2019-12

CYBER AUTOMATED RED TEAM TOOL

Edwards, Preston L.

Monterey, CA; Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

CYBER AUTOMATED RED TEAM TOOL

by

Preston L. Edwards

December 2019

Thesis Advisor: Gurminder Singh
Co-Advisor: Alan B. Shaffer

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2019	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
CYBER AUTOMATED RED TEAM TOOL

5. FUNDING NUMBERS

6. AUTHOR(S) Preston L. Edwards

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

The Cyber Automated Red Team Tool (CARTT) is designed to address the shortage within the Department of Defense (DoD) of trained red teams that are able to conduct security assessments of cyber systems. CARTT, implemented in software, simulates the actions of a red team by automatically identifying and analyzing vulnerabilities in computer systems, and then exploiting those vulnerabilities with cyber-attack actions. It then presents the user with a summary of the results after it conducts its assessment. An ongoing project at NPS has developed the first version of a CARTT, which can conduct vulnerability assessments of computers. This research has extended the capability of the CARTT graphical user interface (GUI), and software to enable an operator to select and automatically execute scripted red team attacks against specified targets to achieve intended cyber effects. The automated nature of these attack scripts allows their use by operators who do not have extensive training in offensive cyber operations (OCO) or red teaming.

14. SUBJECT TERMS	**15. NUMBER OF PAGES**
automation, CARTT, cyber, cyber automated red team tool, cyber physical systems, cyber security, cyber systems, offensive cyber operations, penetration testing, red teaming, vulnerability assessment	71
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

CYBER AUTOMATED RED TEAM TOOL

Preston L. Edwards
Lieutenant, United States Navy
BS, University of Georgia, 1998
MBA, Hawaii Pacific University, 2011

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 2019

Approved by: Gurminder Singh
Advisor

Alan B. Shaffer
Co-Advisor

Peter J. Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The Cyber Automated Red Team Tool (CARTT) is designed to address the shortage within the Department of Defense (DoD) of trained red teams that are able to conduct security assessments of cyber systems. CARTT, implemented in software, simulates the actions of a red team by automatically identifying and analyzing vulnerabilities in computer systems, and then exploiting those vulnerabilities with cyber-attack actions. It then presents the user with a summary of the results after it conducts its assessment. An ongoing project at NPS has developed the first version of a CARTT, which can conduct vulnerability assessments of computers. This research has extended the capability of the CARTT graphical user interface (GUI), and software to enable an operator to select and automatically execute scripted red team attacks against specified targets to achieve intended cyber effects. The automated nature of these attack scripts allows their use by operators who do not have extensive training in offensive cyber operations (OCO) or red teaming.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AEPT	Automatic Executing Penetration Testing
CARTT	Cyber Automated Red Team Tool
CIDR	Classless inter-domain routing
CLI	Command Line Interface
CVE	Common Vulnerabilities and Exposures
DoD	Department of Defense
DoN	Department of the Navy
DOT&E	Director of Operational Test and Evaluation
FY	Fiscal Year
GAO	Government Accountability Office
GSA	Greenbone Security Assistant
GUI	Graphical User Interface
HM&E	Hull-Mechanical-Electrical
IP	Internet Protocol
IT	Information Technology
JTAG	Joint Test Action Group
Lhost	Local Host
MSF	Metasploit Framework
NIPR	Non-Secure Internet Protocol Router
NIST	National Institute of Standards and Technology
Nmap	Network Mapper
NPS	Naval Postgraduate School
NSA	National Security Agency
OCO	Offensive Cyber Operations
OpenVAS	Open Vulnerability Assessment Scanner
OTA	Operational Test Agency
PTES	Penetration Testing Execution Standard

Rhost	Remote Host
SBIR	Small Business Innovation Research
TCP	Transmission Control Protocol
Tkinter	Tk interface
TTPs	Tactics Techniques and Procedures
UART	Universal Asynchronous Receiver-Transmitter
UML	Unified Modeling Language
USB	Universal Serial Bus
XML	Extensible Markup Language

ACKNOWLEDGMENTS

I would like to thank my wife, Sameckia, for her encouragement and support. I would also like to thank our daughters, Morgan and Briana. Special thanks to my advisors, Dr. Gurminder Singh and Dr. Alan Shaffer, for taking me under your wing. Lastly, thanks to the Navy for providing the opportunity to attend the Naval Postgraduate School.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The Department of Defense (DoD) relies on many advanced automated systems to operate in environments that are increasingly threatened by cyber-attacks. These advanced systems are also increasing in complexity to meet the performance requirements of the DoD force. Information must travel almost instantly from the decision makers to the execution officers in each battle domain. As a result, the required automation and interconnectivity of DoD computer systems to enhance performance has also increased the attack surface of these systems.

Cyber threats against these systems also continue to increase. With the interconnectivity of networks, threat actors can seek entry into DoD networks by secondary and tertiary methods (e.g., commercial network service providers or third-party vendors). Nation-states like China and Russia continue to be a persistent cyber threat to DoD networks and systems in search of vulnerabilities to exploit. These countries, among others, continue to sharpen their cyber skills and are increasingly becoming more difficult to defend against. As a result, DoD must remain proactive against cyber threats by discovering and addressing vulnerabilities in systems. These systems must be tested throughout the procurement process and after being deployed in operational environments.

The increased use of automated cyber systems has also increased the demand for red teams to perform vulnerability assessments against these systems. The DOT&E Cybersecurity report for 2018 emphasized the need for "all systems that transmit, receive, or process electronic information" to receive operational tests and evaluations. The report also noted that "the increased demand coupled with the increase in data from the tests is stressing the test community's cybersecurity resources." The DoD has acknowledged the shortage of red team experts to conduct such assessments [1].

A further challenge is that many of the automated systems are not directly connected to the Internet. Thus, assessing a particular system across the Internet would not be feasible. The DoD would need to deploy red teams to onsite locations for vulnerability assessments of its cyber systems. This strain on already limited expertise and resources is

not maintainable. Additionally, the time required to recruit and train new personnel at a level expected of red teams (e.g., formal classroom and on-the-job training) can take years.

A low-cost portable cyber tool designed to perform cyber security vulnerability checks by a novice user would be ideal. Such a tool would have the capability to connect to systems regardless of location. A cyber tool with the capability to simulate cyber-attacks would serve as an additional resource to address the rapid increase in demand. To address the shortage of red team experts, the tool also should have an interface that provides feedback so that it can be operated by a non-expert user. The training time required to employ this tool would be measurably less compared to traditional red team training. An added benefit is that local DoD sites can implement a continuous and repeatable strategy for assessing the vulnerability of their cyber systems. Our goal is to develop a system that can conduct red team assessments during development, operational testing, and normal field operations of DoD systems.

A. PROBLEM STATEMENT

Hypothesis: The reach of Department of Defense (DoD) Red Teams can be significantly expanded by implementation of a portable automated cyber exploitation tool developed for non-experts to conduct red teaming activities against computer systems not directly connected to an external network.

1. Primary Question

How can the capability of the current Cyber Automated Red Team Tool (CARTT) be extended to enhance simulated Red Team attacks against specified computer systems not directly connected to an external network?

2. Secondary Question

How can the CARTT tool be designed so that an operator with minimal training use it to conduct red teaming activities?

B. METHODOLOGY

We will review commercially available and open source tools such as the Metasploit Framework (MSF), Core Impact, and Network Mapper (Nmap) to evaluate their individual capabilities in exploiting computer system vulnerabilities (e.g., mapping the system, identifying the operating system, finding vulnerabilities), and to test whether they can be used for CARTT.

CARTT will incorporate open source software that executes scripted cyber-attacks against a targeted system. The process will not require extensive interaction by the operator. The interface design will provide the user with easy to understand graphical user interface (GUI) elements to execute commands throughout the process. CARTT will provide the user with a list of vulnerabilities and the associated targets. Additionally, a window element will display a menu of exploit modules resulting from a selected vulnerability by the user. Another window element will provide the non-expert user with a description of a selected exploit module. This is so the user may determine if the exploit will have the desired cyber effect. A cyber-attack is conducted if the user selects the button element to initiate the attack. A window element will provide the user with the status of the cyber-attack.

C. SCOPE

The goal of CARTT is to implement a portable system that can automate an entire cyber-attack scenario (i.e., cyber reconnaissance, vulnerability analysis, cyber-attack, and assessment) against computer systems not directly connected to an external network. The results can then be interpreted and understood by a non-expert operator. This thesis will focus on the development and execution of the cyber-attack phase. Additionally, the GUI design will be enhanced to accommodate the extended cyber-attack capability of CARTT. The intent is to provide a limited proof-of-concept tool that can behave as a portable CARTT. Accessing a computer system from an external network or executing multiple attack vectors at once is outside the scope of this thesis.

D. BENEFITS OF STUDY

This tool will enable the DoD to address the increased demand for cyber Red Teams by enabling non-expert operators who do not have extensive training in Offensive Cyber Operations (OCO) or Red Team operations to perform continuous assessments of critical networks and legacy systems. CARTT is a cyber tool that will enable a vulnerability assessment strategy that supplements red teams. This will increase the chance for discovering known and potential (zero day) vulnerability threats. Additionally, red teams would have more time to focus, plan and portray the capabilities of the advanced persistent threat actors.

E. THESIS ORGANIZATION

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

(1) Chapter II: Background

Chapter II provides the current state of operational test and evaluation challenges of DoD systems and current penetration testing policies. It examines how DoD systems are becoming more interconnected and complex, and how these complexities affect cybersecurity. Next, current software tools used to exploit vulnerabilities of target systems and previous CARTT research are covered. Lastly, a brief overview of the CARTT architecture is provided.

(2) Chapter III: Design Methodology

Chapter III describes the CARTT design, to include the Metasploit framework, and discusses the CARTT process flow, from importing of vulnerability reports to execution of a cyber-attack against a specific target. The intuitive nature of the interface design and the decision behind expanding the GUI are also presented. Finally, GUI operations are described and demonstrated in this chapter.

(3) Chapter IV: System Implementation and Testing

Chapter IV presents the implementation of the CARTT proof-of-concept and the test results of a cyber-attack against a known vulnerable target. The backend applications

like Python and Metasploit are discussed in more detail regarding how CARTT executes command requests by the user. The TKinter application is discussed including design decisions for future expansion capability. Next, the steps to initialize the Metasploit framework within Kali Linux for testing and executing a selected exploit in CARTT are presented. Finally, the test results from the experiment are presented.

(4) Chapter V: Conclusion and Future Work

Chapter V summarizes the research efforts (e.g., expanded capability and GUI design) of this thesis. Conclusion to the research questions are provided along with successes and limitations of the research. Finally, future work recommendations are provided to further improve CARTT capability and performance.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

Cyber Warfare is generally accepted as its own warfighting domain. The Department of Defense (DoD) is taking steps to ensure cybersecurity is not an afterthought when it comes to protecting computers, networks and weapon systems. These systems are more integrated and networked in order to enhance the warfighting effort. This interconnectivity requires that the systems are resilient when faced with cyber-attacks from an adversary. The systems need to be constantly assessed before, during and after operationally deployed and used by forces in the field. While the DoD has made significant improvements with respect to ensuring its systems, challenges and threats still remain from advanced persistent adversaries around the world. This chapter presents background relevant to this research in cyber security technologies, tools and techniques.

A. CURRENT STATE OF DOD OPERATIONAL TESTING AND EVALUATION

In 2004, the National Security Agency (NSA) and the military cyber centers developed a program to train and certify DoD red teams [2]. This was an attempt to address the limited resources available for conducting cybersecurity assessments on DoD assets. During the same year, the military service Operational Test Agencies (OTAs) formed a working group to identify inexpensive cybersecurity toolkits. Ten years later, in 2014, The OSD Director of Operational Test and Evaluation (DOT&E) revised procedures for cybersecurity OT&E of acquisition programs to standardize the assessment process [3]. The DoD uses cyber red teams to assess the vulnerability of systems during the development and operational test phases of the systems acquisition process. DOT&E's 2013 Cybersecurity report found that "the majority of cybersecurity problems identified during operational testing in FY13 could have been uncovered and resolved in early phases of development and testing" [4]. The report also highlighted that many vulnerabilities are found well after testing and during fielded operations. Assessments for systems during normal fielded operations usually happen once a year, often during a major exercise.

DOT&E noted in 2018 that "the demand for cyber expertise to plan and execute cyber assessments across the DoD, and for the in-depth analyses of the data produced by these events, is rapidly increasing and stressing available resources." Increased spending on development of new cyber connected systems only exacerbated this problem. For example, DoD spent over a trillion dollars to develop and maintain its weapons systems in fiscal year 2017 [5]. And there is no indication that spending on such systems in the future will decrease. These systems are expected to operate as designed and when needed. However, with increasing automation and connections to networks, DoD weapon systems are not safe from cyber-attacks. Hence, the demand for red teams will remain strong into the foreseeable future. To address this increasing need, the Department of the Navy (DoN) posted a Small Business Innovation Research (SBIR) topic in 2018 for a software tool that could simulate red team actions in the assessment of DoD/DoN systems [6].

B. DEPARTMENT OF DEFENSE WEAPON SYSTEMS AND NETWORKS

The DoD must continue to develop new capabilities to address evolving threats around the world. As new technologies are developed, the complexity of these systems presents several issues to the DoD, as described in this section. Additionally, the military (e.g., Combatant Commanders [COCOMS]) deploys these complex systems in joint military and coalition environments. The requirement for these systems to communicate with one another increases the risk for cybersecurity vulnerabilities. The challenge is to mitigate vulnerabilities before they are leveraged by adversaries.

1. Complexity and Interoperability

Weapon systems and networks continue to increase in complexity to meet performance requirements. Weapon systems may contain multiple smaller embedded systems (e.g., radar system, communication system, and flight system). These embedded systems come with their separate programming coded and rely on various software to integrate the physical and logical control requirements. For example, if the radar system on a Naval vessel detects an inbound threat, it may communicate a recommended countermeasure against the threat. Connecting weapon systems to other computer systems is a result of attempts to address the demand for more information within shorter decision

cycles. Due to their interconnectivity, therefore, weapon systems have become vulnerable to cyber-attacks. The same is true for networks where DoD contractors connect to these networks for information sharing.

Furthermore, many DoD weapon systems and networks are supported by "commercial and open source software" [5]. The reliance on commercial software ensures that future systems and networks will continue to increase in complexity due to interoperability issues between different applications and protocols, which can lead to network vulnerabilities. The DoD contracts with various vendors to procure hardware and software for mission requirements. Additionally, it is known that DoD also employs software that may no longer be supported by the vendor; for example, Microsoft Windows XP installation on the Navy's Aircraft Carrier Gerald Ford (CVN–78) is no longer being actively supported by Microsoft [7]. The 2012 DOT&E Information Assurance and Interoperability report noted that many interoperability issues are not reported, and that many users of such systems usually find "workarounds" to continue to be able to accomplish the mission [8].

Various software and protocol updates can also lead to new cyber vulnerabilities in a weapon system or network. These complexities highlight the need for and usefulness of an automated cyber tool that can test for such vulnerabilities.

2. Factors Affecting Cybersecurity

a. Automation and Connectivity

Automation and connectivity are two factors that have an impact on the cybersecurity of DoD weapon systems and networks. The military has enjoyed many technological advances that provide an advantage over adversaries. However, a DoD Defense Science Board report in 2013 stated that automation and connectivity are vulnerabilities that allow an adversary to conduct an asymmetric attack [9]. Many systems are connected to each other for speed of data and information exchange. This connectivity makes it possible for an advance adversary to gain initial access to one system on a network and potentially pivot to another system on the same or different network. Further, automation is risky because adversaries can inject corrupt data into a system that may

produce an incorrect response or reaction due to a lack of or improper security checks in the system. Adversaries have access to many commercially available tools that can take advantage of this automation and connectivity issue. The more advance threat actors may develop their own tools to exploit such vulnerabilities, as well. These factors will continue to play a role in how DoD address cybersecurity.

b. Lack of Prioritization

According to the Government Accountability Office, cybersecurity was an after-thought by the DoD regarding weapon systems. The focus was more on Information Technology networks. Acquisition program officers and managers did not understand that cybersecurity needed to be a priority for weapon systems [5]. As result, the Defense department is facing the interoperability challenges mentioned earlier among other factors. Cybersecurity of a weapon system needs to be addressed at the beginning of development and continue throughout the testing, operational and fielding phases of acquisition.

c. Cyber Red Team Resources

The DoD continues to demonstrate that overall cyber readiness is an important aspect of defending the nation against cyber threats. The president's fiscal year 2020 budget includes a request for $9.6 billion for the DoD to spend on cybersecurity. However, a very small percentage appears to be allocated to support cyber red team efforts. For example, less than 1% ($75 million) of the fiscal year 2019 budget was allocated toward cyber tools, training. Furthermore, Cyber Command still needs to fill 40% of the positions at its headquarters to support this effort.

The increase in demand for cybersecurity testing has also highlighted the shortage of cyber personnel with the required expertise. This shortage is a result of the DoD having to compete for talent with the civilian sector. Military personnel with cyber expertise can obtain a higher salary by working for industry or contracting with the federal government to perform the same cybersecurity services. Defense department leadership does not have the authority to increase wages or offer higher salaries to servicemembers that have these skills in such a high demand area. Of note, the 2016 National Defense Authorization Act granted DoD the authority to recruit and retain a Cyber Command Workforce by

streamlining the civilian hiring process [10]. The Director Operational Test and Evaluation, during the same year, stated the quality of red teams needs to be addressed through recruitment and retention efforts [11]. Nonetheless, the department has been slow to execute this authority (due to personnel shortages).

Training and skills development are other factors affecting the cybersecurity of DoD systems. Each military service within the DoD provides formal classroom instruction on cybersecurity operations to its servicemembers. Following formal instruction, servicemembers are expected to fulfill assignments (sometimes referred to as payback tours) throughout the DoD. While in these assigned billets, servicemembers also receive on-the-job-training, which helps reinforce the formal instruction and increase efficiency in the red team performance. Unfortunately, the return on this investment may not manifest itself until three to five years, which often exceeds the length of the payback assignment [12]. There is a general understanding that servicemembers' proficiency and knowledge improves over time in a particular assignment. However, the Director Operational Test and Evaluation noted in 2017 that military members do not spend enough time in a particular job to provide the needed continuity and experience in cybersecurity operations [13]. In fact, the average tour length for Sailors and Marines is roughly three and a half years [14]. After that time servicemembers are required to rotate to another job assignment.

3. Access Interface Types

Another factor affecting the cybersecurity of DoD networks and weapon systems is that automation of these systems requires that they become increasingly reliant on connectivity to other devices. The Government Accountability Office noted that DoD weapon systems are designed with various types of interfaces [5]. Some of these interfaces are noticeable upon a physical inspection while others are not so noticeable. Interfaces can be internal and external to the weapon system and can be implemented in hardware (Universal Serial Bus and removable storage) or software (applications and programs). The more interface points a weapon system has, the greater the potential access points an adversary may use to disrupt the integrity of the system. Cybersecurity red teams need to have knowledge of the various types of access interfaces.

C. PENETRATION TESTING

Cybersecurity of networks and weapon systems involves a continuous assessment process and strategy. One such strategy is penetration testing which assesses the cyber-security and resiliency of a system. It allows authorized personnel to actively try and gain access to a particular system. The organization and penetration testers will agree to predetermined rules such as which networks and systems are off limits for the event. Penetration testing, however, is not an activity that should be considered in isolation. It should be part of an organization's comprehensive strategy to assess the cyber-security of a network. This includes passive vulnerability scans as well. The penetration test can be automated or conducting manually.

Red Teaming is another method in which a system's cyber-security is assessed. Red Teaming and penetration testing are similar in nature. Both are part of a comprehensive agenda to test the overall vulnerability of a computer system or network. Penetration testing probes for additional vulnerabilities that may not have been previously discovered. Red Teaming however takes the additional step of creating system effects from a specific adversary's perspective. Both serve as a validation tool for proving that a security issue is real [15]. Paul Paget (Core Security Technologies Chief Executive Officer) and Ron Gula (former Tenable Network Security Chief Executive Officer) stated that penetration testing helps demonstrate the "implication of an intrusion" [16]. For example, an initial vulnerability assessment may identify a vulnerability, however, penetration testing will highlight any system impacts. In other words, if the vulnerability is exploited, does the system behaves differently. A more important implication may be the type of additional access gained by the adversary by exploiting the vulnerability. In this manner, penetration testing is used to confirm or validate that a previous vulnerability that was discovered by a passive scan does have an impact on the security of a system. Another added benefit is that penetration testing can also test the resiliency of a network or computer system. For example, if a system detects an intrusion, does it shut down or continue to operate? If the penetration test injects malware, the system may crash and need to be rebooted. On the other hand, the system may continue to operate in a degraded mode until the restoration is possible. Testing may reveal anomalies that were not present before the testing began.

Aside from vulnerability confirmation, penetration testing can provide some insight into how an attacker may gain access to the system. With such insight, cybersecurity efforts can focus on patching and closing those vulnerabilities discovered during penetration testing. By implementing penetration testing as part of the overall cybersecurity strategy, leadership can prioritize which vulnerabilities to address first and which vulnerabilities present no obvious damage or threat to the system.

Penetration testing does have risks associated with its use as well. The major risk associated with penetration testing is the lack of confidence in the test's ability to not inflict damage upon the system being tested. This is another similarity shared with Red Teaming. Director of Operational Test and Evaluation noted that Combatant Commanders usually place restrictions on red team assessment because of perceived safety and security concerns related to the networks [11]. Leaders need to become comfortable with penetration testing and red team assessments. This will require continuous education for non-cyber experts on the benefits of penetration testing.

1. DoD Penetration Test Policy

The DoD updated its cybersecurity procedures in 2014 to include forms of Black and White box testing throughout the acquisition phases of a system's development [3]. The two assessments types are 1) Cooperative vulnerability and penetration assessment and 2) Adversarial assessment. The Cooperative vulnerability and penetration assessment are designed to identify errors and vulnerabilities in coordination with the system's program manager and design engineers. During this phase, vulnerabilities can be identified and corrected early before reaching the operational and fielding phases. This is a version of White box testing because assessors require in depth knowledge of the system being tested. Thus, full access to the source code is needed in order to conduct the evaluation. Conversely, the Adversarial assessment takes place in an operational environment, where the assessors are conducting the evaluation from the end users' perspective. The assessment not only evaluates if the system operates as designed but can it withstand attacks from an adversary. This process resembles the Black box testing technique. Simply put, the DoD's

white and black box testing phases are designed to test the systems in the pre-operational and post-operational phases, respectively.

Resources and planning, however, still impact the red teams' evaluation and testing process. The Director Operational and Test and Evaluation found that some programs are unable to conduct the white box testing needed in the earlier phases of development to find and correct errors. The result is a more compressed timeline making it difficult for red teams to perform a thorough analysis of the system during its operational and fielding phases. The DoD will benefit from a cyber automated red team tool that can help address some of the resource constraints.

D. RED TEAM TOOLS

There are several cybersecurity tools available in the commercial market. However, most of these tools are designed to remotely test systems over an Internet IP-based outward-facing network interface. Furthermore, these tests are mostly focused on enterprise information technology (IT) networks, not weapon or other specialized systems. Some of these tools offer an open-source platform to support additional code development. This feature can enable red teams to incorporate some of the tool's tactics, techniques, and procedures into their testing and evaluation of DoD systems. These cyber tools may also allow red teams to automate many of the cyber assessment techniques used to perform OT&E. Red teams may also customize the exploit code to address DoD specific system cyber-attack concerns. This section discusses some commercially available and open source tools that automate cyber-attacks, such as Cobalt Strike, Metasploit and Core Impact. A more comprehensive list can be found in Plot's thesis [17].

1. Cobalt Strike

Cobalt Strike is a web application software simulation tool used to emulate adversarial attacks against a specific target. It has a graphical user interface that displays the sessions and targets to the user in an easy to understand fashion. Users would need to familiarize themselves with the toolbar to better facilitate adversary emulations. This tool offers red teams a full range of techniques to select from when designing an attack scenario. Such techniques include spear phishing and embedding malicious code inside a document

as well as red team collaboration capabilities. The red team collaboration allows distributive operations across servers. Team members can execute a phishing campaign from one server location that references a malicious website at another server location. A drawback of this feature is that each server operates independently during the engagement. Post engagement reporting is consolidated at the end of the event to provide a complete picture of the scenario. Another benefit of Cobalt Strike is that it leverages a Cobalt Strike technology known as "Malleable C2" [18]. Malleable C2 allows the red team to alter behavior and indicator signatures while conducting cyber-attacks. This prevents cybersecurity defenders from using known signatures to quickly identify the type of cyber-attack emulated by the red team. However, the red team can use Malleable C2 to slightly alter a known signature to test how quickly the malicious attack is noticed. The focus of Cobalt Strike is to train and enhance the incident response of an organization's Enterprise IT systems. The initial version of Cobalt Strike was built on top of the Metasploit Framework (discussed below) however, later versions were separated from Metasploit. Cobalt Strike comes at a cost of $3,500 per user for one year.

2. Core Impact

Core Impact is another web-based automated cyber tool that provides vulnerability scanning, penetration testing and after-action reports. Though this cyber tool has a friendly user interface, it requires the users to be skilled in conducting penetration testing. It emulates adversary cyber-attacks through its multi-vector capability. An "agent" tunnel is installed on the target providing remoted access to the system or network. For example, if a client on a network opens an attachment inside of an e-mail, the malicious code calls back to Core Impact establishing the agent tunnel. These connection tunnels can be encrypted but are established in the clear by default. Exploits are executed in Python code and are also modifiable by the user. Core Impact allows the user to execute attacks via a "drag-and-drop" interface [19]. This process increases the number of cyber-attacks that can be conducted in a short time period. Red teams are able to pivot between various applications, (e.g., web and email). This tool also allows cyber testers to escalate privileges and cover their tracks after the attack. Systems are returned to their previous state. Agents can be programmed to self-destruct after penetration testing is completed. This option prevents

inadvertently leaving a back door for a real adversary to gain access to a system or network Core Impact is regarded as the top exploitation tool on the market. It is also among the most expensive tools costing $30,000 per year [20].

3. Canvas

Canvas by Immunity Inc. is an automated penetration test scripted in Python. Its users must have an extensive knowledge of penetration testing and exploitation techniques. During testing, a listener shell is established on the target to receive further exploitation commands. Canvas has a feature called “Most definitely” (Mosdef), which is a dynamic shellcode generator [21]. This feature allows penetration testers the ability to pivot between various host on a network regardless of the different operating systems used [22]. Canvas has over 800 exploits but provides limited vulnerability scanning capability. Further, it only conducts scans for IP addresses on a given network. Canvas documentation states that the platform was designed to facilitate the development of other security products. The platform’s MOSDEF session also allows red teams to develop other exploits or payloads to attacks systems [23]. Canvas is well-known for providing the ability for 3rd party exploitation add-ons.

4. Metasploit Framework

Metasploit (also known as Metasploit Framework) is an automated open-source software platform available to penetration testers. A major benefit of Metasploit is that it contains an array of automated modules designed to execute cyber-attacks. Metasploit modules contains various software written to execute a particular function (or purpose) when deployed against a system or network. The five types of modules are: 1) Exploit, 2) Auxiliary, 3) Post-Exploitation, 4) Payload or 5) No Payload [24]. Exploit modules are designed to target specific vulnerabilities and gain access. Auxiliary modules perform functions such as scanning or denial of service. The post exploitation modules further enumerate the target trying to find other avenues to exploit. Exploitation can be executed in automatic or manual mode. This gives the user the option of attacking multiple vulnerabilities at once or one vulnerability at a time [24]. Another feature of Metasploit is Meterpreter. Meterpreter is an advance payload that establishes a command shell and

executes inside of memory. This feature makes detecting the Metasploit session harder for intrusion detection applications [24]. With Metasploit, penetration testers are able to pivot into other processes or sessions to maintain a persistent presence on a system. The Metasploit open-source framework is very popular in the cyber development community. Penetration testers can modify existing modules or developed new modules in an effort to identify vulnerabilities in a system or network. The Metasploit Framework is universal in that many security platforms such as Cobalt Strike, Core Impact and Canvas, among others, can use Metasploit modules as part of their penetration test events. There are also commercial editions of Metasploit beginning from $5,000 per user per year [25].

These adversarial simulation tools provide a variety of techniques that may be used to assess DoD systems. However, they all assume that the user has extensive knowledge of exploits, adversarial attacks, and penetration testing.

E. RELATED RESEARCH

Research teams have been experimenting with various models and techniques to automate all or portions of penetration testing. One benefit of automation is that it can significantly reduce the amount of time needed to conduct an assessment of a system. For example, evaluators will not spend time on rote and tedious task. More time will be dedicated toward discovering the more difficult vulnerabilities and assessing the system impact of those discoveries. This section presents various approaches by researches to automate penetration testing.

1. Penetration Testing in a Box

In 2015, a research team from Northern Kentucky University designed a security assessment tool to address the high cost of conducting vulnerability assessments and to streamline the process. The researchers acknowledge that penetration testing can be a complicated endeavor depending on the complexity of a network. And if an organization wanted a more detail assessment, then the price tag would inherently increase [26].

The “penetration architecture” that the researcher proposes consist of a Pentest Box that is positioned behind the firewall of the organization, a computer on the outside of the

firewall connected by a Virtual Private Server. The Virtual Private Server will allow for the viewing of the vulnerability data via a Secure Shell callback from the Pentest Box to the computer located outside of the organization's firewall [26]. The vulnerability data is accessible via a web application hosted on the Pentest Box. The research team relied on opensource software as part of the Kali Linux suite (i.e., the Metasploit Framework (MSF), Network Mapper (Nmap), and Open Vulnerability Assessment System (OpenVAS)) to conduct the experiment.

The researchers pointed out that the limitations of the Pentest Box is based on available resources in terms of hardware, configurations and budgets. The objective of the box also factors into its design: 1) only conducts vulnerability scans, 2) cyber (exploitation) attacks included in the assessment, and 3) what procedure are automated during the assessment [26].

2. Automated Network Intrusion Process

Researchers at the Technical Educational Institute of Crete demonstrated how combining opensource tools can be used to automate a cyber-attack across the internet [27]. For the demonstrations, the researchers used the MSF, Nmap, and Python to automate the process. They showed how an attacker could either initiate an attack on a single target or on multiple targets at once "under certain circumstances," (e.g., the system or network has to have a valid vulnerability).

The experiment was designed with a computer system vulnerable to a Metasploit module. The vulnerability (CVE 2009–4188) consisted of a hardcoded account located on the 5.5 Tomcat server. The exploit uploaded a JavaServer Page via an http PUT request [27]. This action provided a remote shell back to the automated system. The experiment highlighted that two Metasploit modules were needed to gain access to the server. After gaining access, the researches established a Meterpreter shell session, which allowed them to maneuver away from the previously compromised service onto the operating system.

3. Automated Attack Model for Red Teams

Three graduate students from the Florida Institute of Technology published an article titled “Toward an Automated Attack Model for red teams” analyzing various cyber-attack models used by red teams [28]. The authors described Red Teaming as a necessary process for assessing the vulnerabilities of a system. Plus, it is important to understand how a potential adversary may seek to gain entry into a connected system. One issue with red teams is that there is no universally excepted method for assessing vulnerabilities within a system. Further, the type of red team method is more determined by the requirements and terms set by the organization being assessed. Four cyber-attack models were presented by the authors: 1) Threat modeling, 2) Attack trees, 3) Collaborative attack modeling and 4) Insider threat model.

The Threat modeling approach uses dataflow diagrams to illustrate the layout of the different components of a particular application or system. The idea is that each component of the system is a potential target for an attack by an adversary. For each target, the model identifies various cyber-attacks (i.e., denial of service, buffer overflow) that may be employed against the target. The targets are then displayed in a decision tree format. The path leaving the target represents the decisions an adversary would need to make in order to affect that particular target in the software application or system. User of this model would need to determine a ranking system for the threats based on priorities that are consistent and helps to focus efforts on the more likely threats. The model would also provide recommendations for mitigating the potential threats identified. The dataflow diagram approach assumes red teams will be able to discover more vulnerabilities as more components are added to the diagram.

The Attack tree model is broken down into three branches: “root,” “leaf” and “child” nodes [28]. The cyber-attacker's goal or mission is designated as the root of the tree. The leaf is the pathway in which the attacker travels to reach the goal. The actual cyber-attacks are represented by the child nodes. In addition, logical symbols such as AND or OR can be used to determine the cyber-attack flow. For instance, OR can represent that an event will occur because any of the previous “child” steps occurred. Whereas the AND logic means that all of the previous “child” steps must occur. Another feature is the value

metric. Nodes can contain information like probability, cost or adversary characteristics. Having this information facilitates decision regarding the likelihood and resources required regarding a particular cyber-attack. Again, this model emphasizes the need for red teams to understand the threat in order to assess the security of a system.

The Collaborative attack model is a hybrid between a decision trees network model and a Wiki page. The model uses three terms to describe elements in the model: preconditions, transitions and postconditions. Preconditions may describe the state of a system or the capability of the adversary. The state of the system can be defined as conditions that would make it possible for an adversary to conduct a cyber-attack in the first place. Postcondition describes the state of the systems after a system has been compromised (i.e., what can the adversary do after gaining access). Like a network, the transitions connect the preconditions with the postconditions. The transition is the steps taken to maneuver through to the postcondition. Transitions uses the AND logic similar to the Attack tree model—all preconditions must be met before transitioning to the postconditions [29]. The information is then documented onto a Wiki page with detailed information like code or other vulnerabilities that red team members can access, view or provide updated information. The Wiki page was designed to allow collaboration with knowledgeable professionals as well novice. As such any person can add their experience with cybersecurity to the web page.

The name of the last model, Insider threat model, is self-explanatory. The Insider threat model's point of view is from the access and privileges the user already has to a system or network. Red teams would carry out attacks as if they were insiders. Red teams would have to assume characteristics of a potential insider (e.g., knowledge, disgruntle, malicious intent). The purpose of the model is to document actions that an insider would take to exploit a particular system. Further, red teams can use this information to develop countermeasures to prevent a would-be adversary from conducting an attack.

The authors contend that red teams should have quick access to information regarding cyber-attacks, the system, the adversary and various countermeasures. They propose that the information can and should be automated. The automation method recommend by the authors is the use of Unified Modeling Language (UML) for design

visualization and Extensible Markup Language (XML) to display the attack and defense methods. UML is used by software designers in order to provide greater detail. XML is a simple text file used to describe data. The implication is that red team automation would be more simplified because these languages are more suitable to programming the software. Instead of initially assessing the source code of an application like the Threat model, the automated red team model looks at modeling the particular cyber-attack first. In other words, what is the intent of the adversary and the capabilities required to conduct such an attack. This information is evaluated by the red teams and cataloged into the Extensible Markup Language format for automation. Next, the cyber-attack success matrix is delineated using UML. The XML text descriptions can be parsed using specific software tools to generate code. And the final phase of the model is to develop a countermeasure for the cyber-attack. Once the documentation is complete, red teams can review and automate the attacks by parsing the XML.

4. Automatic Executing Penetration Testing

Five researchers at Beihang University located in China developed a penetration testing tool called Automatic Executing Penetration Testing (AEPT). AEPT was developed to address the high cost of employing penetration testing teams and the inefficiency of the penetration tests themselves [30]. The researchers divided penetration testing into four stages as adapted from the National Institute of Standards and Technology (NIST): 1) Planning contained the guidance regarding what is going to be tested and when (i.e., the scope of the penetration test); 2) Discovery is the enumerating and vulnerability scanning performed on the system or network; 3) The attacking stage is the actual penetration testing. This stage also has the capability to provide dynamic feedback to the discovery stage if a new vulnerability is found.; 4) The reporting stage provides the final report of the test. The researches further divided the stages into two methods: a method to generate the testing scheme (planning and discovery) and a method that executes the testing scheme (attacking and reporting). Both methods are to be automated in nature. The AEPT uses a complex pushdown automata model to automatically generate the testing scheme for which the system will execute on the target. The researchers conducted a “proof of reachability” that concluded the pushdown automata will reach its final state given a define set of input

symbols. The execution method reads in the scheme, sets the exploit module and payload, executes the exploit and displays the result. If the exploit fails, the system calls the next exploit module and payload and repeats the execution for that particular vulnerability. The AEPT also iterates through each vulnerability beginning with the least difficult to the most difficult based on a ranking scheme.

5. Automated Penetration Testing Based on a Threat Model

Although a separate research, two authors, one from Imam Abdulrahman Alfaisal University, Saudi Arabia and the other from University of Southampton, United Kingdom suggest the development of an algorithm that will automate a penetration test based off of a threat model scheme that would be provided [31]. This idea resembles a "hybrid" between the previous two researches. The researchers used a threat model from the IT Innovation Centre; however, several models exist based on design and objectives that can be used to highlight potential cyber-attacks. The automated penetration process proposed follows the techniques provided in the National Institute of Standards and Technology (NIST) 800–115 manual. The design employs a graph format populated with nodes that are linked by edges. The nodes represent the systems to be assessed while the edges are various relationships between the nodes (i.e., controls, connections and other hosts). The algorithm will execute the penetration assessment until each node on the graph has been evaluate against the threat model. The complexity of the algorithm depends on the number of nodes on the graph. A network with multiple nodes will require a more complex algorithm.

6. Efficiency and Effectiveness of Penetration Test Automation

Researchers at Dakota State University Madison, South Dakota designed a proof of concept tool that automates some frequently used penetration testing tools to improve efficiency and effectiveness. The tool is scripted in Python and uses other tools such as Nmap, Nessus, and MSF to name a few [32]. The researchers used the Penetration Testing Execution Standard (PTES) to validate the proof of concept tool. Specifically, the tools focused on automating the information gathering and vulnerability analysis stages of the process. The automated tool scanned for IP addresses and web domains returning information related to e-mail addresses, document files, hosts and service vulnerabilities.

The information is downloaded, parsed and saved to a file for exploitation use later. The file is read into Metasploit for exploitation of vulnerabilities identified through the information gathering and vulnerability analysis stages. After the exploitation stage, Metasploit discontinues the session but leaves open the option for another engagement.

Research work continues in an effort to improve penetration testing and red teaming. assessments. A few academic papers were discussed above, but this area is littered with proof of concepts and testing to at least automate the rote methods of identifying vulnerabilities and attack schemes.

F. CARTT ARCHITECTURE

The Cyber Automated Red Team Tool (CARTT) architecture is composed of open-source software. CARTT was designed using the Kali Linux distribution platform. It allows for vulnerability scanning and penetration testing against systems and networks. CARTT was able to conduct vulnerability scans on 100 hosts using the Kali Linux platform [17]. Python is the programming language used for CARTT. It is a beginner friendly programming language that uses the Tk interface (Tkinter) to display the Graphical User Interface (GUI) to the user. This allows the user to select various options to conduct the vulnerability assessments of a system. Tkinter also allows for the automation of many program commands to be executed behind the seen. The buttons are arranged in a sequential manner for ease of execution by the user. The Network mapper (Nmap) and Open Vulnerability Assessment Scanner (OpenVAS) are the open-source tools used for host discovery and vulnerability analysis of a network, respectively. Finally, the MSF will import the results of a vulnerability scan and attempt to conduct a cyber-attack against a particular vulnerability. For a more detailed discussion regarding any of the open-source tools mentioned above, refer to Plot's thesis work [17].

G. CHAPTER SUMMARY

This chapter discussed factors affecting DoD weapon systems and networks. Furthermore, penetration testing was presented to include the advantages and disadvantages of the various techniques. Lastly, cyber red team tools and previous research

was presented. The next chapter presents the framework for the Cyber Automated Red Team Tool (CARTT).

III. DESIGN METHODOLOGY

In this chapter, we discuss the design methodology and framework of CARTT. Specifically, this chapter describes how CARTT imports vulnerability reports, and then allows the user to select various hosts from the report to conduct automated cyber-attacks against.

A. TEST DESIGN

CARTT enables an operator to replicate the actions of cyber red teams by allowing the CARTT user to select and automate scripted cyber-attacks against specified targets on a network. Figure 1 illustrates a simple network that CARTT can operate on. As discussed in Plot's thesis, the Kali Linux suite is used to implement CARTT host discovery, vulnerability analysis, and cyber-attacks methods [17]. The test platform uses the CARTT system, along with three target hosts with different operating systems: Linux, Microsoft Windows 7 Service Pack 1, and Microsoft Windows XP Service Pack 3. For this thesis' proof-of-concept, a test case experiment is conducted to perform a cyber-attack against the Windows XP target from the CARTT Kali Linux system.

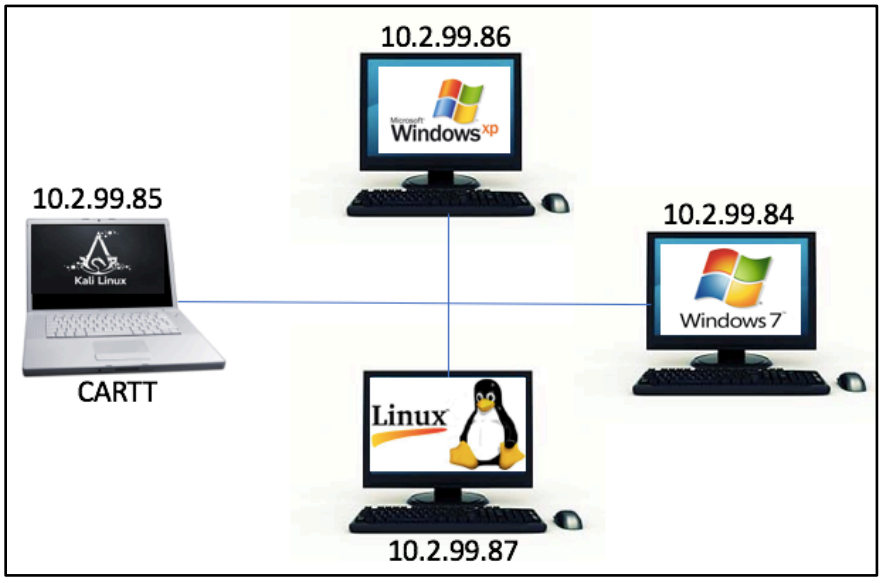

Figure 1. Sample Kali Linux diagram layout

B. METASPLOIT FRAMEWORK ARCHITECTURE

The Metasploit Framework (MSF) architecture provides exploit and payload modules that users may configure and execute within the framework. Exploit modules are comprised of Ruby script commands that perform a sequence of automated steps to exploit a vulnerability previously discovered on a target host. Generally, exploit modules are the delivery vehicles used to establish the "connection" between the attacker and host target. The module options are based on the previously discovered vulnerabilities.

Once a connection is established through an exploit module, a payload module is delivered to the target. Payload modules are Ruby script code (e.g., malware, loaded onto a system or network for execution on a target host), potentially allowing the attacker to gain further access into a system for post-exploitation activities, such as privilege escalation, pivoting, and deeper target analysis. When an exploit is chosen, the MSF automatically selects the best payload to be delivered to the target.

C. CARTT PROCESS FLOW

CARTT is written in the Python scripting language, and is built on top of the MSF architecture. Since MSF (specifically its *mfconsole* utility) is a command-line intensive tool, it can be difficult for inexperience users to become proficient in its usage and full capability. To mitigate this shortcoming, CARTT replaces the command-line interface (CLI) of MSF with a graphical user interface (GUI) to reduce the knowledge needed by the user to perform the procedural steps in a cyber-attack using MSF. To do this, CARTT uses resource scripts to automate complex commands that would normally need to be entered into a terminal window. CARTT also creates files in the background from the MSF results generated by a particular resource code action. This allows CARTT to display the MSF results in the GUI for viewing by the user. Figure 2 shows the entire CARTT cyber-attack process, including identification of vulnerabilities, selection of appropriate exploit modules, and launching of attacks. The following paragraphs describe the overall flow depicted in Figure 2.

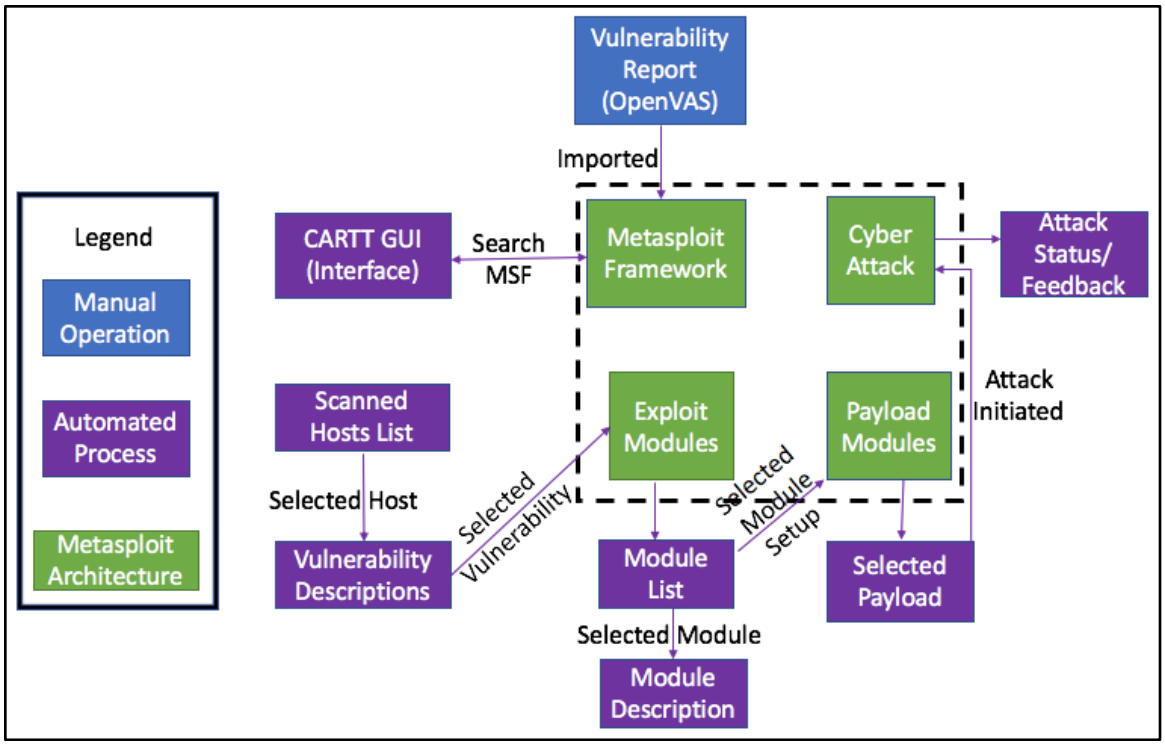

Figure 2. CARTT flow diagram

To start the process, CARTT must receive a vulnerability report generated by the OpenVAS Greenbone Security Assistant (GSA) during the host vulnerability scan. The report is manually downloaded from the GSA GUI into the local file directory. The GSA provides various file downloadable formats; for this development, the file format used is XML. This is the only manual operation currently performed during the proof-of-concept simulation. CARTT then imports the vulnerability report from the local directory into the MSF database. Once imported, each vulnerability is displayed in the Host Vulnerability Descriptions window on the CARTT GUI.

Next, the CARTT GUI allows the user to search the MSF database for corresponding exploit modules for each of the identified host vulnerabilities. The user is allowed to select a particular host and visualize the vulnerabilities identified by the GSA vulnerability scan. The user can then select an individual vulnerability to exploit. Once the vulnerability has been selected, CARTT searches MSF for a list of exploit modules that

may be used to deliver a cyber-attack against the intended host. This list is presented to the user for selection.

CARTT then presents the user with a description of the selected exploit module. This assists the user in determining if the selected exploit is the appropriate exploit for the identified target host. After the user picks an exploit module and communicates the selection to MSF, MSF automatically selects a payload, displays the exploit status in the GUI, and waits for the user to initiate the desired cyber-attack. Once the cyber-attack is initiated, the CARTT interface provides feedback to the user regarding the status of the exploit execution (e.g., if the session failed, or was properly established).

D. CARTT GUI DESIGN

The cyber-attack process described in Section 3.C is presented to the user through the CARTT GUI. This section describes the key design decisions made in designing the GUI.

With the extended cyber-attack capability, the CARTT GUI has evolved from its initial prototype design (Figure 3), however, we have intentionally maintained its simplicity to allow its operation to remain intuitive and easy to understand for the user. The CARTT GUI, as shown in Figure 4, has a menu of buttons located on the top left side of the GUI. Only those buttons which are meaningful at any point in time are activated; the rest are disabled. Initially, only the MSF and Quit buttons are available to the user for selection. A banner stating “Initialize MSF to begin” is displayed above the buttons to prompt the user to begin using CARTT. Otherwise, the user may select the “Quit” button to exit CARTT. Underneath the column of buttons are three small windows. These windows provide the user a quick snapshot of the network that CARTT is connected to. The first window displays the IP address assigned to CARTT while on the network. Second displays the network address in Classless inter-domain routing (CIDR) format of the network. And third displays the number of hosts that were detected on the network by CARTT. Furthermore, immediately below this is a window that list all the hosts in order by their IP addresses.

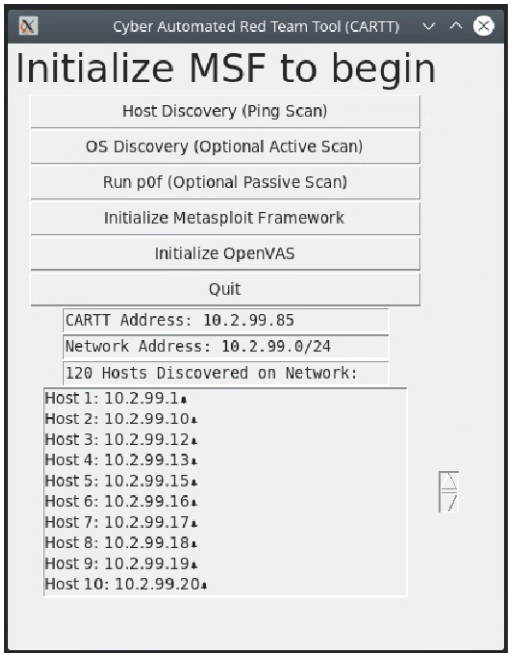

Figure 3. CARTT prototype design

The remainder of the GUI contains windows that display various pieces of information dependent on the actions of the user. The initial GUI screen provides a description to the user of what information each window provides. For example, “Host Vulnerability Descriptions” lists each vulnerability by host. Additionally, if the user were to select a host from the ordered hosts list, only the vulnerabilities associated with that host appear in the “Host Vulnerability Descriptions” window. On the right side of the GUI, are five other windows. The first is an “information bar.” When the user initiates an action within the GUI, the GUI provides “feedback” via the bar indicating that the action was received and in progress. The three remaining bigger windows are self-explanatory: 1) Module List, 2) Exploit Module Description and 3) Exploit Status. The Exploit Status window provides the user feedback during the cyber-attack phase. This is where the user can determine whether to setup, initiate or terminate the cyber-attack.

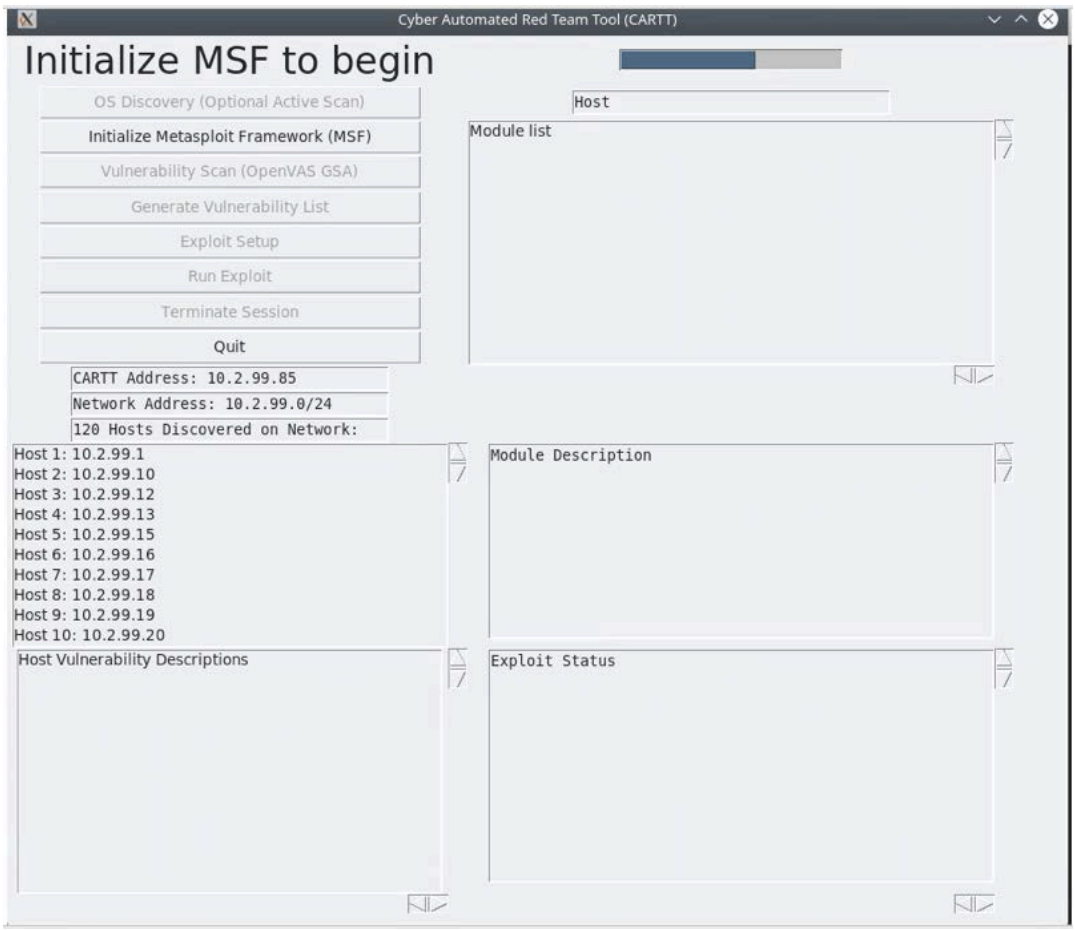

Figure 4. CARTT with extended cyber-attack capability

It is worth noting that the command-line interface provides more options for a user to interact with MSF. However, the user will need to have a higher level of knowledge of MSF and be more familiar with conducting cyber-attacks. The CARTT GUI has been designed with a novice operator in mind (i.e., no extensive MSF or networking knowledge is expected).

E. CARTT GUI OPERATIONS

Once the user has connected to and reviewed the network information, it is time to review the vulnerabilities associated with the hosts on the network. The user must click the “Generate Vulnerability List” button on the GUI. As stated earlier, for this thesis, we will review the vulnerabilities associated with the Windows XP Service Pack 3 machine (IP address 10.2.99.86). Figure 5 shows the list of hosts and associated vulnerabilities

populated inside the description window. The user may use the scroll bars on the side and bottom of the box to either move the window area up, down or side-to-side in order to read the vulnerabilities listed inside the window area.

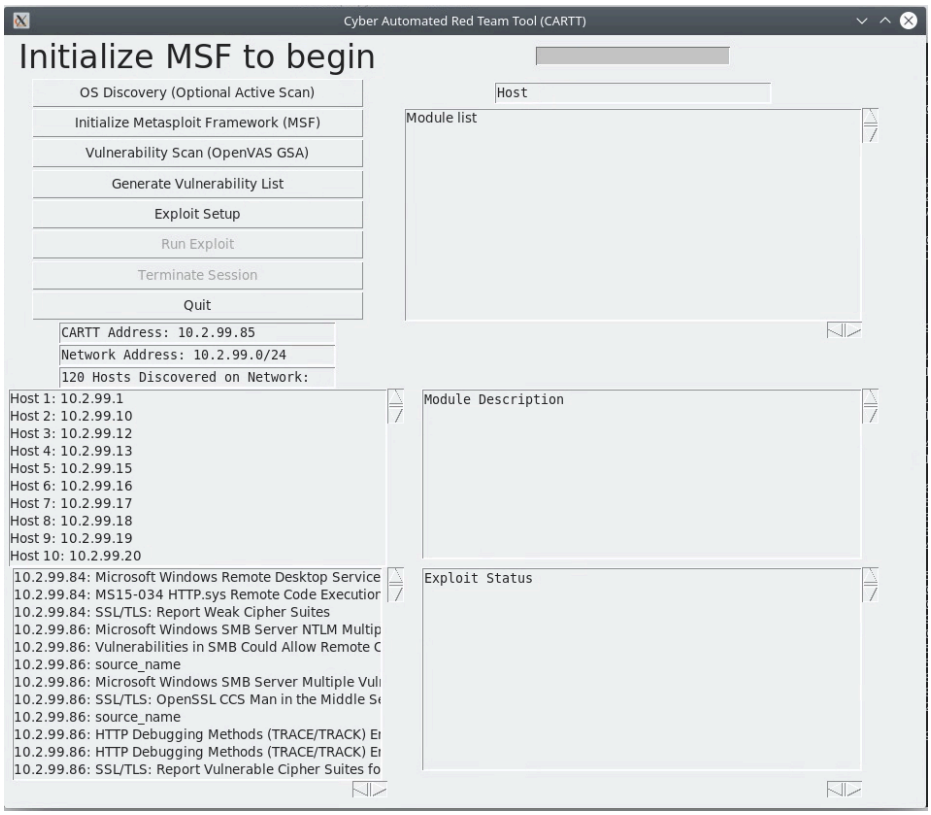

Figure 5. List of vulnerabilities inside CARTT

Instead of scrolling through all the vulnerabilities listed, the user can select a specific host to populate its vulnerabilities in the window. Figure 6 illustrates the user selecting Host 74 with IP address 10.2.99.86. Note that the IP address is also displayed in the top right “Host” window. This gives the user a “quick” reference of which host is in the CARTT selection for further exploitation. As a result of the selection, the corresponding vulnerabilities are listed in the descriptions window located below the hosts. The user can scroll thru the list in search of a particular vulnerability for that host. Once the user selects a vulnerability from this list, CARTT finds and presents the user with a list of exploit modules for the selected vulnerability (see Figure 7).

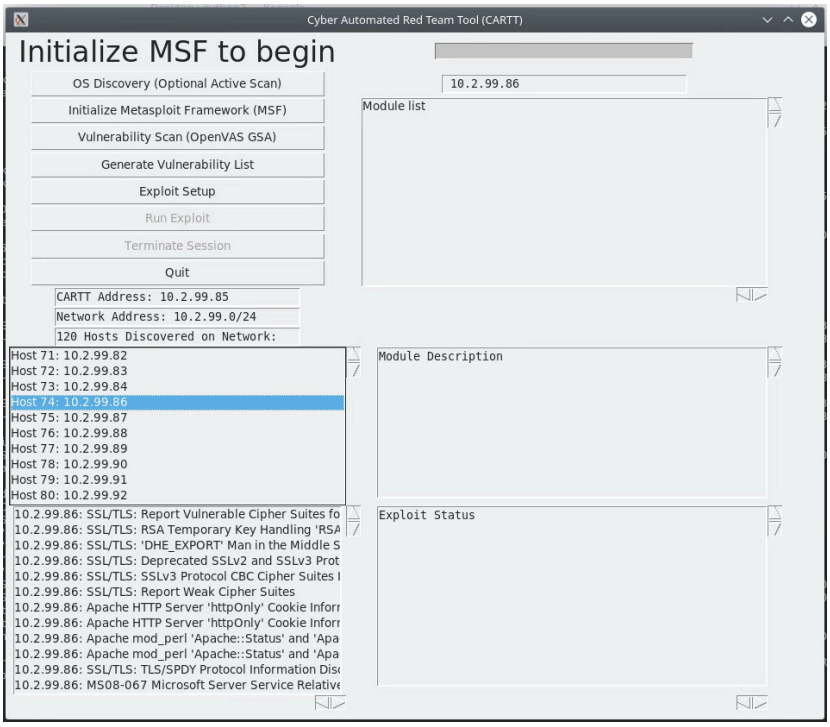

Figure 6. List of vulnerabilities by Host

Figure 7 lists several exploit modules for the Microsoft Windows SMB Server vulnerability. The user can scroll through the list both vertically and horizontally. For the MS08-067 Microsoft Server vulnerability, only one exploit module is returned (see Figure 8). The MS08-067 exploit module will be demonstrated for this thesis project.

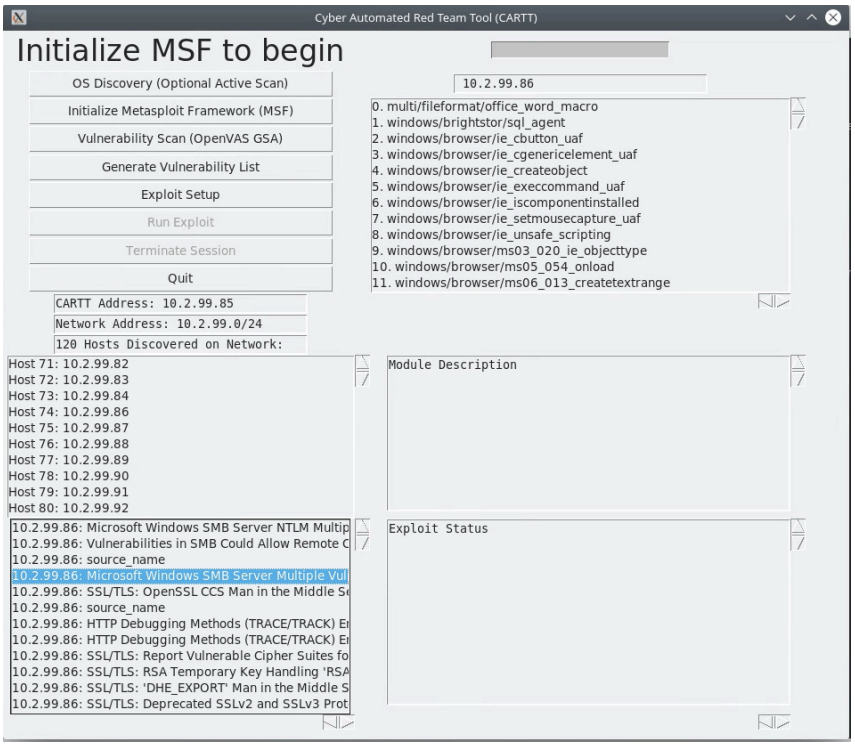

Figure 7. Multiple modules in list window

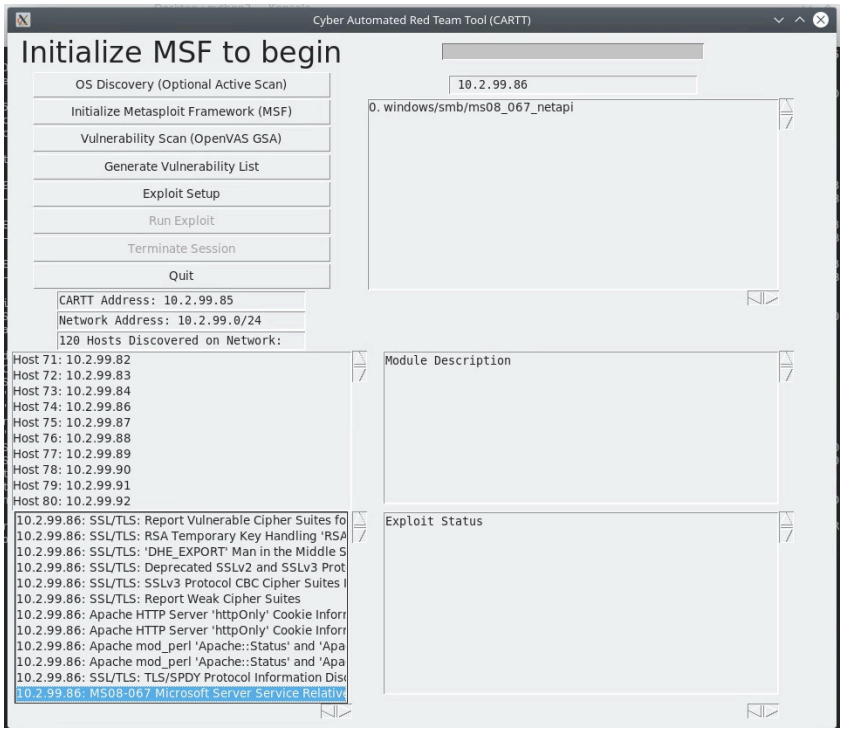

Figure 8. Only one module in list window

To assist the user in determining which exploit module to use, CARTT provides a description of a selected module. Unlike the previous two GUI windows, the "Exploit Module Description" window has only one scroll bar. Since the previous windows' information were an actual list of items, they were designed to display in list format. However, the "Module Description" window "wraps" the text in paragraph format for ease of viewing and reading by the user. Figure 9 shows the description of the *ms08_067_netapi* exploit module. Note the user will still need to scroll vertically to read the full text.

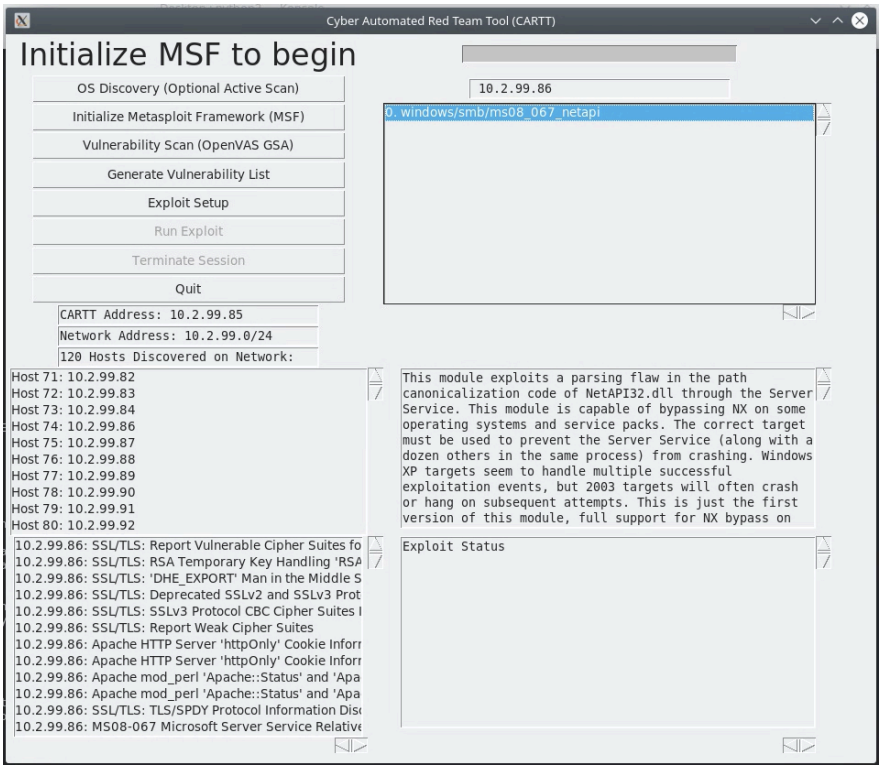

Figure 9. Description of *ms08_067_netapi* exploit module

The last window of the CARTT GUI is the "Exploit Status" window. This is where the user receives feedback regarding the exploit phase of the cyber-attack. After selecting the desired exploit module, the user clicks the "Exploit Setup" button. CARTT uses the selected exploit module and returns a payload status from the MSF (see Figure 10). The user is provided with the type of payload that will be used against the targeted host. CARTT also returns confirmation that the proper exploit module and target IP address is as

previously selected. Next, the “Run Exploit” button is highlighted giving the user the option to initiate the exploit against the target. It is important to note that the user still has the option to select another exploit, vulnerability or host at this point in the operation. The user would only need to select the preferred item in either of the previous windows.

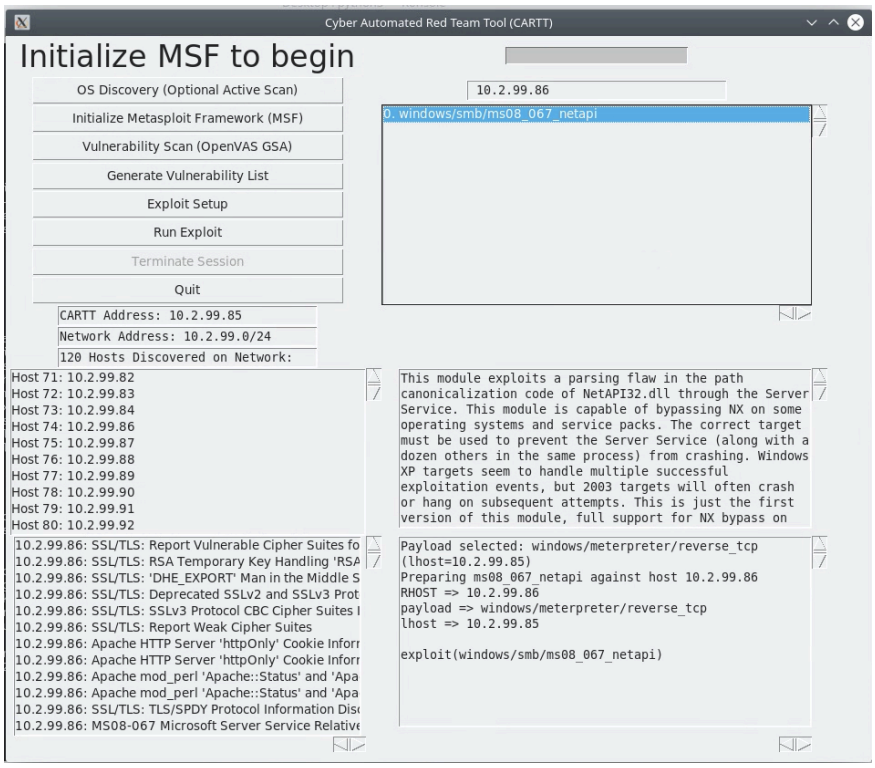

Figure 10. Exploit setup and preparation

After initiating the cyber-attack against the target, CARTT will return another exploit status. In this experiment, CARTT returns several items in the window. The most important being that a “Meterpreter session 1” is open on target IP address 10.2.99.86 (see Figure 11). The user now has remote access to the target. Note that the “Terminate Session” button is now highlighted on the GUI. Again, the user has the option to initiate another exploit against this target. For example, starting another *reverse_tcp* (Transmission Control Protocol) shows a Meterpreter session “2” was open in the Exploit Status window.

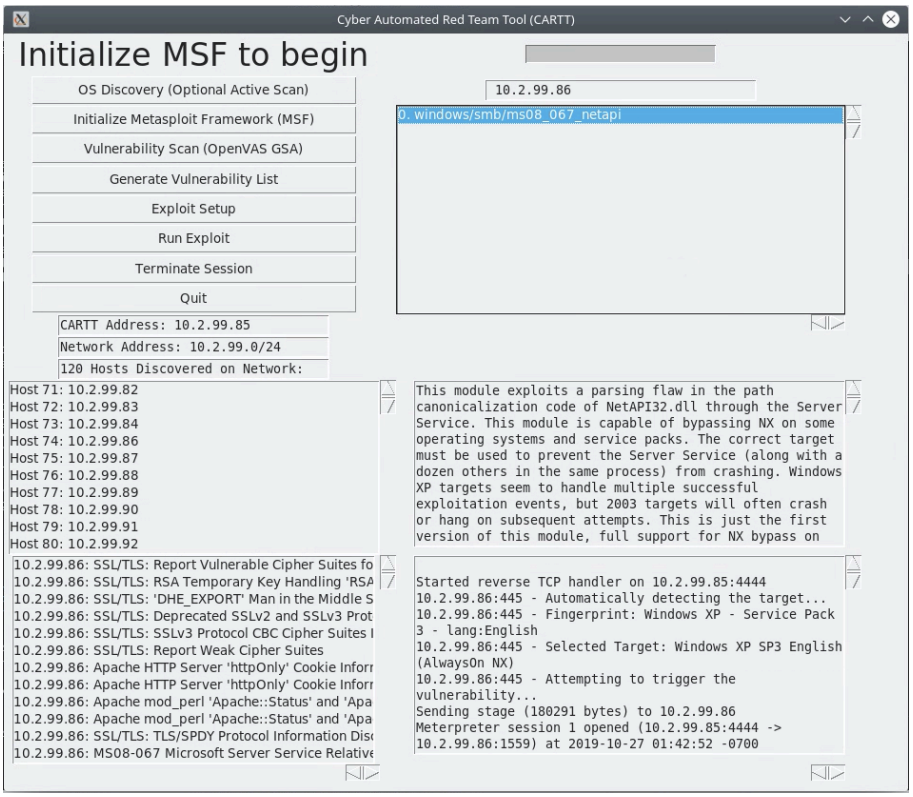

Figure 11. CARTT established Meterpreter access session

At this point, the user may choose to disconnect from the target via the “Terminate Session” button. This action terminates the cyber-attack. Figure 12 shows the feedback acknowledgement in the CARTT GUI window if the user wishes to disengage from the cyber-attack. At this point, the user can select another host and explore more vulnerabilities or exit from the CARTT process by selecting the “Quit” button.

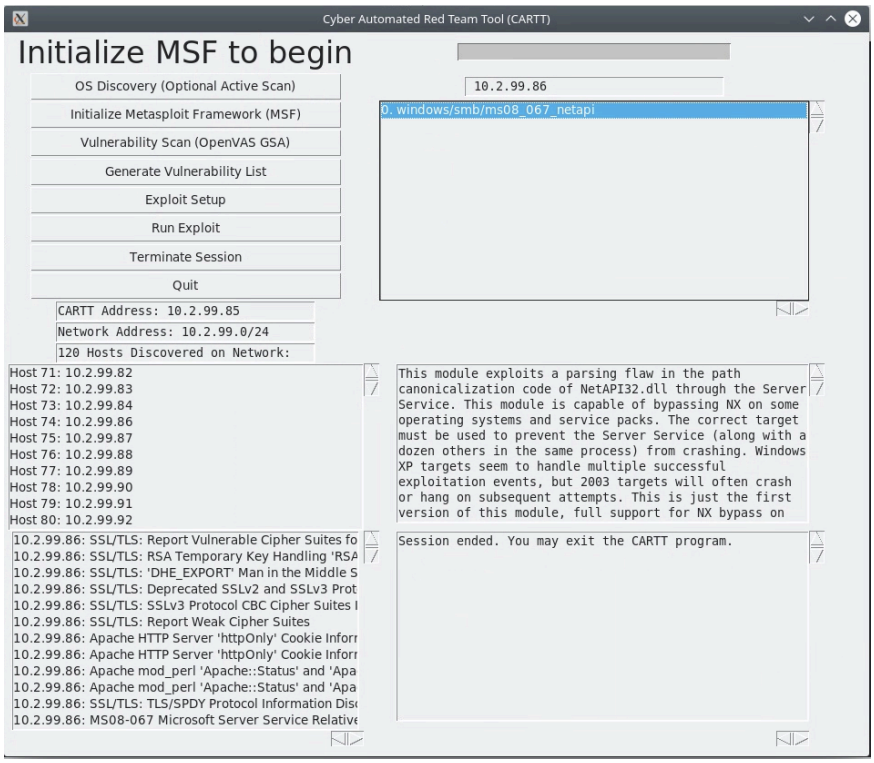

Figure 12. CARTT user terminates cyber-attack against target

F. CHAPTER SUMMARY

This chapter provided an overview of the CARTT design and architecture, and its use in the cyber-attack process. CARTT imports a GSA vulnerability scan report into MSF, then with user input, automates a cyber-attack against a specified host on a target network. The goal of CARTT is to support the process in such a way that a non-expert user can conduct a cyber-attack and receive feedback regarding its success or failure. The next chapter describes the implementation of CARTT and as a proof-of-concept, demonstrates its capability employing a realistic cyber-attack, specifically the MSF *ms08_067_netapi* exploit module against a Windows XP target.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. SYSTEM IMPLEMENTATION AND TESTING

The MSF provides two user interfaces for conducting cyber reconnaissance, exploitations, and attacks: Armitage and the MSF Console. Armitage is a third-party GUI that allows the user to perform cyber actions through a comprehensive set of GUI options. It requires a thorough knowledge of the MSF, however, so a novice user can become overwhelmed with its various options and functions, making Armitage a less than ideal choice for the uninitiated. Conversely, the MSF Console (msfconsole), is an interactive command-line style interface that requires the user to be highly knowledgeable in MSF functionality. As such, being able to navigate the MSF functions in either interface can prove challenging for a user not well-versed in using MSF commands. The CARTT GUI eases this process for the novice user by automating the MSF commands to conduct a cyber-attack as part of a red teaming event.

A. IMPLEMENTATION AND SETUP

1. Backend Applications

CARTT was built using a Python script to interface with the MSF database. Python is a user-friendly programming language that uses scripted code to automate commands in the MSF architecture. The original version of CARTT used Python scripted code to automate the network and vulnerability scans, then used Python to import the scanned results into the MSF [17]. This gave the user the ability to view and select a potential exploit target without needing in-depth knowledge of MSF functionality.

The MSF requires initialization steps before using its exploit capability. It uses a backend relational database called Postgresql, which must be started and initialized before using MSF. Next the user can launch MSF from the Kali Linux command line by simply typing “msfconsole.” CARTT simplifies these setup steps by performing all of them with a single click of the “Initialize Metasploit Framework” GUI button. Of note, CARTT’s initialization button employs the “-q” command-line option when launching Metasploit, which starts msfconsole in “quiet” mode without displaying banner information such as MSF version, current number of exploit and payload modules, etc.

2. GUI Implementation

As mentioned earlier (in Chapter II Section F), Tkinter was the GUI tool chosen for the CARTT architecture. Tkinter is an application programing interface (API) that is a part of the Python library. It is intuitive for the programmer, and easy to implement as a front-end application for a comprehensive tool suite like MSF. For this research, Tkinter allowed us to develop a simple GUI that alleviated the need for the user to have extensive knowledge about MSF operations and command line interface actions to perform cyber-attacks against a target. For more details regarding Tkinter, see Plot's thesis [17].

While developing the CARTT GUI, we wanted to keep the programming simple and easy to understand. For example, the GUI window size is set by a global variable in the code that is easily modifiable if additional features are required. This is done to support future expanded capability of CARTT. Other global variables established in Tkinter for the GUI are the type of FONT, FONT_SIZE, WIDTH, HEIGHT ROW, and COLUMN. Figure 13 shows the global settings used for the test CARTT GUI. The current ROW and COLUMN settings provide an easy offset for adding additional labels, buttons and windows. The title and header labels are offset from these settings.

```
############### [ GLOBAL VARIABLES ] ###################

FONT = "Arial Bold"
FONT_SIZE = 25
WIDTH =  35
HEIGHT = 1
ROW = 0
COLUMN = 0
WINDOW_SIZE = "942x792" #width x height
```

Figure 13. CARTT GUI global setting

The interface buttons, when clicked by the user, invoke an action through the CARTT GUI. The buttons are tied to two actions by the Python lambda function. When

clicked, each button initiates the corresponding action associated with the button, and the progress bar notifies the user that their action was registered on the CARTT GUI. In other words, the Python lambda function creates threads for multiple tasks to take place at the same time. The window features are implemented via a Tkinter interactive "ListboxSelect" widget. The user is allowed to scroll through a line of text and choose one item from the list. The window is configured to call a function that fetches the user's requested item. These features were chosen to make CARTT more user friendly.

B. SYSTEM TESTING

CARTT uses MSF commands to configure and execute cyber-attack actions. CARTT uses the "spool" command to copy the MSF output into a text file in the background. This text file is then parsed by the CARTT script for key vulnerability and exploit information, to be later displayed on the CARTT GUI.

At this point, the MSF is ready to receive information regarding vulnerable targets. The vulnerability report is imported into the MSF database with the "db_import *xml" command. For this testing, the OpenVAS vulnerability scan report was downloaded as an XML file, but MSF has the capability to import other file formats from well-known scanners, such as Nmap and Nessus. The MSF "vulns" command produces a list of the vulnerabilities for all hosts in the XML file. CARTT subsequently copies this output into the GUI's "Host Vulnerability Descriptions" window (see arrow in Figure 14). To find MSF exploit modules related to a particular vulnerability, CARTT combines the MSF keyword commands "name" and "type." First, CARTT parses the text file for the description of the vulnerability. Again, these descriptions are displayed in Figure 14. For example, if a "Microsoft Windows Remote Desktop Service" vulnerability is selected by the user, CARTT will parse the file for the selected text. CARTT then combines the text with two of MSF keywords: "name" for the description, and "type" to return a module for the specific vulnerability description. This entire sequence of characters is then communicated to the MSF database at the command line interface. This combination is passed to the MSF database using the regular expression "search" attached to the CARTT

string value stored in a local variable called “search_cve.” Lastly, the search_cve local variable filters the type of module for CARTT's purpose —”exploit.”

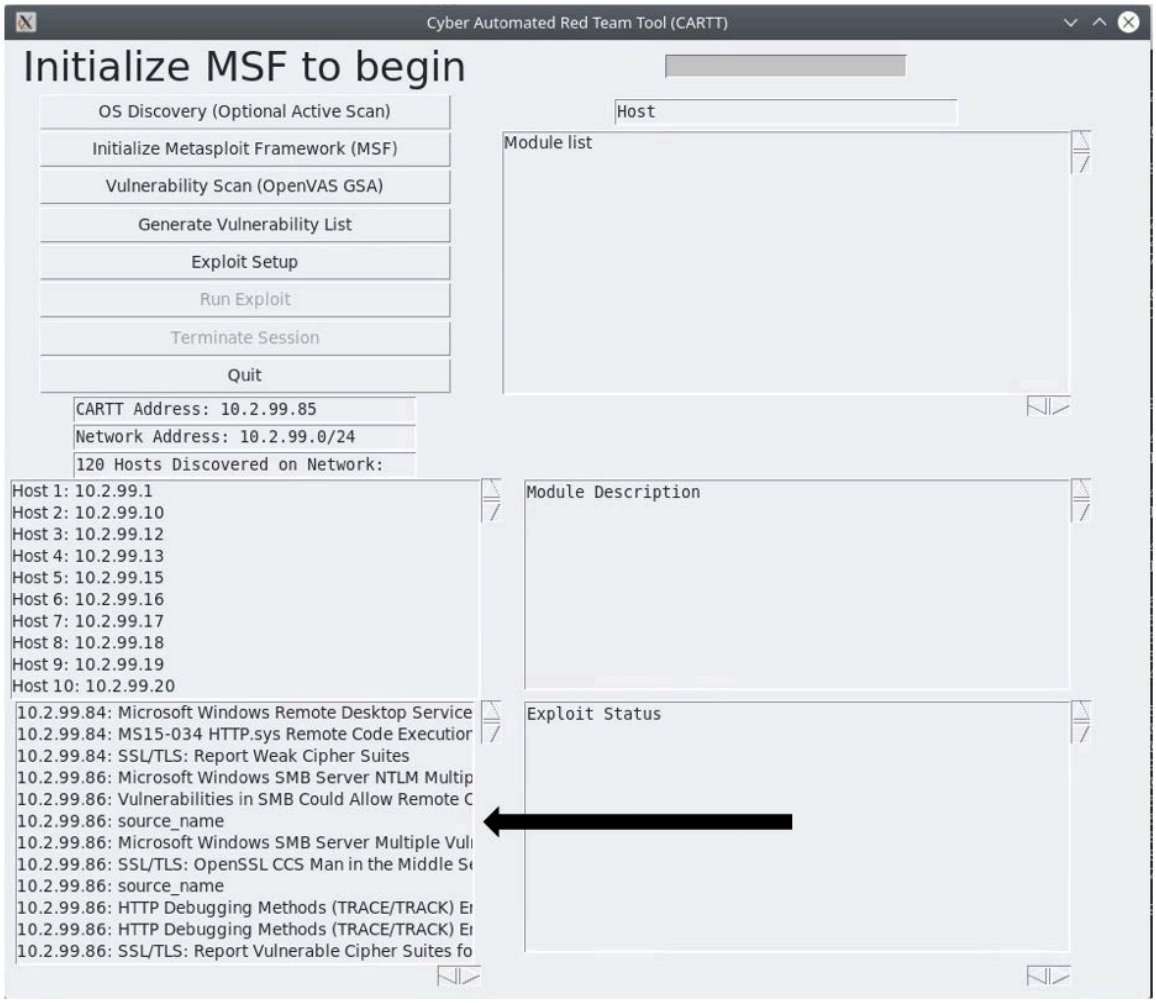

Figure 14. List of vulnerabilities inside CARTT GUI

To display the actual description of the selected exploit module, CARTT places the “info” command at the beginning of the module's name. The output file of 126 lines is parsed and the resulting description is displayed in the CARTT “Module Description” window. Figure 15 shows the result from the command line query executed by CARTT. This is an important step in the process, as it provides detailed information about the exploit that may be of interest to the user, such as possible side effects of the exploit. Because a description length can range from one sentence to multiple lines, a scroll bar is provided to

the user for ease of viewing. If the user is satisfied with the exploit module, they may proceed to the next button — Exploit Setup.

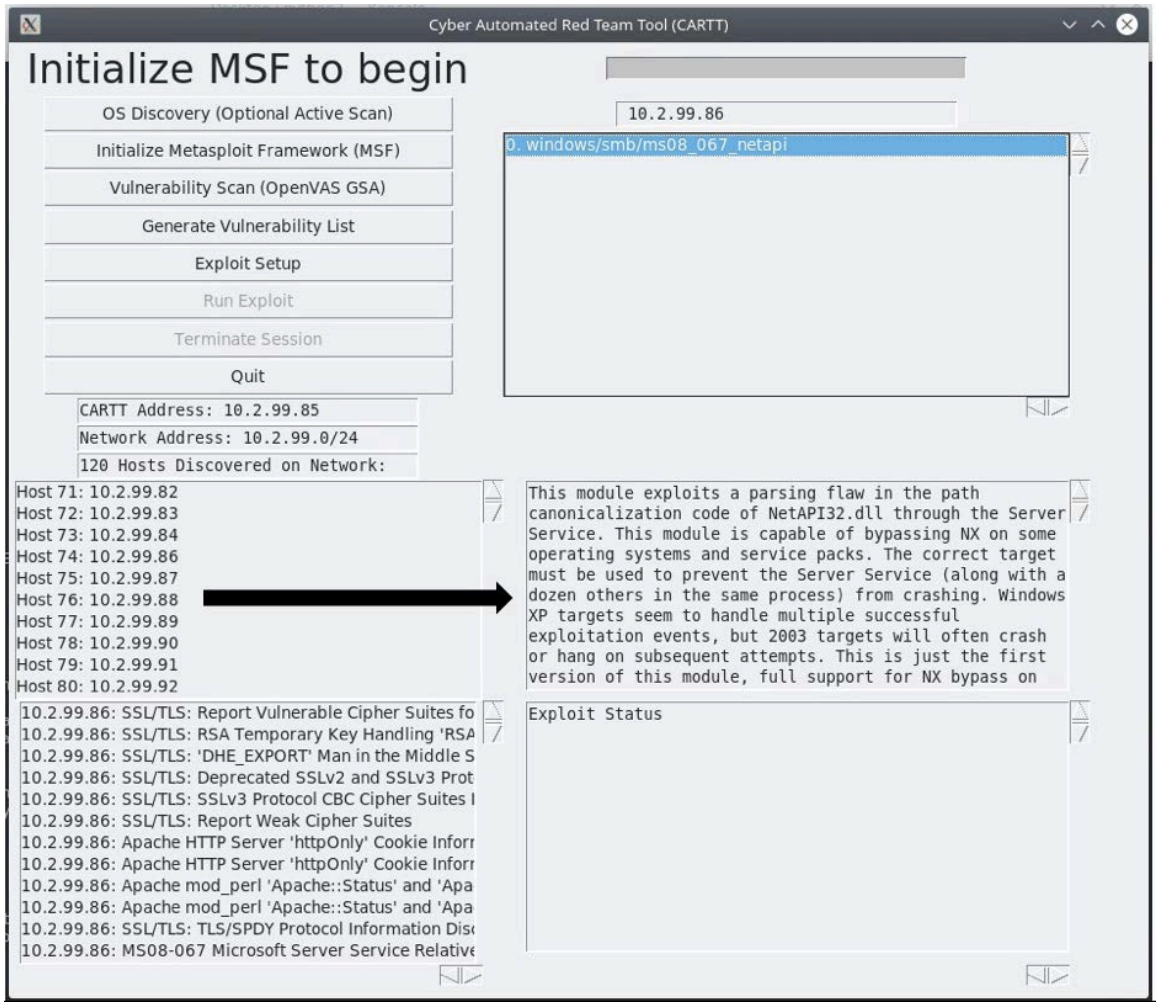

Figure 15. Description of ms08_067_netapi exploit module

To setup an exploit, CARTT automates the execution of a series of otherwise manual steps and commands. The first step is for the MSF to configure the user-selected module for execution. CARTT combines the selected host and exploit module and inputs them into a scripted resources code. The resource code automates the input of the required exploit module parameters (remote host, local host, and payload).

CARTT automates this entire process with the click of the “Exploit Setup” button. Figure 16 shows the exploit status after the setup has been implemented.

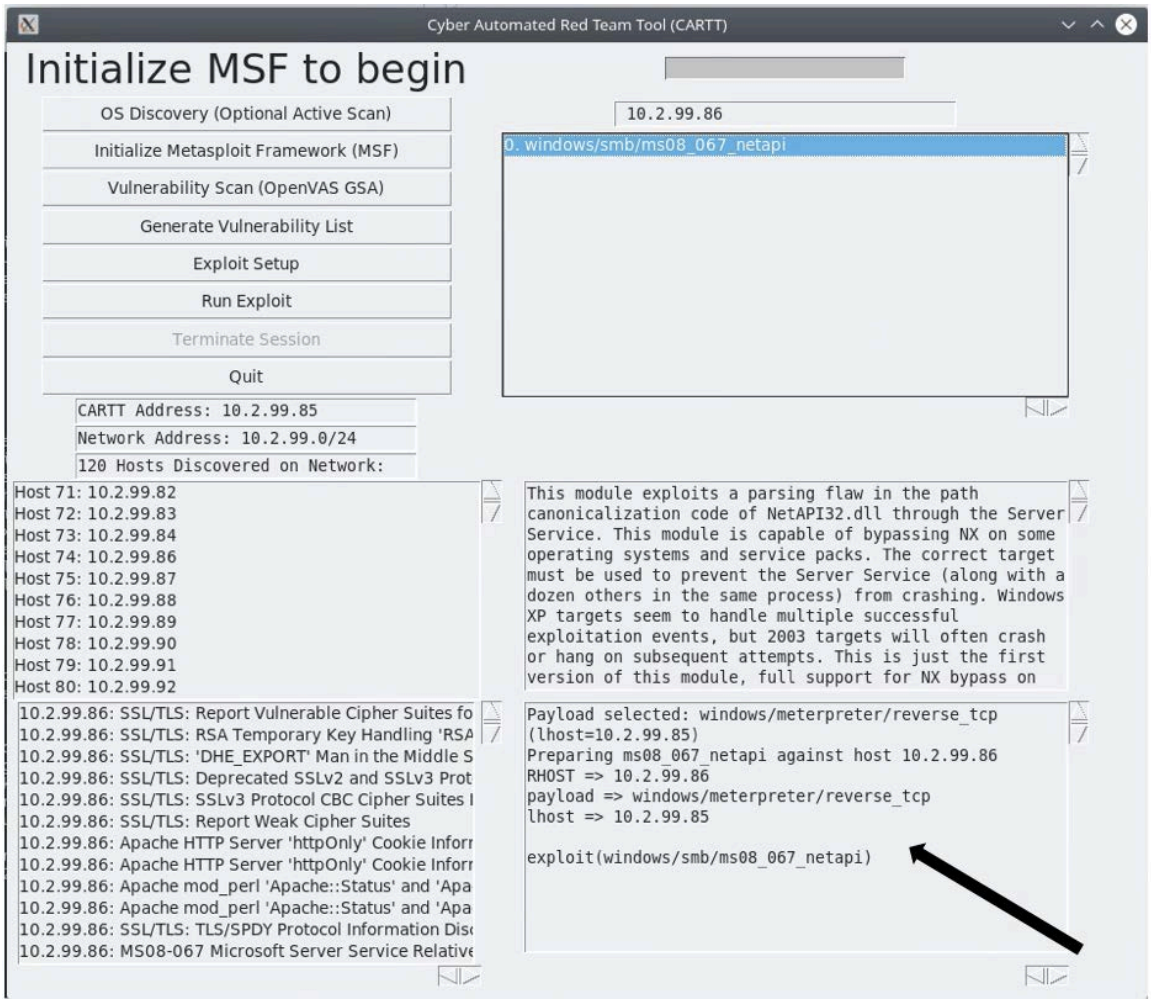

Figure 16. Exploit setup and preparation

CARTT initiates the cyber-attack against the remote host target IP once the user clicks the “Run Exploit” button on the CARTT GUI. The user may terminate the exploit session at any time by clicking the “Terminate Session” button after an exploit has been initiated. When the exploit has completed, CARTT will inform the user that the session has ended, at which time the user may exit the program.

C. RESULTS

For our implementation test, we used CARTT to target an unpatched Windows XP Service Pack 3 system with a known vulnerability identified as a Microsoft SMB server service relative path stack corruption. CARTT used this vulnerability to search MSF for a suitable exploit module and returned the exploit *ms08_067_netapi*. This module is ranked "great" by MSF and considered reliable based on the MSF ranking structure. Upon setup, CARTT displays that a Meterpreter reverse-tcp payload has been selected to couple with the exploit module. Also displayed to the CARTT user is confirmation of the attacker's local host (lhost) IP address and the target's remote host (rhost) IP address. Initiation of the cyber-attack yielded an exploit and payload status stating that the reverse-tcp connection has been established, and that a Meterpreter session has opened. This indicates to the CARTT user that the exploitation against the targeted host was successful. At this point, the user may terminate the cyber-attack session and close the CARTT GUI.

In summary, the msfconsole would require 5 manual commands to be entered at the command-line to conduct this cyber-attack. CARTT automated and simplified this process by reducing the steps required to two button clicks.

D. CHAPTER SUMMARY

This chapter described the MSF interfaces Armitage and msfconsole. It also described how a user can manually conduct a cyber-attack against a desired target. Finally, it demonstrated how CARTT automates the cyber-attack process making it easier for a non-expect user to perform the task with little experience.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS AND FUTURE WORK

A. SUMMARY

The goal of this thesis was to extend the CARTT capability in order to automate a cyber-attack scenario against a designated computer system. To that end, we expanded the user-friendly GUI to allow the user to select various scripted cyber-attacks based on previously identified vulnerabilities. We performed a test experiment using the prototype system against a Windows XP Service Pack 3 system. The test demonstrated that CARTT can import scanned vulnerabilities and allow users to execute cyber-attacks against those vulnerabilities on a targeted host. The portable prototype system was updated to provide real-time feedback that a non-expert user can understand.

CARTT was implemented using the Python programming language, which makes it easy for follow-on programmers to further extend the capabilities of CARTT's front-end applications. The GUI design and the nature of the attack scripts allow CARTT to be used by operators who do not have extensive training in offensive cyber operations or red teaming. These benefits also address the well-known shortage of trained red teams for conducting security vulnerability assessments of cyber systems.

This research aimed to contribute realistic value to DoD testing of computer systems. CARTT reduces the learning curve for red teaming by allowing effective and efficient vulnerability testing during acquisition and operational fielding of computer systems. CARTT is ideal for red team testing legacy systems currently deployed throughout DoD, specifically systems not connected to the internet.

B. CONCLUSIONS

CARTT combines vulnerability scanning and red teaming into one tool for a more complete assessment. The vulnerability scanning is conducted against known cyber vulnerabilities. Conducting red team cyber-attacks is a proactive means of determining whether a previously identified vulnerability is valid and exploitable on a selected target. CARTT was developed, using the MSF open source software suite, as a proof of concept to demonstrate that a vulnerability can be validated as potentially exploitable on a target.

CARTT, by using the MSF suite of pen-testing tools, can have its exploit modules modified to potentially identify new system flaws known as zero days. However, this is something that would require red team assistance for the novice user. CARTT provides a low-cost method for organizations to establish a robust automated testing program for cyber systems. With CARTT as an asset to DoD, red teams will be able to focus their efforts on the growing capabilities of the most urgent adversaries.

This research answered the following questions:

1. Primary Question

How can the capability of the current Cyber Automated Red Team Tool (CARTT) be extended to enhance simulated Red Team attacks against specified computer systems not directly connected to an external network?

We extended the previous CARTT prototype from an automated host discovery and vulnerability assessment tool to one that conducts scripted cyber-attacks against computer systems using the results of cyber reconnaissance. To support this, we expanded the GUI to provide the user with real time feedback in preparing and executing a cyber-attack. We demonstrated that the new prototype can successfully conduct a cyber-attack against a Microsoft Windows XP Service Pack 3 machine. The XP machine had a well-known parsing flaw vulnerability in the Microsoft version API, known as ms08_067. CARTT, using an MSF exploit module, established a Meterpreter session on the vulnerable system, giving the user complete administrative access to the target system. We also extended the CARTT tool functionality by providing the user the option to select a follow-on exploit module or terminate the current cyber-attack at any time during the process.

2. Secondary Question

How can the CARTT tool be designed so that a novice user can interact with it and understand the results?

We expanded the CARTT GUI to provide the user with critical information and feedback throughout each phase of the red team cyber-attack process. For example, the user can scroll through the Host Vulnerability window in CARTT to view all target hosts'

vulnerabilities at once or select a particular host from the target list to view only the vulnerabilities associated with that specific host.

The CARTT interface was also extended to provide the user a description of a selected exploit module. The user can review this information to determine if the exploit module will potentially yield the intended cyber effect. Lastly, CARTT was extended to provide the user with the status of the cyber-attack, confirming for the user the payload, exploit, and remote host information, as well as the cyber-attack success.

Due to limited time availability to conduct this research, we could not conduct a study of user performance with the prototype system. We, however, believe that we have made the process of conducting red team cyber-attacks significantly easier than the earlier manual, time-consuming, error-prone, and technically difficult process.

C. FUTURE WORK

CARTT offers many opportunities for future research. Future work should further expand the capability of CARTT to automate the entire cyber-attack scenario (i.e., cyber reconnaissance, vulnerability analysis, cyber-attack, and assessment) against computer systems. The following are recommendations to work toward that goal.

1. Perform Vulnerability Scans from within the CARTT

CARTT currently launches the OpenVAS GSA GUI from the MSF command line. As a result, the vulnerability scans are conducted outside of the CARTT GUI. The user is required to manually upload the list of hosts discovered by CARTT into the OpenVAS GUI. Future work should configure CARTT to automate such vulnerability scans from within the MSF for the user and provide feedback on the results. This allows the user to only have to interact with one GUI tool.

2. Multiple Attack Vectors

The work in this thesis was a prototype proof of concept, thus it was kept simple with one attacker and one target in a virtual environment. Future work should research the possibility of CARTT conducting multiple attacks against potentially multiple targets in a

single session. Additionally, researchers should consider testing CARTT on real-world systems connected to the Internet, which would need to address network security policies, firewalls, etc.

3. Software Updates

In this thesis, CARTT used a known exploit module against a known vulnerability. A limitation of CARTT is its inability to simulate a red team's ability to "maneuver on the fly" and adapt to a changing situation. Further research should determine an effective and reliable method for CARTT to receive software updates to keep pace with the latest vulnerabilities discovered on target systems. One method is for red teams to develop software that can be uploaded to CARTT for testing against targets.

4. Improved Performance

This thesis focused on CARTT functionality, specifically whether it could execute a cyber-attack. Future work should focus on improving CARTT performance regarding speed of execution and feedback. Research may begin by looking into the NPS cyber battle lab and the Kali Linux suite performance. For example, would the CARTT performance improve if executed in a new environment. The Tkinter GUI interface and MSF are other areas that future work can investigate for possible optimization.

LIST OF REFERENCES

[1] The Office of the Director, Operational Test and Evaluation, "Cybersecurity," Washington, DC, USA, 2018. [Online]. Available: https://www.dote.osd.mil/Portals/97/pub/reports/FY2018/other/2018cybersecurity.pdf?ver=2019-08-21-155613-837

[2] The Office of the Director, Operational Test and Evaluation, *"DOT&E FY 2004 Annual Report,"* Washington, DC, USA, 2004. [Online]. Available: https://www.dote.osd.mil/Publications/Annual-Reports/2004-Annual-Report/

[3] The Office of the Director, Operational Test and Evaluation, "Cybersecurity," Washington, DC, USA, 2014. [Online]. Available: https://www.dote.osd.mil/Portals/97/pub/reports/FY2014/other/2014cybersecurity.pdf?ver=2019-08-22-110127-793

[4] The Office of the Director, Operational Test and Evaluation, "Cybersecurity," Washington, DC, USA, 2013. [Online]. Available: https://www.dote.osd.mil/Portals/97/pub/reports/FY2013/other/2013iaiop.pdf?ver=2019-08-22-111131-863

[5] United States Government Accountability Office, "Weapons 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/assets/700/694913.pdf

[6] The Office of Naval Research, "Red Team in a Box for Embedded and Non-IP Devices," Washington, DC, USA, 2018 [Online]. Available: https://www.navysbir.com/n18_2/N182-131.htm

[7] We Are The Mighty, "The military spends millions to not upgrade computer," Accessed: 19-Aug-2019. [Online]. Available: https://www.wearethemighty.com/news/us-military-old-windows-software?rebelltitem=1#rebelltitem1

[8] The Office of the Director, Operational Test and Evaluation, "Cybersecurity," Washington, DC, USA, 2012. [Online]. Available: https://www.dote.osd.mil/Portals/97/pub/reports/FY2012/other/2012iaiop.pdf?ver=2019-08-22-111536-970

[9] Department of Defense, "Resilient Military Systems and the Advance Cyber Threat," Washington, DC, USA, 2013 [Online]. Available: https://nsarchive2.gwu.edu/NSAEBB/NSAEBB424/docs/Cyber-081.pdf

[10] Serbu, J. “Despite sluggish rollout, DoD vows to expand new hiring system for cyber personnel,” *Federal News Network*, Feb. 27, 2019. [Online]. Available: https://federalnewsnetwork.com/defense-main/2019/02/despite-sluggish-rollout-dod-vows-to-expand-new-hiring-system-for-cyber-personnel/. [Accessed: 13-Jul-2019]

[11] The Office of the Director, Operational Test and Evaluation, “Cybersecurity,” Washington, DC, USA, 2016. [Online]. Available: https://www.dote.osd.mil/Portals/97/pub/reports/FY2016/other/2016cybersecurity.pdf?ver=2019-08-22-105129-640

[12] J. Li and L. Daugherty, “Training Cyber Warriors: What Can Be Learned from Defense Language Training?,” RAND Corporation, 2015 [Online]. Available: http://www.rand.org/pubs/research_reports/RR476.html

[13] The Office of the Director, Operational Test and Evaluation, “Cybersecurity,” Washington, DC, USA, 2017. [Online]. Available: https://www.dote.osd.mil/Portals/97/pub/reports/FY2017/other/2017cybersecurity.pdf?ver=2019-08-19-113538-460

[14] C. Bond, J. Lewis, H. Leonard, J. Pollak, C. Guo, and B. Rostker, “Tour Lengths, Permanent Changes of Station, and Alternatives for Savings and Improved Stability,” RAND Corporation, 2016 [Online]. Available: http://www.rand.org/pubs/research_reports/RR1034.html

[15] P. Engebretson, *The Basics of Hacking and Penetration Testing,* 2nd Edition Rockland, MA, USA: Syngress, 2013.

[16] C. T. Phong, “A study of penetration testing tools and approaches,” M. S. thesis, Sch. Comp. and Math. Sci., Auckland Univ. of Tech., Auckland, New Zealand, 2014. [Online]. Available: https://openrepository.aut.ac.nz/bitstream/handle/10292/7801/ChiemTP.pdf?sequence=3&isAllowed=y

[17] J. A. 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 CS, NPS, Monterey, CA, USA, 2019. [Online]. Available: http://hdl.handle.net/10945/62832

[18] Cobalt Strike, “Raffi’s Abridged Guide to Cobalt Strike,” May 25, 2016 [Online]. Available: https://blog.cobaltstrike.com/2016/05/25/raffis-abridged-guide-to-cobalt-strike/

[19] S. Sidel, “Test center: Core Impact 3.1 automated pen testing tool,” *Techtarget*, Accessed: Jul 28, 2019 [Online]. Available: https://searchsecurity.techtarget.com/feature/Test-center-CORE-IMPACT-31-automated-pen-testing-tool

[20] Core Security, *Core-Impact-Pro-network-testing.pdf* [Online]. Available: https://www.e-spincorp.com/pdf/product/CoreSecurity/CoreImpact/Datasheet/Core-Impact-Pro-network-testing.pdf

[21] R. Harikrishnan, "Three automated penetration testing tools for your arsenal," *ComputerWeekly*, May 24, 2011. [Online]. Available: https://www.computerweekly.com/tip/Three-automated-penetration-testing-tools-for-your-arsenal

[22] E-SPIN, CANVAS Product Overview, E-SPIN Group. Accessed: Jul 29, 2019 [Online]. Available: https://www.e-spincorp.com/canvas-product-overview/

[23] Immunityinc, "Canvas." Accessed: May 15, 2019 [Online]. Available: https://www.immunityinc.com/products/canvas/

[24] E-SPIN, Metasploit-Pro-user-guide.pdf, Rapid7 Accessed: May 16, 2019 [Online]. Available: https://www.e-spincorp.com/pdf/product/Rapid7/Metasploit-Pro-user-guide.pdf

[25] Sectools, "Vulnerability exploitation tools—SecTools Top Network Security Tools." Accessed: May 15, 2019 [Online]. Available: https://sectools.org/tag/sploits/

[26] "] L. Epling, B. Hinkel, and Y. Hu, 'Penetration testing in a box,' Proc of the Information Security Curriculum Development Conference, Kennesaw, GA, USA, 2015. [Online]. doi: 10.1145/2885990.2885996

[27] V. Tilemachos and C. Manifavas, "An automated network intrusion process and countermeasures," in *Proc of the 19th Panhellenic Conference on Informatics, Athens, Greece*, 2015, pp. 156–160 [Online]. doi: 10.1145/2801948.2802001

[28] H. T. Ray, R. Vemuri, and H. R. Kantubhukta, "Toward an automated attack model for red teams," *IEEE Secur. Priv.*, vol. 3, no. 4, pp. 18–25, Jul. 2005. [Online]. doi: 10.1109/MSP.2005.111

[29] J. Steffan and M. Schumacher "Collaborative Attack Modeling" P*roc of the ACM symposium on Applied Computing, Madrid, Spain*, 2002, pp 253–259 [Online]. doi: 10.1145/508791.508843

[30] X. Qiu, S. Wang, Q. Jia, C. Xia, and Q. Xia, "An automated method of penetration testing," *Proc of the IEEE Computers, Communications and IT Applications Conference, Beijing, China*, 2014, pp. 211–216 [Online]. doi: 10.1109/ComComAp.2014.7017198

[31] N. A. Almubairik and G. Wills, “Automated penetration testing based on a threat model,” *Proc of the 11th International Conference for Internet Technology and Secured Transactions (ICITST)*, Barcelona, Spain, 2016, pp. 413–414. [Online]. doi: 10.1109/ICITST.2016.7856742

[32] K. P. Haubris and J. J. Pauli, “Improving the Efficiency and Effectiveness of Penetration Test Automation,” *Proc of the 10th International Conference on Information Technology: New Generations, Las Vegas, NV, USA*, 2013, pp. 387–391. [Online]. doi: 10.1109/ITNG.2013.135

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California