



NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

A CLIENT/SERVER MODEL FOR AUTOMATED RED TEAMING

by

Joseph A. Berrios

December 2020

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 December 2020		3. REPORT TYPE AND DATES COVERED Master's thesis
4. TITLE AND SUBTITLE A CLIENT / SERVER MODEL FOR AUTOMATED RED TEAMING				5. FUNDING NUMBERS
6. AUTHOR(S) Joseph A. Berrios				
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) <p>Red Team testing is a proven method to improve cybersecurity on organizational networks. However, due to the low availability of required expertise in this field, red teaming is prohibitively expensive to conduct on a large scale. In response, the Office of the Secretary of Defense has sponsored research to build a Red Team in a Box (RTIB) tool to perform many of the basic red team functions without requiring the user to have in-depth knowledge of red teaming tools and techniques. This research has resulted in the prototype implementation of CARTT, the Cyber Automated Red Team Tool.</p> <p>This thesis extended CARTT from its current stand-alone host-based implementation to include the ability to identify potential targets on a range network, communicate results to a command node, and respond to orders to attack from the command node. Redesigning the CARTT as a client/server system allows system administrators to access the tool remotely, affording increased cybersecurity throughout the Navy's networks while reducing the cost of red teaming. Additionally, the client/server model mitigates the risk of having Metasploit and OpenVAS installed on machines throughout these target networks. A messaging system was implemented that facilitates a command and control channel between users.</p>				
14. SUBJECT TERMS red teaming, client/server, automated, Red Team in a Box, RTIB, Cyber Automated Red Team Tool, CARTT				15. NUMBER OF PAGES 71
				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.

A CLIENT/SERVER MODEL FOR AUTOMATED RED TEAMING

Joseph A. Berrios
Lieutenant Commander, United States Navy
BS, U.S. Naval Academy, 2011

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

**NAVAL POSTGRADUATE SCHOOL
December 2020**

Approved by: Alan B. Shaffer
Advisor

Gurminder Singh
Co-Advisor

Gurminder Singh
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Red Team testing is a proven method to improve cybersecurity on organizational networks. However, due to the low availability of required expertise in this field, red teaming is prohibitively expensive to conduct on a large scale. In response, the Office of the Secretary of Defense has sponsored research to build a Red Team in a Box (RTIB) tool to perform many of the basic red team functions without requiring the user to have in-depth knowledge of red teaming tools and techniques. This research has resulted in the prototype implementation of CARTT, the Cyber Automated Red Team Tool.

This thesis extended CARTT from its current stand-alone host-based implementation to include the ability to identify potential targets on a range network, communicate results to a command node, and respond to orders to attack from the command node. Redesigning the CARTT as a client/server system allows system administrators to access the tool remotely, affording increased cybersecurity throughout the Navy's networks while reducing the cost of red teaming. Additionally, the client/server model mitigates the risk of having Metasploit and OpenVAS installed on machines throughout these target networks. A messaging system was implemented that facilitates a command and control channel between users.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
A.	RESEARCH PURPOSE.....	1
B.	RESEARCH OBJECTIVES.....	2
C.	RESEARCH QUESTIONS.....	2
1.	Primary Question.....	2
2.	Secondary Questions.....	2
D.	SCOPE OF THESIS	2
E.	BENEFITS OF STUDY.....	3
F.	THESIS ORGANIZATION.....	3
1.	Chapter II: Background.....	3
2.	Chapter III: Design.....	3
3.	Chapter IV: Implementation	3
4.	Chapter V: Conclusions and Future Work	4
II.	BACKGROUND	5
A.	RED TEAMING.....	5
1.	Why Red Teaming Matters to the DoD	6
2.	Status of DoD Red Teams.....	7
B.	CLIENT-SIDE RED TEAMING TOOLS.....	9
1.	CARTT.....	10
2.	Core Impact.....	12
3.	Silent Break Central	13
4.	Shortcomings in Client-Side Tools	14
C.	WEB-BASED AUTOMATED RED TEAM TOOLS.....	14
1.	Benefits of Automated Web-Based Tools	15
2.	SCYTHE	15
3.	Randori Attack Platform	16
4.	Automated, Web-Based CARTT	17
D.	CHAPTER SUMMARY.....	17
III.	DESIGN METHODOLOGY	19
A.	CARTT OPERATION OVERVIEW.....	19
B.	SYSTEM DESIGN.....	20
1.	Architecture.....	20
2.	Operator Process Flow	21
C.	CARTT SERVER DESIGN.....	23
1.	Server Functions	25

D.	CARTT CLIENT DESIGN	26
1.	Login and Registration	26
2.	Commander Interface	27
3.	Operator Interface	29
4.	Administrator Interface	36
E.	CHAPTER SUMMARY	36
IV.	IMPLEMENTATION	37
A.	SECURITY IMPLEMENTATION	37
B.	ACCOUNT AND MESSAGING SYSTEM	38
1.	Account Creation	38
2.	Account Management	39
3.	Messaging System	40
C.	CARTT USE OF METASPLOIT AND OPENVAS	41
1.	Interacting with OpenVAS	41
2.	MSF Exploit Construction	45
D.	CHAPTER SUMMARY	48
V.	CONCLUSIONS AND FUTURE WORK	49
A.	SUMMARY	49
B.	CONCLUSIONS	50
1.	Primary Question	50
2.	Secondary Questions	51
C.	FUTURE WORK	51
1.	Improve the User Interface	52
2.	Advanced User Mode and Post-exploitation Capabilities	52
3.	Improve Logging and Record Keeping	52
	LIST OF REFERENCES	53
	INITIAL DISTRIBUTION LIST	55

LIST OF FIGURES

Figure 1.	CARTT with Extended Cyber-Attack Capability Flow Diagram. Source: [1].....	11
Figure 2.	Sample CARTT Architecture	20
Figure 3.	Process Flow of an Operator Performing a Scan	22
Figure 4.	CARTT Server Diagram	23
Figure 5.	CARTT Login Page	26
Figure 6.	CARTT Registration Page	27
Figure 7.	Messaging Interface	28
Figure 8.	Operator Main Menu.....	29
Figure 9.	Scan Creation Page	30
Figure 10.	Scan Status Page	31
Figure 11.	Sample CARTT Scan Start Page	32
Figure 12.	Scan Import Page	33
Figure 13.	Workspace Selection Page.....	33
Figure 14.	Vulnerability List for Host 17.10.42.239.....	34
Figure 15.	Modules Associated with CVE-2017-0143	34
Figure 16.	Results of PSEXEC Exploit on Host 17.10.42.239	35
Figure 17.	Administrator Menu.....	36
Figure 18.	Session Check Logic in message_page.php.....	37
Figure 19.	Select Statement to Validate Unique User Email	38
Figure 20.	User Registration Password Hashing and Variable Binding	38
Figure 21.	Sample CARTT mySQL Update, Delete, and Insert Statements	39
Figure 22.	Sample Queries Used by the Messaging System.....	41

Figure 23.	Msf_connect() Function used by CARTT to Create MSF Processes	42
Figure 24.	Commands Used to Load and Connect to OpenVAS Inside MSF	42
Figure 25.	Implementation of Target Creation, Task Creation, Task Start, and Task List Commands	44
Figure 26.	Example MSF Workspace Change/Creation, MSF Report Import, and OpenVAS Report Download.....	44
Figure 27.	How CARTT Creates the hosts_\$user.txt File	45
Figure 28.	MSF Search Command	46
Figure 29.	Construction of Exploit Script and Execution of MSF with the Script	47
Figure 30.	Contents of Exploit Script for Target 17.10.42.239 Using Exploit Module ms17_10_eternalblue.....	47

LIST OF ACRONYMS AND ABBREVIATIONS

APT	advanced persistent threat
ASCII	American standard for information interchange
C2	command and control
CARTT	Cyber Automated Red Team Tool
CCDM	combatant command
CI	Core Impact
CIDR	classless interdomain routing
CLI	command-line interface
CSSP	cybersecurity service provider
DCO	defensive cyber operations
DISA	Defense Information Systems Agency
DoD	Department of Defense
DoDIN	Department of Defense information network
DOT&E	Office of the Secretary of Defense's Director, Operational Test and Evaluation
DSB	Defense Science Board
GIAC	Global Information Assurance Certification
GOA	Government Accountability Office
GUI	graphical user interface
IG	inspector general
IP	internet protocol
MSF	Metasploit Framework
NIST	National Institute of Science and Technology
NIWC	Navy Information Warfare Center
Nmap	network mapper
NPS	Naval Postgraduate School
NSA	National Security Agency
OCO	offensive cyber operations
OpenVAS	Open Vulnerability Assessment System
OPFOR	opposing force

OSD	Office of the Secretary of Defense
PCO	persistent cyber operations
PHP	hypertext preprocessor
RAP	Randori Attack Platform
RTIB	Red Team in a Box
RTT	Red Team Toolkit
SaaS	software as a service
SBC	Silent Break Central
SMB	server message block
SQL	structured query language
stdin	standard input
stdout	standard output
TCP	transmission control protocol
TDY	temporary duty assignment
Tkinter	toolkit interface
TTP	tactics, techniques, and procedures
UDP	unicast data protocol
UI	user interface
UIC	unit identification code
UPnP	universal plug and play
USB	universal serial bus
USCYBERCOM	United States Cyber Command
VM	virtual machine
XML	extensible markup language

ACKNOWLEDGMENTS

I want to thank my wife, Kitrina, for putting up with me and listening to me ramble for hours about programming challenges. I also want to thank my advisors, Dr. Alan Shaffer and Dr. Gurminder Singh, for trusting me with this research and for all of the help along the way. I also have to thank the Navy for allowing me to attend the Naval Postgraduate School.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. RESEARCH PURPOSE

Red teaming has a proven history of ensuring cyber security when it is conducted and documented properly. However, the training, expertise, and knowledge of appropriate tools and techniques required to perform effective red teaming come with a significant cost in time and resources. As a result, the Office of the Secretary of Defense (OSD) has sponsored research into developing a tool that performs automated red teaming actions without the need for its user to be an expert in the field. The goal of this research has been to design and implement a software tool that achieves this objective. Previous research has yielded a prototype called the Cyber Automated Red Team Tool (CARTT) [1], [2].

Red teaming requires an in depth understanding of specialized tools, systems, and techniques that allow one to fully assess a target system for vulnerabilities that may be exploitable by adversaries, as well as the target's ability to withstand and respond to those exploitations. Developing red team expertise through training, however, is expensive and time consuming, and many of the tools used for these purposes are expensive or lack intuitive interfaces for non-expert users. [3], [4]. These constraints create high barriers for large organizations such as the Department of Defense (DoD) to fully train and maintain adequate red team experts to test their massive number of systems [5], [6], [7].

Red teaming, if done incorrectly, can result in cyber effects similar to those of a malicious actor. These effects could include denial and/or degradation of services, loss of data, or loss of data integrity. To prevent such outcomes, red teaming must be conducted in a careful, deliberate manner where all personnel involved understand the risks. While implementing the CARTT system as a client/server model will allow greater access to its capabilities, it also may increase the possibility of it being used maliciously to disrupt or degrade network operations. To mitigate this risk, strong authentication is used in the system.

B. RESEARCH OBJECTIVES

The purpose of this research is to implement a simple red teaming system that is accessible through a browser-based interface, as a way to increase cyber security throughout the DoD while limiting the costs associated with training and maintaining cyber red teams. This system should empower local defenders by providing them a deeper understanding of their network security. Additionally, the system should reduce the workload on the limited red testing assets currently employed throughout the DoD.

C. RESEARCH QUESTIONS

In this research, we investigate how a highly automated red teaming tool can be developed using a client/server model that relies on open source software and provides user-friendly functionality while having strong security features. The following are the key questions for this research:

1. Primary Question

How can CARTT be transformed from a stand-alone application into a system that employs a client/server model?

2. Secondary Questions

- (1) How can CARTT employ a command-and-control channel between its Operators and Commanders?
- (2) How can CARTT Operators use the system to remotely conduct red teaming analysis of a network to which they are not attached?

D. SCOPE OF THESIS

This thesis examines previous research on red teaming tools in order to develop a highly automated, user friendly, secure client/server system that performs network discovery, vulnerability analysis, and exploitation testing using open source software on a target network. The focus is to facilitate command and control (C2) between users, manage user accounts, and provide user authentication.

E. BENEFITS OF STUDY

This thesis resulted in an automated red team tool that can be accessed without requiring any additional software deployed on endpoint client systems. Operators can use the tool to generate timely, meaningful vulnerability analysis of their networks that would traditionally require personnel with expensive red team training and experience. Through its implementation, the system can improve the overall cybersecurity of DoD networks.

F. THESIS ORGANIZATION

The remainder of this thesis is organized as follows:

1. Chapter II: Background

Chapter II details the current state of DoD Red Teams, including manpower, funding, and training issues that have contributed to a shortfall in DoD Red Team capability. It also examines commercial and open source products available for conducting red teaming.

2. Chapter III: Design

Chapter III describes the design choices that were made in transforming CARTT from a stand-alone program to a client/server model. We highlight the emphasis that was placed on simplicity and security in designing the client interface and background processes. This chapter concludes by outlining the process flow and system relationships inside of CARTT.

3. Chapter IV: Implementation

Chapter IV discusses the technical implementation of the CARTT system. It describes in detail how the underlying subsystems, MySQL, PHP, MSF, and OpenVAS are used to automate red team assessments, provide security, and present the user with a simple user interface.

4. Chapter V: Conclusions and Future Work

Chapter V summarizes the research conducted in this thesis with analysis of the conclusions drawn from the research questions and hypothesis. Constraints and limitations that were encountered due to design or implementation choices are also discussed. Finally, it provides recommendations for future work to expand CARTT functionality and usability.

II. BACKGROUND

The Department of Defense information network (DoDIN) is under constant barrage from malicious actors; the Defense Information Systems Agency (DISA) blocked over 300 million malicious actions every day in 2019 [8]. As such, the Department of Defense (DoD) has refocused its efforts towards improving the security of its networks and cyber systems. This chapter covers background relevant to the status of the DoD's cybersecurity red teams, some popular red team tools, and the benefits of automated red team solutions.

A. RED TEAMING

The proliferation of cyber-systems throughout the DoD has coincided with an increased need to secure them. The DoDIN is a critical component of the U.S. military's ability to project power and conduct operations. It is massive in scope and globally dispersed, consisting of ships, aircraft, satellites, cloud services, industrial control systems (ICS), tactical communication systems, mobile devices, and commercial and private infrastructure. There are approximately 15,000 networks inside the DoDIN that provide service for ~3 million users spread across 3,500 locations in 26 different countries [9], [10].

While the DoD is making significant efforts to improve cybersecurity, these cannot mitigate all of the existing vulnerabilities present in often decades-old fielded systems. The Government Accountability Office (GAO) found that the DoD more often required security in traditional IT systems than in weapon systems,¹ ignoring or not understanding that weapon systems are going to be integrated into the DoDIN. As a result, the majority of currently fielded equipment has little to no cybersecurity in place, increasing the risk for the entire DoDIN. Additionally, there are legacy systems and applications throughout the DoDIN that no longer receive vendor support or patching but are still allowed to remain in service due to the lack of a suitable replacement [7].

¹ The GAO uses the term "weapon system" to refer to major defense acquisition programs that include a broad range of systems, such as aircraft, ships, combat vehicles, radios, and satellites.

One proven method for improving cybersecurity is through red teaming. A red team is comprised of technically skilled cybersecurity professionals who attempt to penetrate and exploit computer networks using the tactics, techniques, and procedures (TTPs) of a real-world cyber threat. The goal is for the red team to discover vulnerabilities and gaps in the security posture, training, and system configurations of the target systems so that corrective or remediation actions can be taken to prevent a malicious actor from penetrating the network. Additionally, red team activities tend to increase the response and reaction proficiency of system defenders to detect, diagnose, and mitigate cyber-attacks.

1. Why Red Teaming Matters to the DoD

The GAO found that almost every major acquisition program that was tested in a five-year period had mission-critical cyber vulnerabilities; the Office of the Secretary of Defense's Director, Operational Test and Evaluation (DOT&E) and Defense Science Board (DSB) had similar results in their reviews [6]. Examples of common vulnerabilities were weak or default passwords to critical systems and accounts, as well as a lack of or improper application of encryption systems. These are rudimentary issues that the National Institute of Technology and Standards (NIST) provides guidance on managing. This highlights the importance of performing red team assessments to ensure that program offices are properly implementing cybersecurity controls [7].

The primary threats to the DoDIN are nation-state level actors, which are typically more advanced, harder to detect, and more persistent than hacktivists or script kiddies. In order to adequately defend the DoDIN, its network defenders must be highly trained to identify, respond and mitigate cyber-attacks. As such, the DOT&E and the GAO both rely heavily on DoD Red Team assessments to gain perspective on the status of the DoDIN's cybersecurity [5], [6].

The Navy adage of "train like you fight" is frequently applied to the conventional warfare areas, such as Air Defense and Anti-Submarine Warfare. It should be apparent that network defenders must also follow that adage in order to be prepared to battle cyber adversaries. Red teaming offers a unique opportunity for local defenders to "fight the network" as they get an opportunity to learn how their systems, firewalls and anti-virus and

intrusion prevention/detection systems respond to the TTPs employed by the red team. This improves the defender's ability to recognize and respond to real-world events. Additionally, after the completion of the event the red team provides feedback about what mitigation efforts were most and least successful, further improving the response capability of the local defenders.

Even from a non-technical perspective, red teams provide valuable insight into the security posture of the DoDIN. The DoD mandates that all personnel complete annual Cybersecurity Training, which includes information on detecting and preventing insider threats and phishing attacks; this training can normally be completed in less than one hour [11]. One of the DoD Red Teams, in a publicly available interview, stated that, "Most cyber-attacks are user driven... [they're] easiest to get at and yield the most reliable results. We've never had a phishing campaign that has failed" [12]. This statement, even if taken as an exaggeration, clearly demonstrates a shortcoming in the mandated training.

Additionally, the first indication of compromise or data breach will likely be well after an adversary has successfully conducted a cyber-attack. IBM assesses the average time between a breach and its detection at 279 days in the private sector [13]. For an entity like the DoD, where lives are dependent on operational security and information superiority, that type of lag in detection can have disastrous consequences.

2. Status of DoD Red Teams

As the DoD has aggressively expanded the scope of the DoDIN in an effort to improve connectivity to the warfighter, the resources allocated to defend these networks have not kept pace with this expansion. This can cause local defenders to receive training that is insufficient and/or unrealistic for dealing with real-world threats.

There are ten DoD Red Teams certified by the National Security Agency (NSA) and accredited by United States Cyber Command (USCYBERCOM), dispersed between the Army, Navy, Air Force, and Marine Corps [7]. The size, composition, and equipment available to each team varies between services. The DoD Red Teams support exercises as an opposing force (OPFOR), providing testing for development and acquisition programs, and assessing the proficiency of the cybersecurity service providers (CSSP) [12].

Currently, the DOT&E coordinates with combatant commands (CCMDs) to perform cyber readiness campaigns. These campaigns leverage DoD Red Teams to assess network defenders' ability to detect, identify, and remediate cyber-attacks [6]. The GAO, DOT&E, and Navy Information Warfare Center (NIWC) Atlantic all noted that the demand for DoD Red Team services is very high due to the consistently positive results that are produced.

There has been a push from DOT&E to have DoD Red Teams conduct Persistent Cyber Operations (PCO) that more closely align with what a nation-state would attempt. Currently, the majority of DoD Red Teaming occurs during exercises or training events, and as such is limited in scope and duration. Conducting PCO allows the red team to provide a deeper assessment of the target systems, and more realistic training to the network defenders. However, this requires additional personnel, equipment, and funding since it requires the red teams to spend a significantly longer amount of time against one target, often while on temporary duty assignment (TDY) [6].

In their 2018 report, DOT&E explicitly stated that the manning models used for Red Teams across the DoD and the loss of talent to the private sector are serious concerns that must be addressed to meet demand for their services. Additionally, the DOT&E concluded that additional personnel, training, capabilities and other resources are required for DoD Red Teams to adequately perform their mission [6]. The GAO echoes this sentiment, going as far as to claim that the monetary compensation offered by the private sector is so much greater than what the DoD can offer that it places the DoD at a significant disadvantage from a manning and expertise perspective [7]. As a result, DoD Red Teams are falling behind in their ability to accurately emulate nation-state actors. The DOT&E report explains that most of the NSA and DoD Red Teams are only capable of operating at a moderate threat level at best, and that none of the teams can accurately represent a near peer nation-state capability. The report goes on to say that some of the red teams are so ineffective that they cannot be relied upon to perform future assessments [6].

The DoD Inspector General (IG) has a bleak view of the current and future status of the DoD Red Teams. A recent report stated that the DoD Red Teams are not capable of meeting current or future demands for their service [5]. The DoD IG 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. The report recommended the identification and development of a baseline toolkit/capabilities for the DoD Red Teams as well as formalizing a reporting process so that system owners and stakeholders can be more aware of red team findings [5].

B. CLIENT-SIDE RED TEAMING TOOLS

There are various tools available that facilitate red teaming, both open source and commercially licensed. Among the most common tools are Network Mapper (Nmap), Wireshark, Greenbone's Open Vulnerability Assessment Scanner (OpenVAS), Rapid7's Metasploit Framework (MSF), Core Security's Core Impact (CI), and Tenable's Nessus. The Kali operating system is a Linux distribution developed by Offensive Security that is tailored to cybersecurity professionals and includes a wide range of open source red team tools.

Many of these tools are employed from a command line interface (CLI) that requires the user to be well versed in this environment, as well as in the tool itself and the exploitation techniques that it leverages. This creates a barrier for entry that can be difficult to overcome for the average network defender. It is common for network defenders to be unable to receive in-depth training for many of these tools because they are prohibitively expensive for an enterprise as large as the DoDIN. The lowest cost for training and certification from Offensive Security for Kali is priced at \$1,000 [14]; similarly, Global Information Assurance Certification (GIAC) training costs \$1,999 for a single tool or topic [15]. Both organizations project between 30–120 days for individuals to complete training.

Additionally, due to the high demand across the private sector for personnel with this skillset, the DoD has been struggling to retain the personnel it trains. The DoD is simply unable to match the compensation that is offered by the private sector. Taking the average from three different sources, the average penetration tester/red teamer in the United States makes approximately \$93,000 annually [3], [16], [17]. This is well above the compensation enlisted personnel and junior officers can hope to receive, and results in the DoD getting minimal returns on its training investment. The DoD also has to contend with

its duty rotations causing constant turnover in personnel, greatly increasing its training requirements.

The following section focuses on three red team tools: the Cyber Automated Red Team Tool (CARTT) developed at NPS, Core Security's Core Impact, and Silent Break Security's Silent Break Central (SBC).

1. CARTT

The CARTT was developed at the Naval Postgraduate School in previous research in the area of automated red teaming. CARTT was built entirely upon open source software running in Kali Linux, leveraging Nmap, OpenVAS, p0f, and MSF [2]. The application was programmed in Python with the Toolkit Interface (Tkinter) library powering its GUI [1]. The goal was to develop a tool that could, "simulate the actions of a red team by automatically identifying and analyzing vulnerabilities in computer systems, then exploiting those vulnerabilities with cyber-attack actions... by operators who do not have extensive training in offensive cyber operations (OCO) or red teaming" [2]. The CARTT user is able to conduct host discovery, vulnerability analysis, and exploitation from the GUI.

There are currently two components to CARTT [2]: the first deals with host discovery, OS mapping, and vulnerability detection; the second leverages the vulnerabilities that were discovered in order to perform target exploitation. CARTT utilizes ifconfig to determine which host it is running on, then performs a scan to conduct host discovery. Nmap with option `-sn` is used to perform a ping scan of the host network, which is currently set to be the /24 classless interdomain routing (CIDR) of the localhost. The operator can optionally perform two OS discovery scans, using Nmap to actively scan all the hosts with the `-O` option, or by using p0f to passively monitor network traffic to make a determination about the OS of all the hosts communicating on the network [2].

After scanning has completed, CARTT initializes MSF and OpenVAS [2]. The information gained from the earlier host discovery is entered into OpenVAS, which then executes a vulnerability scan of these hosts. In the current implementation, the operator must exit CARTT to interface with OpenVAS in order to populate the target list and

manage the scan results. Ongoing work will fully integrate these steps into the CARTT interface.

The second component of CARTT is its extended cyber-attack capability [1]. The results from the OpenVAS vulnerability scan are downloaded to a local file that is parsed by CARTT. The operator then selects a host from the menu to see the list of vulnerabilities detected on that host. Once the operator has selected a host and vulnerability, CARTT displays a list of applicable MSF exploit modules. The operator can then select a module to receive relevant information about it, and then use the “Exploit Setup” feature to have CARTT build the corresponding exploit in MSF and provide feedback once it is successfully constructed. The operator can then run the exploit within CARTT, which instructs MSF to launch the exploit, with the results parsed from the MSF interface and displayed in CARTT. A flow chart of this process is depicted in Figure 1.

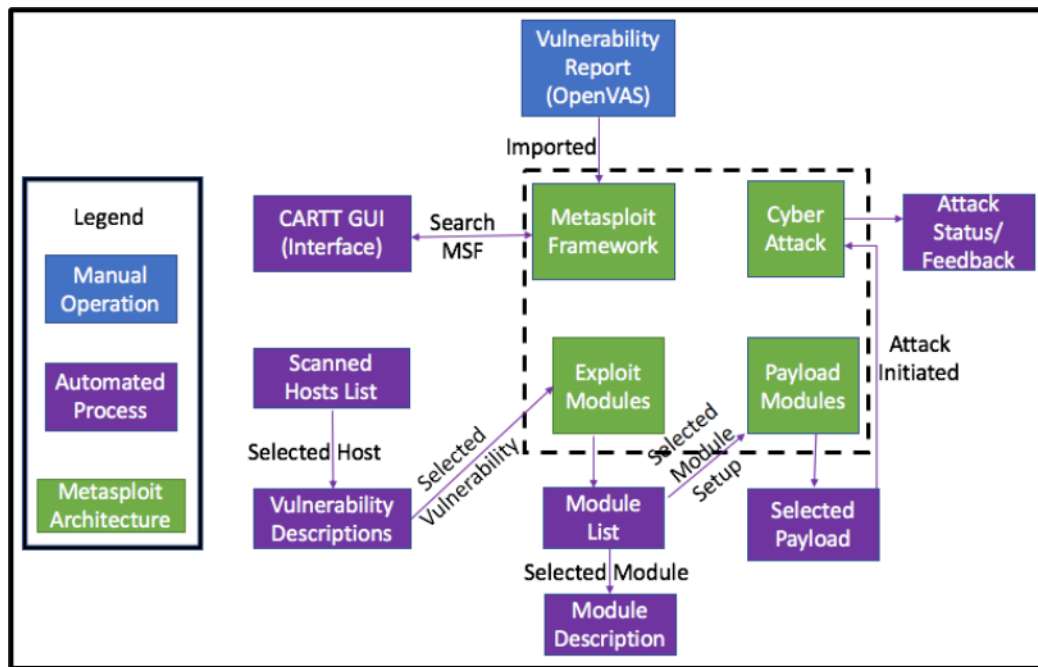


Figure 1. CARTT with Extended Cyber-Attack Capability Flow Diagram.
Source: [1]

The primary benefit of CARTT is its ability to reach standalone networks that are isolated from the DoDIN. A device could be configured with the CARTT software and

then connected directly to air-gapped networks. A more extensible solution, however, should provide this capability without the need for a system administrator to build and maintain a Kali system with the CARTT python script loaded, running OpenVAS with GSA, MSF, p0f, and Python3.x with Tkinter, Subprocess, and Shlex libraries. In order to use CARTT, a system with all of this software must be placed on the target network, which itself presents an entry point for infection by malicious software. Additionally, OpenVAS requires periodic updates from the Greenbone Security/Community Feed to keep its vulnerability library updated. A client/server model for CARTT, which is the focus of this research, would solve these issues by allowing organizations to run a centrally managed server system that local defender “thin clients” could connect to and interact with [1].

2. Core Impact

Core Security provides the Core Impact (CI) penetration testing tool [4]. It features a GUI with step-by-step guides on how to perform various functions of pen-testing. For example, conducting a Network Vulnerability test requires the user to simply press a single button and enter the target IP information (can be an entire network or individual hosts). CI will then perform network discovery, port mapping, and OS scanning using various methods, including universal plug and play (UPnP) protocol, user datagram protocol (UDP) and transmission control protocol (TCP) packets, and server message block (SMB) packets [4].

CI then conducts vulnerability analysis by attempting to exploit the devices on the network based on ports, protocol, and OS determined in the scan [4]. It provides feedback on what exploits were successful and unsuccessful. Additionally, all of the tools, modules, and exploits are modifiable Python scripts, which allows the operator to have a great deal of flexibility to customize the tool. However, this requires a high degree of technical expertise [4].

Operators can also configure the attacks for post-exploitation operations, such as privilege escalation and detection evasion [4]. This allows the operator to better simulate a real-world threat for the network defender to fight against. Additionally, the exploits can be configured to self-destruct once the operation is complete, reducing the work required

to restore the target systems to their previous configuration and ensuring that all malicious payloads are removed [4].

CI also gives the operator the ability to configure and conduct phishing attacks, providing feedback on which users and devices succumbed to the attack [4]. This allows network defenders to train their users in a live environment, and to identify additional training requirements. The NIWC Atlantic Red Team claimed to have never had a phishing campaign fail, underscoring the clear shortcomings that currently exist in DoD cyber defense training. Unfortunately, since CI is such a robust tool, it has a significant cost, with a single license costing \$30,000 per year [4].

3. Silent Break Central

Silent Break Security offers the Silent Break Central (SBC) application [18]. SBC contains an attack simulation module that allows the operator to build an attack from its GUI. The operator selects criteria from a series of menus and enter required information into text fields. This results in a simple process to plan and develop cyber-attacks. The benefit of SBC's approach is that a novice network defender should be able to navigate the GUI to launch attacks and get actionable feedback from the tool. The GUI provides a detailed description of the exploit, the risk factors, links to external resources and recommended remediations. Once the attack has been launched, the GUI will provide feedback on its success, which allows the operator to conduct remediation actions as needed [18].

The SBC exploits and payloads are prepackaged, giving the operator limited freedom to customize some of the attack criteria [18]. As a result, newly discovered vulnerabilities cannot be leveraged by the system until the vendor has provided an update. There is also no functionality for the user to build their own exploits or payloads. Due to this limitation, SBC cannot be used to discover new vulnerabilities. This limits the application to a compliance enforcement function [18].

Silent Break provides other services, such as a password auditor and a post-exploitation agent called Red Team Toolkit (RTT) [19]. The password auditor will attempt to crack passwords on the network and provided detailed feedback. Passwords that are

cracked, duplicated, weak, or set to not expire will be flagged, as well as the accounts associated with them. Alternately, RTT allows the operator to more accurately simulate a hostile actor in the network. While the attack simulation module only provides feedback on if the exploit was successful, RTT agent allows the operator to conduct persistent operations on the target. RTT costs \$7,000 for the initial 1-year license, then \$3,500 annually per user [19].

4. Shortcomings in Client-Side Tools

Client-side red team tools share a common set of shortcomings. First, every client-side red team tool requires that the operator have a fully updated version of the client in order to ensure the latest vulnerabilities and exploits can be detected. This consumes bandwidth to disseminate the updates and hard drive space on the clients to store the ever-growing vulnerability databases. Additionally, if the networks are isolated, it can be challenging to ensure updates occur. Second, every device running a client-side red team tool requires elevated permissions in order to perform the host discovery and network mapping functions. This introduces risk to the organization, if a malicious actor compromises one of the client devices they would be able to conduct reconnaissance with a very low chance of being detected. Third, for an enterprise as large and diverse as the DoDIN, managing commercial licenses and access is a tremendous and expensive challenge. If each network only required two licenses, that would cost the DoD \$105 million dollars annually for the RTT [10], [19]. The high rate of personnel turnover due to periodic rotations in the DoD further complicates the issue from an access management and training perspective.

C. WEB-BASED AUTOMATED RED TEAM TOOLS

The previous section discussed client-side red team tools, including shortcomings with that model. This section discusses the benefits of web-based automated red team tools as well as two current services that are commercially available. These web-based tools were selected due to their highly automated nature as well as their ability to conduct vulnerability assessments and offensive actions against the target network.

1. Benefits of Automated Web-Based Tools

Using a web-based automated system for red teaming would allow the DoD to avoid many of the issues present in client-side systems. A web-based tool would remove the requirements for every command or network to manage its own system configuration and updates. This reduces the administrative cost of managing the tool at the command level and the cost of deploying the tool to thousands of sites at the enterprise level. Additionally, the only device performing red teaming functions would be a centrally managed system, mitigating the requirement to add exceptions or elevated access control permissions to potentially thousands of devices or user accounts across the enterprise. This greatly reduces the risk of the tool being compromised by reducing its surface area. Furthermore, relying on an open source system avoids licensing costs, which could save the DoD hundreds of millions of dollars.

While an automated tool is not a substitute for a full red team exercise, it does allow the local defender to conduct red team operations in support of compliance, remediation, and training. This would allow the DoD Red Teams to focus more of their resources and expertise on emulating real-world adversary threats. Automation is one way to reduce the barrier to entry and allow more personnel to share the workload. However, if the average system administrator is unable to use an automated tool to generate actionable results in a timely manner, then they simply will not use it. Therefore, any solution must prioritize ease of use, which in turn reduces the cost of training, both in terms of dollars and time spent. Training costs will likely never be completely eliminated but reducing them should be a priority.

2. SCYTHE

SCYTHE is a web-based red teaming tool that can be used either on a local server or as software-as-a-service (SaaS) from the vendor [20]. The client operator connects to the server and is presented with a browser-based GUI that allows them to build ‘Campaigns’ that consist of various cyber exploits and attacks. The operator selects from menus the components of the campaign, including payloads, exploits, target services or processes, protocols, timing, etc. Some of the possible cyber-attack options include

ransomware, phishing, and USB malware installation. These give the operator the ability to emulate the TTPs of real-world cyber adversaries [20].

Once a SCYTHE campaign is triggered, the GUI provides real time feedback to the operator that includes what systems and accounts have been compromised, and through what methods [20]. After the attack has been completed, SCYTHE will generate a technical report with mitigation recommendations. These features allow the operator to understand where the gaps are in the cybersecurity posture of the target, and to make technically sound recommendations [20].

SCYTHE advertises a wide range of prebuilt exploits and payloads [20], as well as the ability for the operator to create their own payloads and exploits. This would allow a skilled operator to use SCYTHE to identify zero-day vulnerabilities and vulnerabilities that do not yet have a pre-built module. The actual cost of a license is not publicly available, as the company opts to give quotes directly to interested parties [20].

3. Randori Attack Platform

Randori launched the Randori Attack Platform (RAP) service in early 2020 [21]. Their stated goal of the service is to improve cybersecurity by allowing network defenders to practice defending against automated attacks that emulate real-world threats in a live environment [21]. The platform is designed to simulate the entire attack process, from reconnaissance to persistence. Its advertising suggests that the heavily automated system significantly reduces the cost of the service, however no pricing information is available to the public.

RAP is a large deviation from most other automated red team tools in that it is entirely automated [21]. There is no direct interaction between the defenders and the tool itself, instead the system decides how to conduct the red team operation. While this eliminates the need for operator training, it also prevents the customization of attacks based on organizational priorities. The local defenders must trust that the Randori algorithms are presenting them with a realistic threat from a TTP, targeting, and capability perspective. While this is likely suitable for commercial entities, the DoD is frequently targeted by nation-state actors; which means there are a different set of TTPs the DoD must contend

with compared to the private sector. Additionally, if the DoD or Intelligence Community discovers a change in adversary TTPs, it may be unable to share that information with Randori due to classification issues. This would cause the RAP assessments to be less accurate and possibly foster overconfidence in the security of the network [21].

The system depends on an algorithm that Randori refers to as its ‘hacker logic’ program; this algorithm automates the attack and emulates the TTPs used by real-world advanced persistent threats (APTs) [21]. The platform takes in an email account for the target network and will then begin a series of automated functions. The first is to conduct network discovery, presumably the TTPs used reflect APTs that have been or are current targeting the customer. After discovery the platform will begin a search for gaps by attempting numerous real-world exploits. The platform will provide continuous feedback on what it has determined as the attack surface of the target and what exploits have succeeded [21].

4. Automated, Web-Based CARTT

In comparison to RAP and SCYTHE, this research is conducted using open source tools. This eliminates licensing costs and offers improved transparency into the underlying systems compared to the commercial products. Additionally, while the system we develop is highly automated from a user perspective, the system owner has the ability to create custom exploits and payloads using MSF. This offers greater flexibility than RAP [21], which is fully automated, while also putting guard rails on the operator that SCYTHE lacks [20].

D. CHAPTER SUMMARY

This chapter covered the importance of cybersecurity for the DoD. It also covered the benefits of red teaming to the DoD, and the current state of the DoD Red Teams. Finally, it discussed several notable red team tools and products, including previous work conducted at NPS in developing CARTT.

In the next chapter, we discuss the design of CARTT as a client/server model.

THIS PAGE INTENTIONALLY LEFT BLANK

III. DESIGN METHODOLOGY

This chapter discusses the design methodology used to develop CARTT. It describes in detail the three CARTT user roles as well as the processes for Administrators to manage the system, for Operators to build, perform, and report the status of their cyber actions, and for Commanders to perform command and control over Operators.

A. CARTT OPERATION OVERVIEW

CARTT allows local network defenders to conduct a red team audit of their network through a simple user interface (UI) that automates interactions with MSF, OpenVAS, and other specialized red teaming and pen-testing tools. This alleviates the pressure on Operators to become experts in the use of these tools and the requirement to deploy such tools on every network. Additionally, the automated nature of CARTT allows Operators to perform other work tasks while the scan is being processed.

The Operator logs-in to CARTT via a client/browser interface and proceeds to configure and conduct a scan of a target network (such as their own local network). To do this the Operator enters some basic information, such as the IP range and a name for the scan, and CARTT then performs a vulnerability analysis on the provided IP range, returning the results to the Operator via the CARTT Client.

Once the scan has completed, the Operator has additional options available, for example, with a few simple webforms and clicks, the Operator can select the host(s), vulnerability, and payload to audit. Once the Operator is satisfied with the selections, the audit can be run on the host(s).

CARTT then fires the exploit with the appropriate options, collects the results, and returns the results to the Operator via the Client. To run additional exploits the Operator can select different options for host, vulnerability, and module. This enables the Operator to perform multiple tests without having to navigate multiple menus and subsystems.

CARTT also supports Administrator and Commander roles. The CARTT Administrator manages user accounts, including setting appropriate roles, performing

password resets, and account unlocks. The Administrator is also responsible for maintaining the *commands* table in the database. The CARTT Commander role primarily exists for users at headquarters that do not need to directly perform red team assessments. It provides these users with a channel to send tasking and receive communication from subordinates through CARTT.

B. SYSTEM DESIGN

1. Architecture

CARTT is designed so that a single server runs the processes and applications required to perform a red team assessment; this system will be referred to as the CARTT Server. As described above, the CARTT Client has three user roles: Operator, Commander, and Administrator. The Client is the web-based interface through which an Operator uses CARTT. The Target Network consists of any internet-connected devices on the IP range provided by the Operator. The CARTT Server, CARTT Client(s), and Target Network must be connected via the internet. The user may access the CARTT Client from within the Target Network, but it is not necessary. A notional CARTT architecture is shown in Figure 2.

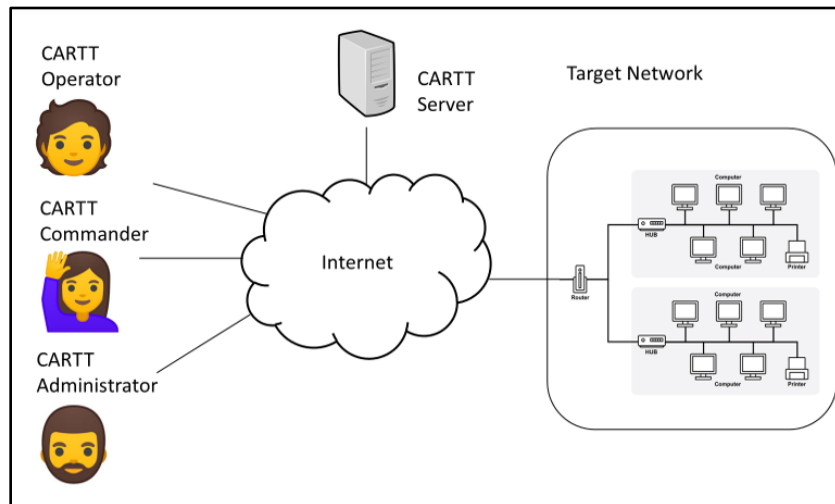


Figure 2. Sample CARTT Architecture

2. Operator Process Flow

The full process flow for an Operator logging into CARTT and attempting exploitation of a target host is detailed in Figure 3. Every interaction with CARTT starts the same way, with the user connecting to the CARTT Client and logging in. The CARTT Server validates the user's credentials and then redirects the user to the correct menu page based on their role.

To start an assessment the Operator fills in a webform to configure and conduct a scan. The data provided by the Operator is used by the CARTT Client to create the commands necessary for the CARTT Server to leverage OpenVAS to conduct a vulnerability scan of the target network. The CARTT Client polls OpenVAS every few seconds in order to provide the Operator with feedback on the status of the scan.

Once the CARTT Client detects that the scan is complete, it requests the import of the scan into MSF. The CARTT Server first downloads the scan as an Extensible Markup Language (XML) file, then imports that file into MSF as a PostgreSQL database workspace designated by the Operator. The Operator is then presented a list of hosts and vulnerabilities in the Client that are based on the scan results. The Operator then selects the host, vulnerability, and payload from the choices provided. The CARTT Client packages these inputs as commands for the CARTT Server to configure the exploit inside MSF. The CARTT Server then attempts to run the exploit against the specified target. The Operator gets feedback in the Client from MSF on the status of the exploit once it has been attempted.

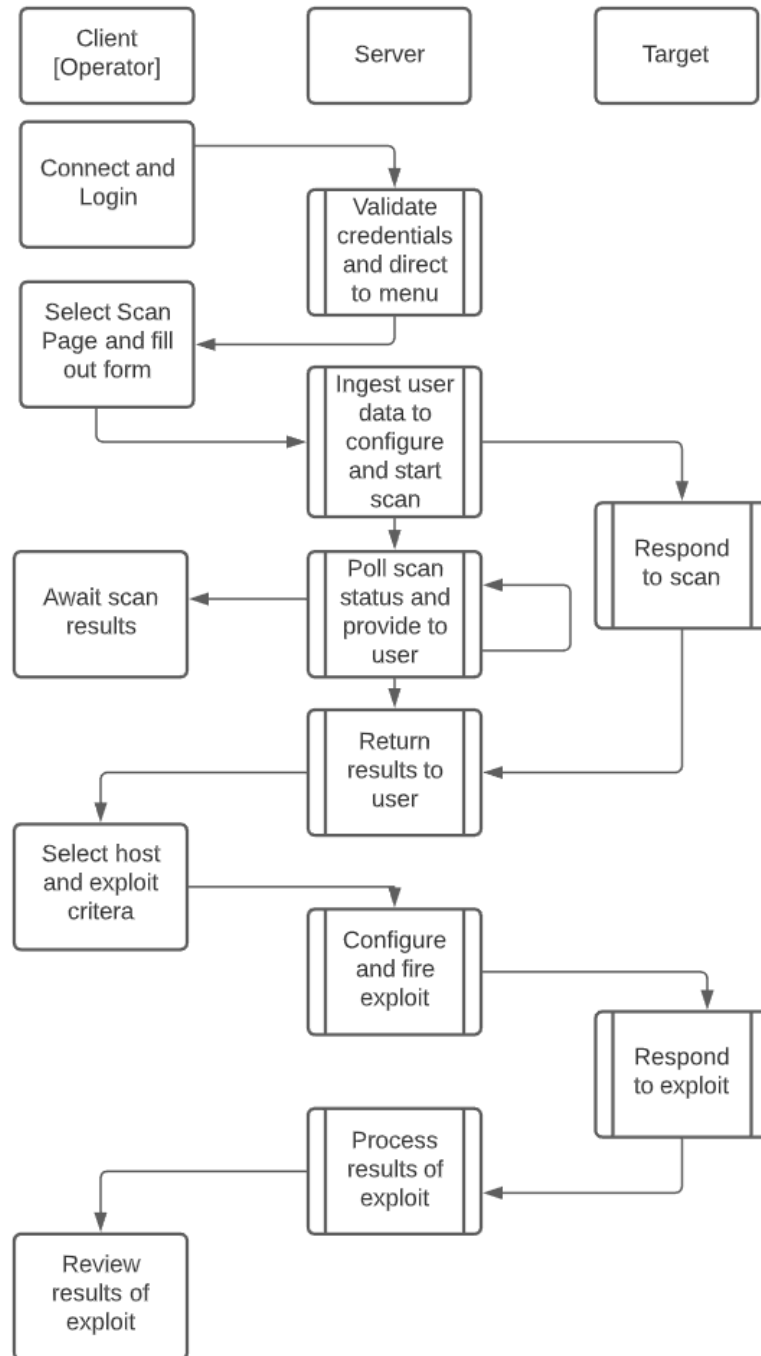


Figure 3. Process Flow of an Operator Performing a Scan

C. CARTT SERVER DESIGN

The CARTT Server consists of a PHP server, a series of PHP scripts, the OpenVAS and MSF programs (with related databases), a report folder, and a mySQL database. A diagram of the CARTT Server is shown in Figure 4.

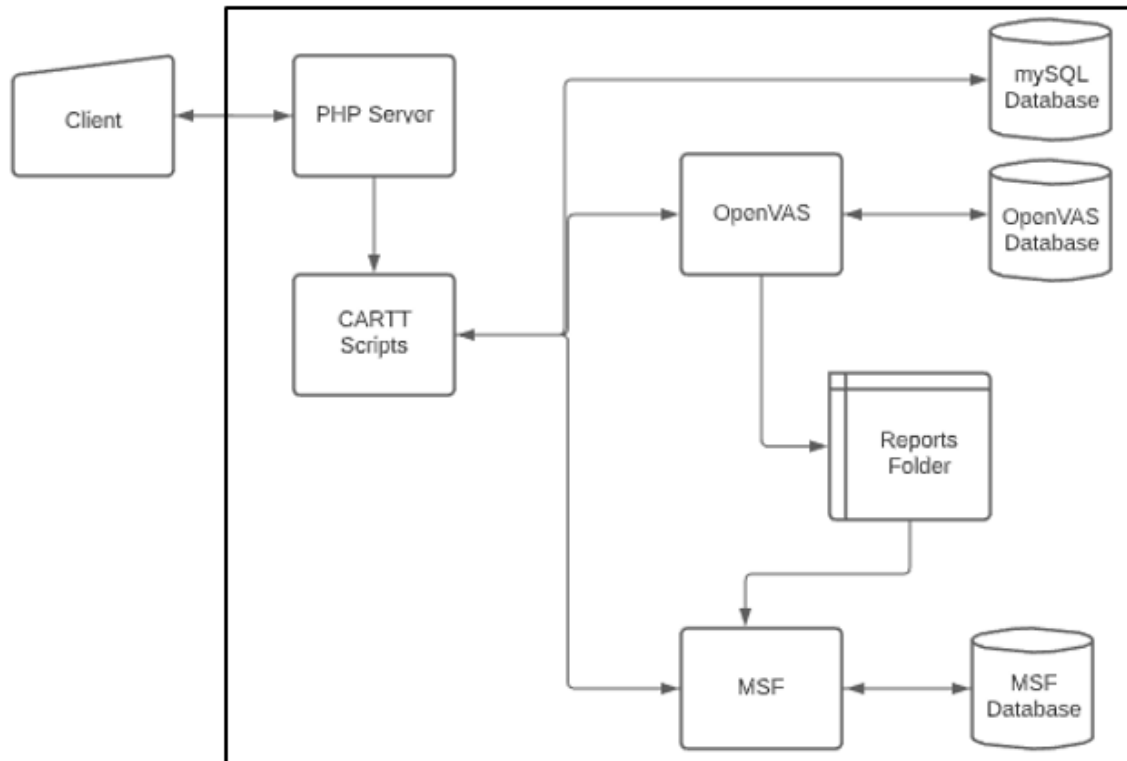


Figure 4. CARTT Server Diagram

The PHP server is the core of CARTT, which is essentially a series of PHP scripts. The scripts that take input and provide feedback to the user make up the Client. PHP was chosen for the scripting language and for the webserver because it is a free, server-side language that is platform-independent and well-documented online. PHP is an interpreted language, making it easy to implement, test, and revise code in real-time. Additionally, it has built-in features to handle dynamic content, integrate with various databases, and perform web security functions. Running PHP on the server side is essential, since the required programs are on the CARTT Server, not on the user's workstation.

OpenVAS is an open source vulnerability scanner provided by Greenbone Networks. It was selected for CARTT primarily due to its ability to integrate with MSF and its lack of cost. OpenVAS has a robust set of customization options, allowing the user to tune the vulnerability scan to their specific needs. OpenVAS also supports a variety of report formats. Additionally, Linux distributions such as Kali and Ubuntu maintain packages or come pre-installed with OpenVAS, making it easy to configure and run the program. OpenVAS uses a PostgreSQL or SQL Lite database to store user information, as well as scan, task, and target configuration, and report data. OpenVAS also uses a Redis database to store temporary data during scans.

CARTT uses MSF to parse the vulnerability report from OpenVAS, perform exploitation, and interact with the OpenVAS scanner. MSF is incredibly well-documented and widely used, which, along with the built-in integration with OpenVAS, were key factors behind the decision to use it in CARTT. Additionally, a free version of MSF is provided by Rapid7 which keeps the project at zero cost. MSF itself relies on a PostgreSQL database to store imported scan data, such as hosts and vulnerabilities, found in an OpenVAS scan.

The concept of a role-based system for CARTT users necessitates the ability for users to send messages amongst themselves, which requires some type of messaging database within CARTT. We chose MySQL for this because it is free, open source, well documented, works on numerous operating systems, and has robust PHP integration. There are also MySQL tools available that allow GUI-based management, simplifying the initial setup and configuration of the database. The MySQL database processes queries quickly and is designed to be scalable, both of which are essential for a wide deployment of CARTT.

In order to facilitate the flow of completed vulnerability scans from OpenVAS into MSF, CARTT instructs OpenVAS to save a local copy of the specified scan. CARTT then commands MSF to import the scan from the saved file. The scan is saved in a folder in the local directory, which allows CARTT to use relative paths to download and import the required reports.

1. Server Functions

In addition to housing OpenVAS, MSF, and the PHP server, the CARTT Server is required to provide systems for communication between CARTT users, administration of CARTT accounts, and security mechanisms.

a. Messaging System

In order to facilitate command and control between the Commander and Operator user roles, CARTT has a messaging system. This system allows users to communicate through the CARTT Server at the command level; there is no direct user-to-user messaging. Commanders can use the system to task subordinate commands to perform scans, report results, provide feedback, etc. Operators and Commanders are able to view messages sent to their command and to send messages, such as acknowledgements, status reports, and updates.

b. Administration

Administration of CARTT user accounts and permissions is handled by users logged into the Administrator role. These users can perform functions such as account unlock, password reset, account creation and removal, and role assignment. Additionally, Administrators manage the *commands* table in the database.

New users request an account by filling in a webform, including the requested role. Administrators then adjudicate the request and assign the user the appropriate role. Due to the difference in the functions of the roles, users will be presented with different web pages and CARTT capabilities based on their assigned role.

c. Security

In order to prevent unauthorized use, CARTT employs built in PHP tools to create and manage sessions, perform password hashing, and perform webpage redirects to prevent unauthorized users from accessing the system. User permissions are also validated to ensure that appropriate permission checks for valid users are performed, for example an Operator cannot access an Administrator page. Upon account creation user passwords are

salted and hashed before being stored in the mySQL database as a 61-character string. Additionally, input validation is performed on user provided text fields in order to mitigate various types of exploits, such as SQL injections.

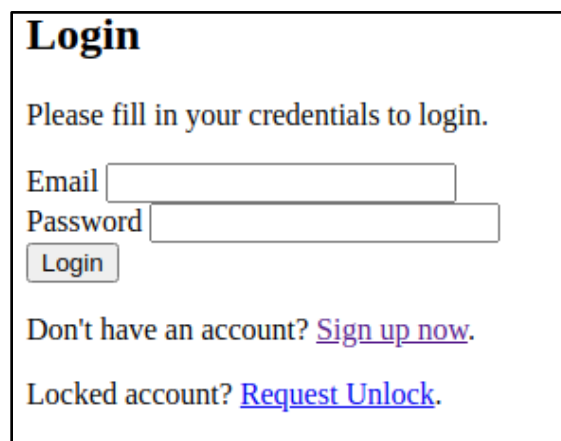
D. CARTT CLIENT DESIGN

As stated in Section B, CARTT has three client modes: Operator, Commander, and Administrator. Separating the users into the three roles makes it easier to conceptualize the relationships that exist between users at various levels in the CARTT Client. Upon logging in users are presented with the menu screen relevant to their assigned role.

In order to maintain simplicity, radio buttons are used whenever possible to allow the user to select their desired action. Text fields are used whenever user input is needed, and these fields are labeled and provide feedback if the user input is invalid or improper.

1. Login and Registration

In order to login to CARTT, a simple web form is provided, as shown in Figure 5. The web form asks for the user's email address that was used during registration and the password via two text fields. Feedback is provided to inform the user if the login attempt fails. If the user fails a defined number of login attempts, the account will be locked and the user redirected to a different page outlining the unlock process. The login page also allows the user to immediately jump to the unlock page.



Login

Please fill in your credentials to login.

Email

Password

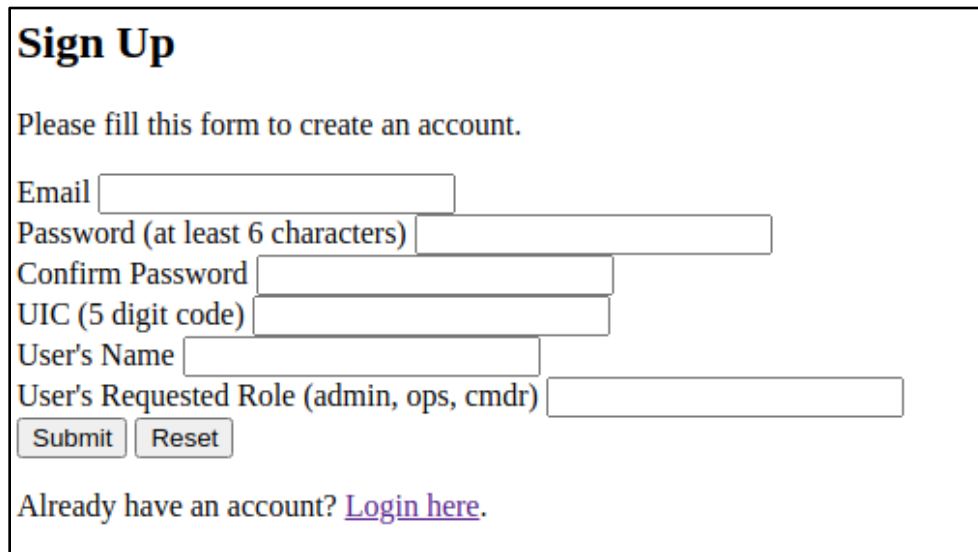
Login

Don't have an account? [Sign up now.](#)

Locked account? [Request Unlock.](#)

Figure 5. CARTT Login Page

The registration page, shown in Figure 6, is also designed with as much simplicity as possible. The fields are clearly labeled and provide feedback if the data entered does not conform to the requirements for that field (such as length or input type). The user is required to enter an email address, which will act as a unique identifier for their account. Email was chosen since every DoD employee has a unique, DoD provided email address. Passwords are required to be a minimum of twelve alphanumeric and special characters. Every command in the DoD has a unique Unit Identification Code (UIC), which is used by CARTT to understand the relationship between commands that is required for the Commander role to function. The user's full name is requested to make it easier to map the DoD email to a specific individual. The requested role field allows Administrators to properly adjudicate and assign the correct role(s) to the new account. This prevents users from directly assigning themselves a role inappropriately.



The screenshot shows a registration form titled "Sign Up" in a large, bold, black font. Below the title is a blue instruction: "Please fill this form to create an account." The form contains six input fields, each with a label to its left: "Email", "Password (at least 6 characters)", "Confirm Password", "UIC (5 digit code)", "User's Name", and "User's Requested Role (admin, ops, cmdr)". The "Email" and "UIC" fields are shorter, while the others are longer. Below the fields are two buttons: "Submit" and "Reset". At the bottom, there is a link: "Already have an account? [Login here.](#)" in blue text.

Figure 6. CARTT Registration Page

2. Commander Interface

The Commander role is designed for users that reside in higher headquarter commands. This role should not have the ability to conduct assessments directly, as its primary function will be command and control of Operator level users. Commanders have the ability to send and receive messages through the UI, which are viewable by all CARTT

users at the receiving command(s) and act as a means to convey tasking or share information. The Commander role shares the messaging interface with the Operator role, shown in Figure 7.

The messaging system allows the user (Operators and Commanders) to send messages to the desired command, identified by its UIC. The UIC is used as the command identifier since it is unique, does not change, and is easy to look up. If the UIC entered into the text field below is incorrect or not in the 'Command' table, the user will receive an error. The messages themselves are submitted by the user in a text box with a max length of 255 characters to keep the tasking and responses concise and easily digestible. The message system also enables the Operator or Commander to view all of the messages addresses to their UIC. Previous messages are displayed chronologically, followed by input fields for the Operator/Commander to create a new message.

Received Messages for command 11111
ID 15
Message: test
Sender: cartt-1
Sender UIC: 11111
Time: 2020-09-11 11:24:32
ID 23
Message: test
Sender: cartt-1
Sender UIC: 11111
Time: 2020-10-16 10:07:15
Message Creation
Receiving Command's UIC:
Message: (max 255 chars)

Submit

Return to Main Menu

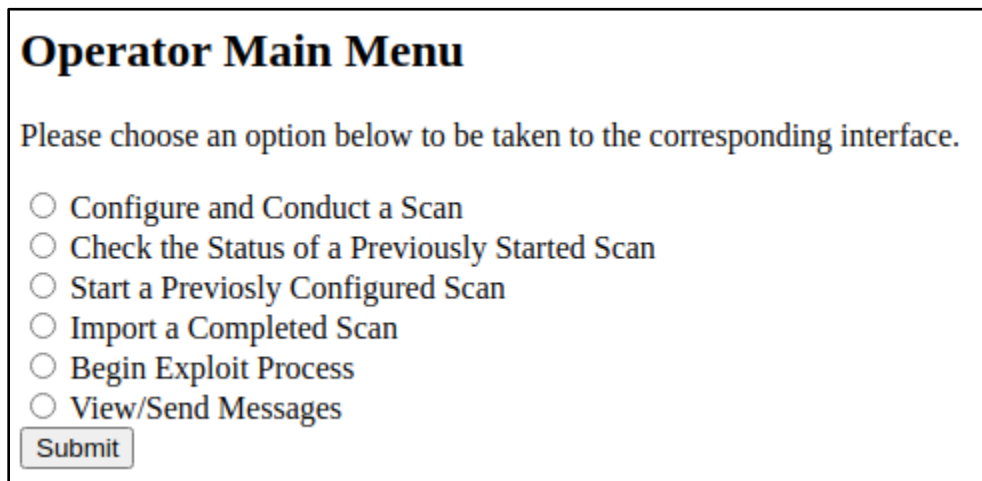
Figure 7. Messaging Interface

3. Operator Interface

The Operator role conducts most of the red teaming functionality in CARTT. This role allows users to perform assessments, fire exploits against the target network, and view and create messages to other CARTT Operators or Commanders. Upon logging in as an Operator, the user will be presented with a menu of task options, as show in Figure 8.

a. Operator Menu

Radio buttons are used in the main menu to ensure the Operator can only select one option at a time. The options are labeled so that Operators should easily understand the task that each represents, show in Figure 8. Additionally, the options are in the same sequence of how the Operator would conduct the tasks, making the menu similar to a checklist for the Operator. Pressing submit will redirect the Operator to the appropriate web page to perform the selected task.

The image shows a web interface titled "Operator Main Menu". Below the title is a prompt: "Please choose an option below to be taken to the corresponding interface." There are six radio button options listed vertically: "Configure and Conduct a Scan", "Check the Status of a Previously Started Scan", "Start a Previously Configured Scan", "Import a Completed Scan", "Begin Exploit Process", and "View/Send Messages". At the bottom left of the menu is a "Submit" button.

Operator Main Menu

Please choose an option below to be taken to the corresponding interface.

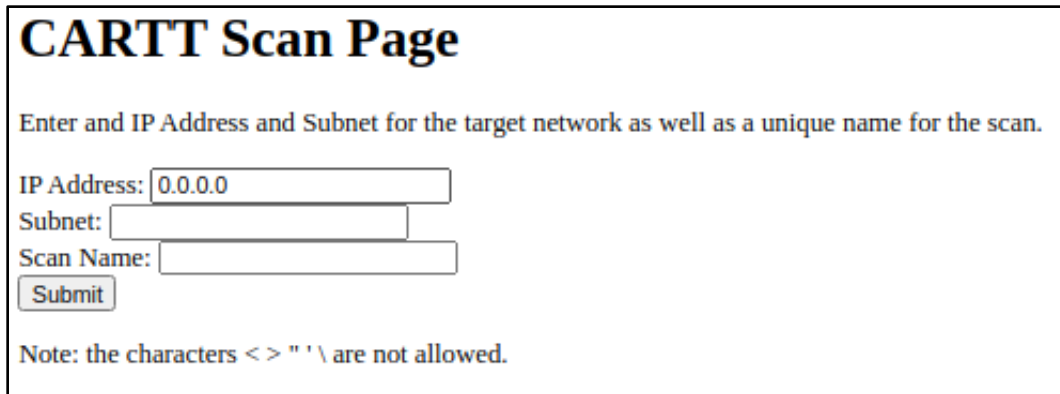
- ☐ Configure and Conduct a Scan
- ☐ Check the Status of a Previously Started Scan
- ☐ Start a Previously Configured Scan
- ☐ Import a Completed Scan
- ☐ Begin Exploit Process
- ☐ View/Send Messages

Figure 8. Operator Main Menu

b. Scan Configuration

The first option in the Operator main menu is the CARTT Scan Page, shown in Figure 9. It presents a series of text boxes that allow the Operator to input a desired target network, scan name, and workspace. It has the same principles applied to it as the registration page (i.e., the fields are clearly labeled and provide feedback if they are filled

out incorrectly), for example, if an invalid IP address is entered the page will update to display the message “<User Entered IP> is not a valid IP Address.” If invalid characters are entered in the Scan Name field, the Operator receives an error message to this effect, and is then able to enter new values into the webform. These scan input fields were chosen because they are the minimum information required by OpenVAS to perform.



The screenshot shows a webform titled "CARTT Scan Page". Below the title is a instruction: "Enter and IP Address and Subnet for the target network as well as a unique name for the scan." There are three input fields: "IP Address:" with the value "0.0.0.0", "Subnet:", and "Scan Name:". Below these fields is a "Submit" button. At the bottom, a note states: "Note: the characters < > ' ' \ are not allowed."

Figure 9. Scan Creation Page

c. Scan Status

The second option in the main menu allows the Operator to check the progress of a scan. This check is performed by scan name, which the user will provide in a simple text field, shown in Figure 10. This supports the Scan Import task (described in the next section) by allowing the user to verify that a previously initiated scan has completed and is ready for import. If CARTT finds the scan to be complete, it will notify the user and then redirect them to the Scan Import page after a 10 second delay. If the scan is not yet complete, its updated progress will be displayed on the page. In order to assist the Operator, a list of all the tasks and their current status are displayed at the bottom of the page.

CARTT Scan Status Page

Enter the name of a scan to check its progress if it is in progress.

Scan Name:

Submit

Note: the characters < > " ' \ are not allowed.

Task List:

Scan: 13octtest Status: Done
Scan: blah Status: Done
Scan: meow Status: Done
Scan: 15oct Status: Done
Scan: 13oct0002 Status: New
Scan: lemon Status: Done
Scan: test26oct Status: Done
Scan: apple Status: New
Scan: house_stark Status: Done
Scan: blackwidow Status: Done
Scan: ubuntu Status: Done
Scan: falcon Status: Done
Scan: scarlettwitch Status: Done
Scan: test Status: Done
Scan: 13oct0001 Status: Done
Scan: tyhn Status: Stopped
Scan: pineapple Status: Done
Scan: oct16scan Status: Done
Scan: blackpanther Status: Done
Scan: 13oct0003 Status: Done

Figure 10. Scan Status Page

d. Scan Start

The third option in the main menu allows the Operator to start a previously configured scan. This is intended to save time when repeatedly conducting scans on the same target network. This page provides the Operator with a list of all previously configured scans in OpenVAS as well as their associated ID numbers. The Operator can simply enter the desired scan Task ID into the text field for CARTT to initiate the scan. This page is shown in Figure 11.

CARTT Scan Start Page

Enter the ID Number of a previously configured scan to start it

Task ID:

Note: the characters < > " ' \ are not allowed.

Task List:

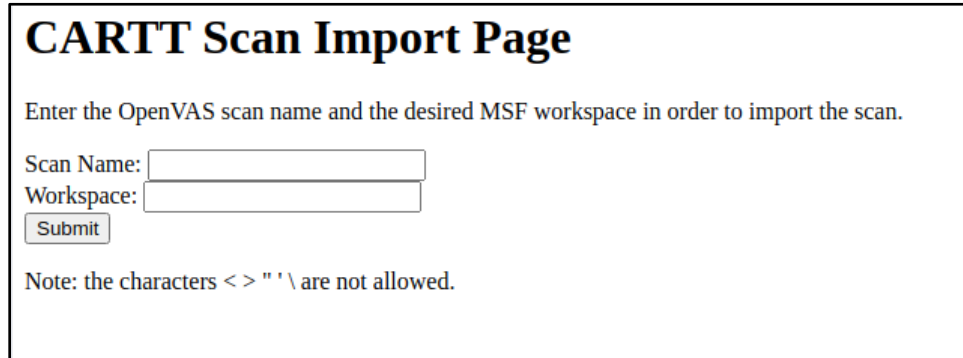
Scan: 13octtest Task ID: 16ec6ca3-65a4-4cbd-8e55-3e4d274b9825
Scan: blah Task ID: 1a8cc2e0-9fed-4874-a72f-797fe1437115
Scan: meow Task ID: 2547cae7-bddc-4d9c-b8cb-6e516cf8fc2c
Scan: 15oct Task ID: 2a5f22b8-a6ec-49ff-9237-ce7f8950b15a
Scan: 13oct0002 Task ID: 39bf7e71-ff7b-4caa-bcce-6b41895701dd
Scan: lemon Task ID: 4a290d68-3d73-40e8-8f5e-f32d69421306
Scan: test26oct Task ID: 4b71b703-3887-4e56-bec5-5675fbbc5284
Scan: apple Task ID: 60d95d48-be73-4424-8c63-7cafe6e8ad0a
Scan: house_stark Task ID: 6ee9549a-4c21-4daa-b723-3b965c052a52
Scan: blackwidow Task ID: 7d440b27-3aee-410c-bdc4-3d6d1a5e6c0b
Scan: ubuntu Task ID: 8db6221a-a96c-4078-b303-7bb09ae97a47
Scan: falcon Task ID: a94cc591-d993-4c37-8c8a-2a9c541e5353
Scan: scarlettwitch Task ID: b363f5d5-5302-45c1-b893-0ec61e428a4e
Scan: test Task ID: bf42e4f6-0e9f-4430-b62e-9ec2bb371677
Scan: 13oct0001 Task ID: c3dbbf28-755d-436d-8409-e4833d733fd5
Scan: tyhn Task ID: ca1ab1b1-742a-4d52-b338-47e4c384538a
Scan: pineapple Task ID: e6dcc24f-a03f-4436-8f47-ffe5a73b7e3b
Scan: oct16scan Task ID: f09311fc-a097-4958-b205-1e17d839cb11
Scan: blackpanther Task ID: f9f68f01-ae5b-4eee-93e4-a397d77140e2
Scan: 13oct0003 Task ID: ff35fe32-6280-47ca-8966-000a5636c0d7

Figure 11. Sample CARTT Scan Start Page

e. Scan Import

The fourth option in the main menu allows the Operator to import an existing OpenVAS scan into a designated MSF workspace, as shown in Figure 11. This task allows Operators to complete the scan import process if it was not completed previously or to set up a separate workspace using a previous scan. This functionality is necessary since large networks or a heavy load on the CARTT Server can cause vulnerability scans to take several hours. Additionally, allowing the Operator to specify their desired workspace, rather than having workspaces assigned by UIC, allows Operators at the same command

to work on scans of different target networks. This requires the Operator to know the exact name of a previously completed scan, which they can verify in the Scan Status Page discussed in Section C.



CARTT Scan Import Page

Enter the OpenVAS scan name and the desired MSF workspace in order to import the scan.

Scan Name:

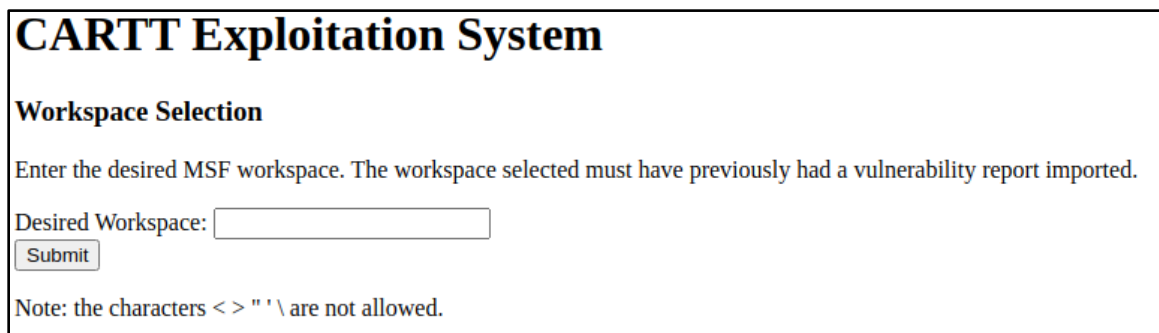
Workspace:

Note: the characters < > " ' \ are not allowed.

Figure 12. Scan Import Page

f. Exploitation System

The fifth option in the Operator main menu allows them to interact with a designated MSF workspace in order to configure and test an exploit. The Operator is prompted for the name of an MSF workspace that has previously had an OpenVAS scan imported into it, shown in Figure 13.



CARTT Exploitation System

Workspace Selection

Enter the desired MSF workspace. The workspace selected must have previously had a vulnerability report imported.

Desired Workspace:

Note: the characters < > " ' \ are not allowed.

Figure 13. Workspace Selection Page

The Operator is then presented with a list of hosts found in the workspace. The Operator is able to enter the IP address of one of those hosts into CARTT for further analysis. CARTT will display the list of vulnerabilities associated with the selected host shown in Figure 14.

Please enter the IP address of the target host.

Target Host IP Address

16 vulnerabilities identified for host 17.10.42.239

Mail relaying CVE-1999-0512,CVE-2002-1278,CVE-2003-0285

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

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

Figure 14. Vulnerability List for Host 17.10.42.239

This allows the Operator to select one vulnerability by its common vulnerabilities and exposure (CVE) number. The selected vulnerability will then be used to search the MSF exploit database, any exploit modules that might be compatible with the selected CVE are displayed for the Operator as shown in Figure 15. At this point the Operator can enter one of the displayed modules into the text field, which completes the configuration process for exploitation. The Operator will be redirected to the Exploit Results Page after a short delay.

Modules for CVE-2017-0143:

exploit/windows/smb/ms17_010_eternalblue

exploit/windows/smb/ms17_010_eternalblue_win8

exploit/windows/smb/ms17_010_psexec

exploit/windows/smb/smb_doublepulsar_rce

Copy and paste a module into the text field to construct an exploit

Module:

Figure 15. Modules Associated with CVE-2017-0143

CARTT attempts to exploit the target host using the Operator provided exploit module. CARTT provides the user with the raw feedback from MSF, allowing the Operator to assess the efficacy of the exploit. Figure 16 is the Operator's view after a successful exploit attempt that leveraged a PSEXEC module under CVE-2017-0143 against host 17.10.42.239.

```
Attempting to use exploit/windows/smb/ms17_010_psexec on host 17.10.42.239

Attempting Exploit.....

Results:

[*] Spooling to file user_data/results_qwert.txt...
resource (user_data/exploit_qwert.rc)> exploit -z
[*] Started reverse TCP handler on 17.10.42.120:4444
[*] 17.10.42.239:445 - Target OS: Windows 5.1
[*] 17.10.42.239:445 - Filling barrel with fish... done
[*] 17.10.42.239:445 - <----- | Entering Danger Zone | ----->
[*] 17.10.42.239:445 - [*] Preparing dynamite...
[*] 17.10.42.239:445 - [*] Trying stick 1 (x86)...Boom!
[*] 17.10.42.239:445 - [+] Successfully Leaked Transaction!
[*] 17.10.42.239:445 - [+] Successfully caught Fish-in-a-barrel
[*] 17.10.42.239:445 - <----- | Leaving Danger Zone | ----->
[*] 17.10.42.239:445 - Reading from CONNECTION struct at: 0x813c3b38
[*] 17.10.42.239:445 - Built a write-what-where primitive...
[+] 17.10.42.239:445 - Overwrite complete... SYSTEM session obtained!
[*] 17.10.42.239:445 - Selecting native target
[*] 17.10.42.239:445 - Uploading payload... SMdKKxXP.exe
[*] 17.10.42.239:445 - Created \SMdKKxXP.exe...
[+] 17.10.42.239:445 - Service started successfully...
[*] 17.10.42.239:445 - Deleting \SMdKKxXP.exe...
[*] Sending stage (175174 bytes) to 17.10.42.239
[*] Meterpreter session 1 opened (17.10.42.120:4444 -> 17.10.42.239:2176) at 2020-10-29 11:34:00 -0700
[*] Session 1 created in the background.
resource (user_data/exploit_qwert.rc)> exit -y
```

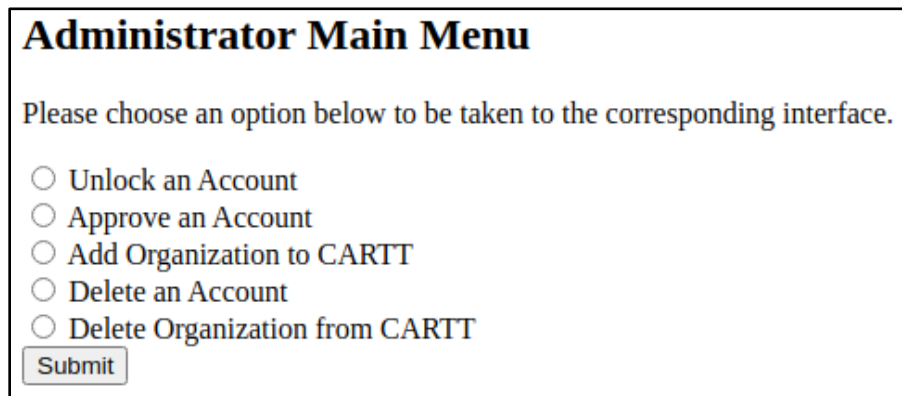
Figure 16. Results of PSEXEC Exploit on Host 17.10.42.239

g. Messaging

The final Operator main menu option allows CARTT messaging, and uses the same functionality and menu as discussed earlier in the Commander section (D.2) and shown in Figure 7.

4. Administrator Interface

Users in the Administrator role will be presented with a menu of radio buttons that will redirect them to relevant webpages for system and user administration of CARTT, as shown in Figure 17. The subsequent administration pages are laid out in a similar manner to the Operator pages, with a combination of clearly labeled radio buttons and text fields to improve ease of use.



The screenshot shows a web interface titled "Administrator Main Menu". Below the title is a prompt: "Please choose an option below to be taken to the corresponding interface." There are five radio button options listed vertically: "Unlock an Account", "Approve an Account", "Add Organization to CARTT", "Delete an Account", and "Delete Organization from CARTT". At the bottom left of the menu area is a "Submit" button.

Figure 17. Administrator Menu

E. CHAPTER SUMMARY

This chapter discussed the design methodology used to develop CARTT. It described the design requirements for the three CARTT user roles, Operator, Commander, and Administrator. It also defined the required processes for Administrators to manage the CARTT system and user accounts, for Operators to build, perform, and report the status of their cyber actions, and for Commanders to perform command and control over Operator actions. The next chapter discusses the implementation of CARTT, specifically how PHP is used to interact with MSF, OpenVAS, and MySQL in order to provide users with the features described in Chapter III.

IV. IMPLEMENTATION

This chapter discusses the implementation of CARTT. It describes in detail the underlying processes and mechanisms used by CARTT to manage users, support a messaging system, provide security, and interact with MSF and OpenVAS in order to conduct vulnerability scans and target host exploitation.

A. SECURITY IMPLEMENTATION

CARTT sessions are used to track logged in users and determine which menu pages and functions are accessible to the session client. Upon logging into CARTT the new session data contains the logged in user's CARTT role (stored as an integer between 0 and 4), email address, and a "loggedin" flag. When attempting to access any CARTT page, the CARTT Server checks if the "loggedin" flag is set for the current session; if not, the user is redirected to the CARTT Login page. If the "loggedin" flag is set, CARTT then checks the "role" field to confirm it matches the permissions for the requested page; if not, the user receives an error message. The error will prevent the rest of the PHP script from running, stopping the unauthorized user from being able to perform the related CARTT function(s). The implementation of this script is shown in Figure 18, where user roles 1, 2, and 3 have access to the page, but user roles 0 or 4 do not.

```
if(!isset($_SESSION["loggedin"]))
{
    header("location: login.php");
    exit;
}
else if($_SESSION['role'] == 4)
{
    echo 'Role not authorized';
    exit;
}
else if($_SESSION['role'] == 0)
{
    echo 'Role not authorized';
    exit;
}
```

Figure 18. Session Check Logic in message_page.php

B. ACCOUNT AND MESSAGING SYSTEM

1. Account Creation

The CARTT account and messaging system leverages PHP to interact with a mySQL database of user credentials. At account creation, CARTT queries the database to ensure that the user-provided email address is not already in the database; the user's email address must be unique since it the primary key for entries in the database's *users* table. This uniqueness check prevents multiple users from having the same login information and is performed with a simple mySQL select statement, as shown in Figure 19.

```
$sql = "SELECT email FROM users WHERE email = $email";
```

Figure 19. Select Statement to Validate Unique User Email

When requesting a new user account, the user is prompted to enter a password that is at least 12 characters; failing to enter at least 12 characters will return an error message. The user must also provide their email, name, UIC, and requested role in their request. Before being stored in the database as part of the *users*, the password is salted and hashed using the `password_hash()` function implemented in PHP. The user inputs, including the salted/hashed password and salt pair, are bound to variables and packaged into a mySQL insert statement. Since user passwords are not stored in plaintext, a leak or breach of the database would not result in a compromise of user passwords. The implementation code for registering a new account is shown in Figure 20.

```
$sql = "INSERT INTO users (email, UIC, rolereq ,username, carttpw) VALUES (?, ?, ?, ?, ?)";  
mysqli_stmt_bind_param($stmt, "sssss", $param_email, $param_uic, $param_role, $param_name, $param_password);  
  
$param_role = $role;  
$param_email = $email;  
$param_uic = $uic;  
$param_name = $name;  
$param_password = password_hash($password, PASSWORD_DEFAULT); // Creates a password hash
```

Figure 20. User Registration Password Hashing and Variable Binding

2. Account Management

All user accounts are implemented in a mySQL database as a table with the following columns: email, username, carttpw, role, rolereq, UIC, and is_locked. Email address is used as the primary key for each user, while username is a mandatory additional identifier to help differentiate similar email accounts (e.g., [jsmith@email.com](#) and [jsmith1@email.com](#)). As discussed in Section A, the carttpw column holds the hashed password and salt pair. For the role column, all accounts initially default to 0, which provides no CARTT functionality. The rolereq column represents a new user's requested role and is collected from the registration form. This data is used by a CARTT Administrator during the account approval process. The is_locked column is used to lock user accounts and is stored as a Boolean value. If the user account is locked (is_locked = True) the user will be unable to log into CARTT and will receive an error message. Finally, the UIC column represents the user's primary organization affiliation and is used by the messaging system.

CARTT Administrators can perform account management functions through their interface, as described in Chapter II Section C. All of their functions utilize mySQL queries to interact with the database, the functions are: approve_user, delete_user, unlock_user, add_organization, and delete_organization. The approve_user and unlock scripts both use update statements that change a value in *users*, role and is_locked respectively. The delete_user and delete_organization scripts use delete statements to remove accounts from the *users* table and organizations from the *commands* table, respectively. The add_organization script uses an insert statement to add a new organization to the *commands* table. The mySQL queries used by the CARTT Server to approve users, delete users, and add organizations are shown in Figure 21.

```
approve_user
    $sql_update = "UPDATE users SET role=$role WHERE email = '$email'";
delete_user
    $sql_update = "DELETE FROM users WHERE email = '$email'";
add_organization
    $sql_update = "INSERT INTO commands (UIC, title, UIC_superior) VALUES('$uic', '$title', '$uic_s')";
```

Figure 21. Sample CARTT mySQL Update, Delete, and Insert Statements

3. Messaging System

The CARTT Messaging system is designed to allow asynchronous communication between organizations, identified by their UIC. The system is dependent on three tables in the MySQL database: *users*, *commands*, and *messages*. As discussed in the previous section, *users* contains all the required information for the user's account to function.

Messages has five columns: id, content, TxUIC, RxUIC, and timestamp. The id is the table's primary key, and is tracked and incremented by the database. It is used to organize messages sequentially, and to deconflict any timestamp collisions. The content column contains the body of the message and is implemented as a 255-varchar. The TxUIC and RxUIC columns correspond to the transmitter's UIC and receiver's UIC, respectively. The timestamp is generated by CARTT immediately before the insert statement is constructed and sent to the database.

Commands has two columns: UIC and title. The UIC column is used as the primary key for the table and represents the messaging organization. The title column is used to store the recognizable organization name. When creating a message, the UIC that is entered into the "Receiving Command's UIC" text field is compared to *commands* and if there is not a match, the message is rejected and an alert is provided to the Operator or Commander. This ensures that messages are not being sent to commands that are not a part of CARTT.

When an Operator or Commander logs into the Message page they are shown all messages received by their organization, i.e., where the *message* RxUIC columns matches the UIC of the session. Once they have crafted their message and it passes the RxUIC validation check, the message is created using an insert statement. Figure 22 shows examples of how some of the various queries used to support messaging are implemented. In the first line the statement that is used to select all of the messages to with the RxUIX column set to the same value as the UIC in the Operator or Commander's session. The second line of code is how CARTT pulls the list of UICs from the database to ensure that the RxUIC the Operator or Commander entered is valid. The final code segment shows how the timestamp is created and the insert statement is packaged using the data provided by the Operator or Commander as well as their UIC from the session.


```

pull messages to user's UIC
$sql = "SELECT * from messages WHERE RxUIC = '$uic_sender'";
validate receiving UIC is in table
$sql_valid_uic = "SELECT UIC FROM commands WHERE UIC = ?";
message creation
$timestamp = date("Y-m-d H:i:s");
$sql = "INSERT INTO messages (content, TxUIC, RxUIC, TIMESTAMP)
VALUES ('$message', '$uic_sender', '$uic_to', '$timestamp')";

```

Figure 22. Sample Queries Used by the Messaging System

C. CARTT USE OF METASPLOIT AND OPENVAS

The Metasploit Framework (MSF) is a powerful cyber red teaming tool suite used by CARTT to conduct vulnerability analysis, through interaction with OpenVAS, and to exploit target hosts. All CARTT scripts that interact with MSF, except for `exploit_test`, invoke it by using PHP's `proc_open()` function, since it allows CARTT to establish two-way communication with the forked MSF process. The `proc_open()` function is passed an array of file descriptors that designate the standard input (stdin) and standard output (stdout) for the new instance of MSF. A PHP function `msf_connect()`, shown in Figure 23, was written to handle the creation of new MSF processes. The function returns an array that stores the new process ID and communication pipes. MSF is then invoked using the `-q` option, which suppresses the ASCII splash screen that MSF displays upon startup.

1. Interacting with OpenVAS

a. OpenVAS Connection

CARTT does not interact with OpenVAS directly, but through MSF, which has prebuilt functions for using the MSF interactive console to perform OpenVAS functions. CARTT uses the communication pipe established by PHP, discussed in Section C, to send MSF instructions to load and connect to OpenVAS with the credentials provided. These credentials are created when OpenVAS is first configured on the CARTT Server and are shielded from the CARTT Client. After CARTT sends the command to connect to OpenVAS, it parses the stdout from MSF, searching for the string "OpenVAS connection successful." The first nine lines provided in stdout are discarded as the success or fail messages from MSF always occur after them. If CARTT does not find the success string

in stdout, it throws an exception and informs the Operator that the connection failed. Figure 24 shows the PHP code that enables CARTT to load and connect to OpenVAS. The two variables \$openvasID and \$openvasPW are the OpenVAS credentials mentioned earlier and are hard coded within CARTT.

```
function msf_connect()
{
    $fd = array(
        0 => array('pipe', 'r'), // stdin
        1 => array('pipe', 'w'), // stdout
    );

    // command for msf
    $cmd = "msfconsole -q";

    // spawn the process and check it was successful
    $msfc = proc_open($cmd, $fd, $pipes);

    if(!is_resource($msfc))
    {
        throw new Exception("Failed to open MSF");
    }

    //must return array, [0] = msf handle, [1] = $pipes array
    $ret_val = array($msfc,$pipes);

    return $ret_val;
}
```

Figure 23. Msf_connect() Function used by CARTT to Create MSF Processes

```
$load_openvas = "load openvas\n";
$connect_openvas = "openvas_connect $openvasID $openvasPW 127.0.0.1 9390\n";

fwrite($pipes[0], $load_openvas);
fwrite($pipes[0], $connect_openvas);

for($i = 0; $i<9; $i++) //read the 9 response lines
{
    $last_line = fgets($pipes[1]);
}

$pos = strpos($last_line, "OpenVAS connection successful");
if($pos === false)
{
    ?><h3>OpenVAS Failed to Connect, try again.</h3><?php
    throw new Exception("Failed to connect");
}
```

Figure 24. Commands Used to Load and Connect to OpenVAS Inside MSF

b. OpenVAS Targets and Tasks

Once a connection to OpenVAS has been established inside of MSF, CARTT uses OpenVAS-specific MSF commands to create, configure, start, and monitor the vulnerability scan requested by the Operator. First, the Operator provided scan name, IP range, and subnet are packaged together to create the target list for the scan, using `openvas_target_create`. Next, `openvas_task_create` is used to create a task in OpenVAS with the same scan name, the newly created target list, and a scan profile. OpenVAS has six predefined scan profiles; among these, CARTT is currently hard-coded to use the “Full and Fast” scan option, although future work could allow the CARTT Operator to select this. After the task has been created, MSF tells OpenVAS to start the scan by calling `openvas_task_start`.

To track the progress of the scan, CARTT sends MSF the `openvas_task_list` command every three seconds and uses the output from this command to provide the Operator with the completion percentage for the scan. A function called `openvas_task_status()` was created to handle this process. It parses every line of stdout from MSF for any instance of the word “Done” and the name of the selected scan. If the scan name is found and “Done” is not seen, then the function tokenizes the string to extract the completion percentage and task status. If “Done” is found on the same line as the scan name the function returns an integer value (-10) to signal the scan is complete.

Figure 25 shows how the OpenVAS commands are sent to MSF. `$pipes [0]` refers to the stdin for the process, while variables `$targetID` and `$taskID` are generated by OpenVAS after the target list and task are created; they are captured from the stdout of MSF and fed back into MSF in order to construct the scan.

```

target creation
$openvas_target = "openvas_target_create $scan_name $ip$subnet none\n";
fwrite($pipes[0], $openvas_target);

task creation
$openvas_task_create = "openvas_task_create $scan_name 'none' $full_fast_ult $targetID\n";
fwrite($pipes[0], $openvas_task_create);

task start
fwrite($pipes[0], "openvas_task_start $taskID\n");

task status
fwrite($pipes[0], "openvas_task_list\n");

```

Figure 25. Implementation of Target Creation, Task Creation, Task Start, and Task List Commands

c. *OpenVAS Reports*

The final CARTT interaction with OpenVAS is for its report download function. CARTT uses the MSF command `openvas_report_download` to have OpenVAS download the report as an XML file – named for the scan that was used – to the reports folder in the CARTT directory. OpenVAS generates a unique `$reportID` for every completed scan, which is required for the `openvas_report_download` function. CARTT collects the `$reportID` information by parsing the output of the `openvas_report_list` function and searching for the same scan name. The PHP code used to download the OpenVAS scan report and import it into an MSF workspace are shown in Figure 26.

```

$workspace_change = "workspace -a $workspace\n";
fwrite($pipes[0], $workspace_change);

$XML = "a994b278-1f62-11e1-96ac-406186ea4fc5"
$openvas_import = "openvas_report_download $reportID $XML reports $report_name\n";

$msf_import = "db_import reports/$report_name\n";

fwrite($pipes[0], $openvas_import);
fwrite($pipes[0], $msf_import);

```

Figure 26. Example MSF Workspace Change/Creation, MSF Report Import, and OpenVAS Report Download

d. MSF Workspace

The MSF commands `workspace -a` and `db_import` are used by CARTT to instruct MSF to import the recently downloaded scan report into the workspace designated by the Operator; if the named workspace does not exist, it will be created. The Operator can choose to work in a new or existing workspace, however once a report has been uploaded to a specific workspace, MSF makes no distinction as to what scan discovered the host or vulnerability. Due to this, it is highly recommended that each target network has its own workspace.

2. MSF Exploit Construction

To begin the exploitation process in CARTT a vulnerability scan must have been imported into the Operator selected workspace. If there are no hosts or vulnerabilities found in that workspace, the Operator will be presented with an error message. For the exploitation process, CARTT creates several local files in which to store the hosts, vulnerabilities, and exploit modules. These files are named using the Operator's emailID and as such are unique. Figure 27 shows how CARTT creates the `hosts_$user.txt` file, and a similar process is used for the vulnerabilities and exploit modules text files.

```
//grab logged in user ID to create/clear storage file
$user = $_SESSION['email'];
$host_file = "hosts_$user.txt";
$fd = fopen("user_data/$host_file", 'w+');

//close file
fclose($fd);

//change workspace
$workspace_change = "workspace -a $workspace\n";
fwrite($pipes[0], $workspace_change);

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

//get host IPs
fwrite($pipes[0], "hosts -c address\n");

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

Figure 27. How CARTT Creates the `hosts_$user.txt` File

Once the Operator selects a workspace the hosts and vulnerabilities files are created using the `spool` command to direct MSF output to a text file, then CARTT uses the MSF `hosts` and `vulns` commands to generate the list of hosts and vulnerabilities, respectively. Once a target host is selected, CARTT parses the vulnerabilities file to find all the matching items and returns those to the Operator. The exploit module file is created in the same manner with the exception that the MSF search function is used to get results related to the selected vulnerability. The PHP code for the search command is shown in Figure 28. For ease of use, CARTT requests the Operator to enter a CVE number, however other values can be entered that could result in valid exploit modules being found. For example, “CVE-2017-0143” refers to the Blue Keep exploit family, so searching for “Blue Keep” or “CVE-2017-1043” would return a similar list of modules.

```
//build search command
$search_string = "search type:exploit $cve\n";

//search for exploit modules that match CVE number
fwrite($pipes[0], $search_string);
```

Figure 28. MSF Search Command

After the Operator enters the selected module into the text field, CARTT builds a script file that will be executed by MSF. The file uses the `.rc` file format and is passed to MSF as an argument with the `-r` option on the command line, as shown in Figure 29. This is the only time in CARTT that MSF is invoked using the PHP `exec()` function; in every other instance, `proc_open()` is called due to the need for two-way communication.

```
//build the .rc script
$script = "$exploit\n
    set RHOST $host\n
    color 'false'\n
    set ExitOnSession true\n
    spool user_data/results_$user.txt\n
    exploit -z\n
    exit -y\n";
fwrite($fd_rc, $script);
fclose($fd_rc);

// command for msf to open and import created rc file
$cmd = "msfconsole -q -r user_data/exploit_$user.rc";

//run msf with rc file
exec($cmd);
```

Figure 29. Construction of Exploit Script and Execution of MSF with the Script

The script uses the set command to select the exploit module and the target host, as well as telling MSF to exit the session upon connection to the target. Color output is disabled in order to make the output from MSF more readable since color encoding adds a lot of unnecessary clutter to the output. An example script is shown in Figure 30.

```
use exploit/windows/smb/ms17_010_eternalblue
set RHOST 17.10.42.239
color 'false'
set ExitOnSession true
spool user_data/results_qwert.txt
exploit -z
exit -y
```

Figure 30. Contents of Exploit Script for Target 17.10.42.239 Using Exploit Module ms17_10_eternalblue

D. CHAPTER SUMMARY

This chapter covered the implementation of CARTT as a Client/Server model. It included a detailed explanation of the processes, functions, and systems leveraged by CARTT in order to provide user account, message, vulnerability scan, and exploit systems.

In the next chapter, we discuss the conclusions and future work.

V. CONCLUSIONS AND FUTURE WORK

A. SUMMARY

The goal of this research was to extend the capabilities of CARTT in order to transform it into a client/server model while also adding user security features and a command-and-control interface. For this work, an Ubuntu 20.02 virtual machine (VM) was configured to act as the CARTT Server; the VM contains MSF, OpenVAS, a MySQL database, and a PHP server. CARTT was constructed using a series of PHP scripts running on the CARTT server. Users interact with CARTT through a web browser client on any remote device, while the Server interacts with the internal processes for MySQL, OpenVAS, and MSF.

To ensure separation of duties in CARTT, users are separated into three distinct roles: Operator, Commander, and Administrator. Commanders and Operators are able to communicate with each other at the organizational level using the CARTT Messaging interface. Further, Operators are able to use the vulnerability analysis and exploit testing features of the CARTT service. Administrators are able to approve new account requests, delete and unlock existing user accounts, and add or delete organizations from CARTT.

To test the end-to-end functionality of CARTT, we demonstrated that a user logged in with the Operator role was able to configure a vulnerability scan, start the vulnerability scan on a remote target host, check the status of the vulnerability scan, import a previously completed vulnerability scan from OpenVAS into MSF, and to perform exploit configuration and execution based on the results of the vulnerability scan. Due to the highly automated nature of CARTT, the Operator was only required to fill in simple text fields at every step, often using information provided by CARTT feedback from previous actions. We also successfully demonstrated that Operators could view and create messages from/to other CARTT users that are logged in as Operator or Commander.

This research has demonstrated that we were able to bridge the gap between vulnerability scans and red team assessments for the DoD, while reducing the cost and barrier to entry of such a capability. CARTT accomplished this using open source software

and by automating the underlying processes required to perform vulnerability scans and cyber exploitation.

B. CONCLUSIONS

CARTT uses open source software to automate vulnerability scanning of a target network and exploitation of a target host. This provides the CARTT Operator with actionable information on whether a detected vulnerability is actually exploitable. It also provides a command-and-control channel for CARTT Operators and Commanders. Additionally, CARTT provides user authentication and a role-based user system, with user and organization management handled by the CARTT Administrator.

The open source and automated nature of CARTT will allow the DoD to widely deploy the system with little overhead and minimal training. While CARTT is not intended to replace a red team assessment, it will reduce the workload on DoD Red Teams by empowering local network defenders to perform automated red teaming tasks. CARTT will allow local defenders to better assess their cybersecurity posture and manage risk, improving DoD cybersecurity.

This research answered the following questions:

1. Primary Question

How can CARTT be transformed from a stand-alone application into a system that employs a client/server model?

We documented the capabilities of the previous version of CARTT, including host discovery, vulnerability scanning, vulnerability analysis, and exploit analysis. We constructed a VM with OpenVAS, MSF, MySQL and a PHP server as our baseline architecture. We then created a series of PHP scripts to handle the interaction between the Users, MSF, OpenVAS, and MySQL that were handled by a python script in the previous version of CARTT. We then demonstrated that an Operator could use the browser interface to conduct a vulnerability scan (which includes host discovery), view the results of this scan, select a target host, select an exploit, and be provided with feedback from CARTT on the efficacy of the exploit. In our system testing, the target device was a Windows XP

Service Pack 3 virtual machine, and the exploit chosen was CVE-2017-143, which relates to the *BlueKeep* family of exploits. The MSF exploit module chosen for this vulnerability used `ps_exec`, and was determined to be successful after a meterpreter session was established on the target host.

2. Secondary Questions

How can CARTT employ a command-and-control channel between its Operators and Commanders?

We used a mySQL database to manage user accounts and messages between CARTT Operators and Commanders. We created a database table, *messages*, to hold the required information for the message: content, message sender (TxUIC) and receiver (RxUIC), ID number, and timestamp. A second database table, *commands*, was created to store the relationship and title of organizations. At message creation, *commands* is checked to verify the receiving command exists in CARTT. The result was that Operators and Commanders are able to send and receive messages between one another at the organizational level from within CARTT.

How can CARTT Operators use the system to remotely conduct red teaming analysis of a network to which they are not attached?

We tested CARTT by logging into the service from a browser opened on Windows 7 and Windows XP devices, and by attempting to configure and conduct a vulnerability scan to perform an exploit on a target host. There was no difference in CARTT's performance between the client devices. This indicates that CARTT Clients should be able to target networks remotely from any browser-capable device. However, due to time constraints we did not set up further test environments to verify this capability.

C. FUTURE WORK

This research focused on the initial steps for transforming CARTT into a multi-user client/server system. Further work should focus on expanding CARTT's capabilities to more closely align with red team operations and improving the user experience. The following recommendations for future work will further those goals.

1. Improve the User Interface

The CARTT Client is functional but spartan, so future work should focus on improving the user experience. Every CARTT page is rendered in a browser using a simple white background with black text, and there is nothing to highlight particularly relevant or important information. In addition, information is displayed as lists of text, which can require the user to have to scroll through a lengthy list of information to find a required input field. Additionally, the messaging system is basic and provides no way to notify users that a message has been received. It also displays received message with no way to prioritize or sort them.

2. Advanced User Mode and Post-exploitation Capabilities

CARTT currently only attempts an exploit in order to determine if it is possible to launch an associated payload against the target host. Any shell session created by the payload between the CARTT Client and the target are terminated as soon as they are established. This was an intentional implementation choice to mitigate the risk of an Operator accidentally carrying out an attack on the target, however, experienced Operators should be provided additional capability within CARTT to take actions on a target after it has been successfully exploited. Future work should focus on creating an Advanced Operator role that allows greater customization of scans and exploits, as well as creating a post-exploitation system for such experienced users.

3. Improve Logging and Record Keeping

CARTT stores vulnerability reports, hosts, vulnerabilities, the exploit script and exploit results. Hosts, vulnerabilities, the exploit script and exploit results are currently stored with the Operator's email as their identifier, this means that CARTT only saves the most recent version, overwriting the files every time an action is taken. Future work to alter the naming convention of these files can create a historical log of the Operator actions. Adding to this, CARTT could then be able to retrieve and display this information, allowing Operators and Commanders to review the logs.

LIST OF REFERENCES

- [1] P. Edwards, "Cyber automated red team tool," M.S. thesis, Dept of Comp Sci, NPS. Monterey, CA, 2019. [Online]. Available: <https://calhoun.nps.edu/handle/10945/64145>
- [2] J. Plot, "Red team in a box (RTIB): Developing automated tools to identify, assess, and expose cybersecurity vulnerabilities in Department of the Navy systems," M.S. thesis, Dept of Comp Sci, NPS. Monterey, CA, 2019. [Online]. Available: <https://calhoun.nps.edu/handle/10945/62832>
- [3] Cyber Degrees, "Penetration tester career overview" 3 March 2020. [Online]. Available: <https://www.cyberdegrees.org/jobs/penetration-tester/>
- [4] Core Security, "Core Impact," 8 May 2020. [Online]. Available: <https://www.coresecurity.com/core-impact>
- [5] 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," Washington, DC, USA, DoDIG Report No. DoDIG-2020-067, 2020. [Online]. Available: <https://www.DoDig.mil/reports.html/Article/2114391/followup-audit-on-corrective-actions-taken-by-DoD-components-in-response-to-DoD/>
- [6] Director, Operational Test and Evaluation, "FY 2019 annual report," Washington, DC, USA, 2019. [Online]. Available: <https://www.dote.osd.mil/Publications/Annual-Reports/2019-Annual-Report/>
- [7] C.T. Chaplain, "Weapon systems cybersecurity DoD just beginning to grapple with scale of vulnerabilities," Washington, DC, USA, GAO Report No. GAO-19-128, 2018. [Online]. Available: <https://www.gao.gov/products/GAO-19-128>
- [8] M. Johnson, "DISA seeks automated tools to identify potential cyber threats, conduct analysis," 20 May 2019. [Online]. Available: <https://disa.mil/NewsandEvents/2019/automated-tools-cyber-threats-analysis>
- [9] C. Craft, "Fight the DoDIN," May 2019. [Online]. Available: https://www.disa.mil/-/media/Files/DISA/News/Events/Symposium-2019/1---COL-Craft_Fight-the-DoDIN_approved-Final.ashx
- [10] DISA, "JFHQ DoDIN capabilities," 2019. [Online]. Available: <https://www.disa.mil/-/media/Files/DISA/Fact-Sheets/DISA-Capabilities.ashx?la=en&hash=058FFF28911BFA504B699B2D3642AB82C292F482>

- [11] DISA, “Cyber awareness challenge 2020,” 2020. [Online]. Available: <https://public.cyber.mil/training/cyber-awareness-challenge/>
- [12] S. Piedfort, SSC Atlantic Red Team: The good ‘bad guys’, Navy Information Warfare Systems Command, 8 March 2018. [Online]. Available: <https://www.public.navy.mil/navwar/News/Pages/NNS180308-12.aspx>
- [13] IBM, “Cost of a data breach report 2020,” 2020. [Online]. Available: <https://www.ibm.com/security/data-breach>
- [14] Offensive Security, “Penetration testing with Kali Linux,” [Online]. Available: <https://www.offensive-security.com/pwk-oscp/>
- [15] Global Information Assurance Certification, “Cybersecurity certifications: pricing,” [Online]. Available: <https://www.giac.org/certifications/pricing>
- [16] Payscale, “Average penetration tester salary,” 2020. [Online]. Available: https://www.payscale.com/research/US/Job=Penetration_Tester/Salary
- [17] Indeed, “How much does a penetration tester make in the United States?,” 2020. [Online]. Available: <https://www.indeed.com/career/penetration-tester/salaries>
- [18] Silent Break Security, “Silent break central,” 2020. [Online]. Available: <https://silentbreaksecurity.com/silent-break-central/>
- [19] Silent Break Security, “Red team toolkit” 2020. [Online]. Available: <https://silentbreaksecurity.com/red-team-toolkit/>
- [20] Scythe, “What is Scythe?,” 2020. [Online]. Available: <https://www.scythe.io/platform>
- [21] Randori, “Randori launches automated ‘attack platform,’ Industry’s First SaaS Solution to Bring Elite Red Team Experience to the Mass Market,” Feb 11, 2020. [Online]. Available: <https://www.randori.com/press/launching-attack-platform-elite-red-team/>
- [22] M. Aharoni, D. Kearns, D. Kennedy, J. O’Gorman, *Metasploit the Penetration Tester’s Guide*. San Francisco, CA: No Starch Press, Inc, 2011.
- [23] G. Weidman, *Penetration Testing a Hands-on Introduction to Hacking*. San Francisco, CA: No Starch Press, Inc, 2014.
- [24] J. Chan, *Learn PHP in One Day and Learn it Well*. Hedge End, Hampshire, UK: LCF Publishing, 2020.

INITIAL DISTRIBUTION LIST

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