



NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

RISK WEIGHTED VULNERABILITY ANALYSIS IN AUTOMATED RED TEAMING

by

Audrey C. Muse

September 2022

Thesis Advisor:

Co-Advisor:

Alan B. Shaffer

Gurminder Singh

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE September 2022		3. REPORT TYPE AND DATES COVERED Master's thesis
4. TITLE AND SUBTITLE RISK WEIGHTED VULNERABILITY ANALYSIS IN AUTOMATED RED TEAMING			5. FUNDING NUMBERS	
6. AUTHOR(S) Audrey C. Muse				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.			12b. DISTRIBUTION CODE A	
13. ABSTRACT (maximum 200 words) The Cyber Automated Red Team Tool (CARTT) automates red teaming tasks, such as conducting vulnerabilities analysis in DOD networks. The tool provides its users with recommendations to either mitigate cyber threats against identified vulnerabilities or with options to exploit those vulnerabilities using cyber-attack actions. Previous versions of CARTT, however, did not consider a risk weighting of identified vulnerabilities before the exploitation phase. This thesis focused on extending CARTT by implementing a risk weighted framework that provides a risk-based analysis of identified vulnerabilities. The framework is based on the Host Exposure algorithm presented by the Naval Research Laboratory and was built into the existing CARTT server using the Python programming language. The resulting risk-based analysis of vulnerabilities is presented to the CARTT user in an easily readable table that provides more complete and actionable information. The implementation of this risk-weighted framework provides CARTT with enhanced analysis of vulnerabilities that pose the greatest risk to a target network.				
14. SUBJECT TERMS advanced cyber operations, ACO, Cyber Automated Red Team Tool, CARTT, Department of Defense, DOD, Department of the Navy, DON, Director, Operational Test and Evaluation, DOT&E, graphical user interface, GUI, offensive cyber operations, OCO, persistent cyber operations, PCO			15. NUMBER OF PAGES 73	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UU	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

**RISK WEIGHTED VULNERABILITY ANALYSIS IN
AUTOMATED RED TEAMING**

Audrey C. Muse
Lieutenant, United States Navy
BS, Troy University, 2007
MS, Troy University, 2009

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

**NAVAL POSTGRADUATE SCHOOL
September 2022**

Approved by: Alan B. Shaffer
Advisor

Gurminder Singh
Co-Advisor

Gurminder Singh
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The Cyber Automated Red Team Tool (CARTT) automates red teaming tasks, such as conducting vulnerabilities analysis in DOD networks. The tool provides its users with recommendations to either mitigate cyber threats against identified vulnerabilities or with options to exploit those vulnerabilities using cyber-attack actions. Previous versions of CARTT, however, did not consider a risk weighting of identified vulnerabilities before the exploitation phase. This thesis focused on extending CARTT by implementing a risk weighted framework that provides a risk-based analysis of identified vulnerabilities. The framework is based on the Host Exposure algorithm presented by the Naval Research Laboratory and was built into the existing CARTT server using the Python programming language. The resulting risk-based analysis of vulnerabilities is presented to the CARTT user in an easily readable table that provides more complete and actionable information. The implementation of this risk-weighted framework provides CARTT with enhanced analysis of vulnerabilities that pose the greatest risk to a target network.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
A.	PURPOSE.....	1
B.	RESEARCH QUESTIONS.....	2
	1. Primary Question.....	2
	2. Secondary Question	2
C.	SCOPE	2
D.	BENEFITS OF STUDY.....	2
E.	CHAPTER ORGANIZATION.....	2
	1. Chapter II: Background.....	3
	2. Chapter III: Design Methodology	3
	3. Chapter IV: Risk Framework Implementation	3
	4. Chapter V: Conclusion and Future Work.....	3
II.	BACKGROUND	5
A.	INTRODUCTION.....	5
B.	CURRENT STATE OF DOD OPERATIONAL TESTING AND EVALUATION	6
	1. DOT&E Cyber Assessments	6
	2. Persistent Cyber Operations.....	6
	3. Advanced Cyber Operations.....	7
	4. Improving DOD Offensive Cyber Operations Capabilities and Processes.....	7
C.	CYBER AUTOMATED RED TEAM TOOL OVERVIEW	8
	1. Architecture.....	8
	2. Prior CARTT Research.....	8
	3. Current Design	9
D.	RISK WEIGHTING METHODS.....	10
	1. Common Vulnerability Scoring System.....	11
	2. National Institute of Standards and Technology, Special Publication (NIST SP800-53B) Control Framework.....	13
	3. Factor Analysis of Information Risk Framework.....	15
	4. Quantitative Bowtie Model	18
E.	QUANTITATIVE RISK WEIGHTING ANALYSIS.....	21
	1. Event Prioritization Framework	21
	2. Target Importance Rank Algorithm.....	22
F.	CHAPTER SUMMARY.....	26

III.	DESIGN METHODOLOGY	27
A.	RISK WEIGHTING VULNERABILITIES IN CARTT	27
B.	SYSTEM DESIGN.....	27
1.	CARTT Server Design.....	27
2.	Process Flow for Vulnerability Identification	30
3.	Vulnerability Results Design.....	31
C.	RISK WEIGHTING FRAMEWORK	32
1.	CVSS Base Vector Score	32
2.	Host Exposure	34
D.	CHAPTER SUMMARY.....	34
IV.	RISK FRAMEWORK IMPLEMENTATION.....	35
A.	ENVIRONMENT SETUP	35
B.	FRAMEWORK SCRIPTING.....	36
1.	Python Functions	36
2.	Hypertext Preprocessor Scripting	40
3.	Responsive Vulnerability Results Table	40
C.	ENVIRONMENT SCENARIO TESTING.....	41
D.	RESULTS	46
E.	CHAPTER SUMMARY.....	47
V.	CONCLUSION AND FUTURE WORK	49
A.	SUMMARY	49
B.	CONCLUSION	49
1.	Primary Research Question	49
2.	Secondary Research Question	50
C.	FUTURE WORK.....	50
1.	GVM Upgrade Integration.....	50
2.	Streamlining to Improve Performance	51
	LIST OF REFERENCES.....	53
	INITIAL DISTRIBUTION LIST	57

LIST OF FIGURES

Figure 1.	CARTT Architecture. Source: [15].....	10
Figure 2.	CVSS Score Ranges. Source: [18].....	11
Figure 3.	CVSS Score Metrics. Source: [19].	12
Figure 4.	NIST Low-Impact Baseline. Source: [25].	14
Figure 5.	NIST Moderate-Impact Baseline. Source: [25].	14
Figure 6.	NIST High-Impact Baseline. Source: [25].....	15
Figure 7.	LEF Factors of the FAIR Taxonomy. Source: [28].	16
Figure 8.	PLM Factors of the FAIR Taxonomy. Source: [28].	17
Figure 9.	Complete FAIR Taxonomy. Source: [28].....	17
Figure 10.	Example FAIR Qualitative Risk Matrix. Source: [28].	18
Figure 11.	Bowtie Structure Example. Source: [31].	19
Figure 12.	Bowtie Structure Overview. Source: [31].....	19
Figure 13.	Outline of QBowtie Model for Risk Classification. Source: [17].	20
Figure 14.	Diagram of the Event Prioritization Framework. Source: [6].....	22
Figure 15.	Linking Factors for Target Importance Rank Criteria. Source: [33].	23
Figure 16.	EP Framework Parameter Definitions. Source: [33].	24
Figure 17.	TIR Parameter Definitions. Source: [33].	25
Figure 18.	CARTT Server Diagram. Adapted from Source: [15].....	28
Figure 19.	MSF Vulnerability Function. Source: [15].	29
Figure 20.	CARTT Script for the Vulnerability Results Function. Source: [15].	29
Figure 21.	Current CARTT Vulnerability Results Page.	30
Figure 22.	CARTT Process Flow Diagram with Risk Weighting Framework. Adapted from [15].....	31

Figure 23.	CARTT Exploitation Configuration Page.	32
Figure 24.	Example XML Report with CVSS Base and Base Vector Scores.....	33
Figure 25.	Implementation Environment.	35
Figure 26.	MSF Vulnerability Results.	36
Figure 27.	Textual Representation of CVSS Base Vector Score in GVM XML Report.....	37
Figure 28.	Mapping Lists for CVSS Base Vector Quantitative Values.	38
Figure 29.	Results Data for the Windows Server 2016 Target Host (192.162.83.26).	40
Figure 30.	PHP Command to Execute the Risk Weighting Python Script.....	40
Figure 31.	PHP Script for Responsive Table Display.	41
Figure 32.	Operator Main Menu.....	42
Figure 33.	Network Scan Information Input Page.	42
Figure 34.	The Check Previous Scan Page Displaying the Operators Completed Scan Status.	43
Figure 35.	Vulnerability Scan Import Page.....	44
Figure 36.	Workspace Verification with Detected Host and Vulnerabilities Display.	44
Figure 37.	New Results Page with Target Exposure Prioritization Based on CARTT's Risk Weighting Framework.....	45
Figure 38.	Output from Risk-Weighting Functions.	47

LIST OF TABLES

Table 1.	HE Characteristic Qualitative Value. Adapted from Source: [34].	23
Table 2.	CVSS Base Vector Metric and Quantitative Values. Adapted from [34].	33
Table 3.	Target Hosts Vulnerability Results.	46

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AC	access complexity
ACA	access complexity authentication
ACCEPT	A Configurable Cyber Event Prioritization Tool
ACO	advanced cyber operations
ATT	attack surface
Au	authentication
AV	access vector
BID	Bugtraq ID
CAP	cyber assessment program
CARTT	Cyber Automated Red Team Tool
CCMD	combatant command
CIA	confidentiality, integrity, and availability
CIDR	classless interdomain routing
CLI	command line interface
CPT	cyber protection team
CSS	cascading style sheet
CVE	Common Vulnerabilities and Exposures system
CVSS	Common Vulnerability Scoring System
DOD	Department of Defense
DON	Department of the Navy
DOT&E	Director, Operational Testing and Evaluation
EPF	Event Prioritization Framework
FAIR	Factor Analysis of Information Risk
GUI	graphical user interface
GVM	Greenbone Vulnerability Manager
HE	host exposure
HTML	hypertext markup language
IC	initial compromise
INFOSEC	information security
ITL	Information Technology Laboratory

LEF	loss event frequency
MDPI	Multidisciplinary Digital Publishing Institute
MI	max impact
MSF	Metasploit Framework
MV	max vulnerability
NIST	National Institute of Standards and Technology
NPS	Naval Postgraduate School
NSA	National Security Agency
NVD	National Vulnerability Database
OCO	offensive cyberspace operations
OIG	Office of Inspector General
OpenVAS	Open Vulnerability Assessment Scanner
OT&E	operational test and evaluation
PCO	persistent cyber operations
PHP	hypertext preprocessor
PLM	probable loss magnitude
POC	proof of concept
QBowtie	Quantitative Bowtie
RTIB	Reb-Team-in-a-Box
SecDef	Secretary of Defense
SP	special publication
TE	target exposure
TEF	threat event frequency
TIR	Target Importance Rank
USSTRATCOM	U.S. Strategic Command
VM	virtual machine
XML	extensible markup language

I. INTRODUCTION

A. PURPOSE

The Cyber Automated Red Team Tool (CARTT) was developed at the Naval Postgraduate School (NPS) as a cybersecurity tool that automates commonly used tasks when conducting red teaming or penetrating testing [1]. The tool was designed for novice users with little to no experience or knowledge in this area. CARTT provides capabilities to identify and assess the cybersecurity vulnerabilities on Department of the Defense (DOD) computer systems, and provides recommendations to mitigate potential threats against vulnerabilities that are identified on a network [1]. To be in line with the DOD Office of Director, Operational Testing and Evaluation (DOT&E) plans of operation for continued improvement of cyber tools, CARTT's capabilities can be enhanced with the addition of a risk weighting approach to its vulnerability analysis.

Risk weighted analysis is a technique that compares different risks in a system to analyze, measure, and evaluate options for the user based on the discovered vulnerabilities and existing threat information. Shahid Suddle describes the process of decision-making about risk as, "very complex and that it is not only technical, but political, psychological, societal, moral, and emotional aspects play an important role" [2]. Risk weighting can provide a mathematical rating, based on a range of severity levels, for the potential risk that an identified vulnerability poses a to a network.

CARTT assists the user in determining potential exploit modules to use against an identified vulnerability, however, it does not provide a risk analysis for each provided exploit. This stage of the analysis is where the risk weighting approach will be beneficial to the user as it will provide extended analysis on which vulnerabilities pose the most risk if they are exploited. Combining CARTT's current vulnerability analysis with a risk weighted framework that considers the threat, vulnerability, and consequences can enhance the tool's ability to support decision-making when deciding on exploits. This extension to CARTT helps improve confidence in its capabilities and processes.

This research has implemented a prototype framework for a risk weighted approach to CARTT's vulnerability analysis. This framework provides the user with a risk-based decision-making technique that addresses all components of a risk assessment: threat, vulnerability, and consequences.

B. RESEARCH QUESTIONS

1. Primary Question

How can the identified vulnerabilities in a target system be weighted such that they can improve follow-on target exploitation?

2. Secondary Question

How can a risk weighted framework assist red teams when using CARTT?

C. SCOPE

This thesis extends previous research on CARTT by integrating a risk weighted framework within its vulnerability analysis capability. Through the implementation of this framework, CARTT can help to improve the overall cybersecurity of DOD networks and their cyber operations.

D. BENEFITS OF STUDY

The primary benefit of this research is that by risk weighting of identified vulnerabilities, CARTT now considers different aspects of those vulnerabilities to better manage risk. The focus is on defending systems, as well as on the ability to exploit vulnerabilities on adversary systems.

An additional benefit to this research is that the implementation of a risk weighted framework in CARTT shows the progression toward the goal of continued development, planning, testing, and execution that will help improve on the identified challenges facing DOD red teams.

E. CHAPTER ORGANIZATION

The remainder of this thesis is organized into the following chapters.

1. Chapter II: Background

Chapter II provides an overview of the purpose and challenges of DOD Cyber red teams. Next, we review the current state of DOD Operational Testing and Evaluations, and its plan of operation for continued improvements pertaining to offensive cyberspace operation (OCO) tools. This is followed by an overview of prior research, architecture, current design of CARTT, and a brief overview of risk weighting methods. Finally, we review related work in this field.

2. Chapter III: Design Methodology

Chapter III discusses the design methodology used to develop the risk weighting framework for vulnerability analysis in CARTT. It describes in depth the need for risk weighting of vulnerabilities by the current CARTT server, and the risk weighting framework design for CARTT.

3. Chapter IV: Risk Framework Implementation

Chapter IV discusses the testing environment setup, and the risk weighting functions implementation within the CARTT server workflow. It also walks through an analysis scenario that details the steps required to conduct, import, and produce the updated vulnerability results.

4. Chapter V: Conclusion and Future Work

Chapter V summarizes the research efforts of this thesis and conclusions to the research questions. Finally, future work recommendations are provided to expand CARTT functionality and performance.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

A. INTRODUCTION

The DOD defines a cyber red team as “a group of DOD personnel that are authorized and organized to emulate a potential adversary’s exploitation or attack capabilities against a targeted mission or capability” [3]. A cyber red team’s mission, according to the DOD [3], is to “identify exposed information and vulnerabilities, improve joint cyberspace operations, and protect the DOD Information Network and DOD weapons systems from vulnerabilities and threats that affect the DOD’s security posture.” Red teams use known vulnerabilities to mimic tactics an adversary may use to infiltrate a system or network [4].

In 2020, the DOD Office of Inspector General (OIG), released results from a subject audit which determined whether DOD red teams took action to resolve issues that were identified in a 2012 report titled, “Better Reporting and Certification Processes Can Improve Red Teams’ Effectiveness” [4]. The report identified challenges facing future red teams and their missions. Additionally, the report included corrective actions to be taken by red teams to mitigate those identified vulnerabilities. One major finding in the audit addressed an enterprise-wide need to develop baseline tools to perform red team assessments. The OIG determined that “the lack of a coherent, unified plan to train, support, prioritize, and fund the DOD red teams has caused this gap in capability” [5]. Concluding the audit results, the OIG recommended to the Secretary of Defense (SecDef) that action be taken to ensure development and implementation of baseline capabilities and processes, which include implementing risk-based toolkits for DOD red team vulnerability assessments [4], [5]. In order for DOD red teams to effectively provide mission support, they need better tools, enhanced methods, and better information [6].

B. CURRENT STATE OF DOD OPERATIONAL TESTING AND EVALUATION

This section discusses the current state of DOD DOT&E cyber assessments, a review of Persistent and Advanced Cyber Operations, and an overview of the ongoing plan of operation for continued improvements pertaining to OCO tools.

1. DOT&E Cyber Assessments

According to the Chairman of the Joint Chiefs of Staff Instruction, as a requirement to maintain certification and accreditation, DOD red teams must report the results of all assessment to the DOT&E, along with other assessed organizations [3]. The DOT&E is responsible for issuing operational test and evaluation (OT&E) policy and procedures for the DOD [7]. The OT&E also analyzes DOD acquisition programs in order to make budgetary recommendations to the SecDef. These recommendations are based on independent assessments of combat defense system requirements to ensure operational effectiveness and suitability [7].

In [8], the DOT&E highlighted gaps in DOD cybersecurity assessments. The article attributed these gaps to the inability of DOD's existing capabilities to mimic complex real-world cyber threats. The article highlighted the Cyber Assessment Program (CAP) and its contribution to developing defenses against advanced cyber threats, but also acknowledged that the risk of disruptions to DOD missions remains high due to the slow development of effective capabilities. Due to the lag in DOD cyber development and resources, red teams have become unable to quickly detect or effectively emulate more sophisticated near-peer attacks [8]. CAP allows the DOT&E to identify critical red team capability gaps to serve as vulnerabilities for improvement of systems, equipment, and operational testing.

2. Persistent Cyber Operations

Red teams are critical to the DOD network systems and structures evaluation. In [9], the Joint Chief of Staff specifies that only DOD red teams certified by the National Security Agency (NSA) and accredited by U.S. Strategic Command (USSTRATCOM) are authorized to operate on DOD networks. DOD red teams have been pushed to conduct Persistent Cyber Operations (PCO) that align with what a nation-state would attempt [5].

In [10], DOT&E's assessment stated that PCO provided red teams with longer durations on DOD networks to simulate complex, long-term adversarial actions without detection. The assessment discovered red teams were able to elevate network privileges to conduct more advanced follow-on operation after gaining network access.

PCO works closely with the Cyber Protection Teams (CPT) to report on current operations. PCO works to identified vulnerabilities and implement effective mitigations during follow-on operations [11]. In the FY 2020 Annual Report [10], DOT&E announced plans to evolve PCO by integrating a campaign-plan element that combines intelligence and other support components. The objective of the campaign is to better portray identified advanced cyber threats and allow PCO to expand across additional combatant commands (CCMD) [10], [12].

3. Advanced Cyber Operations

To handle instances where little to no advanced preparation is available for an assigned mission, the DOT&E created an Advanced Cyber Operations (ACO) team. The ACO is comprised of subject matter cyber experts that augment DOD red teams to provide additional assistance on missions as well as implementing new cyber tools, tactics, techniques, and procedures. The ACO also assists red teams in the portrayal of advanced capabilities that cyber adversaries likely possess [10].

The FY 2020 Cybersecurity Assessment [10], discussed the how cyber capabilities and advancements created by ACO directly correlate with the developments of advanced techniques which are used in planning and execution of PCO operations.

4. Improving DOD Offensive Cyber Operations Capabilities and Processes

In [10], the DOT&E acknowledged that to improve OCO capabilities that enhance operational testing, their teams continue to execute operational cyber assessments with each service representative. The increase of FY22 DOD budget for DOT&E includes identifying initiatives to conduct OCO without significant effects to critical operational capabilities [12]. This budget increase grants the DOT&E to address the challenges that may prevent inadequate operational cyber assessments [10], [12].

C. CYBER AUTOMATED RED TEAM TOOL OVERVIEW

CARTT was developed at the NPS as an automated red teaming tool. It was designed to assist novice users who may have little to no experience or knowledge of conducting vulnerability assessments or penetration testing techniques. To this end, it automates a series of common red teaming tasks, so novice users can perform operations [1]. CARTT is a portable cybersecurity tool that conducts vulnerability assessments on DOD networked systems and embedded devices. Joseph Plot acknowledged in his research in CARTT, that the systems and devices that CARTT scans include both essential and non-essential DOD computer systems [1]. A completed vulnerability assessment provides the user with recommendations to mitigate any identified vulnerabilities [13].

1. Architecture

CARTT is built on open source software. The main cyber operations tools used in CARTT are the Greenbone Vulnerability Manager (GVM), formerly the Open Vulnerability Assessment Scanner (OpenVAS), which is a full-featured vulnerability scanner, and the Metasploit Framework (MSF), a penetration testing platform that enables the user to write, test, and execute exploits against target vulnerabilities [1], [14]. The tool interacts with a graphical user interface (GUI) which enables them to run scripts designed to perform tasks for conducting its vulnerability assessments on a target host. The CARTT GUI was implemented to automate portions of the underlying red team tools and processes so that users would not become overwhelmed by the command line interface (CLI) [1].

2. Prior CARTT Research

The initial CARTT implementation (originally named “Red-Team-in-a-Box” [RTIB]) was developed in June 2019 in a master’s thesis research by NPS student Joseph Plot, which focused primarily on automated vulnerability assessment [1]. In December 2019, NPS graduate student Preston Edwards extended the research by creating functionality to allow the user to select various scripted cyber-attacks based on identified vulnerabilities [13]. This research restructured and improved CARTT by allowing the system to be updated to provide real-time feedback that a non-expert user could understand.

This improved CARTT's design such that a novice user could interact and understand the results.

In 2020, NPS student Joseph Berrios extended CARTT to focus on broadening capabilities from its then stand-alone, host-based implementation to a web-based client-server model. This new architecture identified potential targets on network and communicated the results to the CARTT server, which then provided options for follow-on action for the user [5]. The most recent research was completed in September 2021 by NPS graduate student Ousmane Goumandakoye, who developed a more robust and more usable web-based GUI for accessing CARTT's tools and techniques to identify vulnerabilities and launch exploits against them. The objective was to improve the aesthetics of CARTT and improve responsiveness and efficiency for improving usability and the user experience [14].

3. Current Design

The CARTT GUI was designed for non-expert users with little to no experience or knowledge of using tools such as GVM and MSF. The current web-based GUI provides CARTT users with the essential tools necessary to conduct vulnerability scans on an organization's targeted IT infrastructure by automating tasks that require knowledge and experience in red teaming, to scan the vulnerabilities on an organization's targeted IT infrastructure [14]. The current design still provides three user roles (Operator, Commander, and Administrator), now with improvements for simplicity for the user [14]. Figure 1 shows the users and a simplified architecture for CARTT.

Although the user GUI improvements increased the user's memorability and learnability, and to minimized user errors, the functionality of CARTT's design still has only two main components: IP host discovery and vulnerability exploitation [13], [14]. The CARTT GUI allows the user to create and run an automated vulnerability scan, through GVM, on a specific target based on its IP address and subnet mask. After the scan is complete, the user will import the scan into the MSF [14]. The interface provides the user with a list of the target host's vulnerabilities along with a list of exploit modules for the selected host vulnerability [14]. CARTT assists the user in determining which exploit

module to use by providing descriptions of the selected module [13]. However, CARTT does not provide risk analysis for each provided exploit module. This stage of the CARTT process is where a risk weighting approach will be beneficial as it will provide an extended analysis on which exploitable vulnerabilities pose the most risk before continuing.

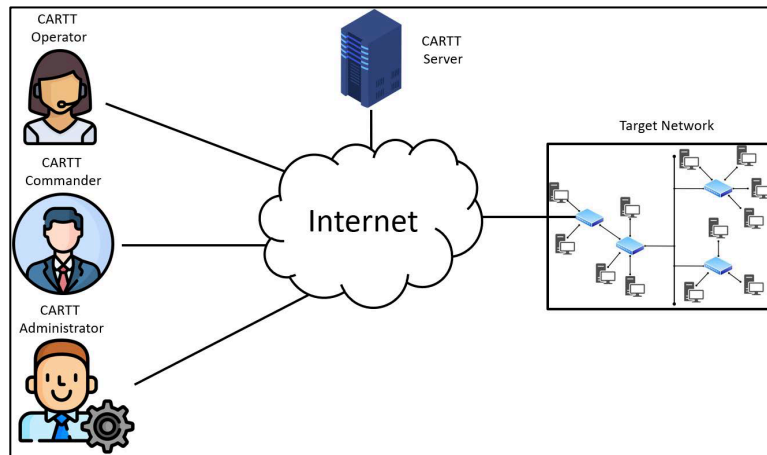


Figure 1. CARTT Architecture. Source: [15].

To be in line with the OIG’s recommendation to SecDef and the DOT&E plan of operation for continued improvement in OCO tools, CARTT’s capabilities can be enhanced with the addition of a risk weighting approach in its vulnerability and threat analysis.

D. RISK WEIGHTING METHODS

Risk weighting analysis compares different risks to analyze, measure, and evaluate options for the user based on the discovered vulnerabilities and existing threat information [2]. Although there are many cybersecurity risk management frameworks that address the threats, assets, and controls that provide guidance for implementing mitigations, these frameworks do not include risk weighted analysis [16]. In [2], Suddle describes the process of decision-making about risk as “very complex and is not only technical, but political, psychological, societal, moral, and emotional aspects play an important role” [2]. According to Barry Sheehan’s 2021 *Journal of Risk Research* article, conventional

frameworks calculate risk based on historical frequency and severity of incurred losses. Sheehan claims that these techniques are “effective for known risks but due to the absence of historical data, prove ineffective for assessing cyber risk” [17].

In [2], Siddle introduced the six steps of a risk analysis: scope definition, hazard identification, modelling of hazard scenarios, estimation of consequences, estimation of probabilities, and estimation of risks. In “An Adversarial Risk Analysis Framework for Cybersecurity,” David Rios et al. simplifies the six steps and defines risk analysis as a methodology for addressing a cyber threat, prioritizing a defense, and implementing security controls [16].

The following sections provide overviews of several notable risk weighting methods.

1. Common Vulnerability Scoring System

The Common Vulnerability Scoring System (CVSS) is an assessment which assigns a score to a known vulnerability based on specific characteristics, including the impact of the attack. CVSS is an open framework that provides an assessment of the severity of known vulnerabilities on a scale of 1 to 10, as seen in Figure 2 [17].



Figure 2. CVSS Score Ranges. Source: [18].

As noted on their website, CVSS scores are used by information security (INFOSEC) teams during assessments to analyze and compare vulnerabilities and then prioritize mitigations [19]. Base, Temporal, and Environmental are the metric groups that

make up the CVSS scoring metric. The underlying scoring components are shown in Figure 3.

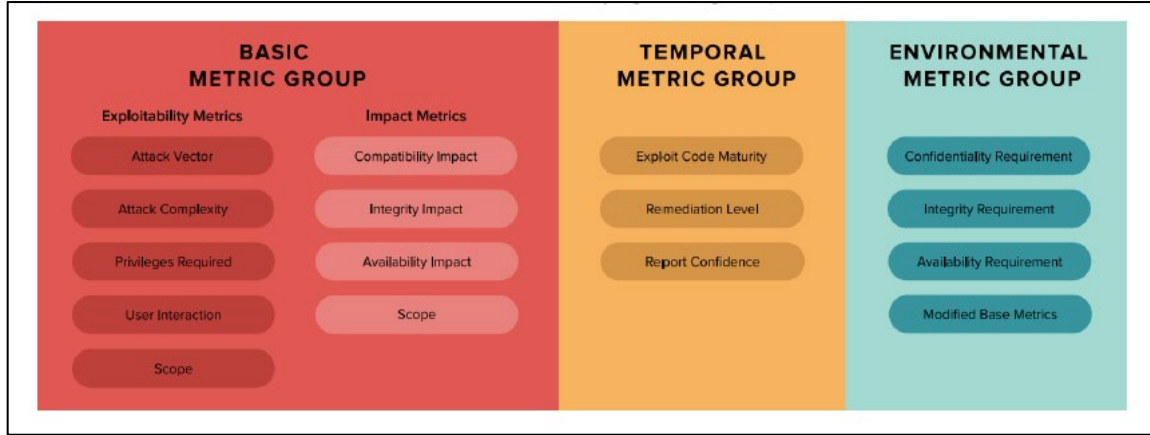


Figure 3. CVSS Score Metrics. Source: [19].

In [18], Jason Sumpter defines the Base metric as the “intrinsic qualities of a vulnerability that are constant over time and across user environments.” The vendor or organization that created the vulnerable product decides the Base metric parameters for scoring. The National Vulnerability Database (NVD) publishes and updates Base scores for known vulnerabilities. The CVSS Base scores are of limited use for assessing a vulnerability by itself because it does not account for other real world or environmental controls that an organization has put into place [19].

Unlike the Base metric group, the Temporal metric group only reflects vulnerability characteristics that change over time [18]. The Temporal metrics calculate the exploitability of the identified vulnerability. Sumpter discusses a new vulnerability exploit proof-of-concept (POC) as an example of an event that can lead to a new temporal metric scoring. There is a vast repository of open source verified vulnerability exploit POCs available online. This poses serious risks for an organization whose critical systems may contain an identified vulnerability. For this purpose, the Temporal metric is designed to ensure the exploitability of an identified vulnerability is captured in the overall CVSS score [18].

The Environmental metric group reflects the vulnerability characteristics that are unique to a specific environment and accounts for any security controls in place. For example, achieving a low environmental metric can be accomplished by mitigating the consequences of a successful exploit against a particular vulnerability [18]. As discussed earlier in this section, the CVSS Base score alone does not provide enough for a complete vulnerability assessment. In addition to identifying a vulnerability, the impact of a successful exploit may have should be identified as well. This explains the importance of the Base and Environmental metric groups within the CVSS scoring system [18].

Sheshan concluded in [17], that although CVSS is considered a comprehensive scoring system for vulnerabilities, the scoring design lacks the ability to weight the risks associated with an identified vulnerability.

2. National Institute of Standards and Technology, Special Publication (NIST SP800-53B) Control Framework

According to the National Institute of Standards and Technology (NIST) website, it was founded in 1901 and is one of the nation's oldest physical science laboratories. NIST was created to change the slow progression of U.S. industrial development compared to the capabilities of the United Kingdom, Germany, and other economic rivals [20]. Congress established NIST as a way of propelling the U.S. into an already competitive race for industrial superiority. Today, NIST provides the technological and measurement standards for products and services used worldwide [20]. NIST creates special publications (SP) that provide recommendations and best practices for information security. The Information Technology Laboratory (ITL) is the research arm behind the NIST SP which provide recommendations and best practices which focuses on overall computer security. In 1990, NIST created the SP800 series in support of the security and privacy needs of U.S. Federal Government information and information systems. This series provides security recommendations, technical specifications, and annual cybersecurity reports [21].

Authored by Joint Task Force, NIST SP 800-53B outlines the security controls for protecting sensitive government and civilian information against potential cyber-attacks [22]. The severity of the security impact is based on three levels: low, moderate, and high.

NIST based the impact levels on the CIA dimensions-confidentiality, integrity, and availability [23]. A security impact is considered low and is expected to have a *limited* effect on operations or assets if all three CIA dimensions are low (see Figure 4) [24], [25].

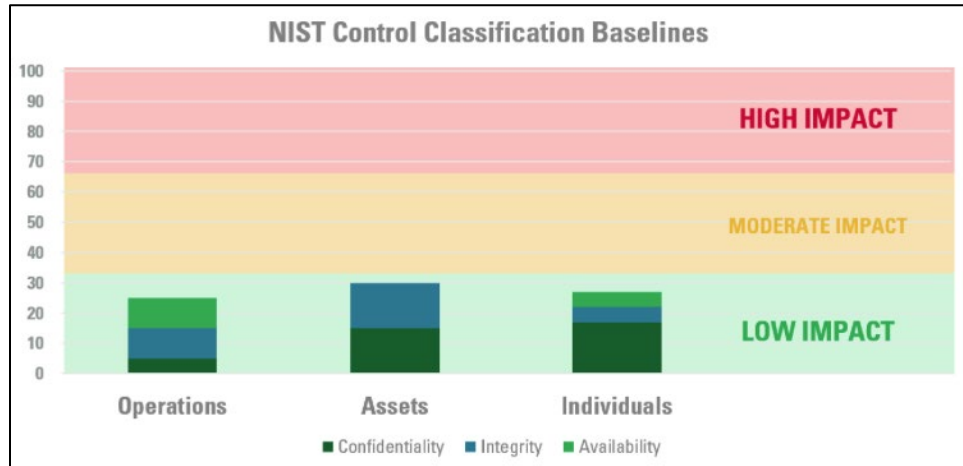


Figure 4. NIST Low-Impact Baseline. Source: [25].

Figure 5 shows an example of a moderate security impact. A security impact is considered moderate and is expected to have a *serious* effect on operations or assets, if at least one of the CIA dimensions is moderate but not high [25].

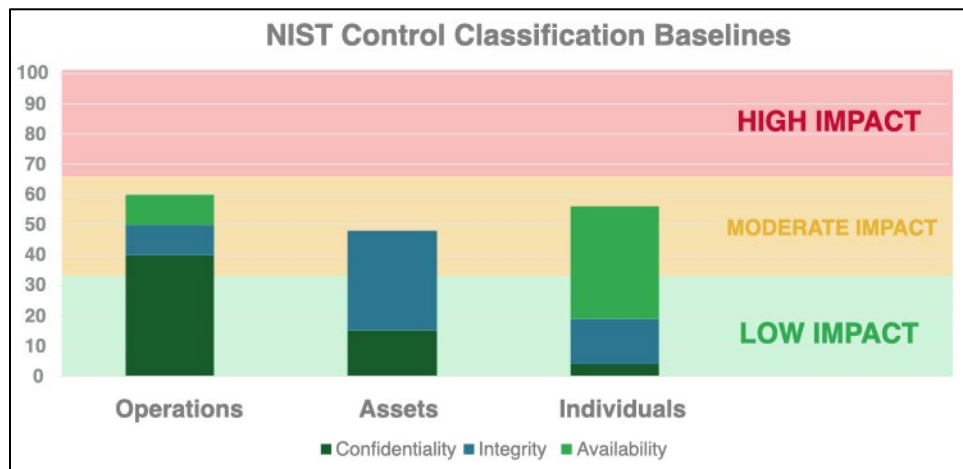


Figure 5. NIST Moderate-Impact Baseline. Source: [25].

A security impact is considered high and is expected to have a *severe* or *catastrophic* effect on operations, if at least one of the CIA dimensions is high [25]. Figure 6 shows an example of a high impact system.

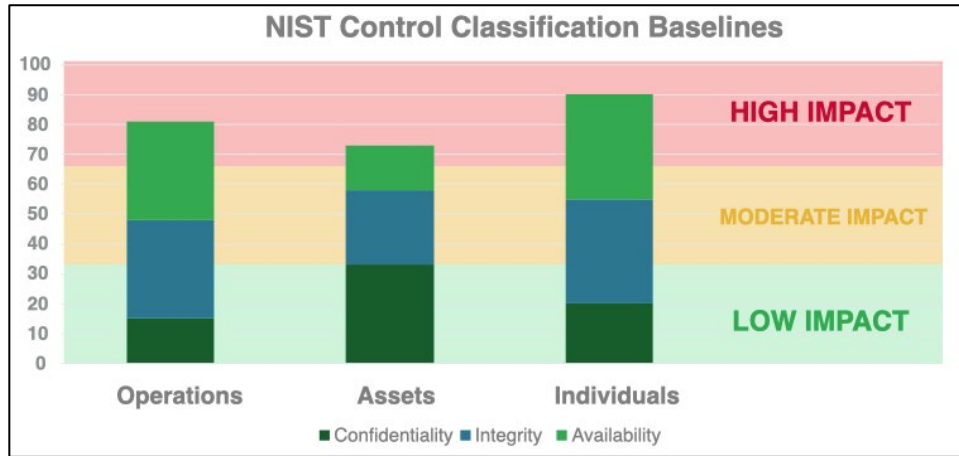


Figure 6. NIST High-Impact Baseline. Source: [25].

Like CVSS, as discussed in Section D.1, NIST SP800-53 recognizes known system vulnerabilities and threats, which are listed in the online NVD [26], [27]. In [17], Sheehan points out that although implementing NIST SP800-53 guidelines can help organizations with defining effective security controls, it lacks the ability to quantify the effects of those controls.

3. Factor Analysis of Information Risk Framework

The Factor Analysis of Information Risk (FAIR) is a risk framework developed by Jack A. Jones [28]. FAIR provides a standard taxonomy and ontology for information and operational risk. FAIR provides best practices which aims to assist organizations understand, analyze, measure, and manage cybersecurity risk [23]. FAIR uses risk factors to measure a system or network impact. These risk factors are described as an event or the likelihood that an event can impact organization.

In [29], Nair states that the first step in the FAIR framework is to define and develop a risk taxonomy by breaking it down to fundamental components. Nair defines a

risk taxonomy as “a comprehensive and stable set of risk categories that are used by an organization.” Nair adds that incorporating a taxonomy into the FAIR framework offers an organization an alternate approach for analyzing risks [29]. According to both Jones and Nair, risk is defined as the probability of loss event frequency (LEF) and probability of loss of magnitude (PLM) [28], [29].

The first part of the FAIR equation describes the factors that drive LEF. LEF is described as the probability that a threat can inflict harm on an asset, within a given timeframe. Jones determined that the threat must take action, resulting in a loss, against an organization’s asset in order to be classified as a loss event [28]. Nair states that a loss event is determined by factoring the threat event frequency (TEF) and Vulnerability. TEF is the probability that a threat can act against an asset, within a given timeframe [29]. Nair describes the driving factors of TEF as *Contact* and *Action*. In [29], Nair defines Vulnerability as the probability of an asset’s inability to resist the actions of a threat. She explains that a vulnerability exists when there is a difference between the force applied by the threat and an asset’s ability to resist that force. There are two factors that drive vulnerability, and they are *Threat Capability* and *Control Strength* [29]. Figure 7 shows all the LEF components added to the taxonomy.

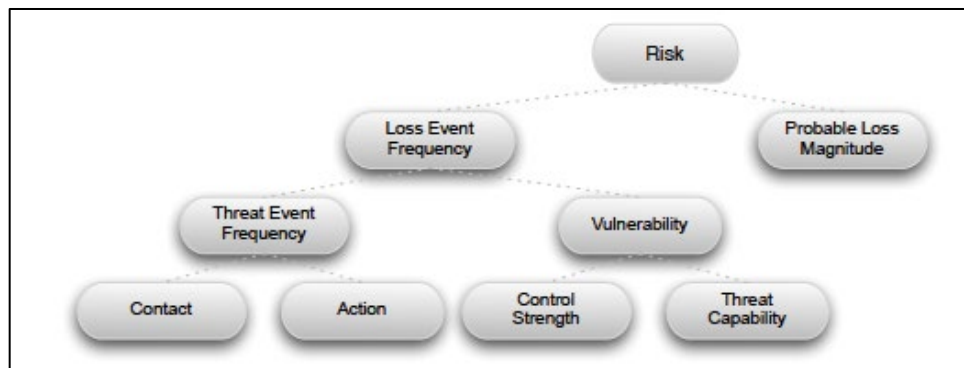


Figure 7. LEF Factors of the FAIR Taxonomy. Source: [28].

The second part of the FAIR equation describes the factors for PLM. PLM is the described as the potential economic loss to an organization due to a risk or loss event [29]. The factors that drive PLM are *Primary Loss* and *Secondary Loss*. Primary loss factors are

characteristics of the organizational environment that can influence the level of loss, whereas the secondary loss factors focus on the organization's external characteristics that can influence the level of loss. Figure 8 shows the PLM components used in the risk equation [28]. Figure 9 shows the complete taxonomy for FAIR.

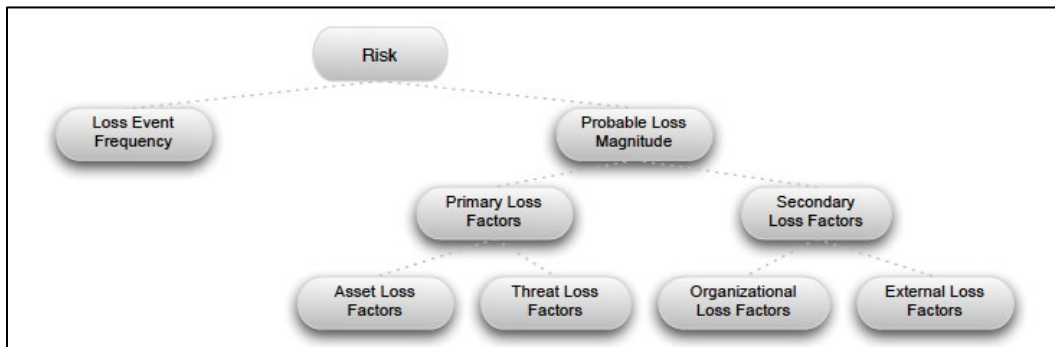


Figure 8. PLM Factors of the FAIR Taxonomy. Source: [28].

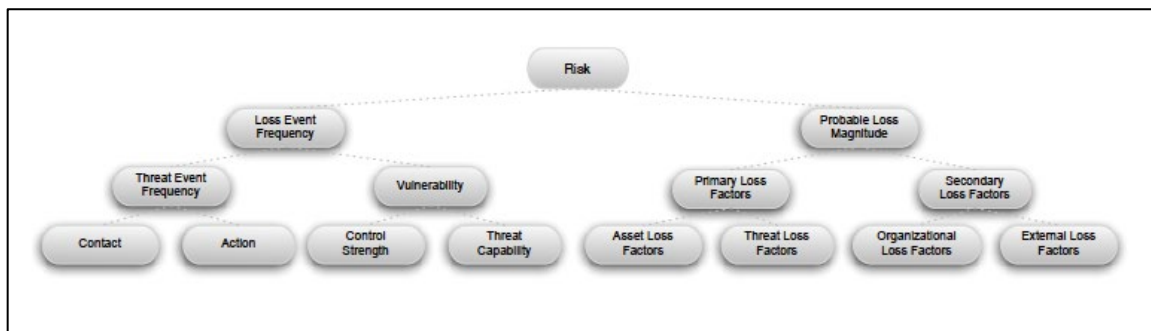


Figure 9. Complete FAIR Taxonomy. Source: [28].

Once the taxonomy is complete, the FAIR analysis can begin. In [29], Nair identifies the four stages to complete the FAIR analysis:

- Stage 1—Identify the asset and the respective threat
- Stage 2—Evaluate LEF
- Stage 3—Evaluate PLM
- Stage 4—Derive and articulate risk.

After stages three and four are complete, stage four concludes the analysis by producing a quantitative analysis matrix to help articulate the risk, as seen in Figure 10.

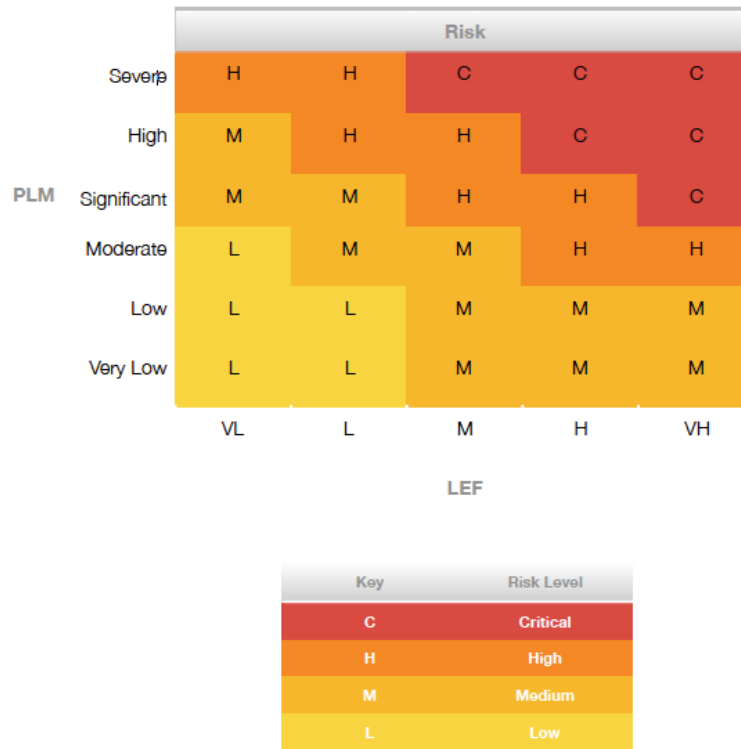


Figure 10. Example FAIR Qualitative Risk Matrix. Source: [28].

4. Quantitative Bowtie Model

Developed by Barry Sheehan from the University of Limerick in 2020, the QBowtie Model is a risk framework that provides a ranked classification for cyber threats and the potential severity incurred by those threats. This is achieved by combining the Bowtie method with a risk matrix [17]. First documented in 1990, the Shell Group adopted the Bowtie method to show comparisons between areas of risk and consequences within the company [31]. Figure 11 shows is a graphical depiction of the Bowtie method. The figure shows the pathway from the identified sources to the consequences of an event or risk [32]. The diagram depicts the relationship between the source of the risk, controls, escalation factors, risk events, and consequences.

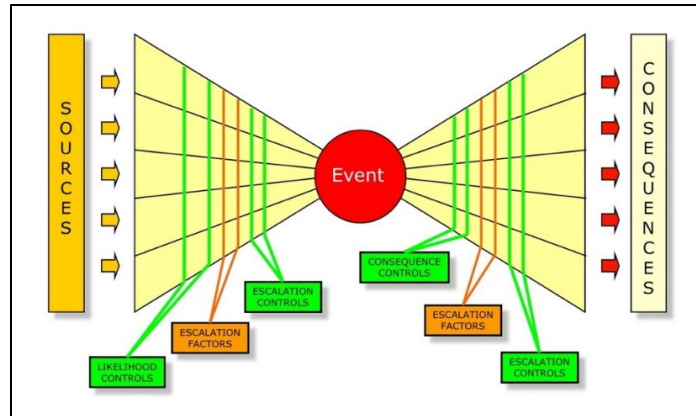


Figure 11. Bowtie Structure Example. Source: [31].

Bowtie analysis can assist an organization in identify security gaps, so that where mitigating controls can be implemented [32]. Figure 12 provides more details on the individual components of the Bowtie model.

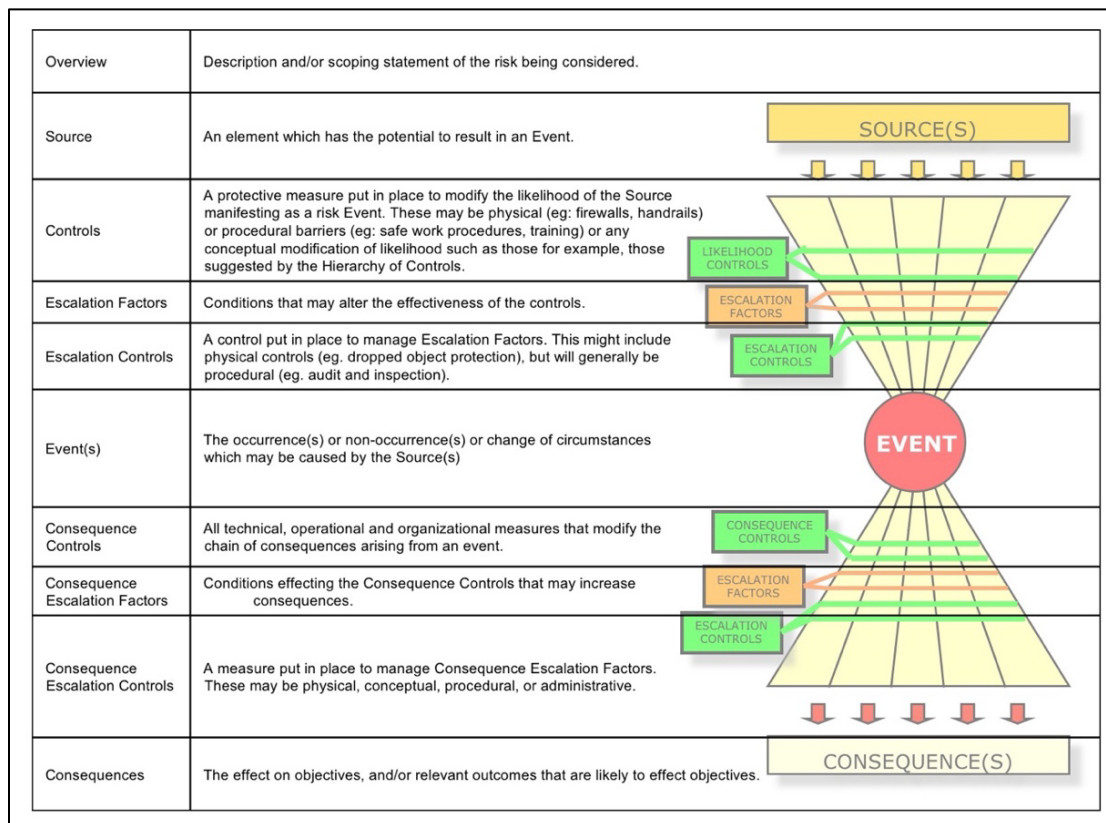


Figure 12. Bowtie Structure Overview. Source: [31].

The QBowtie methodology benefits from the Bowtie model by incorporating unique risk analysis methods during the initial identification process. In [17], Sheehan describes the model's ability to not only accommodate historical data and expert opinion, but also data from other known frameworks. Figure 13 shows the four phases for the QBowtie model.

- Phase 1—Expert opinion
- Phase 2—Bowtie analysis
- Phase 3—Quantification
- Phase 4—Classification

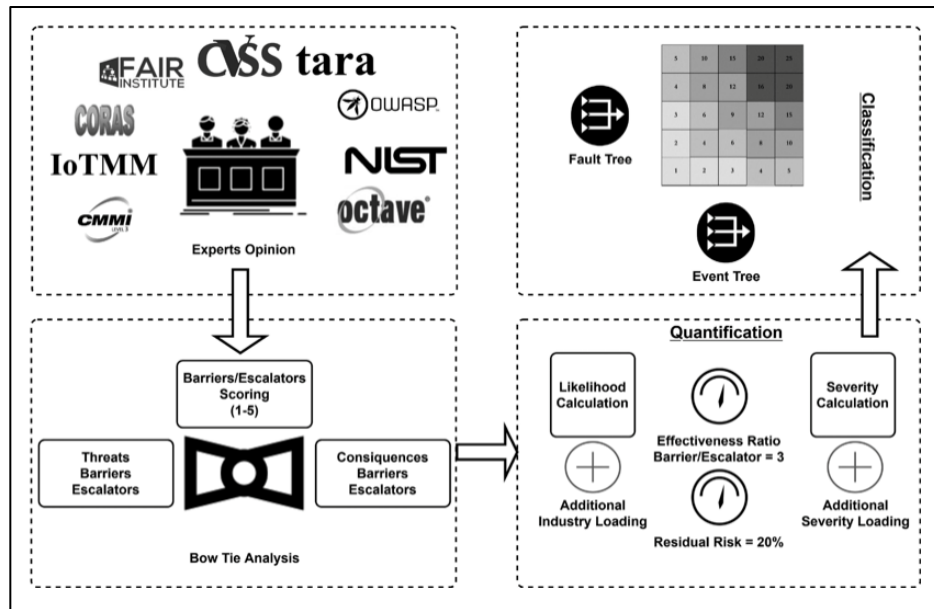


Figure 13. Outline of QBowtie Model for Risk Classification. Source: [17].

The first and second phases of the QBowtie model, Sheehan identifies the risk factors such as, threats, impacts, barriers, and escalators. The third phase scores the risk factors, while the fourth phase ranks scores of the risk factors [17]. This model provides mitigation recommendations in addition to the qualitative and quantitative feedback needed to produce a risk matrix.

E. QUANTITATIVE RISK WEIGHTING ANALYSIS

Previous research has experimented with various frameworks and algorithms that address the need for risk weighting different components of vulnerability scans. This section discusses earlier research in the area of risk weighting frameworks and algorithms.

1. Event Prioritization Framework

To address the need to prioritize and triage cyber events, the Office of Naval Research sponsored and tasked a team from the Naval Research Laboratory to develop a framework to prioritize cyber events based on their potential damage to important hosts and missions [6]. In 2014, Kim et al. developed the Event Prioritization Framework (EPF), which established prioritization of identified vulnerability based on the impact of exploits on a given host. The EPF provides a calculation of a target host's exposure based on asset criticality and the event impact on the host [33]. The final EPF value determines the priority of a host according to the effect of an exploitation. The EPF was implemented as a customizable and user-friendly tool called A Configurable Cyber Event Prioritization Tool (ACCEPT) [6].

The EPF takes in raw input from security appliances in the form of the CIA dimensions, which provides more detailed information for prioritization. This format allowed for the use of the NVD, which also describes vulnerabilities in terms of CIA dimensions. The raw data is then encapsulated into factors such as host exposure, asset criticality, host importance, network connectivity, event impact, and event effect. These factors are then combined to provide an event priority score. The diagram in Figure 14 summarizes the EPF.

The EPF is a step in the direction of addressing the need for an integrated framework to prioritize identified vulnerabilities based on their effect to the target host. The EPF results show that higher impact events will have a higher priority than other events [6]. This integration of the EFP allows cyber warriors to focus on identified and mitigating events, which results in critical and day to day missions being protected.

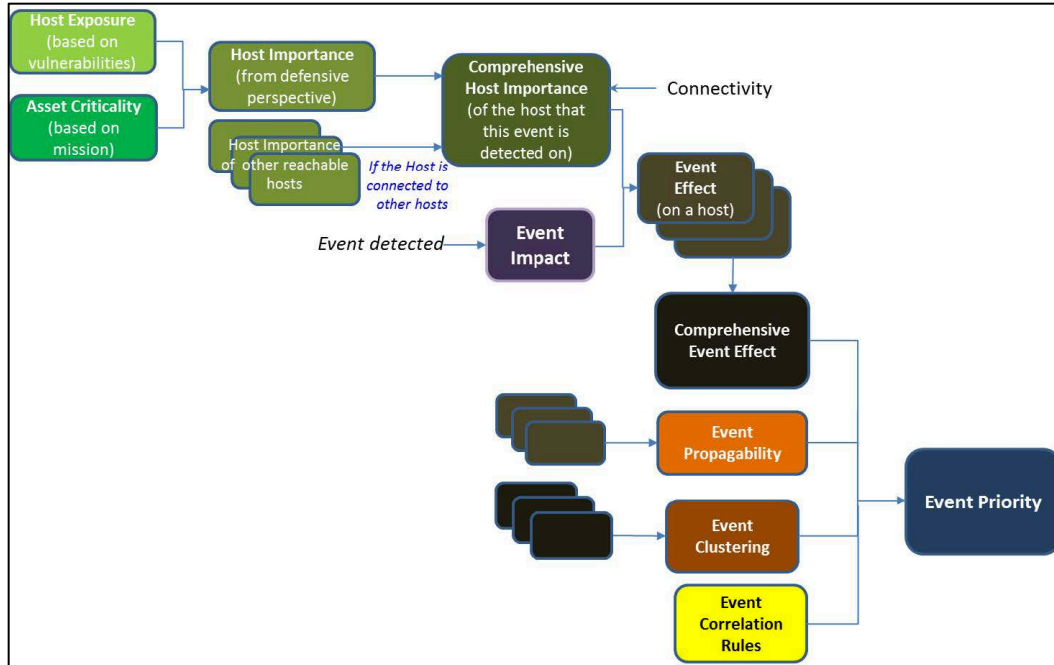


Figure 14. Diagram of the Event Prioritization Framework. Source: [6].

2. Target Importance Rank Algorithm

In 2022, the Multidisciplinary Digital Publishing Institute (MDPI) journal published an article that discussed the development of a method for assigning target importance criteria to calculate target priority [33]. The article by Kim et al. presents the target importance rank (TIR) algorithm, which calculates an importance score for a target using the PageRank algorithm (discussed in the next section), the EPF, and a host criticality evaluation. The TIR algorithm quantifies factors such as the level of connectivity, criticality, and host exposure to create a list of prioritized targets based on the quantified values [33]. Figure 15 shows a diagram of how the factors for TIR criteria are linked, and the following describes each factor.

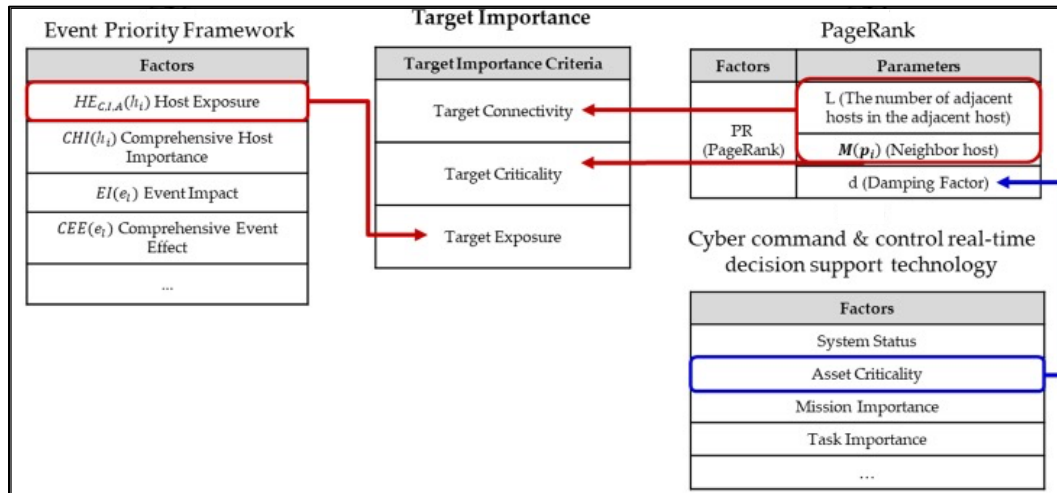


Figure 15. Linking Factors for Target Importance Rank Criteria. Source: [33].

a. Host Exposure

Host exposure (HE) is a factor from the EPF that the TIR algorithm uses to calculate the of risk for a potential exploit on a target [6]. HE is calculated using the CIA dimensions of confidentiality (HE_C), integrity (HE_I), and availability (HE_A). The base vector scores from CVSS (provided from Common Vulnerabilities and Exposures (CVE) data), serve as the input for the calculations [33]. For HE, the impact to the CIA dimensions, the access vector (AV), access complexity (AC), and the authentication (Au) requirements, are used to calculate potential exposure of a host. The HE algorithm assigned qualitative values to the CIA dimensions to represent their qualitative textual representation as seen in Table 1.

Table 1. HE Characteristic Qualitative Value. Adapted from Source: [34].

Characteristic	Qualitative Value	Quantitative Value
Confidentiality, Integrity, and Authenticity	Complete Loss (C)	1.0
	Partial Loss (P)	0.3
	None (N)	0.0

The HE parameters also come from the EPF: Initial compromise level (IC), Max Impact (MI), Max Vulnerability (MV), Access Vector Complexity Authentication (ACA), Attack Surface (ATT), and weight for k (W_k) [33]. Figure 16 provides a short description of each parameter.

Parameter	Definition
IC	Initial compromise level: Potential damage level of host h discovered by vulnerability scanner
MI	Max impact : Potential maximum impact of a vulnerability present in host h for each of the CIA elements
MV	Max vulnerability : The single largest vulnerability in the set of known vulnerabilities for host h
ACA	Access vector: Minimum access required for an attacker to compromise your system Complexity: Average level of complexity required for an attack Authentication: Minimum number of authentications required by a hacker when attacking a system using one or more vulnerabilities
ATT	Attack surface with two parameters x and y : Potential damage to the vulnerability
W_k	Weight for k

Figure 16. EP Framework Parameter Definitions. Source: [33].

The final score for HE ranges from 0 to 10, representing the severity (low to high) of the target host's potential exposure to attack. The equation used to calculate HE is

$$HE_{C,I,A}(h_i) = IC_{C,I,A}(h_i) * (10 - y) * MI_{C,I,A}(h_i)^{W_{MI}} * MV(h_i)^{W_{MV}} * ACA(h_i)^{W_{ACA}} + ATT_{C,I,A}(h_i, x, y).$$

b. PageRank Algorithm

In 1996, Larry Page and Sergey Brin developed the PageRank Algorithm at Stanford University. The algorithm was created to give a quantitative weight to specific elements of a hyperlinked document. The algorithm also and executes iterations to quantitatively determine the importance priority within the set [6]. For its role in calculating TIR, this algorithm was used to determine the number of adjacent hosts for target connectivity and identifying the number of neighboring hosts for target Criticality (Figure 15).

c. Target Connectivity

The first of three criteria for target importance is target connectivity. Target connectivity quantifies the influence that an attack on a host can potentially have on any neighboring hosts. Hosts on any given network may not operate in isolation and, therefore, should not be considered as standalone hosts. However, in [6], the researchers concluded that although hosts may not interact directly, they still can have an influence on each other.

d. Target Criticality

The second of three criteria for target importance is target criticality. This is the quantification of level of influence when attacking a single host. The risk indicators to evaluate the criticality of the exploit are determined by the CIA dimensions [6], [33]. A calculated value for target criticality can be defined as the probability of selecting a different vulnerability if the desired vulnerability is not as high importance from the user's point of view. The parameters for determining target criticality are described in Figure 17 describes the parameters needed to calculate target criticality.

Parameter	Definition
h_i	Host h_i
$M(h_i)$	Set of hosts associated with host h_i
h_j	Hosts associated with Host h_i [Neighbor Hosts] (Can be represented by an edge, connected in both directions to each other.)
$L(h_j)$	A set of close hosts of host h_j connected to host h_i .
d	Probability of moving to another network if the network is not important from the attacker's point of view

Figure 17. TIR Parameter Definitions. Source: [33].

The final target criticality score is referred to as “ d ” in the parameters (see Figure 17). Target criticality determines is the likelihood of the attacker transitioning to host on the network, calculated as shown in the following equation.

$$TC(h_i) = \frac{(1 - d)}{|M(h_i)| + 1} + d \sum_{h_i \in M(h_i)} \frac{TC(h_i)}{|L(h_i)|}$$

e. Target Exposure

The third and final criterion for target importance is target exposure. Target exposure calculates a quantitative value that represents the magnitude to which the target is exposed to the vulnerability. This score is derived from the HE values, which is then integrated into the target importance ranking algorithm [33].

F. CHAPTER SUMMARY

This chapter provided an overview of the purpose and challenges of DOD red teams, including the current state of DOD OT&E, persistent and advanced cyber operations, and the ongoing plan of operation for continued improvements to OCO tools and techniques. It also described the current CARTT architecture, covering prior research in system design and development. Finally, it discussed notable risk weighting methods and related work. In the next chapter, we discuss how combining CARTT's current vulnerability analysis scan with a risk weighted framework that considers the threat, vulnerability, and impacts can enhance a CARTT user's decision-making after performing a vulnerability scan.

III. DESIGN METHODOLOGY

This chapter discusses the need for, and the design methodology used to develop the risk weighting framework for vulnerabilities in CARTT. It also describes the current CARTT design and how the new framework components were implemented.

A. RISK WEIGHTING VULNERABILITIES IN CARTT

Once a target host's exploitable vulnerabilities have been identified, red teams need to make quick, effective decisions on which vulnerabilities to prioritize for mitigation based on the risks they present [6]. Combining CARTT's analysis of its vulnerability scan with a risk weighted framework that considers the threat, vulnerability, and exposure can enhance the user's decision-making when choosing an exploit module against a high priority vulnerable target.

B. SYSTEM DESIGN

This section discusses the system design for producing a vulnerability scan within the CARTT server. Next, we describe the process flow from vulnerability identification to selection. Finally, we will discuss the new vulnerability scan results page in CARTT, including its information, layout, and graphic design.

1. CARTT Server Design

The design of the CARTT server consists of a hypertext preprocessor (PHP) server, mySQL database, GVM (formerly OpenVAS version 9) database, MSF database, and a reports folder, as shown in Figure 18. The user interacts with the CARTT GUI to perform tasks while the components of the CARTT server are working behind the scenes. The core of the CARTT server is the PHP server, which is comprised of several scripts that allow the user to input information for CARTT actions and to receive applicable feedback [15]. The PHP 7.4 scripts also allow the user to communicate with the GVM and MSF (version 6) databases without having to use the command line directly.

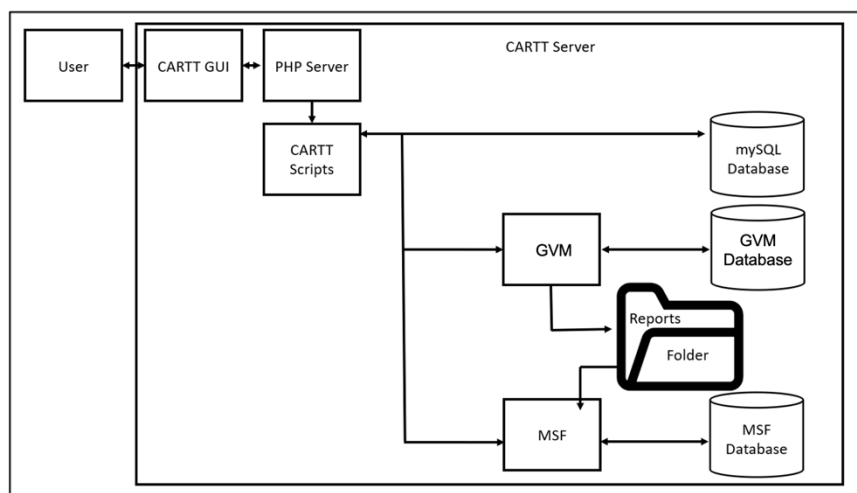


Figure 18. CARTT Server Diagram. Adapted from Source: [15].

Once the GVM scan is complete, the results are downloaded and stored in the reports folder on the CARTT server. The reports are downloaded in extensible markup language (XML) format, which is an acceptable format for importing into the MSF database for parsing. MSF relies on a PostgreSQL database to store imported scan data found in the GVM scan report [5].

A series of PHP scripts help automate functions, from connecting to the MSF database to displaying the results of the vulnerability scans. The `openvas_import_report.php` script imports the vulnerability scan report from GVM into the MSF database for parsing. The GVM scan is imported based on given information such as report identification, import format, and a unique report name, which is specified by the user at the creation of the scan. The script allows for commands to be sent to the MSF database through piping, eliminating the need for manual CLI user input. The GVM import information and the command to import (`db_import`) are piped into MSF where a series of checks occur that will notify the operator if the import was successful or not.

The `msf_vulns` function (see Figure 19) uses the spooling capabilities through MSF to write the imported vulnerabilities and CVE data to a file named `vulns_$user.txt`. The variable `$user` represents the role CARTT is being operated in by the user (Admin, Operator, or Commander), and this text file has read and write permissions which means it

creates a new file if it does not exist or is deleted. The file is overwritten with the latest MSF vulnerability scan results.

```
function msf_vulns($pipes)
{
    //grab email from session and create user var
    //delete previous vulns_$user file
    $user = $_SESSION['email'];
    $vuln_file = "vulns_$user.txt";
    $fd = fopen("user_data/$vuln_file", 'w+');
    fclose($fd);

    //spool output into vulns_$user file
    fwrite($pipes[0], "spool user_data/$vuln_file\n");

    $vulns = "vulns\n";
    fwrite($pipes[0], $vulns);

    //stop spooling
    fwrite($pipes[0], "spool off\n");
}
```

Figure 19. MSF Vulnerability Function. Source: [15].

The `cartt_exploit_config.php` script (see Figure 20) opens and reads in the `vuln_$user.txt` file as variable `$vuln_fd`. The script loops through the file, parsing out any additional whitespace or special characters added by MSF spooling the vulnerability results into the file.

```
if($host)
{
    //grab user info to create handle for vuln file
    $vuln_file = "user_data/vulns_$user.txt";

    //open file or set to error message
    $vuln_fd = fopen($vuln_file, "r") or die("Unable to open $vuln_file");

    $vuln_list = "";
    $count = 0;

    //while not end of file loop and add to list
    while (!feof($vuln_fd))
    {
        $vuln_line = fgets($vuln_fd);
        $pos = strpos($vuln_line, "$host");

        if($pos == true)
        {
            $count++;

            $vuln_line = substr($vuln_line, 38);

            $vuln_list .= $vuln_line."\n";
        }
    }
}
```

Figure 20. CARTT Script for the Vulnerability Results Function.
Source: [15].

The results are presented as output on the CARTT Vulnerability Results Page, which counts the number of vulnerabilities and displays the CVE information for each vulnerability, as shown in Figure 21.

4 vulnerabilities identified for host 192.162.83.17

ICMP Timestamp Detection CVE-1999-0524

Microsoft Windows SMB Server Multiple Vulnerabilities-Remote (4013389) CVE-2017-0143,CVE-2017-0144,CVE-2017-0145,CVE-2017-0146,CVE-2017-0147,CVE-2017-0148,BID-96703,BID-96704,BID-96705,BID-96707,BID-96709,BID-96706

Microsoft Windows SMB Server NTLM Multiple Vulnerabilities (971468) CVE-2010-0020,CVE-2010-0021,CVE-2010-0022,CVE-2010-0231

Vulnerabilities in SMB Could Allow Remote Code Execution (958687) - Remote CVE-2008-4114,CVE-2008-4834,CVE-2008-4835,BID-31179

Please enter CVE of a vulnerability

CVE-xxxx-xxxx

Submit

Note: this search will take several seconds to complete

Figure 21. Current CARTT Vulnerability Results Page.

2. Process Flow for Vulnerability Identification

CARTT assists the Operator in determining which exploit module to use by providing descriptions of the selected module. The CARTT process flow did not previously provide a risk analysis of identified vulnerabilities prior to presenting the Operator with potential exploit modules to employ. This stage of the analysis now implements the risk weighting framework, which is beneficial to the Operator as it will provide extended analysis for exploitable vulnerabilities that pose the most risk. Figure 22 depicts the new process flow, showing where the risk weighting framework has been added.

The framework consists of a series of PHP scripts and Python code which will be located in the CARTT server directory so that it can be easily accessed to run cohesively with the other server components. A series of Python functions analyze the same GVM XML report to parse out specific CVSS Base Vector data which was not provided in the previous MSF vulnerability results. Once the CVSS Base Vector data is extracted and formatted, it will be used within other functions to calculate the results for the components of the framework. These Python functions are imported and executed within the PHP script designated to provide the vulnerability results. This will replace the previous process which

only identified the vulnerabilities. The Operator now has access to additional information as well as a ranking regarding the identified vulnerabilities which assist in determining a possible exploit.

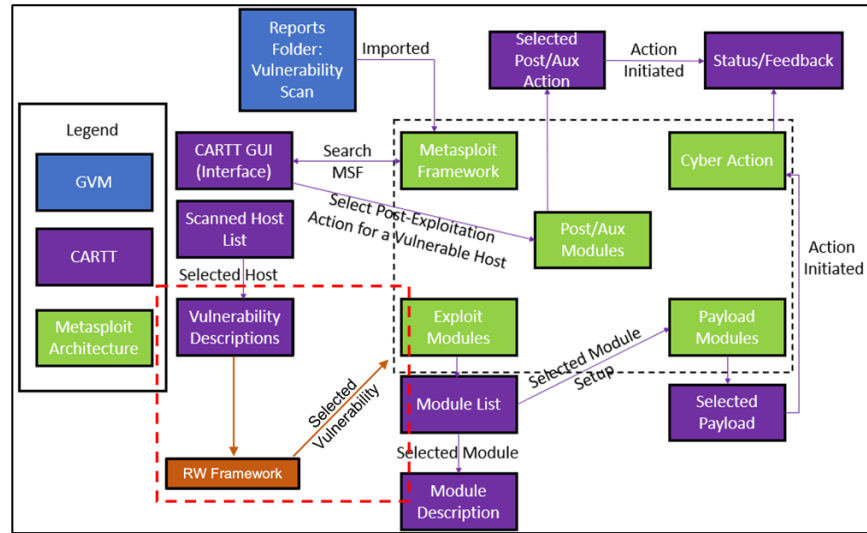


Figure 22. CARTT Process Flow Diagram with Risk Weighting Framework.
Adapted from [15].

3. Vulnerability Results Design

In the current CARTT process flow, after the Operator requests to import the desired report, they are directed to the CARTT Initial Access Exploit Configuration Page (see Figure 23). Once the Operator verifies and submits the scan name on this page, it refreshes and shows a list of identified vulnerabilities imported from the GVM XML report. As discussed in Section B.1, the vulnerability results display for the previous CARTT edition was basic, only providing the Operator with host IP, vulnerability name, and corresponding CVE or Bugtraq ID (BID) for each identified vulnerability. The new CARTT vulnerability results display now provides additional information specific to each vulnerability to include scores for CVSS individual vectors, target exposure components, a host exposure ranking, along with upgraded formatting and design.

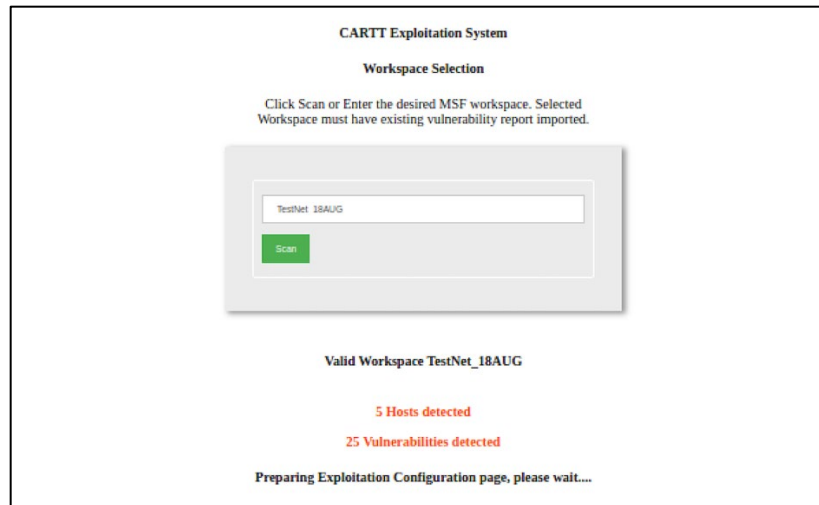


Figure 23. CARTT Exploitation Configuration Page.

C. RISK WEIGHTING FRAMEWORK

This section describes the components of the risk weighting framework implemented into CARTT's analysis of the vulnerability scan. These components consist of the CVSS Base Vector score and HE algorithm.

1. CVSS Base Vector Score

The vulnerabilities identified by a GVM scan are populated from a repository in the NVD. These vulnerabilities are referred to as common vulnerabilities and exposures CVE, which are vulnerability identifiers, whereas CVSS Base scores are qualitative values used to rate the severity of a vulnerability [1]. The XML report downloaded from the GVM vulnerability scan contains both the CVSS Base score and the CVSS Base Vector score for each vulnerability. Figure 24 shows an example XML report, which contains the details of a vulnerability, with each data item enclosed in metatags. The CVSS Base score is shown between the <cvss_base> tag and CVSS Base Vector score is located immediately after the <tags> tag. The positions of these tags are repetitive and identical for each vulnerability, which makes them easy to parse.

```

192.168.83.8</description></result><result id='5ac174cf-c6a7-4aaa-8ce1-8f4b4dafd739'>
<name>Vulnerabilities in SMB Could Allow Remote Code Execution (958687) - Remote</name>
<owner><name>admin</name></owner><comment/><creation_time>2022-02-
03T00:01:07Z</creation_time><modification_time>2022-02-03T00:01:07Z</modification_time>
<user_tags><count>0</count></user_tags><host>192.168.83.8<asset asset_id='b71e9d89-dd71-41cd-
8b94-b68055bf8f0d'></host><port>445/tcp</port><nvt oid='1.3.6.1.4.1.25623.1.0.900233'>
<type>nvt</type><name>Vulnerabilities in SMB Could Allow Remote Code Execution (958687) -
Remote</name><family>Windows : Microsoft Bulletins</family><cvss_base>10.0</cvss_base>
<cve>CVE-2008-4114, CVE-2008-4834, CVE-2008-4835</cve><bid>31179</bid>
<xref>URL:http://www.milw0rm.com/exploits/6463, URL:https://docs.microsoft.com/en-
us/security-updates/securitybulletins/2009/ms09-001</xref>
<tags>cvss_base_vector=AV:N/AC:L/Au:N/C:C/I:C/A:C impact=Successful exploitation could allow
remote unauthenticated attackers

```

Figure 24. Example XML Report with CVSS Base and Base Vector Scores.

As discussed in Section II.D.1., the CVSS Base score ranges from 1 to 10, based on vulnerability severity. That score is calculated from the base vector string, which provides a textual representation of the metric values used for each vulnerability identified. The base vector scores are broken out into six characteristics: AV, AC, Au, C, I, and A. Although the characteristics are reported in textual representation, each metric value has an equivalent quantitative value. These values are used to compute the overall CVSS Base score. Table 2 shows each characteristic metric value and quantitative value for scoring the CVSS Base Vectors.

Table 2. CVSS Base Vector Metric and Quantitative Values.
Adapted from [34].

Characteristic	Metric Value	Quantitative Value
Access Vector	Local (L)	0.395
	Adjacent Network (A)	0.646
	Network (N)	1.0
Access Complexity	High (H)	0.35
	Medium (M)	0.61
	Low (Low)	0.71
Authentication	Multiple (M)	0.45
	Single (S)	0.56
	None (N)	0.704
Confidentiality,	None (N)	0.0

Integrity, and Availability	Partial (P)	0.275
	Complete (C)	0.660

GVM uses CVSS version 2.0 guidelines for assigning the quantitative values for each of the characteristics. The quantitative values of the base vector scores were used as a visual representation in CARTT of the scoring for each characteristic and to calculate the host exposure and target importance rank on the new CARTT Vulnerability Report Page.

2. Host Exposure

As discussed in Section II.E.2.a, HE calculates a numerical measure of risk for a potential exploit on the target host. For CARTT, HE was calculated using the same CIA dimension obtained from the GVM XML report. As with the CVSS Base Vector scores, the CIA dimensions were translated from their qualitative textual representation to a quantitative value. The proposed quantitative values for the CIA characteristics are based on the EPF equations discussed in Section II.E.2.a. The HE algorithm (referenced below) was added to the `cartt_rwf.py` script to provide the foundation for the risk weighting framework for CARTT.

$$HE_{C,I,A}(h_i) = IC_{C,I,A}(h_i) * (10 - y) * MI_{C,I,A}(h_i)^{W_{MI}} * MV(h_i)^{W_{MV}} * ACA(h_i)^{W_{ACA}} + ATT_{C,I,A}(h_i, x, y)$$

D. CHAPTER SUMMARY

This chapter described the risk weighting framework and its components that were incorporated into the CARTT server. In the next chapter, we discuss the implementation of these components to incorporate risk weighting of vulnerabilities into the CARTT analysis.

The next chapter discusses the environment setup, framework scripting, and scenario testing for this thesis.

IV. RISK FRAMEWORK IMPLEMENTATION

This chapter discusses the risk framework implementation environment, additions made to CARTT functions and scripts, a scenario to test the risk weighting framework as discussed in Chapter III, and results of the implementation.

A. ENVIRONMENT SETUP

The CARTT server and target hosts used in the implementation environment were virtualized using VMware Fusion. The environment consisted of one network with six virtually machines (VMs), seen in Figure 25. The CARTT server was housed on an Ubuntu 20.04 VM, and the remaining five VMs were designed to be target hosts running Windows XP, Windows 7, Windows 10, Ubuntu Linux 22.04 LTS, and Windows Server 2016 operating systems. These operating systems were selected as target hosts due to their continued DOD use.

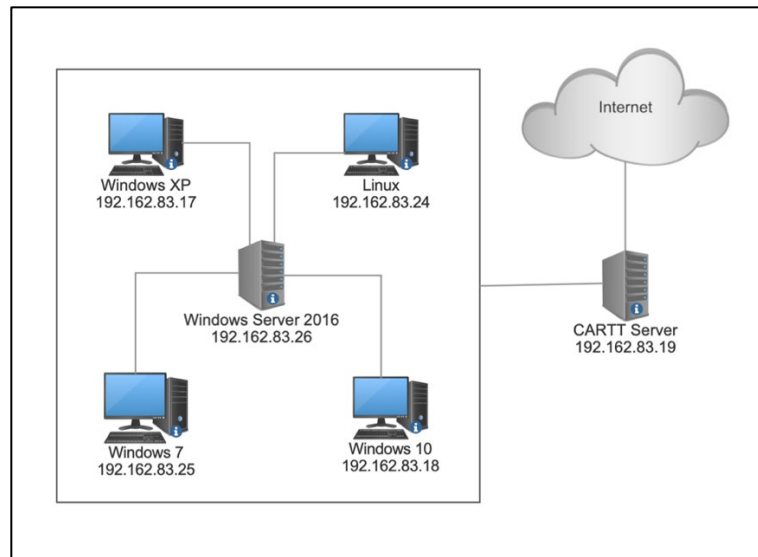


Figure 25. Implementation Environment.

Due to the small size of the environment, all six VMs were configured to use a custom network connection set to a /27 classless interdomain routing (CIDR) on the

192.162.83.0 IP network. An external internet connection was required for operations of GVM and MSF on the CARTT server.

B. FRAMEWORK SCRIPTING

This section describes the scripting methods used to implement the risk weighting framework into CARTT. This framework implementation included the addition of a new Python script, changes to existing PHP and cascading style sheet (CSS) scripts, and GUI redesign of the vulnerability results page.

1. Python Functions

A Python script, `cartt_rwf.py`, was created to handle all the functionality required for the risk weighting framework. This script contains the functions responsible for extracting, parsing, and formatting additional data from the raw GVM XML scan report. The script also contains functions necessary to implement the risk weighting framework within CARTT. The functions used for the framework were based off the Host Exposure Equation discussed in Section II.E.2.a. Python was chosen as the coding language for the risk weighting framework because of its vast library support, algorithm scripting capabilities, and seamless integration with PHP.

a. GVM Report

The MSF vulnerability scan output only identifies the timestamp for the creation of the report, host IP address, vulnerability name, and the CVE/BID references for each vulnerability discovered (Figure 26).

Timestamp	Host	Name	References
2022-08-19 05:09:21 UTC	192.162.83.17	ICMP Timestamp Detection	CVE-1999-0524
2022-08-19 05:09:21 UTC	192.162.83.18	ICMP Timestamp Detection	CVE-1999-0524
2022-08-19 05:09:21 UTC	192.162.83.24	ICMP Timestamp Detection	CVE-1999-0524
2022-08-19 05:09:21 UTC	192.162.83.25	ICMP Timestamp Detection	CVE-1999-0524
2022-08-19 05:09:21 UTC	192.162.83.26	ICMP Timestamp Detection	CVE-1999-0524
2022-08-19 05:09:21 UTC	192.162.83.17	Microsoft Windows SMB Server NTLM Multiple Vulnerabilities (971468)	CVE-2010-0020
2022-08-19 05:09:21 UTC	192.162.83.26	SSL/TLS: Report Vulnerable Cipher Suites for HTTPS	CVE-2016-2183
2022-08-19 05:09:21 UTC	192.162.83.26	SSL/TLS: Report Weak Cipher Suites	CVE-2013-2566
2022-08-19 05:09:21 UTC	192.162.83.26	SSL/TLS: Report Weak Cipher Suites	CVE-2013-2566
2022-08-19 05:09:21 UTC	192.162.83.17	Vulnerabilities in SMB Could Allow Remote Code Execution (958687) - Remote	CVE-2008-4114

Figure 26. MSF Vulnerability Results.

Although this information was helpful to use within CARTT, it did not contain the data necessary to provide functionality for risk weighting. The GVM XML scan report contains a trove of additional data that can be used for deeper vulnerability analysis, and which was used in the risk weighting framework for the CVSS Base Vector score. Therefore, in addition to the MSF vulnerability output, the CVSS Base Vector scores were manually exported for use by the risk weighting framework. Figure 27 shows an excerpt of a GVM XML scan report, where the CVSS Base Vector score is highlighted immediately after the hypertext markup language (HTML) tag named *tags*.

```
</description></result><result id='026e41c5-b22f-496d-b92e-4c195962323e'><name>SSL/TLS: Microsoft Remote Desktop Protocol
STARTTLS Detection</name><owner><name>admin</name></owner><comment/><creation_time>2022-08-19T04:37:51Z</
creation_time><modification_time>2022-08-19T04:37:51Z</modification_time><user_tags><count>0</count></user_tags><host>192.162.
83.26<asset asset_id='0315ef6e-7e89-49e6-9cd8-80604e0ddf28'></host><port>3389/tcp</port><nvt oid='1.3.6.1.4.1.25623.1.0.
140152'><type>nvt</type><name>SSL/TLS: Microsoft Remote Desktop Protocol STARTTLS Detection</name><family>SSL and TLS</
family><cvss_base>0.0</cvss_base><cve>NOCVE</cve><bid>NOBID</bid><xref>URL:https://msdn.microsoft.com/de-de/library/cc240500.
aspx</xref><tags>cvss_base_vector=AV:N/AC:L/Au:N/C:N/I:N/A:N summary=Checks if the remote Microsoft Remote Desktop Protocol
(RDP) service supports the &apos;PROTOCOL_SSL&apos; flag. | qod_type=remote_vul</tags><cert/></nvt><scan_nvt_version>$Revision:
11898 $</scan_nvt_version><threat>Log</threat><severity>0.0</severity><qod><value>99</value><type>remote_vul</type></
```

Figure 27. Textual Representation of CVSS Base Vector Score in GVM XML Report.

In GVM XML scan reports from various vulnerability scans, the HTML tags for each vulnerability are titled and ordered in an identical manner. The GVM XML raw data is formatted as a string, so objects are easily searchable through the HTML tag names. This recurring format structure for data organization, specifically the CVSS Base Vector scores, made it easy to parse and extract the desired data needed for the risk framework functions. We used string manipulation was used to identify, extract, and split the CVSS Base Vector score into its six characteristics and respective textual scores: Access Vector (AV), Access Complexity (AC), Authentication (Au), Confidentiality (C), Integrity (I), and Availability (A). Once split, the six characteristics scores were mapped from their textual metric value to their corresponding quantitative values based on Table 2 in Section III.C.1. Figure 28 shows the mapping lists within the function used for replacing the textual metric values with the quantitative values.

AV	=	{	'L': 0.395,	'A': 0.646,	'N': 1.0}
AC	=	{	'H': 0.35,	'M': 0.61,	'L': 0.71}
Au	=	{	'M': 0.45,	'S': 0.56,	'N': 0.704}
C	=	{	'N': 0.0,	'P': 0.275,	'C': 0.66}
I	=	{	'N': 0.0,	'P': 0.275,	'C': 0.66}
A	=	{	'N': 0.0,	'P': 0.275,	'C': 0.66}

Figure 28. Mapping Lists for CVSS Base Vector Quantitative Values.

With the CVSS Base Vector score now represented by the quantitative values, the risk weighting framework functions were calculated.

b. Risk Weighting Functions

For the purpose of this thesis, the Host Exposure equation will now be rereferred to as the Target Exposure. The TE equation is used to calculate the weighted risk for each host based on identified vulnerabilities. The TE equation is comprised of five variables described below: IC, MI, MV, ACA, and ATT. Each variable calculation was implemented in Python as separate functions in the `cartt_rwf.py` script. The TE equation calculation was also implemented in Python as a separate function, which called the five variable functions. The output of the TE equation function is the weighted risk for each host.

The six equations used to calculate the risk weighted framework are as follows:

- (1) The IC function sets the initial compromise level for each host. The calculation is based on the sum of the CIA quantitative values for each vulnerability identified for that host. For the purposes of this thesis, a complete compromise is equal to 1, no compromise is equal to 0, and a partial compromise is any value between 0 and 1. The pseudocode for IC function is

$$IC = \min(1, (\text{sum}(C, I, A))).$$

- (2) The MI function sets the upper bound, or maximum impact, on the number of vulnerabilities required to completely compromise a host. The MI for a host is the maximum value for the sum of CIA values for each identified vulnerability. The pseudocode for the MI function is

$$MI = \max(\text{sum}(C, I, A)).$$

- (3) The MV function sets the lower bound on the number of vulnerabilities needed to completely compromise a host, across the CIA dimensions. It also tells how much damage one vulnerability can cause on the host when exploited. The MV function was normalized to 1. The pseudocode for the MV function is

$$MV = \max(\sqrt{(\max_vul_c)^2 + (\max_vul_i)^2 + (\max_vul_a)^2}).$$

- (4) The ACA function is the only function that is calculated using the remaining CVSS Base Vector score: AV, AC, and Au. The ACA value was calculated by computing the square root of the sum when combining the maximum AV needed to compromise the host, the average AC required to attack the host, and the maximum number of Au needed when using one or more of the vulnerabilities identified. The pseudocode for the ACA function is

$$ACA = \sqrt{(\max_av)^2 + (avg_ac)^2 + (\max_au)^2}.$$

- (5) The ATT function calculates the number of vulnerabilities that are identified for each host. The pseudo code for the ATT function is

$$ATT = \max(\log_2(avg(sum(C, I, A))), y).$$

- (6) The TE function takes in the results from the IC, MI, MV, ACA, and ATT functions. The TE function output a risk weighting, that identified the host's potential exposure for attack based on vulnerability analysis. The TE value ranges on a scale of 0 to 10, where 10 is the highest severity for exposure. To ensure the final value remained within the desired scoring range, the TE function was normalized to 10. Weighted values for MI (1), MV (0.5), and ACA (1) functions were also implemented as an additional check to ensure normalization was maintained. The pseudocode for the TE function is

$$TE = IC * (10 - y) * MI^{W_{MI}} * MV^{W_{MV}} * ACA^{W_{ACA}} + ATT.$$

The final step for implementing the risk weighted framework was sorting the TE values in descending order, with the highest-ranking host listed first. Figure 29 shows the results of the IC, MI, MV, ACA, ATT, and TE functions for the Windows Server 2016 target host (192.162.83.26). These results are presented in a visual data frame representation for one of seven vulnerabilities identified for that host.

Host IP	Vulnerability Name	CVE	BID	AV	AC	Au	C	I	A	IC	MI	MV	ACA	ATT	Target Exposure
192.162.83.26	SSL/TLS: Report Vulnerable Cipher Suites for H...	CVE-2016-2183, CVE-2016-6329	NOBID	1.000	0.71	0.704	0.275	0.00	0.00	1.0	0.66	1.14	1.38	7	9.92

Figure 29. Results Data for the Windows Server 2016 Target Host (192.162.83.26).

2. Hypertext Preprocessor Scripting

The `cartt_workspace.php` and `cartt_exploit_config.php` scripts are the two most important scripts used by the PHP server because they retrieve, and display identified vulnerabilities to the CARTT Operator. These scripts provided validation checks that confirm proper user access and workspace session. These scripts work together to ensure that before a vulnerability is displayed, the vulnerability report is accurate based on the MSF vulnerability list as discussed in Section III.B.1.

The new `cartt_rwf.py` script is now called by the `cartt_exploit_config.php`, replacing the previous function, which only displayed the identified vulnerabilities from the MSF spooled list. After the validity checks, the `cartt_rwf.py` script is called and executed through the PHP commands as seen in Figure 30.

```
<?php
$command = escapeshellcmd('python3 /home/daz/CARTT/cartt_rwf.py');
$output = shell_exec($command);
echo $output;

?>
```

Figure 30. PHP Command to Execute the Risk Weighting Python Script.

The vulnerability report with the risk weighting framework is now displayed for the Operator using the command “`echo $output.`”

3. Responsive Vulnerability Results Table

Using the additional vulnerability data now available in CARTT, a Responsive Table was implemented for the Operator. This table was designed with ability to

accommodate different screen sizes and provide an aesthetic appearance. Like the other displayed objects within the CARTT GUI, the Responsive Table has a PHP scripting portion, but the styling attributes come from CSS generated content.

The PHP scripting for the Responsive Table was added to the `cartt_rwf.py` script just after the command which calls for the execution of the risk weighting Python functions. Figure 31 shows the semantics for the custom table headers, column groups, and column spacing for the vulnerable output. The body of the table is populated with the output from the risk weighted framework.

```
<div class="table-wrapper">
  <table class="fl-table">
    <thead>
      <tr>
        <th>Host IP</th>
        <th>Vulnerability Name</th>
        <th>CVE</th>
        <th>BID</th>
        <th colspan="6">CVSS Vector Scores</th>
        <th>IC</th>
        <th>MI</th>
        <th>MV</th>
        <th>ACA</th>
        <th>ATT</th>
        <th>Target Exposure<br>(Results)</th>
        <colgroup><col span="1"></col></colgroup>
        <th colspan="4"></th>
        <th scope="col">AV</th>
        <th scope="col">AC</th>
        <th scope="col">Au</th>
        <th scope="col">C</th>
        <th scope="col">I</th>
        <th scope="col">A</th>
        <th colspan="4"></th>
      </tr>
    </thead>
    <tbody>
      <?php echo $output ?>
    </tbody>
  </table>
</div>
```

Figure 31. PHP Script for Responsive Table Display.

The styling attributes for the Responsive Table were created as a separate CSS file called `cartt_rwf.css`. This stylesheet is used in conjunction with the `cartt_exploit_config.css` stylesheet as sources for the entire CARTT Exploitation Configuration Page.

C. ENVIRONMENT SCENARIO TESTING

The primary purpose for the scenario was to perform the preliminary steps required to conduct a vulnerability analysis scan on a given network. The secondary purpose was to

show and discuss the results of combining a risk weighting framework within CARTT's workflow of conducting a vulnerability analysis scan. For our implementation test scenario, we used CARTT to scan the five VM targets on the test network, from the Operator user role.

The test scenario started after the Operator successfully logged in and was directed to the Operators Main Menu page. For the scenario, the Operator selected the *Create New Scan* action button as highlighted in Figure 32.

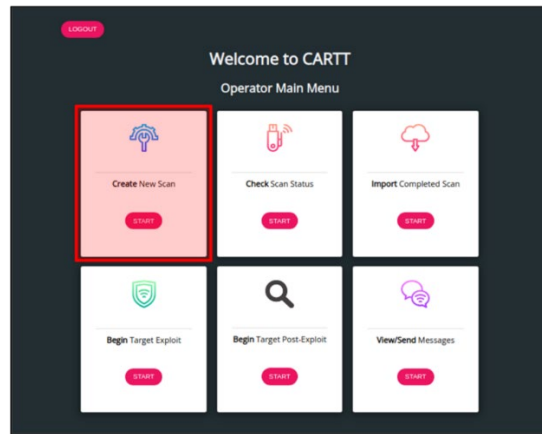


Figure 32. Operator Main Menu.

The Operator was then directed to the *Scan Network* page where they enter the target IP address, subnet, and scan name, as shown in Figure 33.

The image shows a web application interface. On the left is a dark sidebar with a 'Logout' button at the bottom and a list of menu items: 'Create New', 'Check Status', 'Import Completed', 'Begin Exploit', and 'View/Send Message'. The main content area is titled 'Scan Network'. It contains a text instruction: 'Enter an IP Address and Subnet for the target network as well as a unique name for the scan.' Below this instruction are three input fields: 'IP Address' with the value '192.168.83.0', 'Subnet' with the value '/27', and 'Scan Name' with the value 'TestNet_18AUG'. A green 'Scan' button is located at the bottom right of the form.

Figure 33. Network Scan Information Input Page.

To ensure the entire target network was scanned, the *192.162.83.0* network address was entered in the *IP Address* input field, as well as a unique scan name, which is attached to this specific scan throughout the duration of the CARTT workflow. For the scenario, the scan name for the target network was *TestNet_18AUG*. Once the Operator has entered the scan information, they can select the *Scan* action button.

With the vulnerability scan completed, the Operator was directed back to the Operator Main Menu (see Figure 32). From there the Operator can verify if the scan was successful by selecting the *Check Scan Status* action button. On the resulting *Check Scan Status* page (see Figure 34), the Operator can verify that the scan has completed successfully by verifying that the unique scan is listed, and its status shows “Done.”

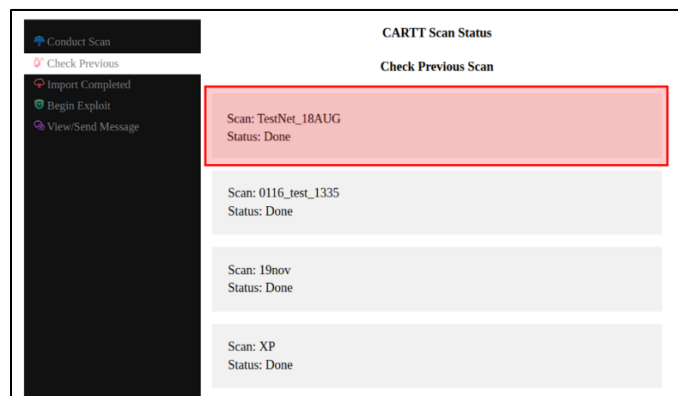


Figure 34. The Check Previous Scan Page Displaying the Operators Completed Scan Status.

For the next step in the scenario, the Operator must import the GVM report into CARTT. When the GVM scan is complete, an XML report is saved to the CARTT reports folder with the unique scan name the Operator assigned. The *Import Completed Scan* action button on the Operator Main Menu was selected, which directs the Operator to the *Report Completed List* page (see Figure 35). Here, the Operator again verifies that the unique scan name matches the information initially provided and selects the *Import* action button. This action initiates the process of the MSF database parsing the *TestNet_18AUG* XML file for any identified vulnerabilities.



Figure 35. Vulnerability Scan Import Page.

When the import has completed, the page reflects the number of hosts and vulnerabilities that were identified by the scan. Figure 36 shows that there were five hosts and 15 vulnerabilities detected in the test scenario. The page is set on a short sleep timer, allowing the Operator to view the information before being redirected to the Exploitation Configuration page.

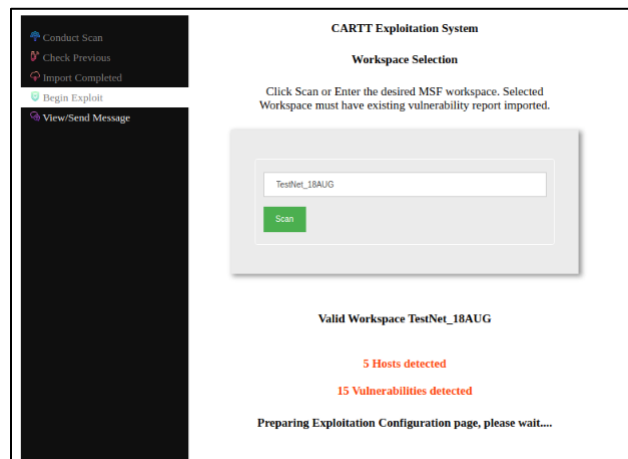


Figure 36. Workspace Verification with Detected Host and Vulnerabilities Display.

The Exploitation Configuration page displays the new risk weighted vulnerability results table as seen in Figure 37.

HOST IP	VULNERABILITY NAME	CVE	BID	CVSS VECTOR SCORES						IC	MI	MV	ACA	ATT	HOST EXPOSURE (RESULTS)
				AV	AC	Au	C	I	A						
192.162.83.17	ICMP Timestamp Detection	CVE-1999-0524	NOBID	0.395	0.71	0.704	0.000	0.00	0.00	1.0	0.66	1.14	1.40	4	9.93
	Microsoft Windows SMB Server Multiple Vulnerabilities-Remote (4013389)	CVE-2017-0143, CVE-2017-0144, CVE-2017-0145, CVE-2017-0146, CVE-2017-0147, CVE-2017-0148	96703, 96704, 96705, 96707, 96709, 96706	1.000	0.61	0.704	0.660	0.66	0.66						
	Microsoft Windows SMB Server NTLM Multiple Vulnerabilities (971468)	CVE-2010-0020, CVE-2010-0021, CVE-2010-0022, CVE-2010-0231	NOBID	1.000	0.71	0.704	0.660	0.66	0.66						
	Vulnerabilities in SMB Could Allow Remote Code Execution (958687) - Remote	CVE-2008-4114, CVE-2008-4834, CVE-2008-4835	31179	1.000	0.71	0.704	0.660	0.66	0.66						
192.162.83.26	ICMP Timestamp Detection	CVE-1999-0524	NOBID	0.395	0.71	0.704	0.000	0.00	0.00	1.0	0.66	1.14	1.38	7	9.92
	Microsoft Windows SMB Server Multiple Vulnerabilities-Remote (4013389)	CVE-2017-0143, CVE-2017-0144, CVE-2017-0145, CVE-2017-0146, CVE-2017-0147, CVE-2017-0148	96703, 96704, 96705, 96707, 96709, 96706	1.000	0.61	0.704	0.660	0.66	0.66						
	SSL/TLS: Report Vulnerable Cipher Suites for HTTPS	CVE-2016-2183, CVE-2016-6329	NOBID	1.000	0.71	0.704	0.275	0.00	0.00						
	SSL/TLS: Report Weak Cipher Suites	CVE-2013-2566, CVE-2015-2808, CVE-2015-4000	NOBID	1.000	0.61	0.704	0.275	0.00	0.00						
	SSL/TLS: Report Weak Cipher Suites	CVE-2013-2566, CVE-2015-2808, CVE-2015-4000	NOBID	1.000	0.61	0.704	0.275	0.00	0.00						
	SSL/TLS: Report Weak Cipher Suites	CVE-2013-2566, CVE-2015-2808, CVE-2015-4000	NOBID	1.000	0.61	0.704	0.275	0.00	0.00						
	SSL/TLS: Report Weak Cipher Suites	CVE-2013-2566, CVE-2015-2808, CVE-2015-4000	NOBID	1.000	0.61	0.704	0.275	0.00	0.00						
192.162.83.25	ICMP Timestamp Detection	CVE-1999-0524	NOBID	0.395	0.71	0.704	0.000	0.00	0.00	1.0	0.66	1.14	1.39	2	9.85
	Microsoft Windows SMB Server Multiple Vulnerabilities-Remote (4013389)	CVE-2017-0143, CVE-2017-0144, CVE-2017-0145, CVE-2017-0146, CVE-2017-0147, CVE-2017-0148	96703, 96704, 96705, 96707, 96709, 96706	1.000	0.61	0.704	0.660	0.66	0.66						
192.162.83.24	ICMP Timestamp Detection	CVE-1999-0524	NOBID	0.395	0.71	0.704	0.000	0.00	0.00	0.0	0.00	1.00	1.08	1	1.00
192.162.83.18	ICMP Timestamp Detection	CVE-1999-0524	NOBID	0.395	0.71	0.704	0.000	0.00	0.00	0.0	0.00	1.00	1.08	1	1.00

Please enter CVE of a vulnerability

Note: this search will take several seconds to complete

Figure 37. New Results Page with Target Exposure Prioritization Based on CARTT's Risk Weighting Framework.

D. RESULTS

For our test implementation, we conducted a vulnerability scan on the *192.162.83.0/27* network. CARTT successfully discovered all five target hosts, on this test network and detected 15 vulnerabilities among them. This was verified by the display on the CARTT Exploitation System page (see Figure 36) and in the *vulns_qwerty.txt* file spooled from the MSF database. Table 3 shows the breakdown of vulnerabilities identified for each target host.

Table 3. Target Hosts Vulnerability Results.

Host IP	Operating System	Vulnerabilities Identified
192.162.83.17	Windows XP	4
192.162.83.18	Windows 10	1
192.162.83.25	Windows 7	2
192.162.83.26	Windows Server 2016	7
192.162.83.24	Linux-Ubuntu 22.04	1

From the new vulnerability results table (see Figure 37) we see not only the vulnerabilities and CVE/BID information, but also the results for the six functions from the *cartt_rwf.py* script, which make up CARTT's risk weighting framework. Figure 38 shows the CVSS Base Vector scores in their quantitative form for each vulnerability, the collective IC, MI, MV, ACA, and ATT scores, and finally, the risk weighted HE rankings for all the hosts. This new results table provides the Operator with more complete and actionable information from a CARTT vulnerability analysis.

CVSS VECTOR SCORES						IC	MI	MV	ACA	ATT	HOST EXPOSURE (RESULTS)	
AV	AC	Au	C	I	A							
0.395	0.71	0.704	0.000	0.00	0.00	1.0	0.66	1.14	1.40	4		9.93
1.000	0.61	0.704	0.660	0.66	0.66							
1.000	0.71	0.704	0.660	0.66	0.66							
1.000	0.71	0.704	0.660	0.66	0.66							
0.395	0.71	0.704	0.000	0.00	0.00	1.0	0.66	1.14	1.38	7		9.92
1.000	0.61	0.704	0.660	0.66	0.66							
1.000	0.71	0.704	0.275	0.00	0.00							
1.000	0.61	0.704	0.275	0.00	0.00							
1.000	0.61	0.704	0.275	0.00	0.00							
1.000	0.61	0.704	0.275	0.00	0.00							
1.000	0.61	0.704	0.275	0.00	0.00							
0.395	0.71	0.704	0.000	0.00	0.00	1.0	0.66	1.14	1.39	2		9.85
1.000	0.61	0.704	0.660	0.66	0.66							
0.395	0.71	0.704	0.000	0.00	0.00	0.0	0.00	1.00	1.08	1		1.00
0.395	0.71	0.704	0.000	0.00	0.00	0.0	0.00	1.00	1.08	1		1.00

Figure 38. Output from Risk-Weighting Functions.

E. CHAPTER SUMMARY

This chapter discussed the implementation of a risk weighting framework for vulnerabilities identified by a CARTT scan. It included the preliminary steps taken to set up the virtual environment, an in-depth review of the new Python script and updated PHP scripts, and the addition of the CSS Responsive Table for the CARTT GUI. This chapter also presented a testing scenario and results for CARTT's risk weighting framework.

The next chapter discusses the summary, conclusions, and future work for this thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSION AND FUTURE WORK

A. SUMMARY

The goal of this thesis was to extend CARTT's capabilities by providing a risk weighted analysis of identified vulnerabilities in a target system to enhance decision-making for subsequent exploitation. To that end, we designed and implemented a risk weighted framework for CARTT that provides target prioritization based solely on identified vulnerabilities. This framework was integrated into the existing CARTT architecture, which is comprised of a GVM and MSF databases, and a PHP server.

To test the framework, we conducted a vulnerability scan on a small virtual network comprised of five target virtual machines with operating systems currently used by DOD. The scan identified vulnerabilities for all five targets and, with the risk weighted analysis, CARTT calculated and prioritized the risks of the target hosts based on the data provided by each vulnerability.

This research demonstrated that a risk weighting methodology can be implemented into CARTT's workflow to provide a rating on vulnerabilities that pose the greatest risk.

B. CONCLUSION

CARTT was designed to assist novice users who may have little to no experience or knowledge of conducting cyber red teaming. Integrating a risk weighting framework in vulnerability analysis benefits users by expanding the analysis of vulnerabilities that pose the greatest risk to a host. CARTT's risk weighted vulnerability analysis now provides the user with actionable information to assist their exploitation decision-making process.

This research answered the following questions:

1. Primary Research Question

How can the identified vulnerabilities in a target system be weighted such that they can improve follow-on target exploitation?

By analyzing the raw XML scan reports, the CVSS Base Vector scores were recognized to be unique to each identified vulnerability. The quantitative value of the CVSS Base score serves as the input variable needed to calculate the Host Exposure algorithm. We applied the algorithm using Python scripting and integrated the script within the existing CARTT server. Through implementation and testing, we successfully created a risk weighting framework.

2. Secondary Research Question

How can a risk weighted framework assist red teams when using CARTT?

The primary goal of this research was to extend CARTT's capabilities by providing its users with more information for identified vulnerabilities to enhance their decision-making. We demonstrated the functionality of parsing through multiple vulnerabilities, calculating and weighting those vulnerabilities to give the user more context when decided which vulnerabilities to exploit. Also, by integrating the risk weighting framework into CARTT, we addressed the DOD OIG's recommendations for enhancing red team tools with risk-based vulnerability processing.

C. FUTURE WORK

To further enhance CARTT the following future work is recommended:

1. GVM Upgrade Integration

GVM is the central management service for conducting vulnerability scans within the CARTT server. CARTT uses an outdated version of GVM (OpenVAS version 9) that is currently at its end-of-life and no longer receives updates. By continuing to use this outdated version, the quality of the CARTT vulnerability scan may be compromised.

Future work should research the possibility of upgrading the GVM database used by CARTT. The updated GVM version has replaced many legacy features, specifically CVSS v2. The most current version supports CVSS v3. This version adds additional components to the CVSS Base Vector scoring. The addition of these new components may be beneficial for finetuning the risk weighting functionality in CARTT.

2. Streamlining to Improve Performance

Numerous students have contributed to the development and evolution of CARTT over several years. As a result, many databases, systems, and services have been added to the server framework, however, some of these are no longer in use and negatively affect CARTT's performance. This was evident when comparing CARTT's remaining available memory with the memory required to execute tasking. Future work should focus on creating a more efficient memory management structure within CARTT and auditing its sub-systems to determine their usability and necessity for CARTT functionality. This improvement would not only create space for further system development within the CARTT server but will also streamline CARTT performance.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

- [1] J. Plot, A. Shaffer, and G. Singh, "CARTT: Cyber Automated Red Team Tool," Calhoun Home, 01-Jan-2020. [Online]. Available: <https://calhoun.nps.edu/handle/10945/66973>
- [2] S. Suddle, "The weighted risk analysis," *Safety Science*, vol. 47, no. 5, pp. 668–679, May 2009, DOI: 10.1016/j.ssci.2008.09.005
- [3] Chairman of the Joint Chiefs of Staff, "Chairman of the Joint Chiefs of Staff manual," <https://www.jcs.mil/Portals/36/Documents/Library/Manuals/m651003.pdf?ver=2016-02-05-175711-083>, 28-Feb-2013. [Online]. Available: <https://www.jcs.mil/Portals/36/Documents/Library/Manuals/m651003.pdf?ver=2016-02-05-175711-083>
- [4] Department of Defense Office of Inspector General, "Followup Audit on Corrective Actions Taken by DOD Components in response to DOD Cyber Red Team-Identified Vulnerabilities and Additional Challenges Facing DOD Cyber Red Team Missions," Department of Defense Office of Inspector General, 17-Mar-2020. [Online]. Available: <https://www.DODig.mil/reports.html/Article/2114391/followup-audit-on-corrective-actions-taken-by-DOD-components-in-response-to-DOD/>
- [5] J. Berrios, "A Client/Server Model for Automated Red Teaming," Calhoun Home, 01-Dec-2020. [Online]. Available: <https://calhoun.nps.edu/handle/10945/66584>
- [6] A. Kim, M. H. Kang, J. Z. Luo, and A. Velasquez, "A Framework for Event Prioritization in Cyber Network Defense," *Naval Research Laboratory*, vol. 5540, no. 15, 15-Jul-2014.
- [7] Director Operational Test and Evaluation, "The DOT&E Mission," Director Operational Test and Evaluation - Home. [Online]. Available: <https://www.dote.osd.mil/About/Mission/>
- [8] Operational Test and Evaluation (DOT&E), "Cyber Assessment Program," Jan-2021. [Online]. Available: https://www.dote.osd.mil/Portals/97/pub/reports/FY2021/other/2021cap.pdf?ver=597qqovFSFg_PajZvaLu_w%3d%3d
- [9] Joint Chiefs of Staff, *Department of Defense Cyber Red Team Certification and Accreditation*, CJCSM 6510.03, Joint Chiefs of Staff, Washington, DDC, USA, 28-Feb-2013. [Online]. Available: <https://www.jcs.mil/Portals/36/Documents/Library/Manuals/m651003.pdf?ver=2016-02-05-175711-083>

- [10] Director, Operational Test and Evaluation (DOT&E), “Cybersecurity assessment,” FY20 Cybersecurity Assessment, Jan-2021. [Online]. Available: <https://www.dote.osd.mil/Portals/97/pub/reports/FY2020/other/2020cyber-assessments.pdf?ver=gZmlHn7--WiLeRdquSMUMw%3d%3d>
- [11] Director, Operational Test and Evaluation (DOT&E), “Cybersecurity Assessment,” FY15 Cybersecurity, January 2016. [Online]. Available: <https://www.dote.osd.mil/Portals/97/pub/reports/FY2015/other/2015cybersecurity.pdf?ver=2019-08-22-105541-220>
- [12] Operational Test and Evaluation (OT&E), FY 2021 Annual Report, Director, Operational Test & Evaluation - Cyber Assessment Program, Jan-2022. [Online]. Available: https://www.dote.osd.mil/Portals/97/pub/reports/FY2021/other/2021_DOTEAnnualReport.pdf?ver=YVOVPcF7Z5drzl8IGPSqJw%3d%3d
- [13] P. Edwards, “DSpace Repository Cyber Automated Red Team Tool,” Dec. 2019. [Online]. Available: <http://hdl.handle.net/10945/64145>
- [14] O. Goumandakoye, “An expanded graphical user interface for web-based cyber automated red teaming tool (CARTT),” Sep. 2021. [Online]. Available: <http://hdl.handle.net/10945/68326>
- [15] R. Benito, “An automated post-exploitation model for offensive cyberspace operations,” Calhoun Home, June-2022
- [16] D. Rios Insua, A. Couce-Vieira, J. A. Rubio, W. Pieters, K. Labunets, and D. G. Rasines, “An Adversarial Risk Analysis Framework for Cybersecurity,” *Risk Analysis*, vol. 41, no. 1, pp. 16–36, Jun. 2019, doi: 10.1111/risa.13331
- [17] B. Sheehan, F. Murphy, A. N. Kia, and R. Kiely, “A quantitative bow-tie cyber risk classification and assessment framework,” *Journal of Risk Research*, pp. 1–20, Mar. 2021, DOI: 10.1080/13669877.2021.1900337
- [18] J. Sumpter, “CVSS scores: A practical guide for application,” ZeroFox, 16-Aug-2021. [Online]. Available: <https://www.zerofox.com/blog/cvss-scores-practical-guide-application/>
- [19] Balbix, “What are CVSS Scores,” Balbix, 31-Jan-2022. [Online]. Available: <https://www.balbix.com/insights/understanding-cvss-scores>
- [20] NIST, “NIST History,” NIST, 15-Nov-2019. [Online]. Available: <https://www.nist.gov/history>
- [21] NIST, “NIST Special Publication 800-Series General Information,” NIST, 21-May-2018. [Online]. Available: <https://www.nist.gov/itl/publications-0/nist-special-publication-800-series-general-information>

- [22] NIST, “NIST SP 800-53r5 Security and Privacy Controls for Information Systems and Organizations,” nvlpubs.nist.gov, Sep-2020. [Online]. Available: <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r5.pdf>
- [23] Cuelogic Technologies, “How to make sense of cybersecurity frameworks,” Cuelogic Technologies Pvt. Ltd., 04-Jul-2019. [Online]. Available: <https://www.cuelogic.com/blog/cybersecurity-frameworks>
- [24] NIST, “Control Baselines for Information Systems and Organizations “ NIST. [Online]. Available: <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53B.pdf>
- [25] NIST, “NIST 800–53 Compliance Simplified: NIST Compliance Software,” Apptega. [Online]. Available: <https://www.apptega.com/frameworks/nist-800-53-compliance>
- [26] Z. M. King, D. S. Henshel, L. Flora, M. G. Cains, B. Hoffman, and C. Sample, “Characterizing and measuring maliciousness for cybersecurity risk assessment,” *Frontiers in Psychology*, 05-Feb-2018. [Online]. Available: <https://www.frontiersin.org/articles/10.3389/fpsyg.2018.00039/full>
- [27] B. Sheehan, F. Murphy, M. Mullins, and C. Ryan, “Connected and autonomous vehicles: A cyber-risk Classification Framework,” *Transportation Research Part A: Policy and Practice*, 08-Nov-2018. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S096585641830555X?via%3Dihub>
- [28] J. Jones, “An introduction to Factor Analysis of Information Risk (FAIR),” yumpu.com, Nov-2006. [Online]. Available: <https://www.yumpu.com/en/document/view/7271140/an-introduction-to-factor-analysis-of-information-risk-fair>
- [29] A. Nair, “Introduction to Information Security Risk using FAIR (Factor Analysis of Information Risk),” Aujas Cybersecurity, 2019. [Online]. Available: <https://www.aujas.com/hubfs/Introduction-to-Information-Security-Risk-Assessment-using-FAIR-1.pdf>
- [30] K. Nguyen, “Introduction to FAIR,” Medium, 23-Dec-2019. [Online]. Available: https://medium.com/@ken_nguyen/introduction-to-fair-bc906751b6d4
- [31] J. Talbot, “Risk bow-tie method,” juliantalbot, 06-Feb-2021. [Online]. Available: <https://www.juliantalbot.com/post/risk-bow-tie-method>
- [32] Broadleaf, “Bow tie Analysis,” Broadleaf, 19-May-2019. [Online]. Available: <https://broadleaf.com.au/resource-material/bow-tie-analysis/>

- [33] K. Kim, S. Oh, D. Lee, J. Kang, J. Lee, and D. Shin, “A Study on Cyber Target Importance Quantification and Ranking Algorithm,” MDPI, 10-Feb-2022.
[Online]. Available: <https://www.mdpi.com/2076-3417/12/4/1833/htm>
- [34] P. M. Mell, K. A. Scarfone, and S. Romanosky, “A complete guide to the Common Vulnerability Scoring System version 2.0,” NIST, 02-Jun-2021.
[Online]. Available: <https://www.nist.gov/publications/complete-guide-common-vulnerability-scoring-system-version-20>

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California