

Towards Scalable Automated Vulnerability Scanning & Exploitation

Submitted in partial fulfillment of the requirements for
the degree of
Master of Science
in
Information Security

Jarrett A. Booz

B.S., Computer Science, Towson University

Carnegie Mellon University
Pittsburgh, PA

April, 2020

Acknowledgements

I'd first like to thank Dr. Vyas Sekar for advising me on this project. Your support is appreciated.

Matthew McCormack, thank you for your support as well. I appreciate your willingness to meet with me several times per week and guide my development and writing process.

Thank you to CMU CyLab for funding for the vulnerable IoT devices that were used in the experiments.

Next, I'd like to thank those that got me where I am today.

First and foremost, to my parents and my sister, thank you for your continued support not only in my academic journey, but in my life.

To Joe Foltz and Harford Technical High School, thank you for beginning to foster my interest in learning about cyber security early in my academic career. Without Mr. Foltz discovering cyber security competitions for high school students and encouraging us to succeed in them, I'm sure I will have been set back quite a bit from where I began my cyber security path.

To Dr. Mike O'Leary and the Towson University Computer Science program, thank you for your dedication to setting students up for success in this field. Dr. O'Leary, without your continued support for the cyber security team at TU and the drive to learn as much as possible that you instill in your students, I and many other students would not be where we are today.

Thank you again to all of my friends, instructors, and family members for the support. Your actions are greatly appreciated.

Abstract

Offensive security assessments, where expert hackers attack a network to document vulnerabilities that can be exploited, are one way to approach network security. These assessments provide useful insights but can often be time consuming and expensive. Automating offensive security assessments can decrease time and monetary expenses. Current works on automating offensive security assessments focus on exploitation or post-exploitation actions, but not both.

We present the Scalable Automated Vulnerability scanning & Exploitation Tool (SAVE-T) which is comprised of additions to an existing automated tool for offensive security assessments. The additions made here add support for various architectures, exploitation capabilities, service fingerprinting, and enhanced decision making conditions. SAVE-T allows an automated system to perform both exploitation and post-exploitation actions to provide more coverage and reliability for automated offensive security assessments in a scalable manner.

We demonstrate the tool on networks of increasing size containing Windows workstation operating systems from XP to Windows 10, Windows server operating systems from Server '08 R2 to Server 2016, Ubuntu, and IoT devices such as Raspberry Pi and IP cameras.

Table of Contents

Acknowledgements	ii
Abstract	iii
List of Tables	x
List of Figures	xii
1 Introduction	1
1.1 Motivation	2
1.2 Goals	3
1.3 Contributions	4
2 Background	7
2.1 Vulnerabilities	7
2.1.1 Industry Standard Repositories	8
2.2 Internet of Things	9
2.2.1 Common Attacks	10
2.2.2 Security State of the Art	10
2.3 Offensive Security Assessments	12
2.3.1 Benefits of Offensive Security Assessments	13
2.3.2 Automating Offensive Security Assessments	13
2.4 CALDERA - A Tool for Automated Offensive Security	14
2.5 Summary	18
3 Problem Overview	19
3.1 Current Limitations	19

3.1.1	Limitations of Offensive Security Assessments	19
3.1.2	Limitations of Automated Offensive Security Assessment Tools	20
3.2	Goals	21
3.3	Challenges	23
3.3.1	Interoperability	23
3.3.2	Emulate Professional Adaptability	23
4	SAVE-T Implementation Overview	25
4.1	Contributions for (G_1) - Integrating Exploits	26
4.1.1	Metasploit Support	26
4.1.2	Custom Exploits	29
4.2	Contributions for (G_2) - Scalability & Reliability	32
4.2.1	Service Fingerprinting	32
4.2.2	Scalable Exploitation	33
4.3	Contributions for (G_3) - Extending the Operation	34
4.3.1	Action Planner	37
4.3.2	Lateral Movement Abilities	38
4.4	Contributions for (G_4) - Diverse Device Support	39
4.4.1	Diverse Device Support	39
4.5	Summary of Abilities	41
4.6	Limitations	44
4.6.1	Services on non-standard ports	44
4.6.2	Architecture Recognition	45
4.6.3	Metasploit Requires Linux With Dependencies	46
4.6.4	Current Support	47
4.6.5	Risk of Target Crash	48
5	Evaluation	49
5.1	Testbed	49
5.1.1	Scenarios	51
5.1.2	Adversaries	53
5.2	Experiment Results	53

5.2.1	Time	54
5.2.2	Scalability	57
5.2.3	Reliability	58
5.2.4	Summary	59
5.3	Top Port Coverage	60
5.4	Comparison to Other Tools	60
5.4.1	SAVE-T vs. Hail Mary Exploitation	61
5.4.2	Capability Comparison	65
5.5	Evaluation Summary	67
6	Future Work	69
6.1	Attack Graphs	69
6.2	Automated Defenses	71
6.3	Plugin Development	72
7	Conclusions	73
7.1	Use Cases	73
7.2	Design Decisions	74
7.3	Final Thoughts	76
	Bibliography	78
	Appendix A Metasploit Variables	91
	Appendix B Device Details	93
B.1	Apache Struts	93
B.2	ThinkPHP	94
B.3	Trendnet TV-IP410W	94
	Appendix C Experiment Hosts	96
	Appendix D Metasploit Exploit Times	98
	Appendix E Number of Exploits	102

List of Tables

Table 3.1	This table maps limitations of current tools to goals we have . .	22
Table 4.1	This table maps goals we have to the coming sections in which our specific contributions are discussed.	25
Table 4.2	The number of exploits in Metasploit, filtered by port number.	33
Table 4.3	The number of port 80 exploits in Metasploit filtered by key- words. Note some results overlap (e.g., Windows and IIS). . . .	33
Table 4.4	Illustration of agents which are tasked to continue an operation	37
Table 4.5	This table summarizes the abilities SAVE-T added to CALDERA. Percentages rounded to the nearest 1%	41
Table 4.6	This table shows the number of CALDERA and SAVE-T abil- ities and their supported operating systems. Percentages are rounded to the nearest 1%.	43
Table 4.7	This table shows the number of CALDERA and SAVE-T abili- ties and their ATT&CK Framework Category. Percentages are rounded to the nearest 1%	43
Table 4.8	Port numbers and their assumed standard services that SAVE-T supports as a proof of concept.	45
Table 5.1	List of hosts, their open ports and known vulnerabilities. Red shows Vulnerabilities that are part of Metasploit exploitation. Blue shows SAVE-T custom exploits. \wedge indicates machines that are joined a common Windows domain	50
Table 5.2	This table shows the number of hosts, the number of open ports, and the device types that were on the network for our 6 tests. .	51

Table 5.3	This table outlines the abilities for each of the adversaries that were used for testing. The “Full Adversary” is simply an expansion to the abilities of the “Windows Only Adversary” . . .	52
Table 5.4	Shown are the abilities and their time to run. Metasploit abilities’ run times are dependent on the number of exploits to be run. Brute Force time is dependent on if credentials are found - worst case is 4 seconds if no credentials are found.	54
Table 5.5	This table shows vulnerabilities found in each of our 6 experiments. Shown is the name of each experiment, the number of MS 17-010_psexec vulnerabilities and the other vulnerabilities which are on the network. The minimum and maximum number of vulnerabilities found by SAVE-T is shown for both the conditional MS 17-010_psexec vulnerability and the other vulnerabilities.	58
Table 5.6	Top 20 open ports on the Internet and SAVE-T’s coverage (Data sourced from Censys [14])	61
Table 5.7	Shown is the top 10 exploit ports on Exploit-DB [74] and SAVE-T’s coverage of these ports.	61
Table 5.8	This table compares the number of exploits that are attempted by Armitage Hail Mary and SAVE-T.	62
Table 5.9	This compares Armitage Hail Mary to SAVE-T with respect to time taken to complete an experiment, the number of hosts that completed the experiment and the number of exploits found. The difference between the number of hosts that were tested and the number of hosts that completed the experiment is the number of hosts which timed out. “Complete” means that the host launched all discovered exploits without timing out.	63
Table 5.10	This table compares the automated capabilities of CALDERA, Hail Mary, and SAVE-T	65

A.0.1	Shown here are the Metasploit exploit variables and their pre-defined default values. Variables with the value of <i>dynamic</i> are variables which are filled in with command line options at run time.	92
C.0.1	This table shows the hosts that were included in each of the experiments with the open ports to the right. RPi is a host running on a Raspberry Pi. ^ indicates that the host is joined to the Windows Domain	97
D.0.1	Shown are the open ports and services for the Metasploitable 2 VM [84]	100
D.0.2	Seconds per Metasploit exploit for the exploits which were chosen based on our network systems. Total Seconds is sourced from the SAVE-T exploit run time data presented in Table 5.9. Exploits Attempted is sourced from the SAVE-T exploits attempted data presented in Table 5.8.	101
E.0.1	Shown is the port and optional keyword filters that are used by SAVE-T and the number of Metasploit exploits that will be attempted as a result.	102

List of Figures

Figure 1.1	Filtering exploits can lead to more scalable exploitation.	5
Figure 2.1	Shown is an example offensive security assessment scenario. (Image from [71])	12
Figure 2.2	A visualization of the CALDERA vocabulary is shown. An ‘Operation’ carries out all the ‘Abilities’ that a given ‘Adversary’ knows how to perform. The ‘Abilities’ can both require and produce various ‘Facts’.	15
Figure 2.3	The CALDERA Server is giving instructions to the Agent on Host A. The Agent on Host A is carrying out the attack instruc- tions and is able to expand the attacker’s foothold.	17
Figure 4.1	Shown is a sequence diagram workflow for a sample operation. Items in black boxes are part of Phase 1 of this operation. Items the red box are part of Phase 2. Items in the blue boxes are part of Phase 3. The green line shows where CALDERA would stop the operation and SAVE-T continues.	35
Figure 5.1	Observed time to complete an operation in relation to the num- ber of open ports on the network with a best fit line.	55
Figure 6.1	Shown is a section of a sample attack graph. The directed graph shows how an attacker with network access to a host running a vulnerable program can exploit the program.	70

F.0.1	Shown is a sample attack graph. The directed graph shows paths that an attacker can take from a starting position on the same subnet as the target device to executing code on the target device. The network illustrated here consists of 4 nodes which all have code execution vulnerabilities.	104
-------	---	-----

1

Introduction

Assessing the security of a computer network is an essential part of network operations. There are various assessment strategies that can help network administrators understand where security vulnerabilities in the network may exist. One such strategy is a “hands-on” approach to network security where security professionals perform attacks on the network to uncover underlying security vulnerabilities.

A hands-on approach to network security can use offensive security assessments to actively find and leverage security vulnerabilities on a network in an attempt to emulate a real-world attacker. Offensive security assessments, however, can take prohibitive amounts of time and money to complete.

The challenges related to prohibitive requirements, such as time and money, can make it difficult for network administrators to fully utilize hands-on network security strategies. This thesis presents the Scalable Automated Vulnerability scanning & Exploit Tool (SAVE-T), which addresses some of the challenges for hands-on network security strategies by automating processes related to offensive security assessments.

1.1 Motivation

Today’s enterprise networks are composed of 1000s of hosts, often representing many different types of devices (e.g., PCs, IoT, printers, switches/routers, etc.). Security assessments of large networks can be difficult to perform because of several prohibitive costs (e.g., time and money). This difficulty has resulted in a call for an automated tool for performing these complex assessments, thereby lessening some of the costs (i.e., an automated tool should reduce time and money costs) [8, 52].

Some semi- or fully automated tools for carrying out portions the offensive security process have been demonstrated [66, 87, 93, 106]. Some works theorize how an automated attacker could exist [106]. Others work towards automating portions of testing such as attack path visualization [87] and attacks/scanning for a specific subset of devices [66, 93].

More closely aligned with our vision is a tool called CALDERA [4, 52]. The CALDERA tool provides a framework for running attacker actions on hosts around a network by utilizing a central command and control server with distributed agents. The server will task agents to perform attacker actions and the agents will report action success, failure and information about the network. The server uses a basic planner and the gathered information about the network to determine which actions it has sufficient knowledge to perform next.

CALDERA provides several functions for finding information about hosts on the network and using credentials to login to hosts but lacks the ability to perform vulnerability exploitation.

1.2 Goals

We aim to lower the prohibitive costs of performing offensive security assessments¹. We present a tool that will automatically gather data variables for and carry out attacks on the network (i.e., eliminating the need for an expert to do so manually).

More specific limitations of the current tools will be discussed in §3. We aim to address some of the limitations of current automated offensive security assessments by building upon current tools to expand capabilities. By adding features to a current state-of-the-art tool (CALDERA), we will demonstrate automated testing for software exploits and other vulnerabilities.

The current CALDERA tool provides a framework for automated offensive security assessments and stands as the base for our implementation. We will present a Scalable Automated Vulnerability Scanning and Exploitation Tool (SAVE-T). SAVE-T aims to address current limitations of CALDERA by adding capabilities which allow for more offensive security actions to be automated in addition to more device types being covered.

In developing SAVE-T, we have the following goals:

- (G_1) Integrate exploitation of vulnerabilities that are included in industry standard tools such as Metasploit [86] and Exploit-DB [73].
- (G_2) Implement capabilities in a scalable and reliable manner.
- (G_3) Use post-exploitation actions to extend the operation beyond a single epoch.
- (G_4) Include IoT devices in the scope of the security assessment.

¹ We use the term “offensive security assessment” to mean an ethical hacking test of a network to determine if security vulnerabilities are present. Other works may use “penetration testing” or “red teaming” interchangeably, but we will use “offensive security assessment”.

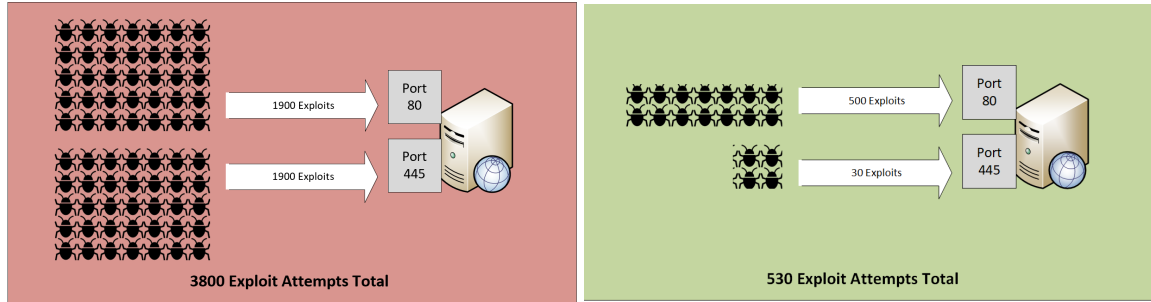
In summary, SAVE-T aims to be a low-required-cost (e.g., time and money) tool which provides the practical benefits of a hands-on offensive security assessment. In doing so, we can increase the circumstances under which offensive security assessments are used. We will demonstrate the tool on a network which contains various types of hosts, including IoT devices, which are popular on networks today.

1.3 Contributions

In our work we present Scalable Automated Vulnerability scanning & Exploitation Tool (SAVE-T). SAVE-T is comprised of additions and changes to the existing CALDERA tool to expand versatility. Our contributions include:

- Adding support for CALDERA to perform service exploitation with the Metasploit Framework [86] and proof of concept custom code (based upon entries in Exploit-DB [73]).
- Performing the exploitation in a scalable and reliable manner.
- Allowing the automated attacker to perform more than one epoch of the planned actions.
- Proof of concept support for including IoT devices in the scope of an automated offensive security assessment.

Exploits that can be used with SAVE-T may be: a) part of the industry standard Metasploit Framework [86], b) an exploit script available via industry standard databases such as Exploit-DB [73], or c) any other custom code that the user wishes to include in the scope of their assessment. Currently, the 1,900+ exploits in the Metasploit Framework [86] are supported by SAVE-T. The over 41,000 exploit code samples available on Exploit-DB [73, 74] and any other custom code samples require



(a) All exploits are tried on all ports. Approximately 1,900 exploits attempted.

(b) Exploits only attempted on open ports. Approximately 530 exploits attempted.

Figure 1.1: Filtering exploits can lead to more scalable exploitation.

a small amount of work by an expert to be incorporated into the tool. Between these sources, there are tens of thousands of potential vulnerabilities that SAVE-T can test for. New exploits can also be added to SAVE-T via Metasploit software updates or new custom code added to the tool’s repertoire by an expert.

SAVE-T can gather information from hosts which will aid in deciding which exploits to attempt. Contextual information can include the ports that are open on a target host and what service is running on a port. This information will greatly limit the search space for potential attacks and therefore increase the scalability of exploit testing. For example, if a host is running an Apache Struts web application on port 80, SAVE-T can carry out two different tests: 1) run exploits which have been shown to work with a default port of 80 or 2) run exploits which are for Apache Struts web applications. Both will greatly limit the exploit search space and make automated exploitation more scalable. For example, there are over 1,900 total exploits in the Metasploit Framework [86]. Filtering these exploits by port 80 gives approximately 500 exploits. Further filtering the port 80 exploits to only include Apache exploits yields 21 results. This is a scale down of about 95x by filtering exploits by a keyword based on the service fingerprint. This concept is illustrated in Figure 1.1 where exploits are scaled down from 1,900 exploits (Figure 1.1a) to only those exploits

which work on the desired port (Figure 1.1b).

The use of offensive security assessments is critical to network security because they demonstrate attacks, rather than simply ticking a box to say that a vulnerability might exist [115]. Offensive security assessments are often costly and time consuming, though [4, 72, 115]. The exploitation features SAVE-T provides allow network operators to fully employ hands-on features for network security assessments with reduced requirements of time and cost that have been prohibitive.

SAVE-T addresses limitations of current tools (discussed in more detail in Chapter 3) by including thousands of exploits in the testing repertoire. SAVE-T also introduces a more scalable attack process by limiting the attack space using contextual information about a host. Lastly, SAVE-T addresses the need for experts to manually interact with exploits by automating these processes.

2

Background

This section will provide background information about key concepts in this thesis including security vulnerabilities and exploits, internet of things security, and offensive security assessments.

2.1 Vulnerabilities

Vulnerabilities in computer systems are software problems that allow an attacker to alter the system's functionality, leading to bypassing security measures or compromising the system [48, 80]. Vulnerabilities can be inadvertently introduced or deliberately introduced into software [41]. Exploits are code which take advantage of vulnerabilities [48]. Exploits which allow remote attackers to run arbitrary code on vulnerable machines, called Remote Code Execution, are especially attractive to attackers because it allows them to load their own custom code for a vulnerable machine to execute [32].

Examples of inadvertent vulnerabilities are poor checking/sanitation of user input and insufficient checking of data boundaries. Poor user input handling can allow

command injections, where an attacker can supply arbitrary commands for a target computer to execute. Insufficient checking of data boundaries can cause overflows which alter program execution [41].

One example of a deliberate vulnerability is called a trapdoor or a backdoor [41]. Backdoors are often introduced purposefully by the developers to bypass normal execution/authentication, often for expedited testing purposes [41]. A common manifestation of this type of vulnerability is a default username and password for a system. Default credentials can ease testing and usability but can also provide a remote attacker authenticated access to the device, often at a high privilege level.

Exploiting software vulnerabilities is not a new concept. Attacks using backdoor accounts and Distributed Denial of Service (DDoS) attacks have been noted by researchers since 2000 [92]. Recent studies show that these types of attacks are still seen. In 2017, default credentials were used to compromise hundreds of thousands of devices and subsequently launch DDoS attacks [2]. Though attacks seen today are often of the same type seen for decades, their sophistication has increased [80].

2.1.1 Industry Standard Repositories

Until late 1999, software vulnerabilities were independently discovered and documented. Today, the MITRE Corporation’s Common Vulnerabilities and Exposures (CVE) Initiative, proposed in January 1999, attempts to centralize the documentation of vulnerabilities [44, 48]. This resulted in the creation of an industry standard database of known vulnerabilities, launched in September 1999 and known as the CVE list [47]. Here, vulnerabilities are assigned names and numbers by worldwide authorities [47, 56].

The Exploit Database (Exploit-DB) [73] and the Metasploit Framework (Metasploit) [86] are industry standards for vulnerabilities and vulnerability exploits. Exploit-

DB is a database kept by the Offensive Security company which houses over 41,000 known exploits for publicly released software [73, 74]. The Metasploit Framework (Metasploit) is a free offensive security tool released by the Rapid7 company [83] which provides over 1,900 fully interactive exploits and exploit payloads [86]. 1,761 of the 1,900+ Metasploit exploits have overlapping entries in Exploit-DB.

By default, exploits in these standard sets require manual interaction, meaning that an expert who is familiar with the exploit must set the required input variables, launch the exploit and interact with any potential output. Prior work looked at automating Metasploit to launch exploits using default variables [28]. However, this work does not take a scalable approach to automation. Rather, it attempts every Metasploit exploit module on all target hosts and ports, potentially wasting time attempting exploits for ports not open on the target.

Armitage, a hacking tool released by Strategic Cyber LLC [102], has a feature called “Hail Mary” which will automatically attempt Metasploit exploits against hosts. This tool, however, also has scalability/reliability concerns when attacking more than 2 or 3 hosts simultaneously. When attempting to use this tool on 4 or more hosts, the database it stores host records in becomes overwhelmed with requests which leads to most actions timing out.

2.2 Internet of Things

The Internet landscape is constantly changing as new types of devices and services are connected. The Internet of Things (IoT) is a recent innovation that will allow millions or billions more devices to be connected to the Internet [49, 64]. This technological advance provides value to consumers and businesses, but also comes with security vulnerabilities that must be considered.

IoT connected devices can collect and transmit data to services and other devices

which are working to achieve a common goal [79, 90]. IoT devices are becoming vital parts of the Internet as they continue to gain traction in various consumer and industrial networks [34]. It is estimated that more than 40 billion embedded devices that make up the IoT will be connected to the Internet by 2020 [49]. IoT devices commonly use popular web frameworks to integrate with cloud services [23, 31]. Web framework integration can introduce vulnerabilities into the IoT ecosystem and open devices up to attack.

2.2.1 Common Attacks

Studies show that IoT devices are increasingly the target of thousands of cyber-attacks [53, 119]. Notably, the Mirai botnet took advantage of vulnerabilities in IoT devices (such as default credentials for exposed services) to build a massive network of attacker-controlled nodes which helped orchestrate Distributed Denial of Service (DDoS) attacks [2, 6, 88]. Mirai evolved and spawned variants of the malware which often exploited vulnerabilities in popular web frameworks in addition to the default credentials that the original malware primarily used [5, 6, 21, 40, 88].

2.2.2 Security State of the Art

IoT security has several areas of research, including vulnerability assessments of IoT devices and defenses for IoT devices. We will discuss these areas and provide a summary of the current state of IoT security.

Vulnerability Assessments

Researchers have been able to demonstrate attacks on IoT Systems [7, 22, 38, 99]. Additionally, surveys of the number and types of attacks and vulnerabilities that appear on real-world systems “in the wild” have been conducted [2, 116]. Common

vulnerabilities from these works include weak default credentials, denial of service attacks, spoofing attacks, and code execution vulnerabilities [2, 116]. The prevalence of vulnerabilities in IoT systems in addition to their growing presence in computer networks has prompted an observed need for an automated vulnerability detection tool for IoT devices [45].

The Open Web Application Security Project (OWASP) is working to keep track of the most seen categories of web vulnerabilities [78] and is working on an automated program for scanning websites for vulnerabilities [77]. Their automated tool may work for scanning IoT devices which utilize web components.

Defenses

Research is looking at various ways to secure networks of connected IoT devices [20, 90]. One approach for securing IoT networks focuses on developing a common gateway that IoT devices can communicate with to establish trusted communications and access control [1, 10, 15, 26]. Another research approach focuses on enhancing cryptographic mechanisms such as protocols and random number generation for IoT devices [105, 113].

Summary

IoT devices have vulnerabilities such as default credentials and vulnerable web frameworks. Defending IoT devices is an active area of research. Today, we are still in need of automated ways to discover vulnerabilities in IoT devices and effective security mechanisms for these devices.

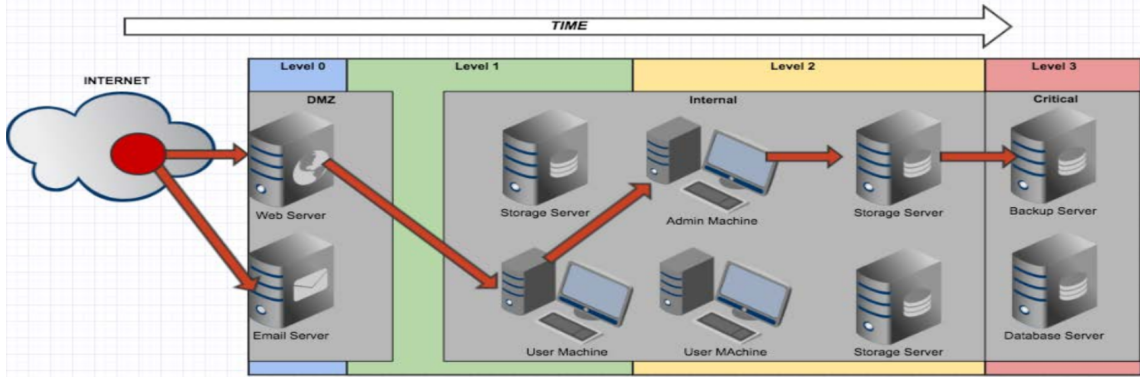


Figure 2.1: Shown is an example offensive security assessment scenario. (Image from [71])

2.3 Offensive Security Assessments

Offensive security assessments are procedures where hired experts aim to identify and exploit security flaws in computer networks before real-world attackers exploit the same systems for malicious purposes [48, 71]. These assessments can be performed by companies or government entities that provide assessments as a service to disclose systems' vulnerabilities [48]. Assessments range in methodology, including assessments such as penetration tests and red team or adversary emulation tests [52, 71]. Penetration tests search for any exploitation opportunities in a network [52]. Red team or adversary emulation tests – employing the tactics, tools, and procedures of a specific adversary in order to accurately emulate a real-world attack scenario [52]. Regardless of the type of test an organization may utilize, these security assessments can find flaws or gaps in security policies or network infrastructure, which may have otherwise gone unnoticed [4, 118]. Offensive security assessments have proven to be successful in identifying security flaws in various applications, such as networks of traditional workstations and servers [71], web programming [13], automotive computer networks [39], network printers [65], and cyber physical systems such as water treatment plants [37].

Figure 2.1 shows an example workflow of an offensive security assessment. In this example, an attacker from the Internet compromises several servers on a network DMZ, then pivots, or use other hosts as stepping stones, through other network nodes to a critical asset. This example demonstrates how vulnerabilities existing in several layers of the network architecture can lead to compromise of a critical asset.

Offensive security assessments can be expensive due to the time and level of expertise that they require [4, 27]. Research has suggested a need to automate offensive security assessments to address the cost and time challenges faced when utilizing them [4].

2.3.1 Benefits of Offensive Security Assessments

Offensive security assessments stand opposite the traditional approach to network defense which often ask subjectively how to secure a network (e.g., examining network maps to determine firewall placement) [4]. These assessments actively find vulnerabilities that may defeat assumptions made by a network designer and will provide a concrete demonstration of attack impact [4, 8]. The concrete demonstration of attacks is an important pro-active step that organizations can employ to find weaknesses or misconfigurations before they are exploited by a real-world attacker [71]. These demonstrations provide a complete view of what steps an attacker was able to follow to carry out an attack. Utilizing offensive security assessments repeatedly after applying mitigations can address this by finding existing attack paths each time new countermeasures are employed [8].

2.3.2 Automating Offensive Security Assessments

Ray et. al have used pre-defined attack strategy information in programmatically readable formats to bring attacks closer to being fully scripted [87]. In this work,

experts will create pseudo-code scripts for penetration testing actions. Similar work by Tilemachos and Manifavas uses scripted actions for penetration tests [106]. A limitation of these works is the requirement to know which actions need to occur and when.

Another tool which has automated exploitation through the Metasploit Framework is Armitage [102]. This tool has a feature called “Hail Mary” which will launch Metasploit exploits at desired targets. A limitation of this tool is that it requires users to have scanned the host previously and have knowledge of the operating system which the target is running. Additionally, there are scalability and reliability issues when using Armitage’s Hail Mary to target more than a few hosts at a time. This is due to the database not being able to handle the number of requests that are coming via the tool.

2.4 CALDERA - A Tool for Automated Offensive Security

In order to improve the realism of automated red team assessments a tool should be able to learn from gathered information and make complex decisions [9, 52]. Work from the MITRE Corporation focuses on automated post-exploitation actions of known adversaries [4, 52]. In this work, the Cyber Adversary Language and Decision Engine for Red-team Automation (CALDERA) tool was developed to automate making decisions around ‘atomic actions’ (i.e., single instructions) [4]. This tool uses the Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) Framework [58] – developed by MITRE to map standard actions taken by known adversaries - and a decision-making engine to automate sequences of atomic actions, such as dumping system credentials and using credentials to login to remote systems [4].

We consider CALDERA to be the current state of the art for automated offensive

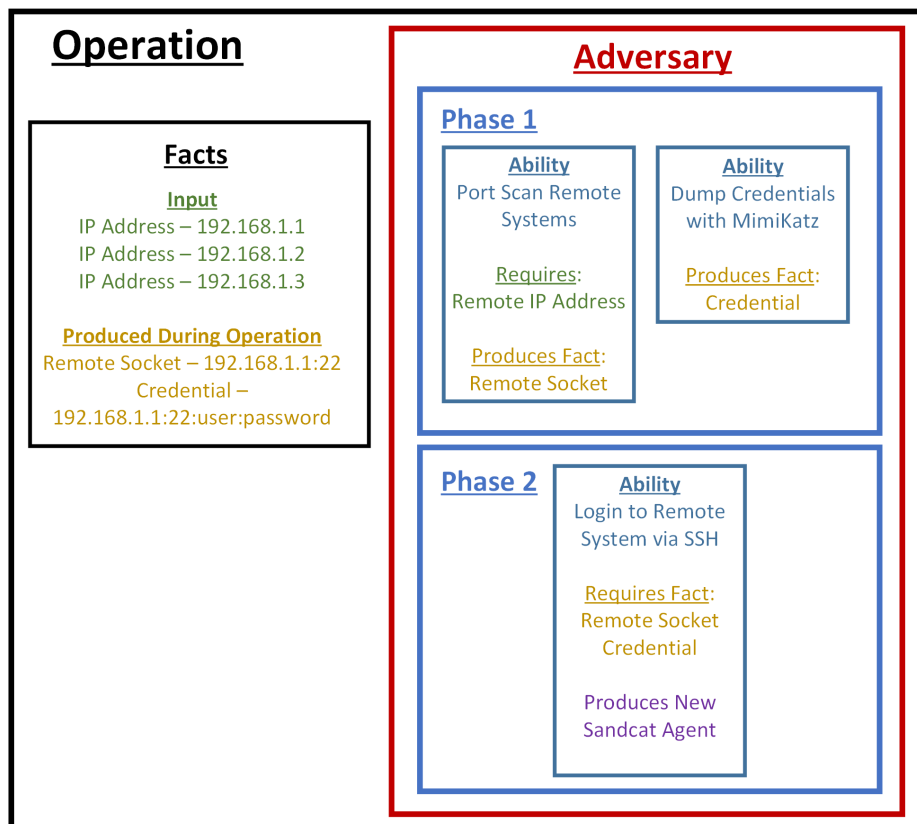


Figure 2.2: A visualization of the CALDERA vocabulary is shown. An ‘Operation’ carries out all the ‘Abilities’ that a given ‘Adversary’ knows how to perform. The ‘Abilities’ can both require and produce various ‘Facts’.

security assessments. We make this judgement based on the tool’s ability to gather knowledge from distributed agents on the network and to make decisions regarding which actions to take.

CALDERA is an open source tool¹ that ships with the ability to automatically discover remote systems, dump the credentials of compromised systems, and log into remote systems with discovered credentials. These ‘abilities’, are provided in the tool’s Stockpile plugin [61].

Abilities can be executed on remote systems via a remote access trojan (RAT or agent) called **54ndc4t** (Sandcat) [60]. A plugin provides executable versions of the

¹ Code for the CALDERA tool is available on GitHub [62]

agent. The agent communicates to the central command and control (C2) server to receive instructions and report results.

Abilities can be combined to form the attack plan of an ‘adversary’. An adversary can carry out an ‘operation’ in which the decision-making engine will perform the actions in the adversary’s list of abilities.

The engine will gather more knowledge (‘facts’) while carrying out an operation. This knowledge is used to fill in required variables for future actions. For example, an ability that dumps system credentials will return username and password facts. The port scanner ability will return remote socket facts. Together, a remote socket, username and password can be used by a remote login ability. The remote login ability cannot be utilized without these required facts as input variables.

Figure 2.2 and Figure 2.3 illustrate an example operation. Figure 2.2 illustrates the relationship between CALDERA terminology for the example operation and Figure 2.3 illustrates the operation workflow over time.

The operation starts with some facts and has 2 phases. The first ability in Phase 1 is a port scan ability which requires an IP address fact to run. This ability will produce ‘socket’, or open port, facts. Another ability will produce credential facts. In Phase 2, an ability will use a combination of socket and credential facts to produce a new agent.

An operation performed using CALDERA’s ‘chain mode’ [59] begins with a minimum of 1 host on the network running a Sandcat agent and optionally includes some facts the system knows at the start of the operation. The initial Sandcat agent(s) will receive instructions to carry out actions that either expand the foothold or the knowledgebase of the emulated attacker.

The sample CALDERA operation workflow is shown in Figure 2.3. In this example scenario, the CALDERA server is communicating with an agent that exists on Host A at the beginning of the operation. The agent is first tasked to port scan

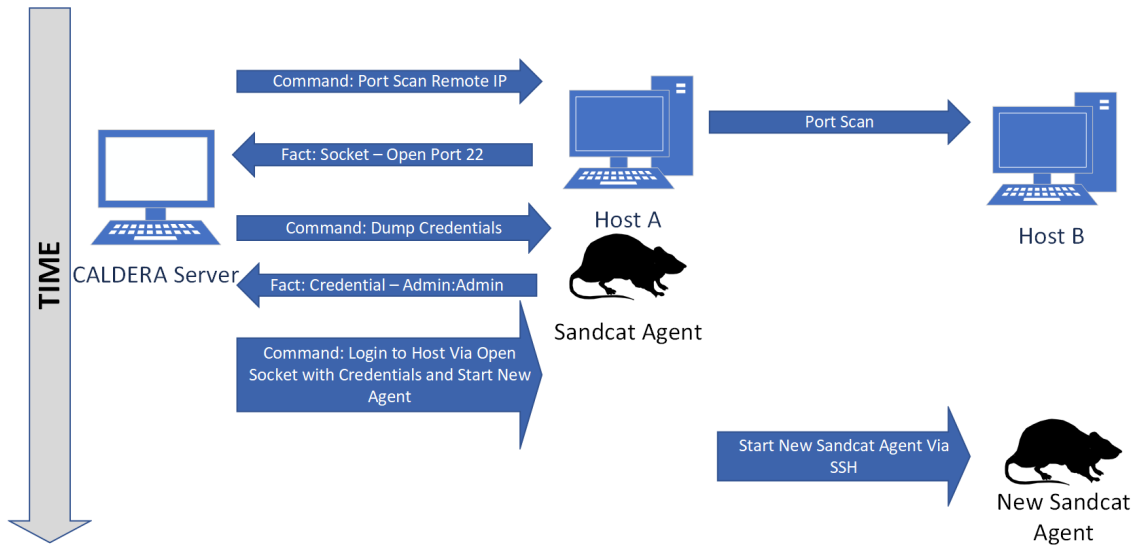


Figure 2.3: The CALDERA Server is giving instructions to the Agent on Host A. The Agent on Host A is carrying out the attack instructions and is able to expand the attacker’s foothold.

Host B and returns a fact which indicates that port 22 is open on the remote host. The agent is then tasked to dump local credentials (the method of doing so is not important here). Dumping credentials produces a fact which indicates that the username/password combination ‘Admin/Admin’ exists on the network. The agent is then tasked with attempting to use those credentials to login to Host B via the open SSH port and start a new agent.

We can call one complete run through of all abilities and all phases one epoch of the operation. In other words, when agents have each run all of the abilities in the operation one time, this is one epoch of the operation. In CALDERA, an operation will stop execution after one epoch. This is the case even when an ability spawns a new agent which has not yet completed all of the abilities in the operation.

Looking at Figure 2.3, the operation will finish execution after the agent has been spawned on Host B. The agent on Host B has not yet been tasked to perform abilities, such as dump credentials. Tasking the new agent to perform all steps of the operation would produce more complete results. Premature operation termination

(i.e., after one epoch) is a limitation of CALDERA.

2.5 Summary

This section discussed security vulnerabilities and exploits, IoT security and offensive security assessments. The CALDERA tool was also detailed as the current state-of-the-art tool for automated offensive security assessments that aligns with our goals. We will detail some limitations of works introduced in this section and define the problem we aim to solve in the next chapter.

3

Problem Overview

This section will provide an overview of the problem this thesis aims to solve. We will state the limitations of current works and define the goals we aim to accomplish by developing SAVE-T. We then present challenges related to accomplishing the goals.

3.1 Current Limitations

We will discuss limitations of several of the tools that were introduced in §2. We will discuss diversity limitations of offensive security assessments and versatility and scalability limitations of automated offensive security assessments.

3.1.1 Limitations of Offensive Security Assessments

Offensive security assessments often have prohibitive requirements (e.g., monetary expense and time) due to the fact they rely on security experts to carry out the exercises [4]. The experts conducting the assessments are often expensive to hire or contract due to the high level of training needed to become an expert in the field [4]. Estimates show that there will be over 3.5 million cyber security job openings by 2021 and that fewer than 1 out of 4 applicants to these positions will be qualified

for the job they are applying to [117].

Additionally, vital parts of an offensive security assessment (e.g., port/vulnerability scanning of networks, learning how to interact with services, etc.) can take weeks for experts to complete [4, 70]. These actions take a lot of time because full network scans are time consuming. Additionally, experts may run into several dead ends while testing a network which consume their time [70].

Research shows that offensive security assessments should be performed several times via the same attack strategy to test the robustness of defensive mitigations [8]. Repetition is also useful because test results can be quickly invalidated due to new exploits and attack techniques that are used by evolving adversaries [55, 71, 103]. Because repetition of offensive security assessments is required to maintain updated results, monetary and time costs are exacerbated. To mitigate exacerbated costs, automated assessments are suggested [52].

There is an additional limitation when testing IoT devices. The need for an automated system to test the security of IoT devices has been stated [45] but has not yet been fulfilled. To address these limitations, an automated system should be produced where the tool will demonstrate vulnerabilities in IoT devices.

3.1.2 Limitations of Automated Offensive Security Assessment Tools

Current tools that automate offensive security actions are limited in their versatility and scalability.

CALDERA version 2.3.0, released August 23, 2019, [62] mainly focuses on post-exploitation actions such as dumping system credentials from processes/files and using credentials via various protocols (e.g., SSH, WinRM, WMI, etc.). Because of the current focus we find CALDERA to be limited in that the system cannot perform attacks along the lines of traditional vulnerability exploits. Its default abilities

resemble finding and using credentials. This automated logging-in may resemble some adversary actions but does not provide the ability to demonstrate exploitation actions (e.g., exploit a vulnerability in an outdated application) which pose a significant threat to networks. There are thousands of attacks on systems documented in the CVE Database [56], the Metasploit Framework [86] and in the Exploit Database [73] that pose a risk to systems. A more complete toolset would include the capability to assess networks for potential susceptibility to the vulnerabilities contained in these datasets.

As introduced in §2.4 with Figure 2.3, CALDERA is not able to extend operations beyond one epoch. This limitation does not allow new agents which are spawned by abilities during an operation to participate. Because of this, operations may not produce complete results.

Version 2.3.0 of CALDERA takes a naïve approach to attempting actions – the system will try every attack it is instructed to attempt on every remote host and port it discovers. For example, if the system is instructed to attempt executing a program via SSH, the system will attempt this $N * M$ times, where N is the number of hosts the system knows about and M is the number of ports each host has open. A more sophisticated approach will only attempt to execute programs via SSH if a remote host is running SSH. This approach requires fingerprinting of services in addition to storing more detailed information about services running on remote ports.

Other tools which can automate launching exploits such as Armitage have scalability and reliability problems when targeting more than 3-4 hosts, resulting in incomplete data.

3.2 Goals

SAVE-T aims to address some of the limitations presented and accomplish the specific goals (G_1) – (G_4) introduced in §1.2. Current works’ limitations are mapped to our

Limitation	Goal
CALDERA cannot exploit hosts using Metasploit/Exploit-DB	(G_1) Add mechanisms to automate exploitation of vulnerabilities that are included in industry standard tools such as Metasploit and Exploit-DB
CALDERA does not implement abilities in a scalable manner Armitage does not perform well when more than a few hosts are targeted	(G_2) Implement capabilities in a scalable and reliable manner
CALDERA does not perform operations for more than one iteration	(G_3) Use post-exploitation actions to extend the operation
CALDERA and IoT exploitation works are not demonstrating an automated system which can support various IoT devices	(G_4) Include IoT devices in the scope of the security assessment

Table 3.1: This table maps limitations of current tools to goals we have

goals in Table 3.1.

(G_1) addresses CALDERA’s limited exploitation capability by adding exploitation capabilities with exploits from Metasploit [86] and Exploit-DB [73].

(G_2) addresses CALDERA’s scalability and Armitage’s reliability limitations by implementing capabilities in a scalable and reliable manner.

(G_3) addresses CALDERA’s single epoch limitation using post-exploitation actions to extend the operation beyond one epoch.

(G_4) addresses CALDERA and other works’ device diversity limitations by including various IoT devices in the assessment scope.

We will accomplish these goals by adding to CALDERA version 2.3.0 [62]. We will add CALDERA abilities to introduce new actions that the agents can complete, change the action planner to allow operations to continue on new agents after each epoch, and alter other processes in the CALDERA backend to facilitate the new features.

3.3 Challenges

Challenges faced when developing SAVE-T include interoperability with existing tools and wanting to emulate the adaptability of professionals.

3.3.1 Interoperability

When making an automated adversary tool, you can start from the ground up. This would produce a tool that meets every requirement you have in the best way for your design strategy. This would take an extraordinary amount of time to design and implement a new system from the ground up.

Instead, we decided to make use of existing tools that each have some of the capabilities we desire. By having existing tools work together, we remove the need to duplicate existing efforts.

Making tools, which were designed separately, operate together is not a trivial task. We must have each tool operate as it was intended while also meeting the purpose of our new design. This could involve accommodating operation limitations and other design factors that each other tool has, and altering existing tools' code to meet the new needs we define.

3.3.2 Emulate Professional Adaptability

When professionals are manually conducting offensive security assessments, they are able to gather information from actions they take and use them to inform future actions. Professionals can also adjust to situations where the tools they usually rely on are not available.

We wish to emulate this adaptability in our tool to enhance the versatility of automated offensive security assessments. To do so we can make use of CALDERA's fact gathering and planning capabilities to utilize information that is gathered during an operation. But we must also make additional enhancements so the tool can

operate on diverse architectures and perform actions without the assumption that certain tools (e.g., command line programs, packages, dependencies, etc.) are present on the target system.

SAVE-T Implementation Overview

This thesis makes several contributions to enhance the capabilities of the CALDERA tool. The goals (G_1) – (G_4) , first introduced in §1.2, are mapped to contributions we made in Table 4.1. The following sections will detail the contributions that were made in relation to each goal and summarize the proof of concept abilities that were designed as part of SAVE-T to realize our goals. We then address some limitations of our work.

Goal	Contribution
(G_1) Add mechanisms to automate exploitation of vulnerabilities that are included in industry standard tools such as Metasploit and Exploit-DB	§4.1.1 - Metasploit Support §4.1.2 - Custom Exploits
(G_2) Implement capabilities in a scalable and reliable manner	§4.2.1 - Service Fingerprinting §4.2.2 - Scalable Exploits
(G_3) Use post-exploitation actions to extend the operation	§4.3.1 - Action Planner §4.3.2 - Lateral Movement Abilities
(G_4) Include IoT devices in the scope of the security assessment	§4.4.1 - Diverse Device Support

Table 4.1: This table maps goals we have to the coming sections in which our specific contributions are discussed.

4.1 Contributions for (G_1) - Integrating Exploits

This section will detail contributions related to (G_1): Adding mechanisms to automate exploitation of vulnerabilities that are included in industry standard tools and databases (for example Metasploit and Exploit-DB). We realized this goal for Metasploit in §4.1.1 and custom exploits in §4.1.2.

4.1.1 Metasploit Support

Related to adding Metasploit support, we will discuss supporting the API and some challenges faced. We then discuss supporting the actual exploits that are coming from the Metasploit API.

Metasploit API

To add support for Metasploit the Sandcat agents need to be able to communicate with the Metasploit API. Because of this, the SAVE-T server startup code needed to be edited such that the Metasploit RPC server would start when starting other server components. Once running, abilities that perform Metasploit actions can communicate with the Metasploit API via the RPC server.

To interact with the Metasploit API, abilities run by agents utilize Python scripts and the PyMetasploit3 Python package [17]. This package allows Python programs to interact with the Metasploit API from Python code. In this way, it is easy to build Python objects which will then translate to Metasploit API calls to carry out exploits.

By having hosts interact with the Metasploit API, SAVE-T has the power to leverage any Metasploit exploit. This allows for more than 1,900 exploits to be used by the system as of release 5.0.70 of the Metasploit Framework [86].

A challenge we saw with the Metasploit API was the likelihood of queries to timeout when a large number of requests are coming in. This caused Metasploit

functions to timeout before the server was able to handle their queries. Typically, when more than about 3-4 hosts are querying the API simultaneously, the API would become overwhelmed with requests and abilities would start failing due to timeouts. We saw a similar problem with Armitage’s Hail Mary feature’s database as well, as described in §3.1.2.

The problem was seen in CALDERA due to the way Sandcat performed tasking. In the original design, agents would schedule all parts of a phase to occur in parallel. This means that if a phase planned to use Metasploit against 10 open ports, all 10 exploitation processes would occur at the same time. We addressed this placing a mutex on Metasploit abilities in the Sandcat agent. This limits each Sandcat agent to running only 1 instance of a Metasploit exploit ability at a given time. This coupled with a restriction that the Metasploit abilities should only run on the same host as the Metasploit and CALDERA servers, will ensure that the Metasploit API does not fail.

In this way, if there are 10 open ports to exploit with Metasploit, the exploitation will happen sequentially by port instead of in parallel. By open ports in this paper, we are talking about IP addresses and port pairs that form a socket. This increases the time it takes for the operation phase to complete, but improves the reliability of abilities executing successfully without timeouts. We think the trade-off of execution time for reliability here is worth the cost.

Metasploit Exploits

For most Metasploit exploits to work properly, there need to be variables filled in at run time. This is done programmatically by SAVE-T. SAVE-T has some pre-defined default variable values and will use dynamic information, such as the target IP address and port, for others.

We defined a set of default values based upon reasonable values that support

our primary set of key exploits. These primary exploits are those that were queried from the Metasploit API when developing the tool based on a number of port filters. When using these exploits, the program should not throw any unexpected exceptions due to required variables not being supplied to an exploit. Other exploits may have required variables which are not covered by the tool defaults and could therefore throw an exception. A default value can easily be added to the list of defaults in order to support any exploit. New exploits which are added in Metasploit updates can also be supported in the same way if they have new required variables.

There are currently 713 exploits (~37% of all exploits in version 5.0.60 of the Metasploit Framework [86]) that are tested to work properly with SAVE-T. These are exploits which have default port values of 21, 22, 23, 80, 443, 445, and 8080.

Appendix A enumerates the complete list of variables and their default values that are currently in SAVE-T. Any exploit that uses these required variables will be fully supported without any changes to the code required.

It should also be noted that an exploit may fail due to an incorrect default value. For example, if a vulnerable website exists at `http://1.2.3.4/my_website` but the default variables are set such that the exploit will target `http://1.2.3.4/`, the exploit will likely fail. This is a limitation of using such default values.

Variables which use provided information are those which set the target IP address, target port, server IP address, and server port. These are passed via command line arguments from the server to the ability. The target IP address and port are gathered from socket or service (SMB, SSH, etc.) facts collected by other abilities prior to the Metasploit ability being called. The server IP address is passed via a variable which instructs the agent running the ability to fill in the address where it knows the server to be. The server port is incremented for each exploit that is attempted as each exploit may require a different payload handler for a shell to be returned to Metasploit.

Currently, SAVE-T covers 28 out of the 48 possible required variables with either default or dynamically supplied values. Exploits which require variables outside of our set of 28 may not work properly.

4.1.2 Custom Exploits

There are thousands of software exploits that exists in databases such as Exploit-DB [73] that are not included in the Metasploit Framework’s repertoire of exploits. Exploits which are not included in the Metasploit Framework can still pose serious risk to network security.

Additionally, many attacks, such as the Mirai IoT botnet, target devices which are using default credentials [2, 6, 88]. In 2012, researchers found more than 1.6 million devices on the Internet which were vulnerable to attack via default credentials [11, 46]. According to Shodan.io, there were over 44,000 Internet connected devices which had default credentials in 2019 [96]. Because default credentials have been used in popular attacks and still have a considerable presence on the internet, we wish to test for the use of default credentials with SAVE-T. For our purpose, we will refer to abusing default credentials as an exploit.

To include various exploits as part of the automated testing that SAVE-T performs, we can include the exploit scripts as payloads of CALDERA abilities. The payload will be downloaded from the CALDERA server by the Sandcat agent when it performs the ability. This allows the agent to execute the exploit code against the target.

While implementing an exploit for use with SAVE-T we have the following two objectives:

(O_1) Determine if a device is vulnerable to the exploit

(O_2) Execute a new agent on the device so it can be used to continue the operation

We choose two web-application frameworks and one network connected device with known vulnerable firmware to demonstrate this capability. The web-application frameworks, Apache Struts [3] and ThinkPHP [107], can be run on any device capable of hosting a website or web-based API. We are running these services on Raspberry Pi devices. The remaining device is an IP surveillance camera which has a vulnerable web portal.

The chosen vulnerable devices and software represent devices that may be seen in real-world networks. Many IoT devices that are in use today utilize a web-based API or other web framework for operation [31].

The chosen frameworks and device types are also representative of those that were targeted by internet-wide botnet attacks such as Mirai [5, 6, 75, 95]. For more information about the specific software and devices we chose, see Appendix B.

We will next discuss the custom exploits that were implemented as part of SAVE-T.

Apache Struts and ThinkPHP

The exploit code-base used for the SAVE-T implementation of the Apache Struts exploit, authored by VEX WOO, was published in the Exploit Database [110]. SAVE-T uses an altered version of the exploit which includes a custom bash script payload that downloads and executes the Sandcat agent.

A nuance of this particular exploit is that the malicious web request containing the Sandcat script never fully returns an HTTP response – the request hangs due to the Sandcat process being spawned and not ending. To account for this, an exception handler was crafted to exploit the target again with a payload to check if Sandcat is running. While it might be valid to assume the first exploit completed successfully if the web request times out, additional certainty is gained by performing this low-cost (1 second) additional query.

The exploit code for the SAVE-T implementation of the ThinkPHP attack found on Exploit Database was authored by a user called VULNSPY [111, 112]. The exploit allows an attacker to provide an arbitrary URL encoded command that will be executed by the vulnerable system. The original payload was altered such that it executes a bash script that downloads and executes the Sandcat agent. Similar to the Apache Struts implementation, a second exploit is sent to verify the first attempt was successful.

With SAVE-T we are able to accomplish both (O_1) and (O_2) for these two web frameworks. Accomplishing (O_1) will produce a fact which indicates that a host is vulnerable to the exploit. Accomplishing (O_2) will result in a new agent being spawned on the vulnerable host. This new agent will then continue the operation with other new agents.

Trendnet TV-IP410W

The code-base used for the SAVE-T implementation of the Trendnet TV-IP410W exploit was authored by Charles Heffner and was released at the Black Hat security conference in 2013 [30]. This exploit code was edited to include a basic authentication header using a backdoor account's default credentials. The injected command is 'ls'. The code launching the exploit will search for a specific filename in the output to determine if the exploit was successful.

Because the Trendnet TV-IP410W device's shell is extremely limited, few commands can be run via the shell interface. Specifically, it lacked a command for downloading a file from a remote server (e.g., wget). This limited our ability to install the Sandcat Agent. Because we are not able to download and install the agent, we are only able to determine if the device is vulnerable to attack. Thus, (O_1) can be accomplished, but (O_2) cannot.

Default Credential Testing

SAVE-T supports testing for default credentials on three different services: telnet, SMB and SSH. These are ports which are commonly targeted by Mirai and other attacks [36, 68].

SAVE-T has a list of default credentials that are tested. This list is sourced from the leaked source code of the Mirai botnet [36]. This list can easily be changed to include additional username/password combinations.

We created three abilities which test for default credentials on SMB, telnet, and SSH. Each requires a specific service fingerprint match to be utilized. This prevents wasting time attempting to use the wrong protocol to login to an open port.

For these abilities, only testing to see if successful login can be achieved is completed. If the ability finds a username/password combination that works to login to a remote device, the connection to the device is terminated and the ability produces a fact which includes the IP address and port of the service and the username/password combination. Other abilities can make use of this credential fact to login to the service and attempt to start a new agent.

4.2 Contributions for (G_2) - Scalability & Reliability

This section will detail contributions related to (G_2): Implement capabilities in a scalable and reliable manner. We realize these goals with service fingerprinting in §4.2.1 and scalable exploits in §4.2.2.

4.2.1 Service Fingerprinting

Scalable exploitation should limit exploit attempts for a specific service to those hosts which are running that service. For example, it would not be scalable to test 100 hosts for SMB vulnerabilities if only 5 out of those 100 hosts are running SMB. Default

Port	Exploits
All	1947
80	504
8080	58
21	48
443	44
445	33
22	18
23	10

Table 4.2: The number of exploits in Metasploit, filtered by port number.

Keyword	Exploits
All Port 80	504
Unix	151
Windows	135
Linux	117
Apache	21
IIS	15
Other*	113

Table 4.3: The number of port 80 exploits in Metasploit filtered by keywords. Note some results overlap (e.g., Windows and IIS).

CALDERA without fingerprinting would use this method for exploiting ports. A more scalable approach would be to determine which hosts are running SMB, then only test those hosts.

To implement this solution, 10 fingerprinting abilities are used by SAVE-T. These abilities use two methods for determining services running on a host. Four abilities will use only the port number to determine if telnet, FTP, SSH or SMB services are running. Five other abilities use the popular banner grabbing technique [101] to determine the service or device that is running. Supported fingerprints include: Trendnet TV-IP410W IP Camera, Asus Routers, Apache Struts, ThinkPHP, IIS, and SMB versions vulnerable to EternalBlue [19, 51, 100].

Each fingerprinting ability produces a fact that contains the IP address and port on which the service is running and the name of the found service.

4.2.2 Scalable Exploitation

To ensure that the Metasploit functionality is scalable, not all exploits are attempted on all hosts. We use contextual filtering to pick the best exploits to attempt against each target. Contextual information is usually gathered from fingerprinting abilities which produce facts related to the type of exploits that will be attempted later. For

example, exploitation in an unscalable manner would perform all Metasploit exploits against all targets; however, scanning the target to determine if a certain port is open can greatly limit the number of exploits performed. Table 4.2 displays scaling down of exploit numbers based on port number filtering. Current tools such as Armitage’s Hail Mary perform similar filtering based on port numbers.

Filtering again based on what program a service is hosting (e.g., Apache, IIS, etc.) or what operating system the exploit works on (e.g., Windows, Linux, etc.) can further limit the number of exploits that are beneficial to attempt. We search the exploit name and description for a keyword based on the service fingerprint we gather. Table 4.3 displays further scaling down exploit numbers based on a keyword filter for port 80 exploits, potentially reducing the number of exploits that should be attempted by at least 3x.

SAVE-T has several abilities which filter based on port number and/or keyword filters which greatly reduces the number of exploits which are attempted on each host. Port filters used with SAVE-T are port 21, 22, 23, and 445. Keyword filters are used with ports 80 and 8080. Keywords include filtering for IIS and Apache Struts. These are filters which are particular to the services which our test environment is running, but additional fingerprints and keyword filters can be added for other services. We chose port 80 as a proof of concept because it has the largest benefit in comparison to other ports.

4.3 Contributions for (G_3) - Extending the Operation

This section will detail contributions related to (G_3): Use post-exploitation actions to extend the operation beyond one epoch. We realized this goal by changing the action planner to account for new agents in §4.3.1 and adding lateral movement abilities which are able to spawn new agents in §4.3.2.

First, we will define the behavior that we would like to occur – continuing the

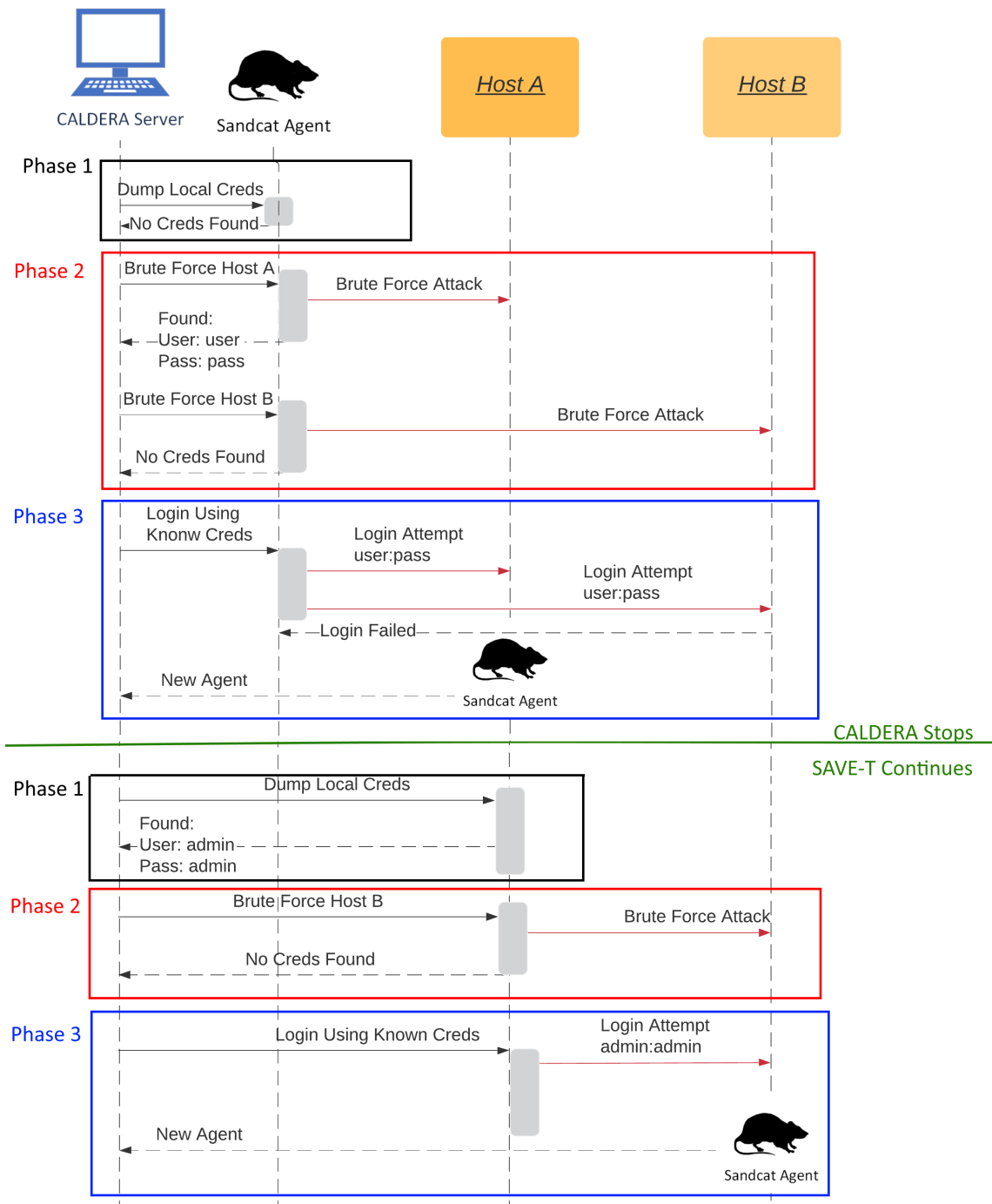


Figure 4.1: Shown is a sequence diagram workflow for a sample operation. Items in black boxes are part of Phase 1 of this operation. Items the red box are part of Phase 2. Items in the blue boxes are part of Phase 3. The green line shows where CALDERA would stop the operation and SAVE-T continues.

operation beyond one epoch. Figure 4.1 illustrates the desired workflow. Here we have a 3-phase operation. Phase 1, in black, instructs an agent to dump local credentials. Phase 2, in red, instructs an agent to brute force passwords on Host A and Host B. Phase 3, in blue, instructs an agent to attempt login to hosts with known credentials.

In the image, we see that the running agent gets no results no dumping local credentials in Phase 1. Move on to Phase 2 and the agent is able to brute force credentials for Host A but not Host B. In Phase 3, the agent uses those credentials to try to login to Host A and Host B (maybe Host B would allow credentials from Host A, but not likely). Host A login succeeds and produces a new agent that checks-in to the CALDERA server. Login attempts fail on Host B. This completes the first epoch of the operation.

In traditional CALDERA, the operation would stop here, as indicated by the green line. With SAVE-T we want the operation to continue tasking the new agent for another operation epoch.

The figure below the green line shows the second operation epoch. The new agent dumps local credentials and finds a new username/password combination. That new combination can be used to login to Host B. This shows that a compromise of Host A can impact Host B, despite Host B not being directly vulnerable to the attacks that were initially attempted. This is an important result to gather because indirect impacts are still security concerns.

We will next discuss the contributions to the planner and lateral movement abilities that facilitate these actions.

Beginning of Operation	After 1 Epoch	Tasked to Continue
Agent 1	Agent 1	No
	Agent 2	Yes
	Agent 3	Yes

Table 4.4: Illustration of agents which are tasked to continue an operation

4.3.1 Action Planner

CALDERA comes with one default planner which allows an operation to continue if there are facts which can be used but have not yet been used. With SAVE-T we also wish for operations to continue if there are new agents which have not yet participated in an operation epoch. This is necessary when a new agent is spawned by an ability. Refer back to Figure 4.1 to see how dumping credentials on a newly compromised system can lead to further compromises. To allow new agents to participate in an operation epoch, the planner logic was changed.

The operation will ask the planner to map abilities to agents which need to perform them during each phase. After one epoch, the operation will ask the planner if it should continue the operation for another epoch. This cycle continues until the operation is told by the planner to stop execution.

When the planner is asked if the operation should continue, it will compare a list of agents that were part of the operation at the beginning of the epoch planning and a list of the agents that exist at the end of the epoch. If there are agents which exist at the end of the epoch which did not exist at the start of the epoch (i.e., agents which were spawned during the operation epoch), then these new agents should be tasked to complete another operation epoch. Only the agents that are new should be tasked to continue the operation – overlapping agents have already completed all phases of the operation and do not need to complete them again.

Table 4.4 illustrates how this list comparison will work. The left column of the figure shows agents which exist at the beginning of the operation. The middle

column shows agents which exist after the first epoch has completed and the planner is asked if the operation should continue. Agents 2 and 3 are new agents which spawned during the first epoch and are tasked with continuing the operation. Agent 1 has already completed all operation actions and should not be tasked to perform them again.

By performing the agent list comparison, we can ensure that all agents have performed all parts of the operation. This will provide efficient and comprehensive coverage.

4.3.2 Lateral Movement Abilities

To continue an operation, we wish to execute new agents on hosts using facts that were gathered during the operation. For example, if the brute force abilities are able to find default credentials for a service, we want to login to that service and spawn a new agent. We can then perform some abilities which occur locally on that new agent, like dump further credentials.

One way to move laterally is through service exploitation. The Apache Struts and ThinkPHP exploit abilities, discussed in §4.1, will automatically start a new agent if the target is vulnerable to the attack.

Metasploit abilities can also be used to move laterally. When a Metasploit module succeeds in exploiting a target, a Metasploit session will be created. The last action in each epoch is to attempt starting a new agent on active Metasploit sessions via the Metasploit API.

Another way is to use SMB credentials that are gathered during the operation. The credentials can either be gathered by brute forcing default credentials or using a credential dumper on Windows ¹.

¹ The credential dumper that is used is called Mimikatz [24, 25]. The implementation utilized is

To move laterally via credentials we utilize a Windows tool called PsExec [91]. More specifically, we utilize the Py-PsExec Python package which allows Python scripts to use the functionality of PsExec [35]. PsExec and the Python implementation allow computers to invoke processes on remote Windows machines if administrator credentials are provided ² [91].

The SMB login ability will take in either local credentials which were gathered by brute force, or local/domain credentials that are dumped via the Mimikatz ability. If domain credentials are provided, then the credentials can most likely be used on any machine which is also a domain member. If local credentials are attempted, the exploit may succeed on machines which have the required configurations and the same local administrator credentials.

4.4 Contributions for (G_4) - Diverse Device Support

This section will detail contributions related to (G_4): including IoT devices in the security assessment. We realize this in §4.4.1.

4.4.1 Diverse Device Support

In order to support a more diverse set of devices that may be seen in networks today, we added support for devices with an ARM architecture and rethought what kinds of abilities/payloads should be delivered to agents.

To add support for the ARM architecture the Sandcat Agent needed to be recompiled with the appropriate settings to produce a binary executable file that is able to run on ARM. Additionally, ARM devices wishing to download and run Sandcat

the Powershell script known as `Invoke-Mimikatz.ps1` from the PowerSploit tool set [81]. A similar tool does not exist on Linux, so this is Windows only

² A caveat for PsExec arises when systems are not joined to a Windows domain, or domain administrator credentials are not provided. In order for local administrator credentials to be used with PsExec, a registry setting has to be changed from the default value.

needed to be able to request the appropriate binary from the CALDERA server. To accommodate this, the API endpoint for downloading the Sandcat agent from the server was edited to appropriately serve the requested binary.

To support a wide array of devices on the network, we had to rethink what payloads should be delivered to agents. Traditional CALDERA ability payloads were command line scripts that were highly dependent on the platform. To make a cross-platform solution, we decided to use Python scripts. These scripts can be run on many different architectures, allowing one script to be delivered to all types of hosts.

One problem with using Python scripts is that the code cannot run unless Python and all script dependencies are present on the host running the script. We did not want to require that Python and script dependencies be installed on all machines. As a workaround for this, we used the PyInstaller [82] tool which packages the Python program and all dependencies into a standalone executable file. We can use PyInstaller on any Windows or Linux device to compile the script into a binary for the appropriate operating system.

Delivering binaries as payloads is not in itself a novel contribution, however, it does go against what was traditionally delivered in CALDERA. CALDERA abilities were traditionally “Live off the Land”, where they would only utilize command lines and other tools which are already installed on the machines.

By changing ability payloads from using only tools native to Windows/Linux to Python compiled binaries, we are able to add versatility and reduce the amount of work required to use abilities. Abilities can now perform actions that Python can do, that the OS might not be able to perform natively. We also make it such that only one script needs to be written for the ability to work on all platforms, instead of needing a different script for each target OS.

Type of Ability	Number	% of Total
Altered from CALDERA	1	~3%
Start Sandcat on Remote Host	2	~7%
Brute Force	3	~10%
Naive Exploit of IP address	5	~16%
Fingerprinting Services	10	~32%
Targeted Service Exploit	10	~32%
SAVE-T Total	31	100%

Table 4.5: This table summarizes the abilities SAVE-T added to CALDERA. Percentages rounded to the nearest 1%

4.5 Summary of Abilities

This section will summarize the contributions we made to the ability catalog by comparing CALDERA and SAVE-T added abilities. We will compare abilities that are included as part of CALDERA and abilities that were designed and added as part of SAVE-T.

CALDERA was released with 74 abilities. As part of SAVE-T development, 30 new abilities were written and 1 was improved from the release of CALDERA. SAVE-T is able to use any of the CALDERA abilities in addition to those that were designed as part of SAVE-T. This means that SAVE-T has access to 104 abilities that it can use for testing networks.

Abilities which were added as part of SAVE-T are summarized in Table 4.5. The SAVE-T abilities include 3 credential brute forcing abilities, 10 service fingerprinting abilities, 10 abilities which exploit based on a fingerprinted target (targeted exploit), 5 abilities which exploit based on IP addresses alone (naïve exploit), and 2 abilities which start the Sandcat agent on a remote system.

Table 4.6 shows a comparison between CALDERA abilities and SAVE-T added abilities with respect to their supported operating systems. Note that SAVE-T has a much higher percentage of abilities which can be run on all operating systems compared to CALDERA. This is due to the fact that SAVE-T uses Python scripts

as ability payloads rather than built-in operating system tools. Using one script standardizes actions that can occur in abilities and eliminates the need to write different scripts for different device types.

Traditional CALDERA abilities are written in an operating system dependent format. Abilities for Windows are either written in Powershell or Batch scripts. Abilities for Darwin and Linux are often written in shell script. This results in a requirement to rewrite each ability for each operating system, dependent on the command line language that the operating system supports and the command line tools which are built-in on each operating system.

In order to have more traditional CALDERA abilities to support diverse operating systems, the abilities would need to be rewritten for each operating system as described. In some cases, an operating system may not have the required functions to perform a particular ability.

The diverse device support shown by the SAVE-T added abilities is due to the new abilities being written in an operating system agnostic format. The abilities added by SAVE-T are all written in Python and can be delivered to agents via standalone binary executables with all of the required dependencies packaged. By packaging the code into executables, we remove the need for the host to have Python and the code dependencies installed.

Designing abilities in this way allows the abilities to be delivered to the agent in a format that the agent can run, regardless of the operating system or what command line tools the host has installed. This design choice accounts for the SAVE-T added abilities generally supporting more operating systems.

More than 50% of the CALDERA abilities were designed to work only on Windows machines. 78% of abilities added by SAVE-T can work on Windows, Linux and Darwin (Mac OS) hosts. This is indicative of the operating system flexibility of the SAVE-T abilities. The hosts on which SAVE-T abilities are able to run are

OS Supported	CALDERA		SAVE-T	
	# of Abilities	% of CALDERA Total	# of Abilities	% of SAVE-T Added
Linux Only	0	0%	6	~19%
Darwin Only	7	~9%	0	0%
Windows Only	39	~53%	1	~3%
Darwin & Linux	9	~12%	0	0%
Darwin, Linux, Windows	19	~26%	24	~78%
Total	74	100%	31	100%

Table 4.6: This table shows the number of CALDERA and SAVE-T abilities and their supported operating systems. Percentages are rounded to the nearest 1%.

ATT&CK Category	CALDERA		SAVE-T	
	# of Abilities	% of CALDERA Total	# of Abilities	% of SAVE-T Added
Privilege Escalation	1	~1%	0	0%
Command & Control	1	~1%	0	0%
Persistence	2	~3%	0	0%
Evasion	4	~5%	0	0%
Collection	7	~10%	0	0%
Execution	12	~16%	1	~3%
Credential Access	6	~8%	4	~13%
Discovery	34	~46%	10	~32%
Lateral Movement	7	~10%	16	~52%
Total	74	100%	31	100%

Table 4.7: This table shows the number of CALDERA and SAVE-T abilities and their ATT&CK Framework Category. Percentages are rounded to the nearest 1%

more diverse than those provided by CALDERA. We tested our abilities by targeting Windows, standard Linux hosts, and IoT devices which are running stripped down Linux kernels, supporting (G_4).

Table 4.7 shows a comparison between CALDERA and SAVE-T abilities with respect to their ATT&CK Framework categories. Comparing the ATT&CK Framework categories will highlight what kinds of actions are the focus of each (e.g., actions are focused on collecting information, or actions are focused on exploiting new hosts).

Only 9% of CALDERA abilities are categorized as ‘Lateral Movement’, or abilities which further an attacker’s reach by compromising new hosts. Comparatively, 52% of SAVE-T abilities are in the category. This demonstrates SAVE-T’s attention to service exploitation. By adding many abilities which exploit services, we demonstrate accomplishment of (G_1).

Using lateral movement abilities also inherently supports (G_3). The credential

access abilities provide additional support to (G_3) by providing credentials that can be used by lateral movement abilities to login to remote hosts and start new agents. By having abilities which spawn agents on new hosts, we support the workflow of tasking the new agents to continue the operation for an additional epoch.

Looking at Table 4.7 again, we see that SAVE-T added abilities are also concentrated in the ‘Discovery’ ATT&CK Category. The SAVE-T added abilities in this category are the fingerprinting abilities. These abilities allow SAVE-T to parse down the number of exploits attempted by the Metasploit abilities. The fingerprinting abilities in the ‘Discovery’ ATT&CK Category support our (G_2) of scalable exploitation.

4.6 Limitations

This section will describe the limitations SAVE-T currently has. Limitations include detection of services on non-standard ports, correct architecture recognition, and launching Metasploit exploits only from Kali. We discuss an additional limitation of currently supported ports and exploits and a risk of crashing target machines.

4.6.1 Services on non-standard ports

SAVE-T currently makes the assumption that services will be running on their standard port number³. Table 4.8 shows the port numbers and services which are currently assumed by SAVE-T for proof of concept.

While it is common to use the standard port number for a service, standard port numbers may not be in use on all networks. If non-standard port numbers are in use, SAVE-T may make incorrect assumptions and/or miss vulnerabilities which exist on the network.

³ Standard port number used is the port/service pair identified by IANA as standard [108].

Port	Service
21	FTP
22	SSH
23	Telnet
80	HTTP
443	HTTPS
445	SMB
8080	HTTP

Table 4.8: Port numbers and their assumed standard services that SAVE-T supports as a proof of concept.

SAVE-T can be customized to include any port number in a port scan. The non-standard service port limitation could be mitigated by having the fingerprinting abilities not rely on port number for common services. Other banner grabbing mechanisms can be used to detect services which occur on non-standard ports in the same way programs such as Nmap [68] work.

4.6.2 Architecture Recognition

In order to properly deliver the correct executable binary to an agent, the Sandcat Agent needs to provide the C2 server with the operating system and the architecture that the device is using. Currently, the Sandcat agent can only provide the operating system to the server. This results in an edge case where IoT devices running Linux on an architecture such as ARM are delivered the incorrect binary and therefore cannot execute the ability payload.

We circumvent this currently by simply delivering the ability payloads to Linux hosts as Python scripts. This adds the requirement that Linux hosts have Python and the script dependencies installed to be able to perform abilities which are written as Python scripts.

We can mitigate this limitation by having the Sandcat agent report its operating system and CPU architecture when checking in with the server. This can be done

by using the built-in GOARCH runtime variable for the GoLang language which the agent is written in. However, making this change is not as simple as it may seem due to the complexity of service interactions in CALDERA. In order to accommodate the updated information passing from the Sandcat agent to the CALDERA server, we would have to change the Sandcat Agent and API code in addition to the CALDERA service which handles requests from the Agent for file downloads. We then must add conditional logic to the file service which will serve a different file based not only on the operating system, but on the architecture.

This problem was noticed late during testing and as a result we were not able to invest the time needed to make the change.

4.6.3 Metasploit Requires Linux With Dependencies

For most abilities, we were able to package the script as a binary with all required dependencies that would function on most architectures. However, we had issues when packaging the Metasploit scripts into binaries. When these scripts were packaged, the script would appear to be running properly, but no packets were being sent on the network.

For example, we packaged the ability to perform Metasploit exploits on the SMB service. When this packaged binary was executed, the appropriate debugging information was being printed to the console. Despite this, the packets containing the exploits were not visible leaving the host from which the program was running. In other words, the program appeared to be running properly based on debugging statements, but the program was not sending the exploit traffic over the network. When running the code as a Python script instead of the compiled binary, we were able to see the exploit traffic on the network.

We are unsure what caused this problem when packaging the binary; it could either be a problem with the pymetasploit package or with the pyinstaller packaging

tool, but we are unsure which. Due to the network problem with the packaged Metasploit scripts, we limited the Metasploit abilities to only functioning on our Kali system. The ability would also function on any other Linux distribution with the required dependencies installed, but Kali is the easiest since it comes with the required dependencies already installed.

4.6.4 Current Support

As with any software, there are some things that are possible, but simply aren't implemented yet. For this thesis, our goal was to implement proof of concept abilities, not to implement every feature that could be desired.

As described in §4.2.1, we have 10 fingerprinting abilities for 7 different standard ports. These abilities are used for scalable exploitation as a proof of concept. Additionally, as described in 4.1.2, we implemented 3 custom exploits as proof of concept. Support for additional ports and custom exploits may be added by an expert.

To add support for additional ports, the expert can edit the Python script that is used to scan IP addresses for open ports to include any of the desired ports. These ports may be supported by the Metasploit ability which will apply a port filter to all Metasploit exploits. The user might also have to adjust the Metasploit default variables if an exploit they wish to test for uses one that is not currently supplied by SAVE-T. The user can add additional fingerprinting support for this port by editing the service fingerprinting Python script and writing a new ability that will run the script with the appropriate options.

To add support for additional custom exploits, the user must write their own code or find code from a source like Exploit-DB [73]. The code should be tailored such that the only output of the code is in the format of a fact. It may get tedious to add many exploits for specialized devices or networks.

The need for exploit code to already exist for SAVE-T to be able to test for the

vulnerability can make it difficult for the tool to be adopted in the IoT community. The community support for exploits against Windows and Linux Machines is currently more prevalent than the support for attacking IoT devices. This is because IoT devices are very diverse and often require specialized exploits, whereas Windows systems are widely deployed with similar software versions. In order for SAVE-T to be of more use to a wide array of IoT devices, community support for finding and releasing exploits for those devices needs to be increased.

4.6.5 Risk of Target Crash

SAVE-T exploits and scans target hosts with a risk of the host crashing. While SAVE-T is using Metasploit or custom exploits there could be a risk of memory corruption that leads to a system crashing. We observed this in our results with a Windows Server 2016 virtual machine that crashed one time during our 20+ tests due to memory corruption from the MS 17-010_psexec exploit from the Metasploit Framework [85].

SAVE-T has no way of realizing that a target system has crashed due to an exploit. The server will notice that the system has stopped replying, but will not have information about what caused the system to crash or otherwise stop responding.

Crashing systems is never a desired effect of running a security analysis tool and is often not acceptable. For networks where a reboot/reset of a crashed system is an acceptable risk, SAVE-T can perform any exploits. For cases where crashing a system is not acceptable, exploits that SAVE-T runs should be limited to those known to not cause systems to crash.

5

Evaluation

This section will describe the experiment testbed we used to test SAVE-T and how we increased the testbed complexity to assess SAVE-T’s scalability. We will then discuss the experiments we conducted and their results, including the time it took experiments to complete and an analysis of SAVE-T’s ability to consistently find vulnerabilities. Next, we examine how well SAVE-T covers the Internet’s top ports. We then compare SAVE-T to base CALDERA and Armitage Hail Mary. We conclude this section with a summary of SAVE-T’s evaluation.

5.1 Testbed

To test SAVE-T, we designed a testbed consisting of Windows and Linux virtual machines in addition to our chosen representative IoT devices. We have these hosts configured in a flat network.

Table 5.1 summarizes the hosts we used in our experiments, their open ports and their known vulnerabilities. The Windows domain joined hosts have the Host Type column labeled with “^”. Other hosts are not joined to the Windows domain.

There are 5 different Metasploit vulnerabilities shown in Table 5.1. MS 08-067

Host Type	Open Ports	Known Vulnerabilities
Server 2016^	445	MS 17-010_psexec
Server 2012^	445, 8080	MS 17-010_psexec
Server 08R2^	21, 80, 445	MS 17-010_psexec
XP^	445	MS 08-067
Win 7^	445	MS 17-010
Win 8.1^	445	MS 17-010_psexec
Win 10^	445	MS 17-010_psexec Default Password - SMB
XP	445	MS 08-067 Default Password - SMB
Win 7	445	MS 17-010 Default Password - SMB
Win8.1	445	Default Password - SMB
Win10	445	Default Password - SMB
TrendNet Camera	80	Custom TrendNet Exploit
RPi - Apache Struts	22, 8080	Custom Apache Struts Exploit
RPi - ThinkPHP	22, 23, 80	Custom ThinkPHP Exploit Default Password - SSH Default Password - Telnet
Ubuntu	21, 22, 23, 80, 445	VSFTPD Backdoor Samba Code Execution Default Password - SMB Default Password - Telnet

Table 5.1: List of hosts, their open ports and known vulnerabilities. **Red** shows Vulnerabilities that are part of Metasploit exploitation. **Blue** shows SAVE-T custom exploits.

^ indicates machines that are joined a common Windows domain

[50] and MS 17-010 [51] are SMB vulnerabilities in Windows systems. The VSFTPD exploit is a backdoor vulnerability in VSFTPD that results in code execution on the target [67]. The Samda exploit is a code execution vulnerability in the Samba service for Linux hosts [54]. The list of known vulnerabilities on the network is up to date and complete as of writing.

There are two versions of the MS 17-010 exploit shown in Table 5.1. One version, known as the “psexec” version can exploit targets that are higher versions than Windows 7 if and only if there is a named pipe open that can be accessed anonymously

Network Name	Total Hosts	Total Open Ports	Total Vulnerabilities	Device Type(s)
Standalone Host	1	1	1	Windows
Minimum Enterprise	2	2	2	Windows
Small Enterprise	7	10	8	Windows
Medium Enterprise	11	14	14	Windows
IoT Enterprise	14	20	19	Windows and IoT
IoT Enterprise plus Linux	15	25	23	Windows, IoT, and Linux

Table 5.2: This table shows the number of hosts, the number of open ports, and the device types that were on the network for our 6 tests.

[85]. This is common on domain connected hosts, but it is not always guaranteed to exist [85]. This special exploit condition cannot be predicted or planned for. We will see consequences of this unpredictable required condition in our tests.

The SAVE-T server was hosted on a Kali VM that is connected directly to the flat network. This VM served not only as the C2 server, but also as the location of the first agent on the network. In other words, a Sandcat agent would be started on the Kali machine manually before initiating the operation. This agent served as the starting point.

5.1.1 Scenarios

To show scalability of the system (i.e. how the system performs when more hosts are added), we tested the number of open ports on the network incrementally. We did this by limiting the number of hosts that are on the network at a time. We scaled the network up from 1 host with 1 open port to 15 hosts with 25 total open ports distributed across the hosts in 6 increments. Table 5.2 shows how the network was scaled up based on the number of hosts and the number of open ports. The table also shows what kinds of devices were on the network at the different sizes. “Windows” indicates a Windows VM. IoT devices (i.e., the Trendnet camera and the Raspberry Pi’s with the web frameworks) were included in later network versions. The last network also included a vulnerable Ubuntu VM. See Appendix C to view a more detailed table of which hosts were included for each experiment.

Phase #	Windows Only Adversary	Full Adversary
Phase 1	Scan IP for Ports Dump Local Credentials	Scan IP for Ports Dump Local Credentials
Phase 2	Fingerprint SMB Fingerprint Apache Struts Fingerprint IIS	Fingerprint SMB Fingerprint Apache Struts Fingerprint IIS Fingerprint ThinkPHP Fingerprint Trendnet Fingerprint SSH Fingerprint FTP Fingerprint Telnet
Phase 3	Brute Force SMB Exploit Apache - Metasploit Exploit IIS - Metasploit Exploit SMB - Metasploit	Brute Force SMB Brute Force SSH Brute Force Telnet Exploit Apache - Metasploit Exploit IIS - Metasploit Exploit SMB - Metasploit Exploit FTP - Metasploit Custom Exploit Apache Struts Custom Exploit ThinkPHP Custom Exploit Trendnet
Phase 4	Metasploit Start Sandcat Execute Sandcat SMB Execute Sandcat SMB (Domain Credentials)	Metasploit Start Sandcat Execute Sandcat SMB Execute Sandcat SMB (Domain Credentials)

Table 5.3: This table outlines the abilities for each of the adversaries that were used for testing. The “Full Adversary” is simply an expansion to the abilities of the “Windows Only Adversary”

We choose to increase the network size and complexity slowly over time with our scenarios. The first scenario is the most basic of a single standalone Windows host. We then have two iterations of a Windows domain of size two and seven hosts. We increase the complexity by adding four additional standalone Windows hosts alongside the Windows domain¹. We increase complexity again by introducing our chosen IoT devices. Finally, we increase the complexity by adding one additional Linux VM which has several vulnerabilities to ensure the tool can support identifying multiple vulnerabilities on a single host.

¹ It is important to note that the standalone Windows hosts will behave differently than domain joined hosts. The difference is due to shared domain administrator credentials and the MS 17-010_psexec exploit condition that do not exist on standalone Windows hosts.

5.1.2 Adversaries

We used two different adversaries for testing. These adversaries are the “Windows Only Adversary” and the “Full Adversary”. The Windows Only Adversary was used while testing networks which are labeled as “Windows” in Table 5.2. The Full Adversary is an expansion on the Windows Only Adversary and was used with tests which included the IoT devices and the Linux host.

The adversaries and the abilities that were part of each are shown in Table 5.3. Note from this table that the Full Adversary has the same abilities as the Windows Only Adversary, plus additional abilities to accommodate the net devices that were added to the network.

The abilities shown in Table 5.3 have 4 basic phases with common actions. In phase 1, the adversary scans for new open ports on remote hosts and attempts to dump local credentials. In Phase 2, the adversary will fingerprint any found open ports. In Phase 3, the adversary will attack the open ports based on their assigned fingerprints. In Phase 4, the adversary will use Metasploit sessions and SMB credentials to attempt to start new agents on remote hosts.

Two different adversaries were used to reduce the experiment complexity when targeting the networks with only Windows hosts. In these experiments, the abilities targeting the IoT devices, SSH, and Telnet would not have run anyway because those ports are not open on the network. By removing those abilities from the operation menu, we reduced the debugging and implementation complexity of the first experiments without detracting from the value delivered by the experiment.

5.2 Experiment Results

Our experiments can be defined as running a SAVE-T adversary on a given controlled network. Our testing consisted of running an experiment 3 times on each network

Ability	Average Time
Metasploit	0.82 seconds per exploit
Credential Dump	3 seconds
Scan IP for Ports	1 second
Brute Force	4 seconds (no login found)
Struts Exploit	1 second
ThinkPHP Exploit	1 second
Trendnet Exploit	1 second
Fingerprint	1 second

Table 5.4: Shown are the abilities and their time to run. Metasploit abilities' run times are dependent on the number of exploits to be run. Brute Force time is dependent on if credentials are found - worst case is 4 seconds if no credentials are found.

named in Table 5.2. The state of virtual machines was reset to a known good snapshot state before each experiment. Hardware devices were manually restored to a known good state.

This section will detail the results of our experiments. We will consider the amount of time taken to complete a SAVE-T operation and the consistency with which SAVE-T finds vulnerabilities.

5.2.1 Time

We will first discuss two aspects of time for SAVE-T. We will discuss how long it takes each ability to run and examine the duration of SAVE-T operations.

Ability Run Times

Table 5.4 shows the amount of time it takes to run each ability. Notice that most abilities only require a maximum of between 2 and 4 seconds to complete. Metasploit abilities are a special case, because the run time depends on the number of exploits that will be attempted. The time it takes for a Metasploit ability will be $0.82 * N$ where N is the number of exploits that are going to be attempted. We calculate this average exploit run time by timing how long it takes to run all Metasploit exploits

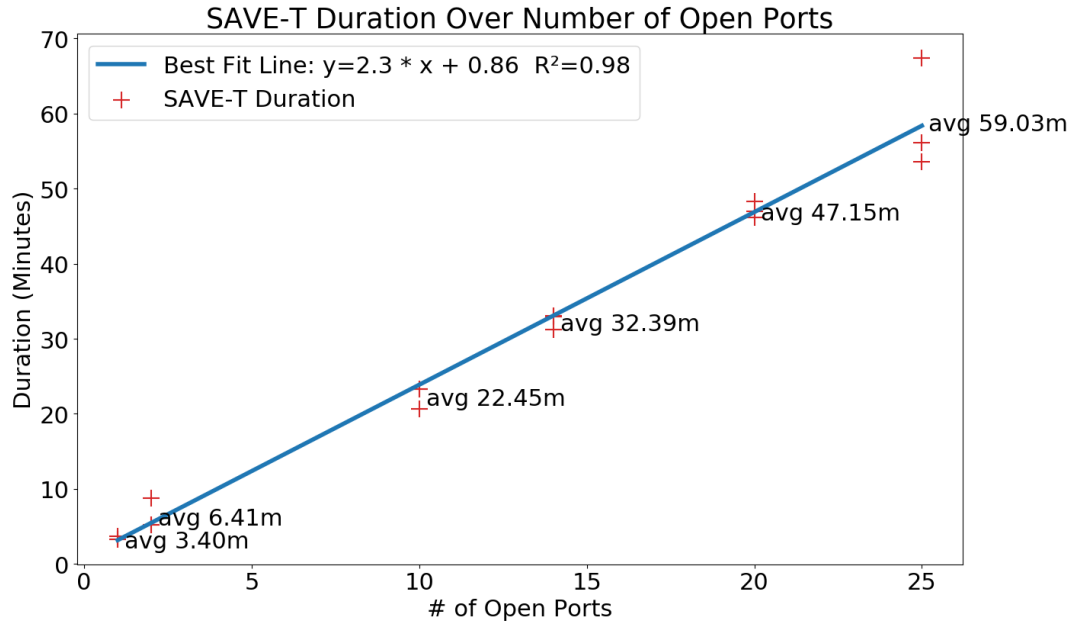


Figure 5.1: Observed time to complete an operation in relation to the number of open ports on the network with a best fit line.

and dividing by the number of exploits that were run to get an average number of seconds per exploit. This is detailed further in Appendix D. Because we are going to attempt a minimum of 12 exploits (see Appendix E for exact numbers) when using a Metasploit ability, Metasploit abilities will take the longest amount of time to complete. We will see that the SAVE-T operation time is impacted by the number of Metasploit exploits that are attempted on open ports.

SAVE-T Operation Duration

When running the SAVE-T operations with our described adversaries on the testbed network, we observed how long the operation took to complete. SAVE-T operation duration is an important metric to observe because if operation time is increasing exponentially, then we would not have built a scalable tool. We can also use this in the future to estimate how long a SAVE-T operation would take, given a certain number of known open ports on the network.

Figure 5.1 shows the observed operation duration for each of our experiments. The x-axis shows the number of ports that were open for the experiment and the y-axis shows the duration of the SAVE-T operation in minutes. The annotations show the average of the three observed durations for each experiment.

The standalone host experiment, with 1 open port, takes the least amount of time, taking an average of 3 minutes 24 seconds to complete. The Enterprise IoT Plus Linux experiment, with 25 open ports, took almost 1 hour to complete. There is a linear trend with observed experiment durations between the smallest and the largest experiment. The slope of the best fit line is 2.3, indicating that a SAVE-T operation will take approximately 2 minutes and 20 seconds per open port on the network plus the an additional 50 seconds added from the y-intercept value of the linear equation. The R^2 for this best fit line is 0.98 indicating that the data is highly correlated with the given line.

This linear trend is to be expected with respect to the number of open ports on the network. This is because when we increase the number of open ports on the network, we also increase the number of Metasploit and other abilities we are attempting on each of those ports. Exploiting the network with Metasploit takes a significant amount of the operation time. Therefore, with more open ports, there are more actions to be performed which lengthens the operation.

There is one outlier in the data. This is in the Enterprise IoT Plus Linux network where one of the 3 experiments took over an hour to complete. This longer time is due to an exploit causing one of the Windows hosts to crash, perform a memory dump, and reboot. This caused several SAVE-T abilities to timeout (based upon 5 minute timeout intervals). This was the only time we recorded a host crashing during testing, but serves to show that there is risk of device crashes while exploiting targets. The results of this run were not skewed other than the time taken for the operation to complete. When the abilities failed due to timeout, they were successfully attempted

again.

5.2.2 Scalability

The trend for the time taken to run SAVE-T is linear, as was discussed in §5.2.1 and Figure 5.1. Our experiments might be especially correlated with the given best fit line because many of the open ports in each experiment happen to be the same port. All of the Windows hosts have port 445 open because this is a Windows default and is required on domain joined systems. Based on the heterogeneous nature of the open ports on our networks, we may have more linear results than is seen in more diverse networks.

Open ports on our test networks which are not port 445 have had the number of exploits to be attempted filtered down using either port filters or the keyword filters. This results in all ports on the network having a similar number of exploits attempted by SAVE-T. The range in number of exploits attempted per port for our experiments is 43 and the average number of exploits per port is 31.33 (see Appendix E for the exploit attempt numbers). Filtering open ports on the network to have a small range in the number of exploits attempted provides a more uniform distribution of time spent to exploit each port, which may also be influencing our results.

Users may want to extrapolate from our data to predict how long a new operation on a new network might take. Extrapolation can occur based on the number of open ports on the network. This can be done via the linear equation for the best fit line shown in Figure 5.1 $y = 2.3 * x + 0.86$ where x is the number of open ports on the network. This linear trend fits well as an estimate for our experiments, but the trend cannot be guaranteed to be true for all sample sizes. This trend line estimate should serve only as an estimate, and not as a guarantee as to how long an operation might take.

Network Name	MS 17-010_psexec Vulnerabilities	MS 17-010_psexec Vulnerabilities Found (Min — Max)	Other Vulnerabilities	Other Vulnerabilities Found (Min — Max)
Standalone Host	0	0 — 0	1	1 — 1
Minimum Enterprise	1	1 — 1	1	1 — 1
Small Enterprise	5	1 — 5	3	3 — 3
Medium Enterprise	5	1 — 5	9	9 — 9
IoT Enterprise	5	0 — 4	14	14 — 14
IoT Enterprise Plus Linux	5	3 — 4	18	18 — 18

Table 5.5: This table shows vulnerabilities found in each of our 6 experiments. Shown is the name of each experiment, the number of MS 17-010_psexec vulnerabilities and the other vulnerabilities which are on the network. The minimum and maximum number of vulnerabilities found by SAVE-T is shown for both the conditional MS 17-010_psexec vulnerability and the other vulnerabilities.

5.2.3 Reliability

We will next discuss the vulnerability discovery reliability of SAVE-T. For each of our experiments, we know the ground truth of how many vulnerabilities exist on the network. We will compare the ground truth to how many vulnerabilities SAVE-T reports at the end of the operation. Again, for these experiments, we run each 3 times.

Table 5.5 shows the name of each of our networks along with the number of MS 17-010_psexec Vulnerabilities which require the special exploit condition in order to succeed and the number of other (i.e. not MS 17-010_psexec) vulnerabilities which exist on the network. Also shown is the minimum and maximum number of MS 17-010_psexec and other vulnerabilities which were found by SAVE-T across the three experiment runs. The vulnerability numbers in this table represent those that should be found by Metasploit and those that should be found by our custom exploits. Refer to Table 5.1 to see the exact vulnerabilities that exist on each target host.

Notice from Table 5.5 that SAVE-T is able to reliably discover vulnerabilities which are not the conditional MS 17-010_psexec vulnerability. For our tests, we were able to find all of the existing vulnerabilities which are unconditional (i.e. do not rely on a special condition to succeed) in all cases.

In the base cases, SAVE-T is also able to find the MS 17-010_psexec vulnerability. Due to the unpredictable exploit condition, however, SAVE-T was not always able to identify every host on the network that is vulnerable to that attack. The data shows this when the number of MS 17-010_psexec vulnerabilities found is lower than the total number of those vulnerabilities which exist.

The experiments on the IoT Enterprise and IoT Enterprise Plus Linux networks show that even in the best of three repetitions, SAVE-T cannot always identify vulnerabilities that exist on the network. This is because SAVE-T must rely on exploit conditions to be met on the host for the exploit to succeed. This is a drawback to using actual exploitation to find vulnerabilities rather than using service signatures in a scan, much like Nessus [104] does.

A more robust system might use a combination of service signature scanning like in Nessus, and vulnerability exploit validation like in SAVE-T. This type of system would provide consistent results via the signature scanning and validate results via exploitation of the vulnerability.

5.2.4 Summary

This section showed the time and reliability results of running SAVE-T in our experiments. SAVE-T has a strong correlation to a line of best fit which estimates that SAVE-T takes approximately 2 minutes 20 seconds per open port to run, plus an additional 50 seconds. We did not see large variation in the time it took to run SAVE-T between our three repetitions of the same experiment. One outlier to this observation is caused by a host which crashed during the SAVE-T operation. SAVE-T is also able to reliably identify vulnerabilities on the network which do not depend on special conditions to succeed.

5.3 Top Port Coverage

It is important that SAVE-T provides support for the most used services on the Internet. While we do not have data relating to the most used ports on private networks, open ports on the Internet can provide insights about common open ports. While the most used ports on the Internet might not be perfectly representative of the ports in use in private networks, we believe that supporting these ports allows SAVE-T to be used in many networks.

Table 5.6 shows the top 20 open ports on the Internet according to Censys [14]. Table 5.7 shows the top ports which are indicated on exploits published on Exploit-DB [74]. In Table 5.7, 'Null' indicates that the exploit does not have a default port entered. This can mean that the exploit is a local exploit and does not require a port, or that the author of the exploit did not include a default port. The check marks in each table indicate the ports that are currently supported by SAVE-T.

SAVE-T can currently provide coverage for 7 out of the top 20 open ports on the Internet, including the top 2 ports, according to Censys [14]. This coverage overlaps with the ports which have the most exploits published on Exploit-DB [74], where SAVE-T provides coverage for 5 out of the top 10 most popular ports.

SAVE-T does not provide coverage for all of the top ports listed in Table 5.6 and Table 5.7. The ports we chose for SAVE-T were chosen because of their popularity and only serve as a proof of concept. Despite not currently providing coverage for all of the top ports, SAVE-T can easily be adjusted to provide coverage of more of these ports.

5.4 Comparison to Other Tools

This section will compare SAVE-T to Armitage Hail Mary and base CALDERA. First, we compare the exploitation actions of SAVE-T to those of Hail Mary. We

Port/Service	# of Devices	SAVE-T	Port	# of Devices	SAVE-T
80/HTTP	53.34M	✓	143/IMAP	4.22M	
443/HTTPS	42.69M	✓	993/IMAPS	4.1M	
7547/CWMP	21.78M		587/SMTP	4.03M	
22/SSH	18.54M	✓	995/POP3S	3.92M	
21/FTP	9.76M	✓	465/SMTP	3.69M	
8080/HTTP	6.98M	✓	3389/RDP	3.6M	
53/DNS	6.84M		23/Telnet	2.69M	✓
25/SMTP	5.99M		8888/HTTP	2.46M	
3306/MySQL	4.75M		161/SNMP	1.98M	
110/POP3	4.38M		445/SMB	1.22M	✓

Table 5.6: Top 20 open ports on the Internet and SAVE-T’s coverage (Data sourced from Censys [14])

Port	Exploit-DB Entries	SAVE-T Coverage
80	2,043	✓
21	158	✓
443	107	✓
8080	107	✓
143	48	
25	42	
8000	29	
22	27	✓
69	24	
Null	39,227	

Table 5.7: Shown is the top 10 exploit ports on Exploit-DB [74] and SAVE-T’s coverage of these ports.

then compare the capabilities of SAVE-T to those of Hail Mary and base CALDERA.

5.4.1 SAVE-T vs. Hail Mary Exploitation

When considering scalable and reliable exploitation, we want to compare SAVE-T with Hail Mary. For both Hail Mary and SAVE-T we will look at the number of exploits that will be attempted given their respective filtering mechanisms, time taken to complete our experiments, and the number of exploits found for each experiment. In these cases, an experiment is running Hail Mary or SAVE-T.

First, consider the number of exploits that each Hail Mary and SAVE-T will

Network Name	Hail Mary Exploits Attempted	SAVE-T Exploits Attempted
Standalone Host	33	33
Minimum Enterprise	66	66
Small Enterprise	838	307
Medium Enterprise	970	439
IoT Enterprise	1,681	498
IoT Enterprise Plus Linux	2,243	607

Table 5.8: This table compares the number of exploits that are attempted by Armitage Hail Mary and SAVE-T.

attempt during the experiments. Table 5.8 shows these numbers. The number of exploits attempted for Hail Mary are based on the port-only filtering that Hail Mary uses. The number of exploits attempted by SAVE-T is based on the combination of port and keyword filters that SAVE-T uses.

The increase in the number of Hail Mary and SAVE-T exploits is due to how the network changes in each experiment. The increase is not consistent for each experiment because each open port has a different number of applicable Metasploit exploits. Refer to Appendix E to see the number of exploits that are applicable for each port and optional keyword filter.

At first glance, it may seem as though Hail Mary is better because it is testing more exploits. However, this is not the case as Hail Mary is testing exploits which are not applicable to the service that is running. For example, if there is a web server hosted on port 80 with IIS on Windows, Hail Mary will try over 500 exploits for services which run with port 80 as the default. Over 200 of these exploits will only work on hosts running Linux/Unix, so they are not applicable.

Our keyword filtering will not miss any applicable exploits for a port/service. We base this on an assumption about the conventions Metasploit exploits follow. Metasploit exploits all must include a name and detailed description when they are added to the framework. We assume that if an exploit targets IIS, for example, that

Network Name	# Exploits Exist	Armitage Hail Mary			SAVE-T		
		Time	Hosts Completed	Exploits Found	Time	Hosts Completed	Exploits Found
Standalone	1	3:31	1	1	2:11	1	1
Minimum Enterprise	2	4:06	2	2	3:52	2	2
Small Enterprise	7	9:45	1	1	17:37	7	6
Medium Enterprise	9	9:01	1	1	25:48	11	7
Enterprise IoT	9	10:38	2	0	26:32	14	8
Enterprise IoT Plus Linux	11	5:48	2	1	29:01	15	11

Table 5.9: This compares Armitage Hail Mary to SAVE-T with respect to time taken to complete an experiment, the number of hosts that completed the experiment and the number of exploits found. The difference between the number of hosts that were tested and the number of hosts that completed the experiment is the number of hosts which timed out. “Complete” means that the host launched all discovered exploits without timing out.

this keyword (“IIS”) would be included in either the exploit name or the description. If there exists an exploit that works for a service, and that service is not listed in the exploit name or description, then our assumption is false and SAVE-T will currently not test the service for the exploit.

Using a keyword filter, we can remove exploits which are not directly applicable to the service that is running. SAVE-T still does not perfectly filter out exploits which will not work due to version mismatch or other conditions that an exploit may require. Because SAVE-T filtering is still not perfect, exploits which are not applicable to the service may still be attempted; however, there are fewer inapplicable exploits that are attempted with SAVE-T than with Hail Mary.

Table 5.9 compares the run times and reliability (i.e. the ability to correctly identify the correct number of vulnerabilities that exist on the network) of Hail Mary and SAVE-T for each of our experiments. By comparing the time taken for each to complete an experiment, and the number of Metasploit exploits found by each tool for each experiment, we can compare the scalability and reliability of Hail Mary and SAVE-T.

The first column of the table shows the experiment name. The second column

shows the number of Metasploit exploits that exist on the network. This number only counts exploits that can be found with Metasploit. The remaining columns show the average time it took for the tool to mark the experiment as complete, the number of hosts that finished the experiment without timeout errors, and the number of exploits the tool found for each experiment.

Notice that the number of exploits on the network does not increase between the Medium Enterprise experiment and the Enterprise IoT experiment. This is because the vulnerabilities that are on the IoT devices that are added to the network for the Enterprise IoT experiment are not able to be found by Metasploit, therefore those vulnerabilities are not counted in this table.

The first two rows of the table show that SAVE-T is faster than Hail Mary for the first two experiments. SAVE-T is faster than Hail Mary in these first two cases because Hail Mary has a hard-coded 60 second sleep time after all exploits are launched. This sleep time is designed to allow exploits to finish executing before the tool marks the experiment as complete. Because of the hard-coded sleep in Hail Mary, it was sleeping for some amount of time after all of the exploits had finished running, and therefore wasted some time before marking the experiment as complete.

SAVE-T does not use a hard-coded sleep. Instead, SAVE-T will query the Metasploit API every second to determine if there are any exploit jobs still running. Once all jobs are done, SAVE-T will mark the experiment as complete. This allows SAVE-T to finish running as soon as all exploit jobs are finished without having to sleep.

With the larger experiments, SAVE-T takes a longer time to complete than Hail Mary. This is because most of the hosts Hail Mary targeted didn't actually finish the experiment. The database service became unresponsive and most hosts failed. Despite taking longer to complete, the SAVE-T experiments completed for all targeted hosts. Because SAVE-T was able to complete the tests for all hosts, the tool was able to find more vulnerabilities that exist on the network than Hail Mary was

Automated Capability	Hail Mary	CALDERA	SAVE-T
Scan For Open Ports	✓	✓	✓
Exploit Services - Metasploit	* requires user interaction	X	✓
Exploit Services on a Large Network - Metasploit	* low reliability	X	✓
Gather/Dump Credentials	X	* parser is not reliable	✓
Lateral Movement	X	✓	✓

Table 5.10: This table compares the automated capabilities of CALDERA, Hail Mary, and SAVE-T

able to find.

In some cases, SAVE-T was not able to find all of the Metasploit exploits that existed on the network for the experiment. This was the case in the Small Enterprise, Medium Enterprise and the Enterprise IoT experiments. The unfound vulnerabilities here are due to the MS 17-010_psexec exploit condition not being met by one or more of the hosts on the network.

5.4.2 Capability Comparison

We will now look at a capability comparison between Hail Mary, base CALDERA and SAVE-T. We will qualitatively compare the capabilities that each tool has.

Table 5.10 shows a comparison of the capabilities of Hail Mary, CALDERA, and SAVE-T. The capabilities that we are considering for this comparison are the following: scan an IP address for open ports, exploit network services with Metasploit on a single machine and a large network, gather credentials either by finding default credentials or dumping credentials from memory, and lastly utilize lateral movement.

In Table 5.10, a ‘✓’ indicates that the tool provides support for the capability and a ‘**X**’ indicates that the tool does not support the capability. Some cells have a comment which will clarify why the tool does not provide the capability well.

From Table 5.10, we see that Hail Mary provides support for scanning devices for

open ports. Exploiting services does not provide fully automated support because it requires a user to interact with the interface. In order for Hail Mary to exploit some targets, the user must click to start a port scan on the target; then when the port scan is finished, the user must click another dialogue to start the exploits. We color this yellow because it is not a fully automated workflow in that capacity, it requires user interaction to proceed. Exploiting services on a large network with Metasploit is also colored as yellow for Hail Mary. We make this determination based on the highly unreliable results we gathered from Hail Mary in Table 5.9. Hail Mary does not support any actions for gathering or dumping credentials from a target or moving laterally from one target to another.

CALDERA provides support for both scanning IP addresses for open ports and using known information to move from one host to another. CALDERA also provides a limited credential dumping capability. The Mimikatz credential dumping ability that CALDERA implements has an unreliable parser which leads to erroneous usernames and passwords. The erroneous results can lead to CALDERA attempting to use text which is not a username or password in one of those fields. This behavior wastes a lot of time on the CALDERA system. As mentioned in §3.1.2, CALDERA does not provide support for exploit actions from Metasploit.

SAVE-T provides all of the capabilities displayed in Table 5.10. SAVE-T utilizes the same port scanning ability as base CALDERA. We demonstrated in §4.1 how SAVE-T integrated exploitation using Metasploit and custom exploits. We further demonstrated the use of Metasploit exploitation on networks of increasing size in §5.2. The adversaries described in Table 5.3 demonstrated reliable credential dumping and credential brute forcing capabilities that we used in the SAVE-T experiments. And lastly, the use of lateral movement abilities that are added by SAVE-T is discussed in §4.5 and illustrated with Table 4.7.

5.5 Evaluation Summary

In this section, we have described our testbed which included Windows devices ranging from Windows XP to Windows 10 and Windows Server 2016, 3 different IoT devices, and a Linux Host. We described our 6 experiments and how each was chosen to gradually increase the complexity of the network. We finally illustrated the results of the experiments, showed SAVE-T’s coverage of the top Internet ports, and compared SAVE-T to other tools.

SAVE-T had a linearly increasing run time with respect to the number of ports that were open on the network. The slope of this line indicates that SAVE-T will take approximately 2 minutes and 20 seconds per open port on the network, plus a nearly 1 minute value for the y-intercept. Our data has a strong correlation with the best fit line shown in Figure 5.1.

SAVE-T was able to demonstrate consistent discovery of vulnerabilities which do not require unpredictable conditions to successfully exploit. For exploits which do require special conditions to succeed (e.g., MS 17-010_psexec), SAVE-T is not able to find that the vulnerability exists every time the tool is run.

We examined the top open ports on the Internet according to Censys [14] and the most exploited ports on Exploit-DB [74]. In the current proof of concept form, SAVE-T is able to support 7 out of the top 20 open ports on the Internet and 5 out of the top 10 most exploited ports on Exploit-DB. While these top ranked ports may not be representative of all networks that are in use today, these two data sets provide a reference point for us to use. Coverage of more ports can be added to SAVE-T if a user desires.

We demonstrated how SAVE-T chooses exploits to attempt in a much more scalable way than Hail Mary for the proof-of-concept ports chosen. While Hail Mary performs port filtering to choose which exploits to attempt against a target, SAVE-T

will often use additional keyword filters to reduce the number of exploits that are attempted. With exploits taking a measured average of 0.82 seconds each to launch, SAVE-T's methods of choosing exploits to attempt will result in exploitation taking less time.

When comparing the number of vulnerabilities found by each Hail Mary and SAVE-T during experiments on identical networks, SAVE-T provides a more reliable result. Hail Mary experiences database timeouts when the tool attempts to target more than a small number of hosts simultaneously, which results in many of the target hosts not finishing the experiment. SAVE-T takes a longer time to complete the experiments when the network size increases, but none of the hosts failed to complete the experiment due to timeouts, resulting in more vulnerabilities being found.

Finally we compared the capabilities of Hail Mary, CALDERA and SAVE-T. We saw that while Hail Mary and CALDERA each provide some of the desired capabilities listed in Table 5.10, only SAVE-T was able to provide them all.

6

Future Work

In order to continue the development of an effective tool, we propose two future works. We will discuss future work which includes attack graphs, automated network defenses and plugin development steps.

6.1 Attack Graphs

Attack graphs are security analysis tools that traditionally use manually generated expert input about all known parameters of a computer network (e.g., network connections, running network services, and software vulnerabilities), then generate a graph of potential attack scenarios [69]. The graph will show series of compromises that may occur to allow an attacker to achieve their goal (e.g., compromise a specific host) [33]. Generating accurate attack graphs relies on extensive information about the network [42].

MulVal is one example of a scalable solution for modeling attack scenarios where an expert wishes to determine the potential impact of security vulnerabilities [76]. The software uses a modeling language to map interactions between network nodes by taking into consideration system configuration and software vulnerabilities [76].

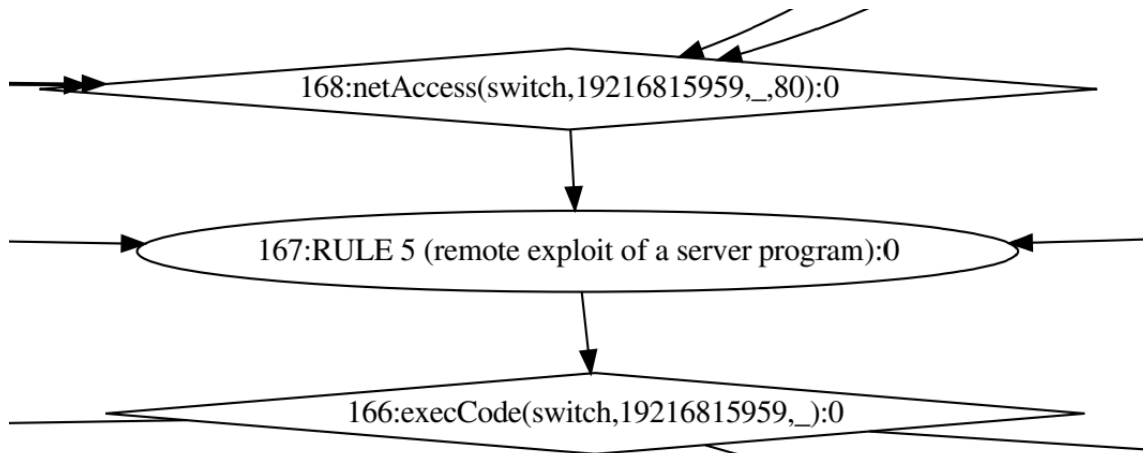


Figure 6.1: Shown is a section of a sample attack graph. The directed graph shows how an attacker with network access to a host running a vulnerable program can exploit the program.

Figure 6.1 shows a section of an attack graph that was generated with MulVal. The section of the graph shows how an attacker with network access to a vulnerable program running on port 80, can exploit the vulnerability to execute code. The full attack graph, shown in Appendix F, is a large file with many complex paths.

MulVal requires an input file(s) which detail the network configuration and known software vulnerabilities/exploits that exist. Producing an input file requires extensive knowledge of the devices on the network and how each device is configured. The input files for MulVal are usually manually generated, however there is some work on automating the generation of input files for other attack graphing software [33].

SAVE-T currently collects information about the network, including hosts, open ports and vulnerabilities that exist on the network. Utilizing SAVE-T's automated collection of this information, a future work could demonstrate translating the SAVE-T operation report into an input file for an attack graphing tool. Automating the workflow of collecting information about the network, then generating the input file for an attack graphing program will address MulVal's requirement for extensive network knowledge and tedious creation of input files.

We have a proof of concept version of a program which will translate CALDERA operation reports to MulVal input files. Our proof of concept program will parse the JSON report that is generated by a CALDERA operation. It is able to parse IP addresses and open ports out of the report. It also has limited capabilities for parsing vulnerabilities. After some design changes to our implementation of exploit abilities, the parser needs additional work to be fully functional with the current version of SAVE-T.

A complete tool will function as a plugin for CALDERA which has it's own user interface tab on the CALDERA web interface. The web interface should allow a user to choose an operation report to parse, then display an attack graph that is automatically generated.

6.2 Automated Defenses

Current IoT research is considering the use of security gateway devices to protect devices from attacks [1, 10, 15, 26]. These gateway devices require the user to program in the rules, signatures, or other items that are used to protect the devices from attack. This implicitly requires the user to know what types of vulnerabilities exist on each device that is protected by the gateway.

SAVE-T automatically gathers vulnerability information. It would be beneficial for network defenders if SAVE-T could also automatically suggest or even implement security rules that would protect the devices from attack at the network level.

A SAVE-T operation report parser could use a taxonomy of vulnerabilities and known defenses to automatically suggest defenses. An additional plugin for this program could automatically generate rules configuration files for a specific IoT security gateway device, or use the gateway's API to programmatically apply network level patches to the vulnerabilities that are found. This would remove the burden of determining and implementing the appropriate mitigations from the network defenders.

6.3 Plugin Development

SAVE-T is built off of CALDERA version 2.3.0, released in August of 2019 [62]. Since we started development of SAVE-T, the main branch of CALDERA has continued rapid development, and has released up to version 2.6.65 as of this writing [63].

Since version 2.3.0, the CALDERA repository has matured not only in its code base but also in its documentation. SAVE-T development was started as proof of concept code that was simply applied on top of existing CALDERA code. We took this approach due to a lack of documentation describing how to properly write a plugin for CALDERA. Now, with proper documentation in place, it will be useful to adjust SAVE-T's code to be a proper plugin for the existing CALDERA system.

By having SAVE-T as a plugin, we can keep separate the code and assumption changes that SAVE-T makes. This will improve the likelihood that our changes will be accepted by the main repository maintainers due to having little impact on the way their existing tool functions. This will take some additional development, however, as the CALDERA code has evolved a lot since SAVE-T development began.

Conclusions

This section will provide some envisioned use cases for SAVE-T, some further insight into design decisions we made and provide final concluding thoughts.

7.1 Use Cases

We envision several use cases in mind for SAVE-T. SAVE-T can be used as a supplement or replacement for traditional offensive security assessments, or it could be used for network defender training.

SAVE-T might be used as a supplement or replacement for traditional offensive security assessments. SAVE-T provides automated actions that an attacker is likely to perform on a network and will inform users of vulnerabilities that exist. Despite this, SAVE-T cannot replace the inventiveness of human actors fully, so a traditional offensive security assessment may still be beneficial.

An additional use case is for training exercises. In exercises where network defenders are scored on their ability to defend against ongoing attacks, human attackers are traditionally used. This can lead to an uneven playing field based on the way different human attackers would perform the same/similar actions. An automated

attacker provides a level playing field in these cases.

Additionally, if a network defense team is to be certified complaint to organization standards, then an automated attacker provides a great way to ensure exact attacker actions are logged along with the success of the action and the time the action was taken. SAVE-T provides an automated attacker which might make certification of abilities easier to standardize.

Lastly, SAVE-T can be used in an on-demand training labs. In labs where users are learning or practicing skills to identify and respond to an ongoing threat, SAVE-T can act as the attacker and perform actions that the student must detect and respond to. Student lab assessment could be performed with a script that determines if SAVE-T was able to perform certain actions and if the student was able to detect those actions. This scenario would allow for a simulated ongoing attack where students learn how to respond.

7.2 Design Decisions

For SAVE-T we often made design choices that went against what the MITRE engineers had originally decided. This section will serve to give some insight into why those choices were made and why our choices might not be ideal for all scenarios.

One design choice we made was storing information that is found together in the same fact. This showed in how we stored credential facts (username and password pairs) as opposed to separate username and password facts. We chose this method of storing credentials to reduce the algorithmic complexity of using the credentials. In the case that usernames and passwords are stored as separate facts, using known credentials to login to a system becomes an exponential problem of trying all combinations of known usernames and passwords. Storing credentials together is a linear problem where you must only try all known combinations. This can save execution time during an operation and prevent wasted effort on invalid

username/password combinations. However, storing credential pairs comes with an additional requirement that all abilities must have a parser which can separate the pair into the appropriate tokens. This may not be desirable in all cases. Especially if parsing a string into tokens is impossible on a limited command line. We overcame this requirement by using Python scripts for our abilities, but this may not be desired either.

We made the design choice to package ability code as Python scripts/binaries. This makes developing new abilities more streamlined because there is no need to rewrite scripts for different command shells. However, this comes with the requirement that a Python script or binary must be planted on a system which is running an agent. Base CALDERA's abilities utilize a more "Live off the Land" approach where they simply use the command line tools that are already on the system. As was mentioned, this makes development more of a hassle, but it comes with the added benefit that no additional tools need to be placed on the system. This could be desired on systems where the network operators do not want additional software to be placed on their systems. Both the streamlined Python development and the CALDERA "Live off the Land" development have their advantages and disadvantages.

These differences in design choices go to show that different methodology might be more suited for different use cases. In MITRE's base CALDERA implementation, maybe they wanted to minimize the impact to hosts on the network. In SAVE-T's implementation, we were not worried about impacting hosts on the network, but more concerned about finding all of the vulnerabilities possible. When designing future tools or improvements for automated offensive security systems, it is imperative to think about the desired use cases and the trade-offs of certain design decisions.

7.3 Final Thoughts

Offensive security assessments use professional hackers to actively find the vulnerabilities which exist in computer networks, but the assessments are expensive and time consuming [4, 71]. To address the high cost of the assessments, there is a call for an automated offensive security assessment [4].

To address the limitations of current automated offensive security assessment tools, we develop SAVE-T, a scalable automated vulnerability scanning and exploitation tool. SAVE-T is built on top of the current state-of-the-art automated offensive security assessment tool, CALDERA. Which we extended it by adding exploitation actions and support for analyzing IoT devices. Furthermore, we strengthen the tool's utility by improving the scalability and reliability when analyzing a large number of hosts and allowing adversary operations to occur for multiple epochs.

SAVE-T has been demonstrated to perform exploitation and lateral movement on networks which included Windows, Linux and IoT devices. While testing networks of increasing size, SAVE-T's runtime scales linearly with respect to the number of open ports on the network. The proof of concept implementations shown in this thesis demonstrates exploitation of Windows, Linux, and IoT devices. We demonstrate use of commonly available exploits in the Metasploit Framework [86] and customized exploit code from Exploit-DB [73]. We also demonstrate testing hosts for default credentials and using those credentials to move laterally around the network. Our abilities cover 7 out of the top 20 open ports on the Internet [14] and 5 out of the top 10 most exploited ports [74]. We also demonstrated that SAVE-T is able to consistently identify vulnerabilities based upon the exploit's reliability.

SAVE-T can be used by professionals to automate offensive security assessments on their networks and can be useful in other training scenarios. SAVE-T will perform an automated offensive security assessment on the network's standard Windows and

Linux hosts in addition to IoT devices. SAVE-T reduces the traditionally high costs of offensive security assessments by automating the processes that professionals would perform.

The SAVE-T source code can be viewed and downloaded from the following GitHub link: https://github.com/jbooz1/SAVE-T_INI-Thesis

Bibliography

- [1] T. Adesina and O. Osasona, “A Novel Cognitive IoT Gateway Framework: Towards a Holistic Approach to IoT Interoperability,” in *IEEE 5th World Forum on Internet of Things, WF-IoT 2019 - Conference Proceedings*. Institute of Electrical and Electronics Engineers Inc., apr 2019, pp. 53–58.
- [2] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, “Understanding the Mirai Botnet,” *Proceedings of the 26th USENIX Security Symposium*, pp. 1093–1110, 2017. [Online]. Available: <https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis>. [Accessed 2019-06-16].
- [3] Apache Software Foundation and Apache, “Releases,” 2020. [Online]. Available: <https://struts.apache.org/https://struts.apache.org/releases.html>. [Accessed 2019-10-19].
- [4] A. Applebaum, D. Miller, B. Strom, C. Korban, and R. Wolf, “Intelligent, automated red team emulation,” in *Proceedings of the 32nd Annual Conference on Computer Security Applications - ACSAC '16*, vol. 5-9-Decemb. New York, New York, USA: ACM Press, dec 2016, pp. 363–373. [Online]. Available: <http://dl.acm.org/citation.cfm?doid=2991079.2991111>. [Accessed 2019-09-21].
- [5] I. Arghire, “IoT Botnets Target Apache Struts, SonicWall GMS — SecurityWeek.Com,” 2018. [Online]. Available: <https://www.securityweek.com/iot-botnets-target-apache-struts-sonicwall-gms>. [Accessed 2019-10-19].
- [6] —, “Mirai, Gafgyt IoT Botnet Attacks Intensify — SecurityWeek.Com,” 2018. [Online]. Available: <https://www.securityweek.com/mirai-gafgyt-iot-botnet-attacks-intensify>. [Accessed 2019-10-19].
- [7] Q. M. Ashraf and M. H. Habaebi, “Autonomic schemes for threat mitigation in Internet of Things,” *Journal of Network and Computer Applications*, vol. 49, pp. 112–127, 2015.

- [8] F. Baiardi, “Avoiding the weaknesses of a penetration test,” *Computer Fraud and Security*, vol. 2019, no. 4, pp. 11–15, apr 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1361372319300417>. [Accessed 2019-05-30].
- [9] T. Bao, Y. Shoshitaishvili, R. Wang, C. Kruegel, G. Vigna, and D. Brumley, “How Shall We Play a Game?: A Game-theoretical Model for Cyber-warfare Games,” in *Proceedings - IEEE Computer Security Foundations Symposium*. IEEE, aug 2017, pp. 7–21. [Online]. Available: <https://ieeexplore.ieee.org/document/8049648/>. [Accessed 2019-06-05].
- [10] P. Bull, R. Austin, E. Popov, M. Sharma, and R. Watson, “Flow Based Security for IoT Devices using an SDN Gateway,” in *IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*. IEEE, 2016, pp. 157–163.
- [11] CarnaBotnet, “Internet Census 2012,” 2012. [Online]. Available: <http://census2012.sourceforge.net/paper.html>. [Accessed 2020-03-21].
- [12] J. Castro, “ThinkPHP 5.x Remote Code Execution,” 2019. [Online]. Available: <https://blog.sucuri.net/2019/04/thinkphp-5-x-remote-code-execution.html>. [Accessed 2019-10-19].
- [13] M. Ceccato and R. Scandariato, “Static Analysis and Penetration Testing from the Perspective of Maintenance Teams,” in *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM '16*. New York, New York, USA: ACM Press, 2016, pp. 1–6. [Online]. Available: <http://dl.acm.org/citation.cfm?doid=2961111.2962611>. [Accessed 2019-06-10].
- [14] Censys, “Censys,” 2020. [Online]. Available: <https://censys.io/ipv4>. [Accessed 2020-03-22].
- [15] H. Chen, X. Jia, and H. Li, “A brief introduction to iot gateway,” *IET Conference Publications*, vol. 2011, no. 586 CP, pp. 610–613, 2012.
- [16] C. Cimpanu, “Chinese websites have been under attack for a week via a new PHP framework bug — ZD-Net,” 2018. [Online]. Available: <https://www.zdnet.com/article/chinese-websites-have-been-under-attack-for-a-week-via-a-new-php-framework-bug/>. [Accessed 2019-10-19].
- [17] CoalfireResearch, “pymetasploit3 · PyPI,” 2020. [Online]. Available: <https://pypi.org/project/pymetasploit3/>.

- [18] Console-Cowboys, “Multiple Trendnet Camera Products Remote Security Bypass Vulnerability,” 2012. [Online]. Available: <https://www.securityfocus.com/bid/51922>. [Accessed 2019-11-23].
- [19] S. Dillion and D. Davis, “MS17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption,” 2017. [Online]. Available: <https://www.rapid7.com/db/modules/exploit/windows/smb/ms17{-}010{-}eternalblue>. [Accessed 2020-03-21].
- [20] P. Dorey, “Securing the internet of things,” in *Smart Cards, Tokens, Security and Applications: Second Edition*, 2017.
- [21] Editorial, “TRENDnet answers to the problems of vulnerable code in their IP cameras - Digital Security Magazine,” 2012. [Online]. Available: <https://www.digitalsecuritymagazine.com/en/2012/02/08/trendnet-responde-ante-los-problemas-de-vulnerabilidad-de-codigo-en-sus-camaras-ip/>. [Accessed 2019-10-19].
- [22] G. Francia, D. Thornton, and T. Brookshire, “Wireless vulnerability of SCADA systems,” in *Proceedings of the 50th Annual Southeast Regional Conference*. ACM-SE, 2012, p. 331. [Online]. Available: <http://www.colasoft.com/packet{-}builder..> [Accessed 2019-06-03].
- [23] V. Gazis, M. Gortz, M. Huber, A. Leonardi, K. Mathioudakis, A. Wiesmaier, F. Zeiger, and E. Vasilomanolakis, “A survey of technologies for the internet of things,” in *IWCMC 2015 - 11th International Wireless Communications and Mobile Computing Conference*. Institute of Electrical and Electronics Engineers Inc., oct 2015, pp. 1090–1095.
- [24] GentilKiwi, “gentilkiwi/mimikatz: A little tool to play with Windows security,” 2020. [Online]. Available: <https://github.com/gentilkiwi/mimikatz>. [Accessed 2020-04-05].
- [25] —, “mimikatz — Blog de Gentil Kiwi,” 2020. [Online]. Available: <http://blog.gentilkiwi.com/mimikatz>. [Accessed 2020-04-05].
- [26] P. Giura and T. Jim, “Sapphire: Using network gateways for IoT security,” in *ACM International Conference Proceeding Series*. Santa Barbara, California, USA: Association for Computing Machinery, oct 2018.
- [27] Hacken, “How much does Penetration Test Cost, or Price of your Security — Hacken,” 2018. [Online]. Available: <https://hacken.io/research/education/>

- how-much-does-penetration-test-cost-or-price-of-your-security/. [Accessed 2020-04-04].
- [28] Hawul, “hahwul/metasploit-autopwn: db_autopwn plugin of metasploit.” [Online]. Available: <https://github.com/hahwul/metasploit-autopwn>. [Accessed 2020-01-07].
 - [29] D. Hedley and M. Jacobs, “The shape of things to come: the Equifax breach, the GDPR and open-source security,” *Computer Fraud and Security*, vol. 2017, no. 11, pp. 5–7, nov 2017.
 - [30] C. Heffner, “Exploiting Surveillance Cameras Like a Hollywood Hacker,” in *Black Hat*. Las Vegas, NV: Black Hat USA, 2013. [Online]. Available: <https://media.blackhat.com/us-13/US-13-Heffner-Exploiting-Network-Surveillance-Cameras-Like-A-Hollywood-Hacker-Slides.pdf>. [Accessed 2019-10-19].
 - [31] H. Hejazi, H. Rajab, T. Cinkler, and L. Lengyel, “Survey of platforms for massive IoT,” in *2018 IEEE International Conference on Future IoT Technologies, Future IoT 2018*, vol. 2018-Janua. Institute of Electrical and Electronics Engineers Inc., mar 2018, pp. 1–8.
 - [32] H. Holm, T. Sommestad, U. Franke, and M. Ekstedt, “Success Rate of Remote Code Execution Attacks Expert Assessments and Observations,” *Journal of Universal Computer Science*, vol. 18, no. 6, pp. 732–749, 2012.
 - [33] K. Ingols, R. Lippmann, and K. Piwowarski, “Practical Attack Graph Generation for Network Defense,” in *22nd Annual Computer Security Applications Conference (ACSAC’06)*. IEEE, 2006, pp. 121–130.
 - [34] A. Iqbal, F. Ullah, H. Anwar, K. S. Kwak, M. Imran, W. Jamal, and A. ur Rahman, “Interoperable Internet-of-Things platform for smart home system using Web-of-Objects and cloud,” *Sustainable Cities and Society*, vol. 38, pp. 636–646, 2018.
 - [35] Jborean93, “pypsexec · PyPI,” 2019. [Online]. Available: <https://pypi.org/project/pypsexec/>. [Accessed 2020-04-05].
 - [36] Jgamblin, “Mirai-Source-Code: Leaked Mirai Source Code for Research/IoC Development Purposes,” 2017. [Online]. Available: <https://github.com/jgamblin/Mirai-Source-Code>. [Accessed 2020-03-21].

- [37] E. Kang, S. Adepu, D. Jackson, and A. P. Mathur, “Model-based security analysis of a water treatment system,” in *Proceedings of the 2nd International Workshop on Software Engineering for Smart Cyber-Physical Systems - SEsCPS '16*. New York, New York, USA: ACM Press, 2016, pp. 22–28. [Online]. Available: <http://dl.acm.org/citation.cfm?doid=2897035.2897041>. [Accessed 2019-06-09].
- [38] C. Kolias, A. Stavrou, J. Voas, I. Bojanova, and R. Kuhn, “Learning Internet-of-Things Security ”Hands-On”,” *IEEE Security and Privacy*, vol. 14, no. 1, pp. 37–46, jan 2016. [Online]. Available: <http://ieeexplore.ieee.org/document/7397713/>. [Accessed 2019-05-21].
- [39] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Snachám, and S. Savage, “Experimental security analysis of a modern automobile,” in *Proceedings - IEEE Symposium on Security and Privacy*. IEEE, 2010, pp. 447–462. [Online]. Available: <http://ieeexplore.ieee.org/document/5504804/>. [Accessed 2019-06-16].
- [40] P. Kulkarni, V. Gandhi, and S. Desai, “A Sneak Peek into Recent IoT Attacks — Zscaler Blog,” 2019. [Online]. Available: <https://www.zscaler.com/blogs/research/sneak-peek-recent-iot-attacks>. [Accessed 2019-10-19].
- [41] C. E. Landwehr, A. R. Bull, J. P. McDermott, and W. S. Choi, “AD-A274 500 A Taxonomy of Computer Program Security Flaws, with Examples,” United States Space and Naval Warfare Systems Command, Washington, D.C., Tech. Rep., 1993.
- [42] R. Lippmann and K. Ingols, “An annotated review of past papers on attack graphs,” Massachusetts Institute of Technology Lincoln Laboratory, Lexington, MA, Tech. Rep., 2005.
- [43] J. Luszcz, “Apache Struts 2: how technical and development gaps caused the Equifax Breach,” *Network Security*, vol. 2018, no. 1, pp. 5–8, jan 2018.
- [44] D. E. Mann and S. M. Christey, “Towards a Common Enumeration of Vulnerabilities,” MITRE Corporation, Bedford, MA, Tech. Rep., 1999. [Online]. Available: <https://cve.mitre.org/docs/docs-2000/ceries.html>. [Accessed 2020-02-13].
- [45] M. Markovic, W. Asif, D. Corsar, N. Jacobs, P. Edwards, M. Rajarajan, and C. Cottrill, “Towards automated privacy risk assessments in IoT systems,” in *Proceedings of the 5th Workshop on Middleware and*

- Applications for the Internet of Things - M4IoT'18*. New York, New York, USA: ACM Press, 2018, pp. 15–18. [Online]. Available: <http://dl.acm.org/citation.cfm?doid=3286719.3286723>. [Accessed 2019-06-09].
- [46] G. Markowsky and L. Markowsky, “Scanning for Vulnerable Devices in the Internet of Things,” *IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, vol. 1, pp. 463–467), 2015. [Online]. Available: <https://www.researchgate.net/publication/294457975>. [Accessed 2020-03-21].
- [47] R. Martin, S. Christey, and D. Baker, “A Progress Report on the CVE Initiative,” MITRE Corporation, Tech. Rep., 1999.
- [48] R. A. Martin, “Managing Vulnerabilities in Networked Systems,” *IEEE Computer*, vol. 34, no. 11, pp. 32–28, 2001. [Online]. Available: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp={&}arnumber=963441>. [Accessed 2019-10-18].
- [49] M. Medwed, “IoT Security Challenges and Ways Forward,” in *6th International Workshop on Trustworthy Embedded Devices*. Association for Computing Machinery (ACM), 2016, pp. 55–55.
- [50] Microsoft, “Microsoft Security Bulletin MS08-067 - Critical,” p. 1, 2008. [Online]. Available: <https://docs.microsoft.com/en-us/security-updates/SecurityBulletins/2008/ms08-067>. [Accessed 2020-04-16].
- [51] —, “Microsoft Security Bulletin MS17-010 - Critical — Microsoft Docs,” p. 1, 2017. [Online]. Available: <https://docs.microsoft.com/en-us/security-updates/SecurityBulletins/2008/ms08-067{%}0Ahttps://docs.microsoft.com/en-us/security-updates/securitybulletins/2017/ms17-010https://docs.microsoft.com/en-us/security-updates/SecurityBulletins/2017/ms17-010>. [Accessed 2020-03-21].
- [52] D. Miller, R. Alford, A. Applebaum, H. Foster, C. Little, and B. Strom, “Automated adversary emulation: A case for planning and acting with unknowns,” MITRE Corporation, McLean, VA, Tech. Rep., 2018. [Online]. Available: <https://www.mitre.org/sites/default/files/publications/pr-18-0944-1-automated-adversary-emulation-planning-acting.pdf>. [Accessed 2019-04-14].
- [53] Y. Minn, P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, “IoTPOT: Analysing the Rise of IoT Compromises,” in *USENIX Workshop on Offensive Technologies (WOOT)*. USENIX, 2015.

- [54] MITRE, “CVE - CVE-2011-2523.” [Online]. Available: <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-2523>. [Accessed 2020-04-16].
- [55] —, “CVE - CVE ID Syntax Change (Archived).” [Online]. Available: <https://cve.mitre.org/cve/identifiers/syntaxchange.html>. [Accessed 2019-10-20].
- [56] —, “CVE - Home.” [Online]. Available: <https://cve.mitre.org/about/>. [Accessed 2019-10-18].
- [57] —, “CVE - CVE-2017-5638,” 2017. [Online]. Available: <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-5638>. [Accessed 2019-10-19].
- [58] —, “MITRE ATT&CK™,” 2019. [Online]. Available: <https://attack.mitre.org/>. [Accessed 2019-04-14].
- [59] —, “mitre/chain: A CALDERA plugin,” 2019. [Online]. Available: <https://github.com/mitre/chain>. [Accessed 2019-10-20].
- [60] —, “mitre/sandcat: A CALDERA plugin,” 2019. [Online]. Available: <https://github.com/mitre/sandcat><https://github.com/mitre/sandcat/>. [Accessed 2019-10-19].
- [61] —, “mitre/stockpile: A CALDERA plugin,” 2019. [Online]. Available: <https://github.com/mitre/stockpile/>. [Accessed 2019-10-20].
- [62] —, “Release 2.3.0 · mitre/caldera,” 2019. [Online]. Available: <https://github.com/mitre/caldera/releases/tag/2.3.0>. [Accessed 2019-10-20].
- [63] —, “Release 2.6.65 - mitre/caldera,” 2020. [Online]. Available: <https://github.com/mitre/caldera/releases/tag/2.6.65>. [Accessed 2020-04-21].
- [64] J. Morgan, “A Simple Explanation Of ‘The Internet Of Things’,” *Forbes*, pp. 2014–2017, 2014. [Online]. Available: <http://www.forbes.com/sites/jacobmorgan/2014/05/13/simple-explanation-internet-things-that-anyone-can-understand/>. [Accessed 2019-10-20].
- [65] J. Muller, V. Mladenov, J. Somorovsky, and J. Schwenk, “SoK: Exploiting Network Printers,” in *Proceedings - IEEE Symposium on Security and Privacy*. IEEE, may 2017, pp. 213–230. [Online]. Available: <http://ieeexplore.ieee.org/document/7958579/>. [Accessed 2019-06-16].

- [66] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, “Demystifying IoT Security: An Exhaustive Survey on IoT Vulnerabilities and a First Empirical Look on Internet-Scale IoT Exploitations,” *IEEE Communications Surveys and Tutorials*, vol. 21, no. 3, pp. 2702–2733, 2019.
- [67] NIST, “NVD - CVE-2007-2447,” 2007. [Online]. Available: <https://nvd.nist.gov/vuln/detail/CVE-2007-2447>. [Accessed 2020-04-16].
- [68] Nmap, “Port Specification and Scan Order — Nmap Network Scanning,” 2020. [Online]. Available: <https://nmap.org/book/man-port-specification.html>. [Accessed 2020-03-21].
- [69] S. Noel, M. Jacobs, P. Kalapa, and S. Jajodia, “Multiple coordinated views for network attack graphs,” in *IEEE Workshop on Visualization for Computer Security 2005, VizSEC 05, Proceedings*. IEEE, 2005, pp. 99–106. [Online]. Available: <http://ieeexplore.ieee.org/document/1532071/>. [Accessed 2019-06-29].
- [70] J. Oakley, “Improving Cyber Defensive Stratagem Through APT Centric Offensive Security Assessment,” *International Conference on Cyber Warfare and Security*, pp. 552–560, 2017. [Online]. Available: <https://search.proquest.com/docview/2018924218?accountid=9902>. [Accessed 2019-04-14].
- [71] —, “Improving offensive cyber security assessments using varied and novel initialization perspectives,” *dl.acm.org*, pp. 1–9, 2018. [Online]. Available: <https://dl.acm.org/citation.cfm?id=3190673>. [Accessed 2019-04-14].
- [72] —, “Towards Improved Offensive Security Assessment Using Counter APT Red Teams,” Ph.D. dissertation, Towson University, 2018.
- [73] Offensive Security, “Exploit Database - Exploits for Penetration Testers, Researchers, and Ethical Hackers.” [Online]. Available: <https://www.exploit-db.com/>. [Accessed 2019-10-18].
- [74] —, “Exploit Database Statistics.” [Online]. Available: <https://www.exploit-db.com/exploit-database-statistics>. [Accessed 2019-11-23].
- [75] C. Osborne, “Mirai, Gafgyt IoT botnets stab systems with Apache Struts, SonicWall exploits — ZD-Net,” 2018. [Online]. Available: <https://www.zdnet.com/article/mirai-gafgyt-iot-botnets-stab-systems-with-apache-struts-sonicwall-exploits/>. [Accessed 2020-03-23].

- [76] X. Ou, S. Govindavajhala, and A. W. Appel, “MulVAL: A Logic-based Network Security Analyzer.” in *USENIX Security Symposium*. USENIX, 2005. [Online]. Available: https://www.usenix.org/event/sec05/tech/full_{-}papers/ou/ou_{-}.html. [Accessed 2019-04-14].
- [77] OWASP, “Category:Vulnerability Scanning Tools - OWASP.” [Online]. Available: https://www.owasp.org/index.php/Category:Vulnerability_{-}Scanning_{-}Tools. [Accessed 2019-12-03].
- [78] —, “OWASP Top Ten,” 2019. [Online]. Available: <https://owasp.org/www-project-top-ten/>. [Accessed 2020-03-23].
- [79] P. N. Pawar, S. Ramachandran, N. P. Singh, and V. V. Wagh, “A Survey on Internet of Things Based Home,” *2019 International Conference on Intelligent Sustainable Systems (ICISS)*, no. Iciss, pp. 76–81, 2016.
- [80] B. Potter and G. McGraw, “Software security testing,” *IEEE Security and Privacy*, vol. 2, no. 5, pp. 81–85, 2004.
- [81] PowershellMafia, “PowerSploit/Invoke-Mimikatz.ps1 at master · PowerShellMafia/PowerSploit,” 2016. [Online]. Available: <https://github.com/PowerShellMafia/PowerSploit/blob/master/Exfiltration/Invoke-Mimikatz.ps1>. [Accessed 2020-04-05].
- [82] PyInstaller, “PyInstaller Quickstart — PyInstaller bundles Python applications,” 2020. [Online]. Available: <https://www.pyinstaller.org/>. [Accessed 2020-04-05].
- [83] Rapid7, “Metasploit: Penetration Testing Software.” [Online]. Available: <https://www.rapid7.com/products/metasploit/>. [Accessed 2020-01-07].
- [84] —, “Metasploitable 2,” 2017. [Online]. Available: <https://metasploit.help.rapid7.com/docs/metasploitable-2>. [Accessed 2020-04-14].
- [85] —, “metasploit-framework/ms17_010_psexec.md at master · rapid7/metasploit-framework,” 2018. [Online]. Available: https://github.com/rapid7/metasploit-framework/blob/master/documentation/modules/exploit/windows/smb/ms17_{-}010_{-}psexec.md. [Accessed 2020-04-08].
- [86] —, “Release 5.0.70 · rapid7/metasploit-framework,” 2020. [Online]. Available: <https://github.com/rapid7/metasploit-framework/releases/tag/5.0.70>. [Accessed 2020-03-17].

- [87] H. T. Ray, R. Vemuri, and H. R. Kantubhukta, "Toward an automated attack model for red teams," pp. 18–25, jul 2005.
- [88] A. Remillano II and J. Urbanec, "New Mirai Variant Uses Multiple Exploits to Target Routers and Other Devices - TrendLabs Security Intelligence Blog," 2019. [Online]. Available: <https://blog.trendmicro.com/trendlabs-security-intelligence/new-mirai-variant-uses-multiple-exploits-to-target-routers-and-other-devices/>. [Accessed 2019-10-19].
- [89] A. Remillano II and M. Vicente, "With Mirai Comes Miori: IoT Botnet Delivered via ThinkPHP Remote Code Execution Exploit - TrendLabs Security Intelligence Blog," 2018. [Online]. Available: <https://blog.trendmicro.com/trendlabs-security-intelligence/with-mirai-comes-miori-iot-botnet-delivered-via-thinkphp-remote-code-execution-exploit/>. [Accessed 2019-10-19].
- [90] R. Roman, P. Najera, and J. Lopez, "Securing the Internet of Things," *IEEE Computer*, no. 9, pp. 51–58, 2011.
- [91] M. Russinovich, "PsExec - Windows Sysinternals — Microsoft Docs," 2016. [Online]. Available: <https://docs.microsoft.com/en-us/sysinternals/downloads/psexec>. [Accessed 2020-04-05].
- [92] J. Scambray, S. McClure, and G. Kurtz, *Hacking Exposed, 2nd Ed.*, 2nd ed. McGraw-Hill Professional, 2000. [Online]. Available: <http://portal.acm.org/citation.cfm?id=572741{%&dl={%&coll={%&CFID=15151515{%&}CFTOKEN=6184618{#}}>. [Accessed 2019-10-29].
- [93] N. Schagen, K. Koning, H. Bos, and C. Giuffrida, "Towards Automated Vulnerability Scanning of Network Servers," in *EuroSec*, Porto, Portugal, 2018, pp. 1–6. [Online]. Available: <https://doi.org/10.1145/3193111.3193116>. [Accessed 2019-05-30].
- [94] L. Segal, "Threat Actors Rapidly Adopt New ThinkPHP RCE Exploit to Spread IoT Malware and Deploy Remote Shells," 2018. [Online]. Available: <https://www.f5.com/labs/articles/threat-intelligence/threat-actors-rapidly-adopt-new-thinkphp-rce-exploit-to-spread-i>. [Accessed 2019-10-19].
- [95] Y. Shen and G. Stringhini, "ATTACK2VEC: Leveraging Temporal Word Embeddings to Understand the Evolution of Cyberattacks," Symantec Research Labs & Boston University, Tech. Rep., 2019.

- [96] Shodan.io, “default password” - Shodan Search,” 2019. [Online]. Available: <https://www.shodan.io/search?query={%}22default+password{%}22>. [Accessed 2019-12-03].
- [97] —, “netcam - Shodan Search,” 2019. [Online]. Available: <https://www.shodan.io/search?query=netcam>. [Accessed 2019-11-23].
- [98] —, “What is Shodan? - Shodan Help Center,” 2019. [Online]. Available: <https://help.shodan.io/the-basics/what-is-shodan>. [Accessed 2019-10-19].
- [99] V. Sivaraman, H. H. Gharakheili, A. Vishwanath, R. Boreli, and O. Mehani, “Network-level security and privacy control for smart-home IoT devices,” *2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications, WiMob 2015*, pp. 163–167, 2015. [Online]. Available: <https://www.researchgate.net/publication/281275810>. [Accessed 2019-05-21].
- [100] SLEEPYA, “Microsoft Windows 7/2008 R2 - ‘EternalBlue’ SMB Remote Code Execution (MS17-010) - Windows remote Exploit,” 2017. [Online]. Available: <https://www.exploit-db.com/exploits/42031>. [Accessed 2020-03-21].
- [101] R. Spangler, “Analysis of Remote Active Operating System Fingerprinting Tools,” University of Wisconsin - Whitewater, Tech. Rep., 2003. [Online]. Available: <http://lcantuf.coredump.cx/newtcp/>. [Accessed 2020-03-21].
- [102] Strategic Cyber LLC, “Armitage - Cyber Attack Management for Metasploit,” 2014. [Online]. Available: <http://fastandeasyhacking.com/index.html>. [Accessed 2020-03-21].
- [103] Symantec, “Internet Security Threat Report Internet Security Threat Report,” Symantec, Tech. Rep., 2016.
- [104] Tenable, “Download Nessus Vulnerability Assessment — Tenable®,” 2020. [Online]. Available: <https://www.tenable.com/products/nessus>. [Accessed 2019-07-07].
- [105] C. Thirumalai and H. Kar, “Memory efficient multi key (MEMK) generation scheme for secure transportation of sensitive data over cloud and IoT devices,” in *2017 Innovations in Power and Advanced Computing Technologies, i-PACT 2017*, vol. 2017-Janua. Institute of Electrical and Electronics Engineers Inc., 2017, pp. 1–6.

- [106] V. Tilemachos and C. Manifavas, “An automated network intrusion process and countermeasures,” in *Proceedings of the 19th Panhellenic Conference on Informatics - PCI '15*. New York, New York, USA: ACM Press, 2015, pp. 156–160. [Online]. Available: <http://dl.acm.org/citation.cfm?doid=2801948.2802001>. [Accessed 2019-06-10].
- [107] TopThink, “top-think/framework: ThinkPHP Framework,” 2018. [Online]. Available: <https://github.com/top-think/framework>. [Accessed 2020-03-19].
- [108] J. Touch, E. Lear, A. Mankin, M. Kojo, K. Ono, M. Stiernerling, L. Eggert, A. Melnikov, W. Eddy, A. Zimmermann, B. Trammell, J. Iyengar, A. Mankin, M. Tuexen, E. Kohler, and Y. Nishida, “Service Name and Transport Protocol Port Number Registry,” 2020. [Online]. Available: <https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml>. [Accessed 2020-04-08].
- [109] TrendNet, “SecurView Wireless N Pan/Tilt/Zoom Network Camera - TRENDnet TV-IP410WN,” 2019. [Online]. Available: <https://www.trendnet.com/support/support-detail.asp?prod=185{ }TV-IP410WN>. [Accessed 2019-10-19].
- [110] VEX WOO and Offensive Security, “Apache Struts 2.3.5 < 2.3.31 / 2.5 < 2.5.10 - Remote Code Execution - Linux webapps Exploit,” 2017. [Online]. Available: <https://www.exploit-db.com/exploits/41570>. [Accessed 2019-10-19].
- [111] VULNSPY, “ThinkPHP 5.x (v5.0.23) GetShell — VULNSPY.” [Online]. Available: <https://www.vulnspy.com/cn-thinkphp-5.x-rce/>. [Accessed 2019-10-20].
- [112] VULNSPY and Offensive Security, “ThinkPHP 5.0.23/5.1.31 - Remote Code Execution - PHP webapps Exploit,” 2018. [Online]. Available: <https://www.exploit-db.com/exploits/45978>. [Accessed 2019-10-19].
- [113] K. Wallace, K. Moran, E. Novak, G. Zhou, and K. Sun, “Toward Sensor-Based Random Number Generation for Mobile and IoT Devices,” *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1189–1201, dec 2016.
- [114] J. Wang, Y. Li, and C. Wang, “Research on ThinkPHP Framework Based on the Mode of MVC,” *Electronic Science and Technology*, vol. 4, p. 45, 2014.

- [115] S. Waterman, “Air Force Managers Can Call in Hackers-for-Hire Under New Contract - Air Force Magazine,” 2020. [Online]. Available: <https://www.airforcemag.com/air-force-managers-can-call-in-hackers-for-hire-under-new-contract/>. [Accessed 2020-04-13].
- [116] R. Williams, E. McMahon, S. Samtani, M. Patton, and H. Chen, “Identifying vulnerabilities of consumer Internet of Things (IoT) devices: A scalable approach,” in *2017 IEEE International Conference on Intelligence and Security Informatics: Security and Big Data, ISI 2017*. IEEE, jul 2017, pp. 179–181. [Online]. Available: <http://ieeexplore.ieee.org/document/8004904/>. [Accessed 2019-06-29].
- [117] E. Winick, “A cyber-skills shortage means students are being recruited to fight off hackers - MIT Technology Review,” 2018. [Online]. Available: <https://www.technologyreview.com/s/612309/a-cyber-skills-shortage-means-students-are-being-recruited-to-fight-off-hackers/>. [Accessed 2020-02-16].
- [118] B. J. Wood and R. A. Duggan, “Red Teaming of advanced information assurance concepts,” *Proceedings - DARPA Information Survivability Conference and Exposition, DISCEX 2000*, vol. 2, pp. 112–118, 2000.
- [119] I. Yaqoob, E. Ahmed, M. H. ur Rehman, A. I. A. Ahmed, M. A. Al-garadi, M. Imran, and M. Guizani, “The rise of ransomware and emerging security challenges in the Internet of Things,” *Computer Networks*, vol. 129, 2017.

Appendix A

Metasploit Variables

Variable Name	Default Value
Username	admin
Password	admin
User	admin
Pass	admin
WP_Username	admin
WP_Password	admin
OatsPassword	admin
SysaxUser	admin
SysaxPass	admin
EFW_Username	admin
EFW_Password	admin
SSH_Username	admin
SSH_Password	admin
TargetURI	/
Form_Path	/
Page	/
Secret	admin
Collector	0
Domain	.
CheckScanner	auxiliary/scanner/smb/smb_ms17_010
CMD	pwd
LHOST	<i>dynamic</i>
RHOST	<i>dynamic</i>
RHOSTS	<i>dynamic</i>

RSRVHOST	<i>dynamic</i>
SRVHOST	<i>dynamic</i>

A.0.1: Shown here are the Metasploit exploit variables and their pre-defined default values. Variables with the value of *dynamic* are variables which are filled in with command line options at run time.

Appendix B

Device Details

This section will provide more details about each of the devices we chose for the experiments.

B.1 Apache Struts

The Apache Struts web-framework is an open source collection of software packages released by the Apache company, designed to host Java web-applications and APIs [3].

One of the most notable CVE entries for the Apache Struts Framework is CVE-2017-5638, a remote code execution vulnerability [43, 57]. This vulnerability is due to an exception handling bug in the Apache Struts framework in versions 2.2.x before 2.3.32 and 2.5.x before 2.5.10.1, both released in March of 2017 [3, 57]. The bug allows a remote attacker to execute arbitrary commands on a system utilizing the file-upload feature by sending specially crafted HTTP headers [57]. This particular vulnerability gained significant attention when attackers were able to exploit it to expose sensitive personal data on 143 million customers of the

Equifax company [29, 43]. It was later discovered that the Mirai IoT Botnet and its variants began attacking this vulnerability to expand their infectious reach [5, 6, 95].

B.2 ThinkPHP

ThinkPHP is a widely used web-application framework designed to speed up the development process of web-apps written in PHP [107, 114]. There are over 45,000 publicly addressable websites utilizing this framework, most of which are in China [16, 94]. Chinese developed IoT devices often utilize this framework to implement management APIs [40].

A Proof of Concept (PoC) remote code execution exploit for ThinkPHP versions less than 5.0.23 and 5.1.31 was published on Exploit Database in late 2018 [16, 112]. This exploit allows a remote attacker to execute arbitrary commands on a vulnerable machine [94]. This exploit is due to poor filtering of user input in the URL [112].

Thousands of websites were targeted by attackers less than 24 hours after the release of the exploit PoC [12, 16, 94]. Similarly, the exploit was quickly included in Mirai botnet variants [12, 40, 89, 94].

B.3 Trendnet TV-IP410W

The Trendnet TV-IP410W is a home/business WiFi surveillance camera. This product has been discontinued by Trendnet and is no longer supported, with a final firmware update in 2017 [109].

A search on the popular IoT search engine Shodan [98] reported that there were over 28,000 devices with IP addresses that matched this type of camera at the time the exploit was reported in 2013 [30]. Today, there are approximately 7,500 of these devices reported by Shodan [97].

A vulnerability in the Trendnet TV-IP410W (and 21 other IP camera products from Trendnet [18]) allows an authenticated attacker to take full control over the device via command injection [21, 30]. A backdoor account included on the device allows an attacker to authenticate themselves before exploiting the command injection by using the hardcoded, default username and password [30].

Appendix C

Experiment Hosts

Network Name	# Hosts	Hosts	Open Ports
Standalone Host	1	Windows 7	445
Minimum Enterprise	2	Windows Server 2016 ^ Windows 7 ^	445 445
Small Enterprise	7	Windows Server 2016 ^ Windows Server 2012 ^ Windows Server 2008R2 ^ Windows 10 ^ Windows 8.1 ^ Windows 7 ^ Windows XP ^	445 445, 8080 21, 80, 445 445 445 445 445
Medium Enterprise	11	Windows Server 2016 ^ Windows Server 2012 ^ Windows Server 2008R2 ^ Windows 10 ^ Windows 8.1 ^ Windows 7 ^ Windows XP ^ Windows 10 Windows 8.1 Windows 7 Windows XP	445 445, 8080 21, 80, 445 445 445 445 445 445 445 445 445

IoT Enterprise	14	Windows Server 2016 ^	445
		Windows Server 2012 ^	445, 8080
		Windows Server 2008R2 ^	21, 80, 445
		Windows 10 ^	445
		Windows 8.1 ^	445
		Windows 7 ^	445
		Windows XP ^	445
		Windows 10	445
		Windows 8.1	445
		Windows 7	445
		Windows XP	445
		RPi - Apache Struts	22, 8080
		RPi - ThinkPHP	22, 23, 80
		Trendnet TV-IP410W	80
IoT Enterprise Plus Linux	15	Windows Server 2016 ^	445
		Windows Server 2012 ^	445, 8080
		Windows Server 2008R2 ^	21, 80, 445
		Windows 10 ^	445
		Windows 8.1 ^	445
		Windows 7 ^	445
		Windows XP ^	445
		Windows 10	445
		Windows 8.1	445
		Windows 7	445
		Windows XP	445
		RPi - Apache Struts	22, 8080
		RPi - ThinkPHP	22, 23, 80
		Trendnet TV-IP410W	80
		Ubuntu	21, 22, 23, 80, 445

C.0.1: This table shows the hosts that were included in each of the experiments with the open ports to the right.

RPi is a host running on a Raspberry Pi.

^ indicates that the host is joined to the Windows Domain

Appendix D

Metasploit Exploit Times

This appendix will detail how we come to our calculation that each Metasploit exploit takes an average of 0.82 seconds to run.

For this calculation, we used a script which will run all Metasploit exploits with default configurations. We perform attacks against a Metasploitable 2 Ubuntu virtual machine [84] that is running on the same host as the attacker script is running. The Metasploitable 2 VM is a highly vulnerable machine that is released by the Rapid7 company to allow users to practice exploiting services with Metasploit. We choose this because it is a highly vulnerable machine with many services already configured in a vulnerable state. The ports and services that are open on this machine are shown in D.0.1.

We run all 1947 Metasploit exploits that were available at the time for Metasploit Release 5.0.70 [86]. Our program which ran all 1947 exploits against the Metasploitable vulnerable target with default exploit configurations took 1597 seconds to complete execution. We take 1597 seconds to complete all exploits and divide by 1947 total exploits to determine an average of 0.82 seconds per exploit.

This method approximates an average time that it would take an arbitrary Metasploit exploit to succeed, but does not accurately represent all random samples of exploits. For example, see D.0.2. This table shows the average seconds per exploit only for those exploits which were chosen by SAVE-T based on the network designs we had in our test bed. Notice that the average seconds per exploit in these experiments is much higher than the previously calculated 0.82 seconds per exploit. This is likely because our network results in a lot of SMB exploits, which are often complex exploits which require a series of packets over several seconds to complete. Some of the more prevalent exploits on Metasploit, such as the high number of web port exploits, can often be a simple one HTTP request exploit which take much less time.

A more accurate way to estimate the time taken to launch each exploit might be to individually test each exploit being used against a target which is vulnerable to that exploit and timing how long it takes for the exploit to complete. With each exploit's known completion time on a vulnerable target, we can average the completion time for all 1947 exploits. This method is not feasible due to the amount of time it would take to setup the vulnerable targets for all 1947 exploits and run each exploit manually.

Port	Service
21	FTP
22	SSH
23	Telnet
25	SMTP
53	DNS
80	HTTP
111	RPCBind
139	NetBIOS
445	SMB
512	Execution
513	Login Session
514	Shell
1099	RMI Registry
1524	Ingress Lock
2049	NFS
2121	CCProxy FTP
3306	MySQL
3632	DistCCD
5432	PostGreSQL
5900	VNC
6000	X11
6667	IRC
6697	Unknown
8009	AJP13
8180	Unknown
8787	Unknown
39292	Unknown
43729	Unknown
44813	Unknown
55852	Unknown

D.0.1: Shown are the open ports and services for the Metasploitable 2 VM [84]

Network Name	Total Seconds	Exploits Attempted	Seconds / Exploit
Standalone Host	131	33	3.97
Minimum Enterprise	232	66	3.52
Small Enterprise	1057	307	3.44
Medium Enterprise	1548	439	3.53
IoT Enterprise	1592	498	3.20
IoT Enterprise Plus Linux	1741	607	2.87

D.0.2: Seconds per Metasploit exploit for the exploits which were chosen based on our network systems. Total Seconds is sourced from the SAVE-T exploit run time data presented in Table 5.9. Exploits Attempted is sourced from the SAVE-T exploits attempted data presented in Table 5.8.

Appendix E

Number of Exploits

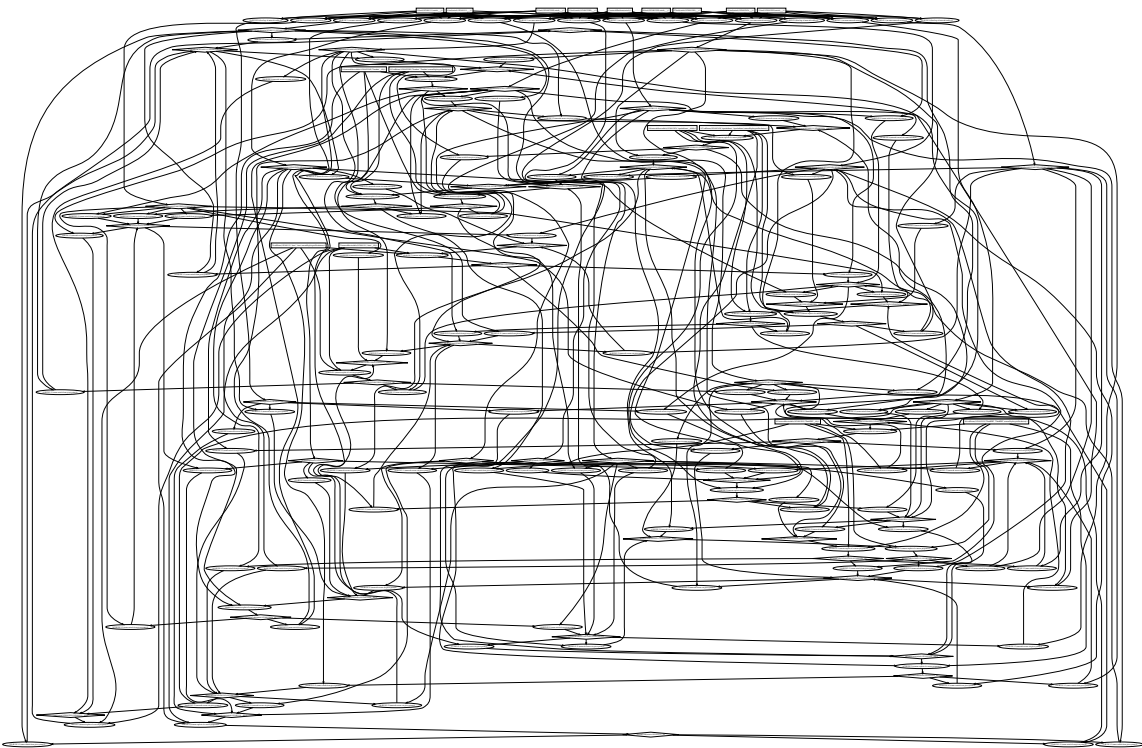
Port (– keyword filter)	# of Exploits
8080	58
21	48
445	33
80 – Apache	21
80 – IIS	15
8080 – Apache Struts	13

E.0.1: Shown is the port and optional keyword filters that are used by SAVE-T and the number of Metasploit exploits that will be attempted as a result.

Appendix F

Attack Graph

Shown in F.0.1 is a large attack for a network that has 4 hosts. Each of the hosts on the network has code execution vulnerabilities. The graph shows many paths that an attack could take from one host to another through the network to reach a goal of executing code on one of the hosts.



F.0.1: Shown is a sample attack graph. The directed graph shows paths that an attacker can take from a starting position on the same subnet as the target device to executing code on the target device. The network illustrated here consists of 4 nodes which all have code execution vulnerabilities.