figure 41: Manually Running a Pipeline

If you want to enable the pipeline to start automatically when there is a push or pull request created for the main branch, edit the pipeline file found at: `.github > workflows > pipeline.yml` and remove the extra space after “main.” See Figure 34.

```
1 name: Deploy VSMS to AWS
2
3 # Currently, you have to manually trigger the GitHub Actions pipeline to deploy to AWS.
4 # remove the space after main to enable automatic deployment.
5 on:
6   push:
7     branches: [ "main "]
8   pull_request:
9     branches: [ "main "]
```

Figure 42: Running a Pipeline Automatically

Additional Notes

- Self-hosted runners are automatically removed from GitHub if they have not connected to GitHub Actions for more than 14 days. [\[Source\]](#) If a runner is automatically removed from GitHub, remove the old actions-runner directory on the EC2 instance and rerun the above steps.
- SQL user passwords should be stored using **Actions secrets and variables**. You can then use these secrets in your pipeline.yml file using the following format: `{{ secrets.VTMECS_DB_USER_PASSWORD }}`. See Figure 35.

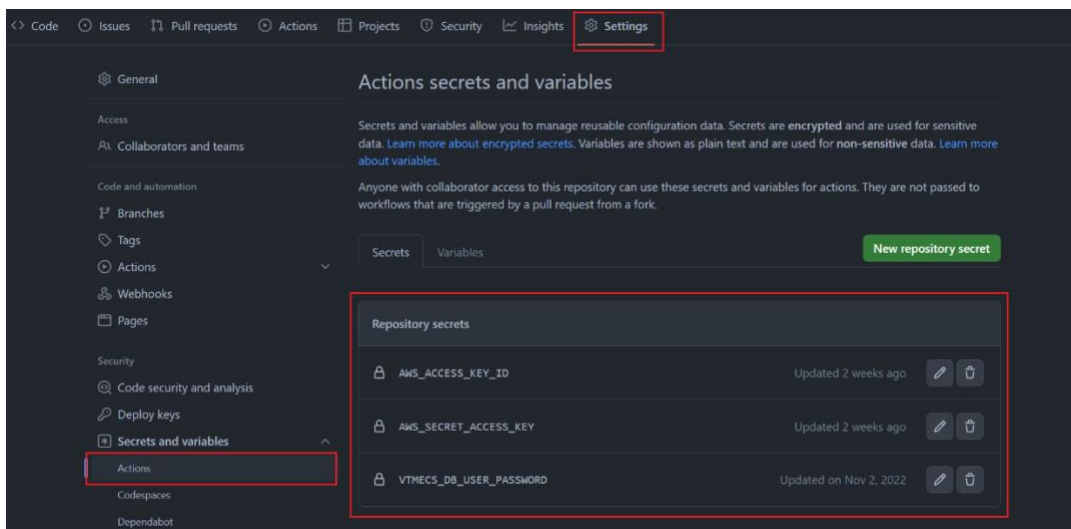


Figure 43: Action Secrets and Variables