ATHABASCA UNIVERSITY

A FEDERATED APPROACH TO IDENTIFYING ADVANCED PERSISTENT SECURITY

THREATS ON ENTERPRISE COMPUTER NETWORKS

BY

WADE W. WESOLOWSKY

A PROJECT submitted in partial fulfillment

Of the requirements for the degree of

MASTER OF SCIENCE in INFORMATION SYSTEMS

Athabasca, Alberta

October, 2019

DEDICATION

There are two types of student, the self-taught, and the hopeless.

Our job is to turn the hopeless into the self-taught.

— Piotr Rudnicki

This work is dedicated to my family, friends, and teachers for guiding me through life.

ABSTRACT

Computer Security is an intense flash point of concern in the modern computing landscape. Many threats to computer systems both known and unknown prey on the mind of computer professionals. Advanced Persistent Security Threats (APST) are a species of clandestine attack which infiltrate computer systems to exfiltrate data and struggle to maintain an everlasting foothold inside the target network. They are of particular concern thanks to their use and sponsorship by state actors in proxy battles and covert operations. In this landscape we chart the rise of the threat and search for free and open source software which can be used on an enterprise network to detect the threat. Uncovering the threat is achieved with a collaboration between the security tools cooperating together as components within a Federated Security Module (FSM) built using the Elasticsearch, Logstash, and Kibana (ELK/Elastic) Stack. A federated aggregation between pfSense, Snort, Sweet Security (Zeek), Wazuh (OSSEC), and HoneyTrap will offer a centralized data source from which their combined knowledge can be accessed. The efficacy of the federation is evaluated using a test suite on a simulated enterprise network to provide us with opportunities for improvement and lessons learned for future research.

Keywords: Intrusion/anomaly detection and malware mitigation; Malware and its mitigation; Intrusion detection systems; Malware / spyware crime; Advanced Persistent Threat; Combination, fusion and federated search.

ACKNOWLEDGEMENTS

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS, ABBREVIATIONS AND NOMENCLATURE

Symbol	Definition
APST	Advanced Persistent Security Threat
APT	Advanced Persistent Threat
API	Application Programming Interface
C&C	Command and Control
C2	Command and Control
CIA	Central Intelligence Agency
COTS	Commercial Off The Shelf
DHCP	Dynamic Host Configuration Protocol
DMZ	Demilitarized Zone
DNS	Domain Name System
DSL	Domain Specific Language
ELK	Elasticsearch, Logstash, and Kibana
FSM	Federated Security Module
HIDS	Host-Based Intrusion Detection System
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
IDS	Intrusion Detection System
IT	Information Technology
JSON	JavaScript Object Notation
LAN	Local Area Network
MAC	Media Access Control
MITM	Man-in-the-Middle Attack
NAT	Network Address Translation

NIC	Network Interface Card
NIDS	Network Intrusion Detection System
NSA	National Security Agency
OS	Operating System
OSINT	Open-Source Intelligence
OSSEC	Open Source HIDS Security
OSSIM	Open Source SIEM
RAT	Remote Access Trojan
SIEM	Security Information and Event Management
SMS	Short Message Service
SQL	Structured Query Language
SSL	Secure Sockets Layer
TCP/IP	Transfer Control Protocol/Internet Protocol
TLS	Transport Layer Security
WAN	Wide Area Network

CHAPTER 1

INTRODUCTION

1.1 Statement of Purpose/Goal

The research objective is the study of existing tools and methods which can be leveraged to detect and remove APST within enterprise computer networks. After a review of the literature, a federated security model (FSM) design will be proposed and implemented to integrate select tools and techniques together in a federated manner which will help to mitigate threats and vulnerabilities within an enterprise environment. The primary goal of the FSM will be detection of intrusions and vulnerabilities, while the subjugate goal will be removal or elimination of the identified threats or vulnerabilities.

Creating, building, or customizing software to perform the federation of various software security tools will be a key focus of our work. Due to the federation of components, there will be problems of overlapping, non-deterministic tool behaviour, and mismatch between the tools while still having the requirement to maintain interoperability between separate components [3].

Another goal of our research will be to prototype our FSM on a simulated enterprise network. This will require the setup of a lab environment running similar services and software that might be present in an enterprise network environment. Then we would run the components of our FSM which will be used for the detection of APST. Since these threats are often directed by a human intelligence, the research will implant several types of malware within the enterprise environment to observe the efficacy of the FSM at finding the threats. An analysis of how the FSM performs during the simulation should suggest additional avenues for future research.

1.2   Research Problems

A plethora of research problems are inspired by our research project. We need to ask questions regarding why APSTs are created and what motivates their creators? Do APST have a standardized life cycle and well defined stages with certain behaviour at each stage? What is the composition of a typical enterprise network which an APST might attack? What characteristics do APSTs possess which can be used to detect them? Investigation of these questions should allow us to design a FSM which can assist in the detection of APST. Once some methods of detection are outlined and tested, we will speculate on some possible future directions for our research.

1.3   Rationale for the Research

Curiosity surrounding ideal ways of protecting Information Technology (IT) assets from security threats is a growing area of study. The interconnected nature of computer systems in the modern world makes this problem even more difficult due to the multitudinous of ways devices can communicate and exchange data on a network. All the components in a computer network, including the human operators, introduce attack vectors for a determined adversary to exploit. The system administrator and other computer professionals tasked with protecting the network will often take baseline precautions to mitigate these threats. These precautions might take the form of frequent patches, password policies, user education, running anti-virus software, or other defensive measures. Often baseline protections are adequate when trying to prevent damage from automated, routine security threats like spam and botnets. However, they often fall short when there is a human intelligence directing the attack - when the attacker takes the time to learn the network and tailor the attack to the vulnerabilities therein. Companies will often hire a penetration tester to attempt a break-in of their network in order to expose and report on potential vulnerabilities [4]. Often, the attempt is successful at illustrating systemic problems that make the total compromise of the computer system possible. The custodians of the network often ask how this is possible when baseline security precautions were taken and studiously followed.

The inevitable conclusion must be that baseline network security is inadequate to the challenges present in a modern computer network. Even when a minor system glitch is detected, it can be a warning indicator of a much larger and deeper issue [5]. Without the ability to police these security threats, there is little chance of the enterprise network remaining secure. While many threats exist, we would like to focus on one of the most salient threats which is the Advanced Persistent Threat (APT). APT can be defined as "a sophisticated attack composed of several steps and performed by experienced and highly skilled actors with a great determination to achieve their goal [6, p. 1][7]." These threats are *advanced* due to their sophisticated nature and multi-staged methods used to attack the IT system. They are *persistent* because their foothold within the network is non-volatile and tends to be of a permanent nature. They are a *threat* due to their sneaky, often undetected, exploitation of vulnerabilities within the computer system. Within our terminology we have scoped the definition of APT as Advanced Persistent Security Threat (APST) to highlight our focus on computer security threats versus other threats which maybe be present on computer systems such as coding errors, bit rot, data loss, etc.

Since APST are so difficult to mitigate there is ample opportunity to explore available options for their detection and removal. This field is also exploding with the rise of malware and hacking tools created by state actors [8], and we see only opportunity for academic exploration within this domain. Some writers even suggest the deployment of systems to monitor your employees to watch for insider threats [9]. There is no lack of innovation, and we hope that our Project will add further ideas to this endeavor.

1.4   Potential Impact and Contributions of the Research

There has been a consistent growth of security threats which target computer systems and networks over the years [10]. Each year many new and previously unknown threats are discovered by security researchers, who studiously document them along with potential software patches to fix them. This happens on such a regular basis that anti-virus software must be updated daily and

most software vendors have moved towards software patching models which allow for a consistent delivery of updates and upgrades to their software over the Internet [11].

Specifically in the realm of APST the sophistication of threats has slowly advanced as researchers find and analyze threats on a continual basis. When a new threat is discovered often it is reverse engineered by security researchers [12] in order to get a better idea of how it is constructed. This gives valuable insights into how the APST functions and sometimes even gives information about its source. However, it also allows researchers to learn new techniques of software exploitation which can be used to create new malware.

We would strongly believe that most corporate and government networks have already been compromised in some way and they are simply not aware of it. This is because one of the important features of APST is avoiding detection[13] in order to succeed at maximizing data exfiltration from the compromised network. Without tools and techniques to at least find this threat it will continue to plague the network [14] and most likely allow long term loss of corporate secrets and other confidential data.

A quick browse of WikiLeaks is all that should be required to justify the present research. It is quite clear that software exploitation has become a central focus within the Central Intelligence Agency (CIA), especially in relation to the Vault 7 leaks which have been ongoing [15]. There is an important section regarding how to evade forensics and anti-virus software [16], [17], [18], [19]. This information demonstrates that state actors are well versed in evading detection while exploiting target systems and exfiltrating data. There can be no doubt that such practices are ongoing today.

We have argued that the risks to network security are only growing and shown how one state actor (CIA) may be developing exploitation tools. Within this space there is an urgent need to for better defense techniques which will help to alert security professional to potential risks within their enterprise networks. Furthermore, the arguments presented tend to suggest that this defense should be able to handle unknown threats. The impact of our research is greatly dependent on the

programming skill of researcher, how astute they are to federate tools together, and the novelty of their ideas!

1.5 Technical Background of the Research

In this section we explore some of the reasons why APST have become such a large nuisance in enterprise networks along with some of the reasons they are becoming an even more prevalent threat. The reasons explored include the expansion of network medium, the perpetual discovery of vulnerabilities, the ongoing value of data contained within the networks, the specter of cyberwarfare, complexity of enterprise networks, the evolution of countermeasures, and aging IT infrastructure. All these areas collectively illuminate reasons for the proliferation of APST and highlight some of the problems and opportunities present in that space. We do not claim these potential causes are exhaustive or complete.

1.5.1 Enterprise Computer Networks

Computer networks were historically limited by physical wires which connected the computer systems. Presently the network media could be cellular, satellite, or wireless signals. There is less reliance on wired connections and an increasing focus on non-wired devices. This means that any computer system that is able to send and receive is at risk from nefarious signals. This risk can even extend to computer microphones and speakers which can be used to transmit and receive in a local area [20]. In general, an air gap between systems is no longer a deterrent to information leaving your computer [21]. Additionally, these types of attacks are difficult for the user to know about because humans cannot perceive radio frequency signals or hear the ranges that the malware transmits at, and thus there is no warning signs that malware is working against them.

1.5.2 The Value of Data

All organizations contain valuable data. This data often has technological and commercial value, which means that other people would like to gain access to this information. APST would be a great way for an actor to gain and maintain access to this data. The type of data which might be stolen can be almost anything from passwords to behavioural profiles.

There is an argument made that malware may even be used in a stealing-reality attack [22]. This means that malware may be present in a system specifically to slowly spread and steal data for behavioural analysis. This type of attack is deadly because it can be used to build profiles on specific human targets on patterns of their behaviour that may not be easily changed.

Malware can even be used to gain political [23] advantage over opponents. We can imagine the computer accounts of powerful people being compromised by an attack in order to track or blackmail these people. It is apparent from the Edward Snowden leaks [24] that this type of surveillance has already been wildly implemented by the National Security Agency (NSA). APST definitely plays a role in these types of attacks, as often computer systems must be compromised over the long term in order for reliable intelligence to be extracted gradually. Furthermore, often the adversary has an IT environment that is configured in such a way that generic attacks would be unsuccessful, so customized advanced attacks are necessary.

Another recent article [25] points out that cracking computer networks to monitor news reports in order to find information which could affect stock prices, and then using that information to place stock trade requests is a viable way of using APST to make money. When criminal enterprises are able to gain access to information exchanges they have the ability to use that information to gain monetary advantage.

1.5.3 Vulnerabilities and Threats Everywhere

Each day software vulnerabilities are discovered and exploited. All one has to do is glance at the Bugtraq Mailing List or the Full Disclosure Mailing List to get a sense of how frequently

this occurs. The onslaught of such notices gives even the most stoic reader pause, and highlights just how frequently software contains bugs, undocumented features, or other quirks which allow it to be compromised in many ways. The sheer number of vulnerabilities shows the dangers of using an unpatched system, and even if you are using a patched system it shows that it would be impossible to trust your software to be bug free. The simple fact is that no matter which system you are running someone will have an exploit for it.

As a software consumer, there is even the risk that the software you have purchased contains back doors or other side-channels which allow covert access and surveillance. One long running debate is that your should have access to the source code of products which you run on your computer and whether that will contribute to the security of the software [26]. Without auditing the source code, it is very difficult to trust that your computer is executing the code you would expect.

Even areas of the computer system which historically have been considered safe are now coming under attack. There are well known techniques for re-flashing the BIOS of a computer system and embedding malware inside it [27]. There are also many places for malware to hide, "between Linux [the Operating System] and the hardware [because there is]...at least 2 1/2 kernels [between them] [28]." A more recent scare has broken out regarding the use of the Minix operating system within Intel processors, "Thanks for putting a version of MINIX inside the ME-11 management engine chip used on almost all recent desktop and laptop computers in the world [29]."

1.5.4   The Source of Vulnerabilities: System Complexity and Aging Infrastructure

The enterprise network environment is always trying to push the boundaries to offer new services to its users [30]. We can see the shift in the last few years to bring-your-own devices and allowing workers to work from home using company provided computer systems. Within the network itself, systematic design of VLANs and other interconnectedness is often poorly understood [31]. This means that often the corporate network environment is in a constant state of redesign engineering flux where new threat avenues may be created from many different angles. Furthermore,

7

the sheer complexity of the environment often means that no single individual has a full picture of how all the services work together to create a unified whole. Often there are many specialists which oversee their own small parts of the enterprise network. Problems can be exacerbated because of inadequate communication that exists within corporate structures. Should a security threat find its way into an enterprise network removing it will be a difficult challenge and it may be impossible to verify that it has been removed completely.

No analysis would be complete without the mention of aging legacy IT systems present within any computing environment. From rumors about a forgotten server hiding in a back room or the necessity of keeping an aging computer system operational to ensure business continuity, often the enterprise depends on legacy IT systems. These systems present a unique challenge and their modernization is a concern for computing professionals [32]. However, these systems present a soft target for malware as they are often poorly understood and sparsely maintained. Furthermore, skill sets within industry for their operation are often fading and difficult to find [33] so they may be maintained by system administrations who do not understand their full complexities.

1.5.5 Countermeasures

APST have undergone many different evolutions over the years, and their writers have come up with more and more advanced countermeasures to prevent their detection and removal [34]. This has been briefly studied using such systems as BEAGLE [35] which shows how malware authors refine their malware over time. It is important for us to realize during our research that new techniques will be created which may make our detection tools ineffective or obsolete and thus the current research cannot arrive at a perfect solution. Rather a balance will need to be reached which allows adequate protection on the enterprise network.

1.5.6 Cyberwarfare

With nation states taking notice of the importance inherent in Information Technology infrastructures and their ability to subvert these infrastructure for various ends, there has been some

dialogue about cyberwarfare. The digital frontier can be thought of as trench warfare in many respects, as it is still new and untested. The best defenses are multi-layered [36], however the visual of many trenches might be more apt. An aerial bombardment which bypasses the trenches, shows how inadequate this method of defense might be. We definitely see that superpowers [36] often blame each other for network intrusions and lack of cooperation on these issues.

While the specter of cyberwarfare is often maligned, the threat of other types of operations, like PsyOps [37] show that gaining control of a system might allow you to plant disinformation. So, while stealing information might be important - planting information might be just as effective. Therefore, securing systems often means that you need to make sure the information in them is not modified or altered in ways which benefit your adversary.

Cyberwarfare may not be limited to state groups, and many of the current threats originate from elsewhere [38]. The conflict between hacker groups (which happen to be different national groups) are in some cases international conflicts of ideology. This paper accepts that using patriotic groups within the country to advance the national agenda is possible.

Irregardless of the actors in question, it is clear that cyberwarfare is a discussed topic. While at this time there is scant evidence of a true cyber-war happening between nation states. There is evidence for skirmishes between long time rivals, like the USA and China. Any large company tied to a nation state would be wise to consider the possibility they may be targeted in the future and plan accordingly.

1.6 Definition of Terms

Advanced Persistent Security Threat - a species of clandestine attack which infiltrate computer systems to exfiltrate data and struggle to maintain an everlasting foothold inside the target network.
Command and Control - malware which runs on a target system and accepts remote commands to direct the behaviour of the host system, usually associated with commands being sent from one central site to many infected hosts.

Domain Name System - a method for resolving a human readable address to an IP address.

Elasticsearch, Logstash, and Kibana - a software suite used for collecting logs from multiple sources, indexing them, and making them available for real-time search and visualization.

Federated Security Module - a software federation built around the Elasticsearch, Logstash, and Kibana (ELK) Stack through the cooperation of pfSense, Snort, Sweet Security (Zeek), Wazuh (OSSEC), HoneyTrap, and Sysmon (Windows Event Logs).

Host-Based Intrusion Detection System - endpoint monitoring system designed to detect any intrusion or hostile attacks against a host system.

Intrusion Detection System - a device designed to detect any intrusion or hostile network traffic.

Local Area Network - a computer network which spans a small area like a small office, or one floor in an office tower.

Man-in-the-Middle Attack - when an attacker intercepts network transmissions to modify or alter them.

Network Intrusion Detection System - a network device designed to detect any intrusion or hostile network traffic. Often intrusion detection system and network intrusion detection system are used interchangeably.

Open-Source Intelligence - intelligence gathered from public sources like the Internet, newspapers, and other general access databases.

Open Source HIDS Security - an open source host based intrusion detection system which runs on multiple host operating systems.

Open Source SIEM - a specific SIEM implemented using an amalgamation of open source software and tools.

Remote Access Trojan - malware installed on a device which allows local administrative functions to be accessed remotely.

Security Information and Event Management - "is a platform that enables the collection and real-time analysis of security events from different components and applications in order to detect

and prevent internal and external threats [39]."

Secure Sockets Layer - is a secure method of negotiating and creating a private connection between a web server and a browser.

Transfer Control Protocol/Internet Protocol - "a suite of protocols which provides a language that can be used for computers to talk to each other [over the Internet] [40]."

Transport Layer Security - is a secure method of negotiating and creating a private connection between a web server and a browser, the predecessor to Secure Socket Layer.

Wide Area Network - a computer network which spans a large distance, often consisting of multiple local area networks.

1.7 Organization of Remaining Chapters

This work describes how we can detect APST on a Windows network using various sensors which form part of a software federation. In the introduction we have described why this research is important, some of the research problems we would like to investigate, why our research is valuable, and talk about some of the technical reasons why our research is necessary.

Chapter 2 is an overview of historical research in this area and looks at some of the technical and academic publications by other authors. We also investigate what other software has attempted to look into our research questions. We draw all this together using a series of mind maps which are a visual representation of ideas and concepts pertinent to our research. A brief overview of the limitations of current approaches is outlined with a discussion on potential solutions.

Chapter 3 looks at our research methodology and mentions how we did our literature and technical review. We discuss how we selected the specific tools which form the components of our federation and a brief summary of them. A high level overview of our penetration testing lab network setup is detailed along with how our security software is located among our network components. The development of our FSM Kibana Plugin is highlighted along with some discussions of tests for the federated system.

Chapter 4 talks about the detailed layout of the penetration testing lab.

Chapter 5 details how the selected software tools were installed and configured which is a high level detailed process of getting the entire software ecosystem setup. We also give some guidelines to navigate past difficulties with setup in future project iterations.

Chapter 6 does a deep dive in to the programming of the Kibana plugin and gives some more detailed technical analysis of how it works.

Chapter 7 presents an overview of APT Hunter which is a command-line tool written in Python to search the various indexes for indicators of APST.

Chapter 8 talks about the penetration tests we did to give some idea of how well our tools detect various network attacks. We use some baseline configurations and then use to software tools APTSimulator and FlightSim to approximate APST attacks on network hosts and devices. We provide frequent examples of data returned by our tests to give the reader an idea about what our tools detect in a practical sense.

Chapter 9 deals with the strengths and weaknesses of our current approach. We also mention any issues which were unresolved after our current investigation.

Chapter 10 has concluding remarks about the significance of our results and some forward looking statements about future work.

The appendices contain raw JSON log files which are returned by Elasticsearch queries. The query DSL examples show the queries we used in our Kibana plugin to return useful information from our federation. Finally, the output from running FlightSim is given.

CHAPTER 2

REVIEW OF RELATED LITERATURE

2.1   Academic and Technical Publications

APST have been documented to have several distinct phases in their life cycle [2], [41], [42], [43]. While these stages may differ between published papers, it is widely accepted that humans are exploited in the early stages of the attack [44]. The stages also provide differentiation for the strategies that can be deployed for detection as summarized in Table 2.1. Therefore, we will briefly look at the various stages as identified in the papers for completeness of the literature review and to make sure the reader is familiar with them. Depending at which stage an infection is detected will suggest potential areas for removal or containment of the malware within the enterprise network. For a real-world analysis of APST stages of attack and some of the related actions taken by the malware at each stage please see [45].

The APT attack stage model adopted during our investigation will be the one contained in [2]. This model is selected because it is compact with four stages, each stage is clearly laid out in the paper presented. It is not as in depth as the phases identified in [43], but nevertheless we believe it is sufficient for our purposes and you can refer back to the comparison between the models as a reference. This model has four main phases: prepare stage, access stage, resident stage, and harvest stage. Within each of these main stages there are several sub-stages which the threat actors will take [46].

One of the high level methods of mitigating APST is to model the network and the various vulnerabilities that exist on it. One method which could be used by a security analyst is the Optimized Attribute Attack Graph which models the network and potential attack vectors and methods [47]. This graph can show the potential attack avenues for multiple hosts in the network, however the authors do not present an automated method of generating these graphs. The idea of presenting a

holistic view of the network lead us to another paper which presents an ontology based approach where various data providers can contribute to behaviour detection on the network. This approach known as TAON gives, "an OWL-based ontology offering a holistic view on actors, assets, and threat details, which are mapped to individual abstracted events and anomalies [48]." This method will aggregate data from various data providers which are software components which provide information. This is a breed of behavioral detection systems which are posited as the next generation of APST detection methods because they model the stages of attack and try to use those as methods of detection. Unfortunately, this approach did not come with a corresponding software implementation so while the idea is novel we cannot implement it during our project. However, it does highlight the use of semantic web capabilities for the detection of threats.

The life cycle of the APST is important when considering different detection and removal methods for them. There is much research done into detection, but much less research was discovered for removal.

2.2   Available Tools and Software Systems

Most detection methods usually rely on network traffic analysis and inspection [49]. One paper which stood out was [50] which compared the network monitor implementations which were broken into packet capture, deep packet inspection, and flow-based observation. The paper brought Zeek to our attention because it is flexible and allows the creation of various detection methods. Another method of APST network detection is to detect traffic flows for Remote Access Tools running within internal enterprise networks [51]. The inspection of traffic by a host-based detection module on each IP enabled device was an effective way of finding over 90% of the threats on infected systems. It is important to monitor internal hosts because often defensive strategies use an eggshell approach where the perimeter firewall is the only place where attacks might be stopped. The collaborative monitoring of hosts within the network is important to stop the lateral movement of APST within the network [52]. A final paper presents various real world recommendations

14

for detection of threats on the network [53] however it is only a proposal and does not present real-world results for analysis.

Another proactive defense strategy is the use of honeypots which sit on the network and mimic real systems to lure APST to them [54]. The use of KFSensor was cited as the selected solution by these authors, for our uses we will need to select more open source tools for our investigation. However, the main idea is the use of proactive notifications which are sent to the network administrator when the honeypot receives incoming network traffic. A recent source of information regarding honeypots can be found in [55], [56], and [57] which provide many examples of existing honeypots to choose from.

The removal of APST was not an easy find in our investigation. When a network is compromised it would be prudent to simply take all infected hosts off-line and rebuild them from known good backups. The detection components can be used to show when a host is compromised. "Across the first organisation's estate of 5,000 plus machines, there were four that were infected with the malware [58]," they could be fixed by using a backup solution to restore the affected hosts after forensic analysis. Removal becomes even more difficult because often the best malware authors build on the successes and mistakes of other malware authors when creating their malware designs [34].

During the search for APST detection tools we came across Open Source Security Information Management (OSSIM) which is a collaboration of open source tools to perform security and asset monitoring. While not specifically tailored towards APST they may use some tools which are similar to our proposed solution. Also, their solution is more broadly based to take into account a wider variety of threats.

2.3   Mind Maps

The mind map is a visual representation of interconnected concepts to assist in finding non-obvious connections. We have used it here to summarize some of our initial investigation into

the properties of APST. All mind maps were creating by using CmapTools and heavily relied on research from [59]. In order to fit on the page, we have had to scale Figure 2.1 down to make it fit. We also have extracted some of the high level concepts into subgraphs to make viewing them easier. The main concept in each subgraph corresponds to a concept bubble in the main Figure 2.1.

Figure 2.1 shows the high level properties of APST and interconnected components.

Figure 2.2 shows some of the common initial attack vectors of APST.

Figure 2.3 shows some of the famous APST cases mentioned in academic literature which can be used as reliable case studies of attacks.

Figure 2.4 shows some indicators that could be detected on a host or network to suggest APST infection.

Figure 2.5 shows some of the software tools that would be helpful in detecting APST on an enterprise network.

Figure 2.6 shows the stages in the APST life cycle.

2.4   Problems with the Existing Technical Approaches and Tools

Many of the existing solutions for detecting APST are difficult and costly to implement, often consisting of commercial software systems [60]. Even when solutions are non-technical, like in the case of user education, the focus is often not on APST and more on general personal behaviours to improve overall system security while avoiding potential pitfalls. Often detection methods are not tailored to advanced APSTs and are only concerned with stopping common malware threats [61]. This means that there is still a need for cost effective detection technology specifically tailored towards APST.

By examining the available life cycle models of APST it is possible to figure out which stages different software components may be useful [43]. Furthermore, due to the multiple phases, "...a single technical solution may not be sufficient to make an organisation immunised against the APT attacks without wanting to neglect their roles to align with good practice and compliance
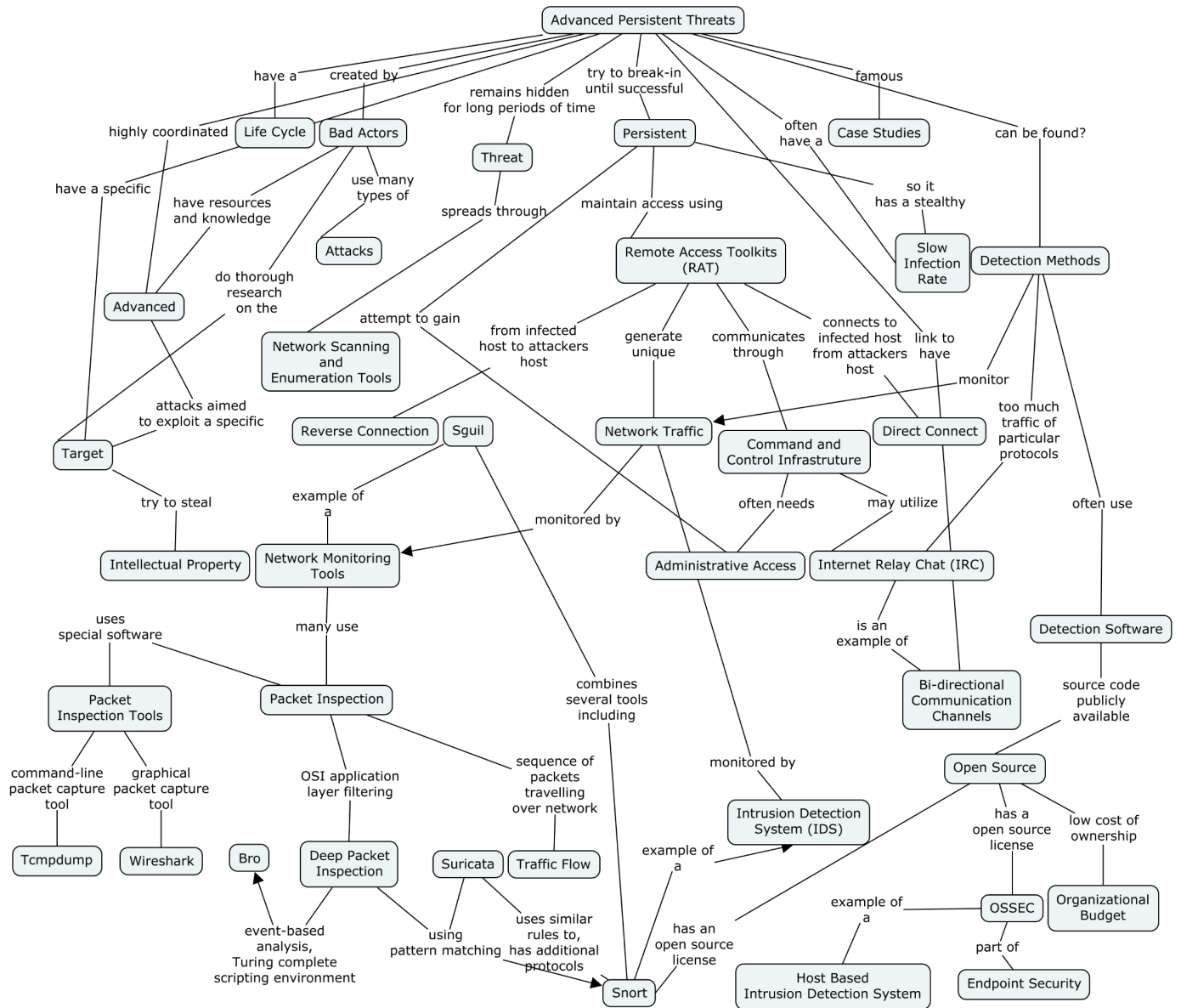
16

Figure 2.1: APST Mind Map

Figure 2.2: Mind Map Subgraph of Attacks
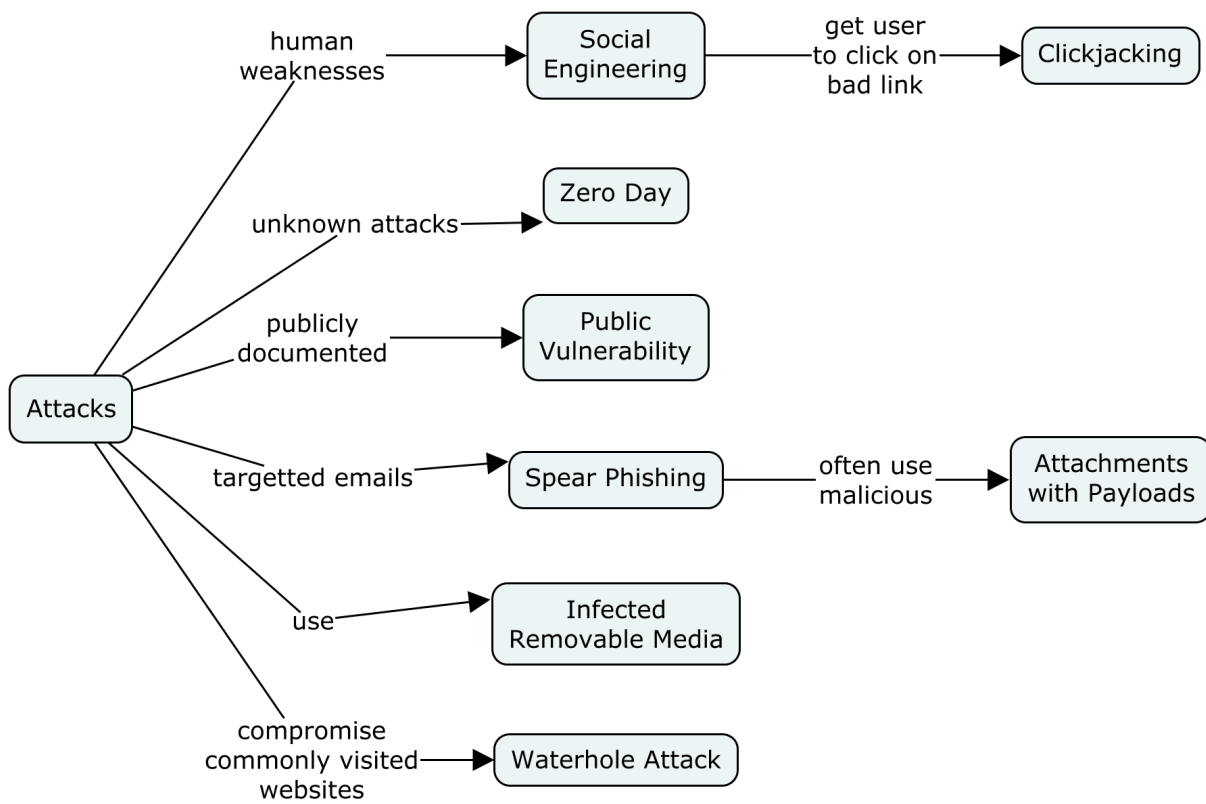
Figure 2.3: Mind Map Subgraph of Case Studies
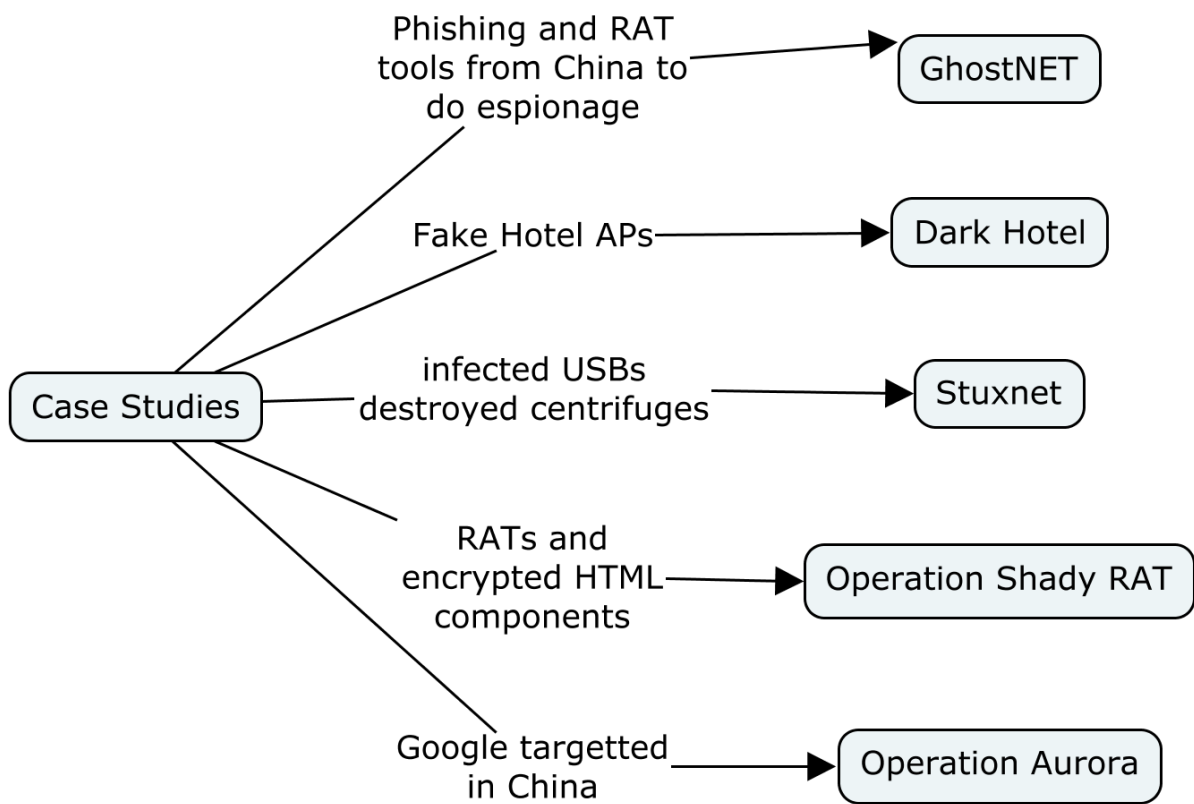
Figure 2.4: Mind Map Subgraph of Detection Methods
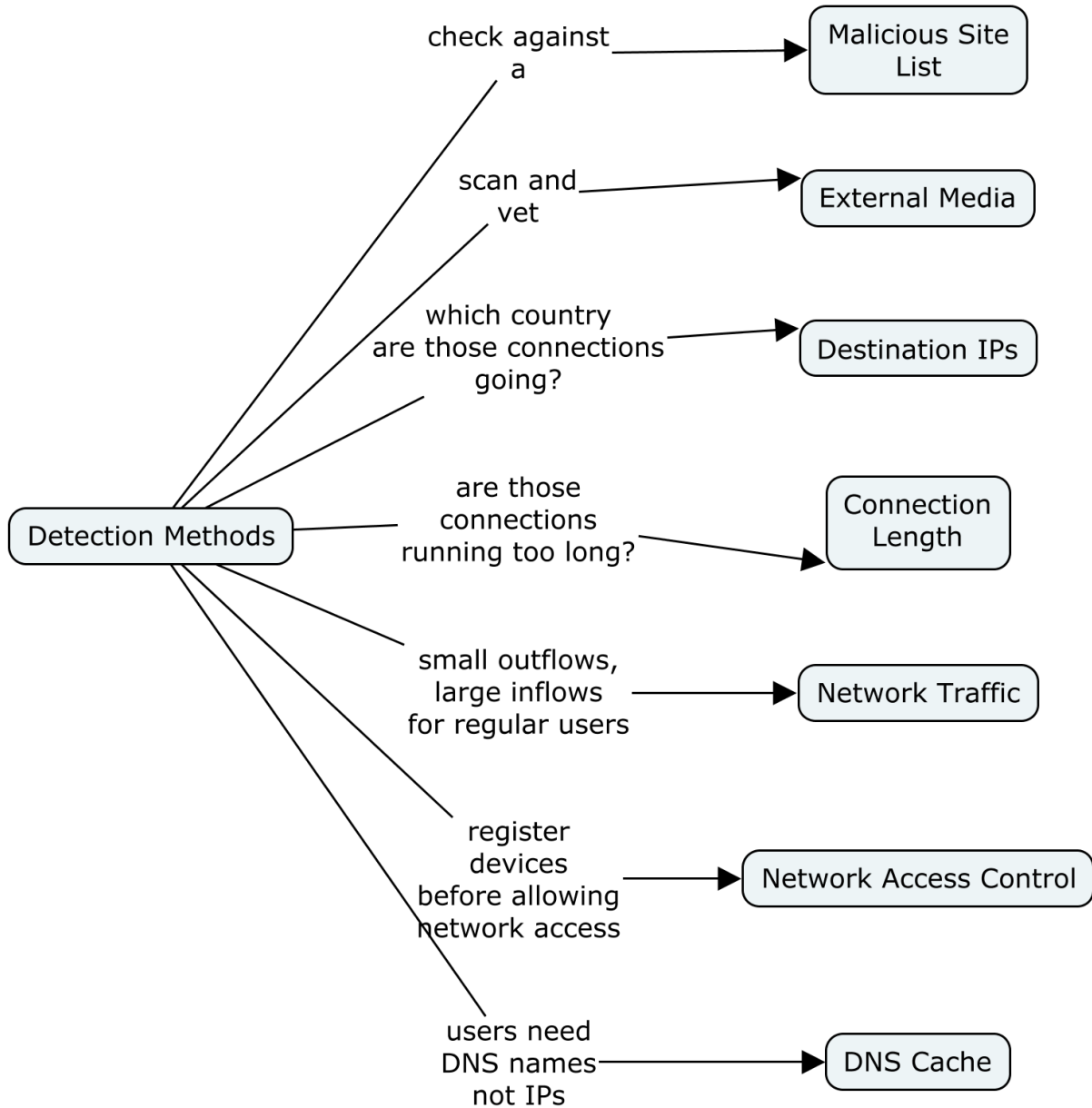
Figure 2.5: Mind Map Subgraph of Detection Software
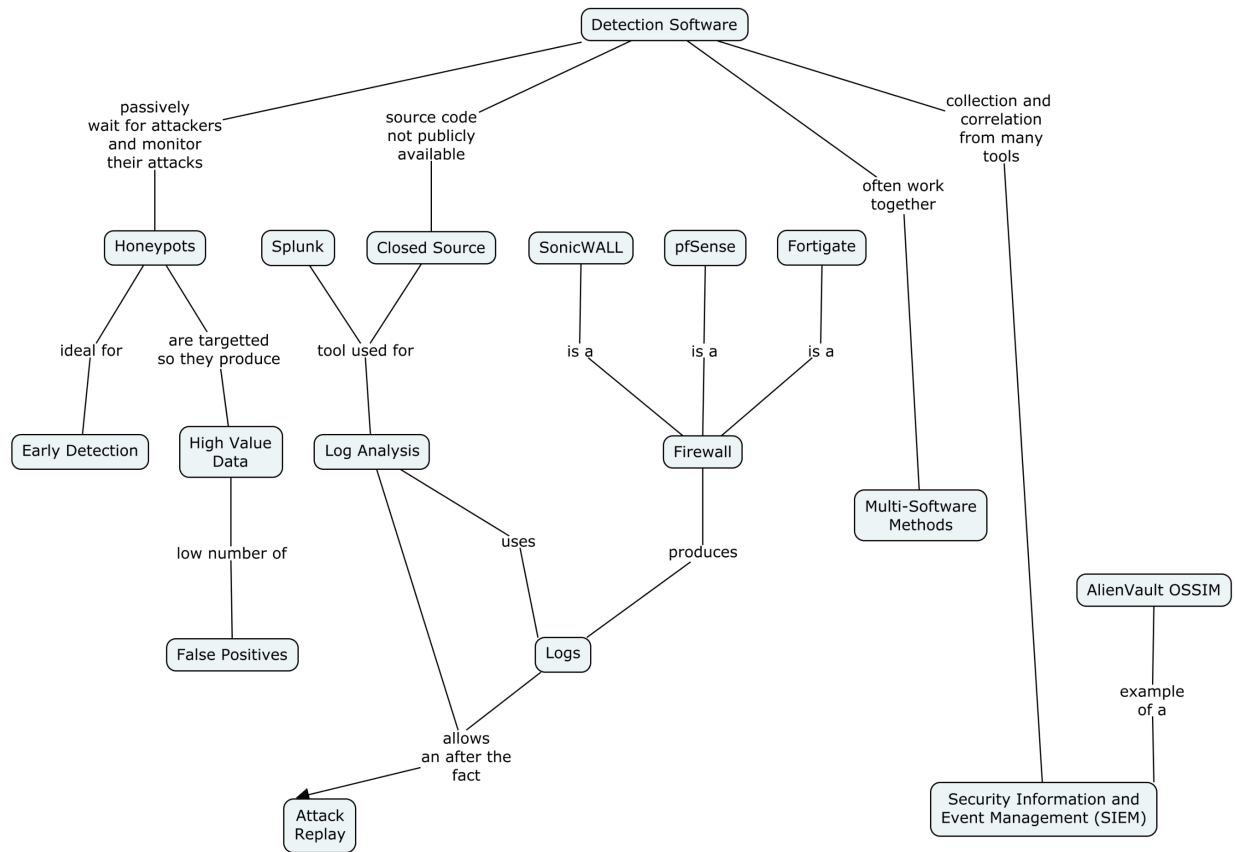
Figure 2.6: Mind Map Subgraph of APST Life Cycle
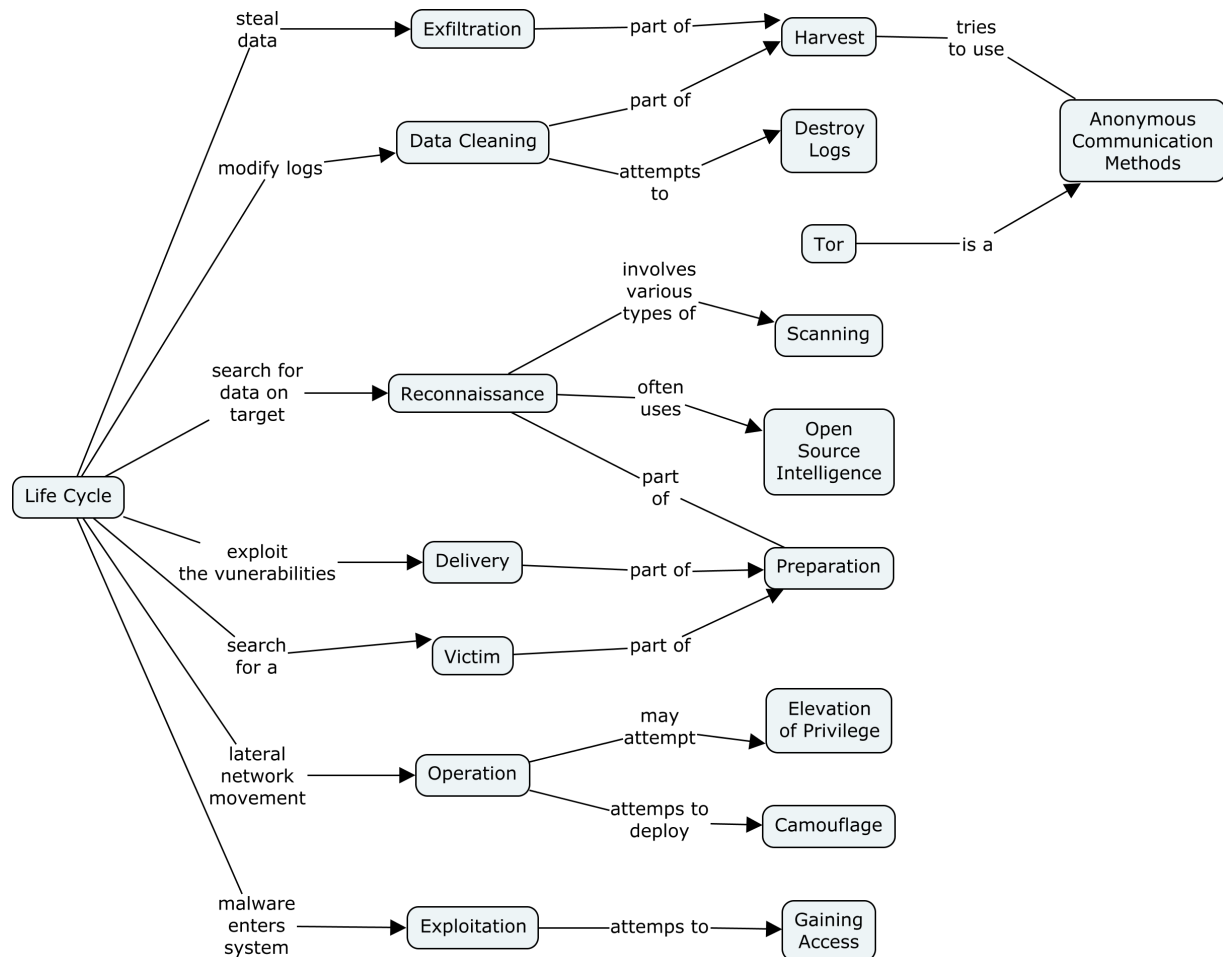
requirements imposed by certain regulations or security standards [43]." So many possibilities in detection methods [59] can produce a quizzical security analyst who is confused about which methods would be most effective on their particular network.

It is clear that current approaches for dealing with APST are inadequate and there is room for improvement within each phase of the attack life cycle. Many authors have examined selections of open source tools [62] which is helpful, but we feel that a method of auditing would be ideal. As noted in one paper, "Logging has becoming an essential element of any application, system, etc. It can help in auditing to track about what happened as a document that consists of records of all the events, what resources used with details with the timestamp and the source and destination addresses and more information about the users and their events, which can be very useful if that information used properly. [63]" By maintaining log files we will both have evidence available to detech APST and data available to track their spread through our network. Most tools do not offer an advanced logging functionality which we believe would be helpful for our investigation.

2.5 Potential Solutions to the Problems

There are many claimed solutions to the APST scourge, but many of them are difficult and costly to implement. We will attempt to secure our simulated enterprise network using free and open source tool. Therefore, while some proprietary security products perform well in this space we discount them from our analysis [60]. The main reason for this is due to funding constraints and to make the research more accessible to a wider audience.

In this section we will explore some solutions suggested by the literature review which might help to assist us. One of the most cited defense strategies is simply *user education about the dangers of cyberspace* [1]. However, we are going to focus on specific technologies we can implement within the enterprise in an attempt to enhance security.

Some initial suggestions are promising, and a first look presents some novel areas to consider. A PhD thesis we located gave voluminous suggestions without recommending any specific pieces

of software [64]. A high level analysis provides many examples of signature based, anomaly based, and specification based detection methods [65]. An Intel presentation gives us some future considerations about how Intel Chips may be protected against these threats [66]. We also encountered general advice that all security appliances and software should be kept up-to-date [67], furthermore we should limit physical access to all secure facilities. These are excellent starting suggestions, but do not provide the concrete recommendations we need for our purposes.

Force the DNS servers on the local network to make use of fast-flux DNS domain [68]. Also, by limiting access that malware has to Command and Control servers prevents it from updating itself. The [68] paper gives us a list of several countermeasures which are highly recommended, which include: patch management, network segregation, white listing common Internet traffic, controlling access to software and hardware within the environment.

In [59] additional suggestions are given at the end of the paper. They include controlling all external media, making sure the endpoint security is kept up-to-date, and implementing network access control. Some of the tools recommended in this paper for network monitoring are OSSEC, Snort, Sguil, and Splunk. However, some papers [69] only give general advice on how to monitor the network but do not provide us with a software tool recommendation for the endeavor. An additional paper expands on these tool recommendations and offers more granular suggestions [70].

Other papers [14] offer some suggestions about using the Microsoft Enhanced Mitigation Experience Toolkit to stop attacks, but this toolkit is currently end of life, so is not useful for us. This paper does suggest, "we propose to implement a detection approach that monitors if a process requests a handle to the LSASS process and performs suspicious function calls with the help of this handle." When we searched the paper though, no source code is offered for this task. The final recommendation for Windows log analysis is indeed good, and we think having a tool which will be able to look at all the logs in a reasonable period of time could be useful.

The search for APST[1] must take into account event anomaly detection and data loss preven-

Table 2.1: Attack techniques and countermeasures in each stage of an APST attack taken from [1, Table. 3]

Stages	Attack techniques/tools	Countermeasures
Reconnaissance and Weaponization	OSINT, Social engineering Preparing malware	Security awareness training, Patch management, Firewall
Delivery	Spear phishing, Watering hole attack	Content filtering software, NIDS, Anti-virus software
Initial Intrusion	Zero-day exploits, Remote code execution	Patch management, HIDS, Advanced malware detection
Command and Control	Exploiting legitimate services, RAT, Encryption	NIDS, SIEM, Event Anomaly detection
Lateral Movement	Privilege Escalation, Collecting data	Access control, HIDS, NIDS, Event Anomaly detection
Data Exfiltration	Compression, Encryption, Intermediary Staging	Data Loss Prevention

tion. This paper presents an excellent summary of the defense strategies that can be employed at each stage of the APST life cycle, we have reproduced it in Table 2.1. This discussion illustrates the ongoing observation that different countermeasures should be used to stop APST depending on the attack stage it is currently in.

As mentioned in the literature review, we became aware of a similar product, Open Source Security Information Management (OSSIM) [71], which has some of the features were are examining in our current project. OSSIM is an amalgamation of various open source tools which were brought together to give a network administrator an overview of their entire network. There is a subset of tools used by OSSIM which we have mirrored including Snort and OSSEC (part of Wazuh) [72]. While we do not consider this a competing project, we mention it for completeness and an example of a real-world implementation currently used by organizations.

2.5.1 APST Removal

Our literature review of the removal possibilities of APST had a huge focus on harm reduction and not harm prevention. As one project paper told us, "There is no right answer on how you

should react on APT attack detection. APT is not [a] one time occurrence, it is a war [73]." Further on it tells us, "...full re-imaging guarantee[s] total remediation if the image was made before an infection." Which presents the unfortunate answer that only restoring the computer to its pre-infection state by restoring from known good back-ups or total re-installation of the base OS and all software will stop the threat. However, even that is not always a solution as it points out because you may often want to watch a n APST move through your network to see what hosts are compromised before removing it.

Another resource was consulted on removal [74] and recommends the use of an incident response methodology. One part of this response will be to review logs as evidence of what has happened. The six-step process recommended to be followed is: preparation, identification, containment, eradication, recovery, and lessons learned. Eric Cole presents a clear definition of all the steps, but we will only briefly mention them here. Preparation involves getting your organization ready before the attack, which requires making sure your employees are properly trained in procedures, and you have developed solid links with with law enforcement and other relationships which can assist you. Identification is about who will report the attack, and reports may often arrive from a third party. Containment makes sure the threat does not spread and isolates infected systems or infrastructure in a way that will not alert the attackers. Eradication is the investigation into how your systems were compromised and what needs to be done to bring them back to a working state. Recovery wants to make sure that everything is restored to a pre-attack state with a critical focus on making sure that re-infection does not occur. This may include modifying security systems to make the organization more resistant to attacks in the future. Lessons learned looks back over the entire cycle and creates documented evidence of what went wrong, how it was handled, and what could be done better in the future.

CHAPTER 3

RESEARCH METHODOLOGY

3.1   Doing Literature and Technical Review

Our literature review used online databases to search for topics related to "Advanced Persistent Threats" and "APT". The five main databases queried were: ScienceDirect, IEEE Xplore Digital Library, ACM Digital Library, Google Scholar, and SpringerLink. We found the Google Scholar "related article" feature particularly helpful in find tangentially related articles which were targeted to our topic of interest.

3.2   Tool Selection

The selection of tools for our project was important for the successful detection of APST. We wanted to find tools which met several criteria which included: open source driven, active development, supportive user base, and consisted of well known and proven technologies. Moreover it was imperative to make sure that we selected tools which were supported by our literature review to ensure they had the backing of our peers when dealing with APST. Please note that the four stage lifecycle model [2] was our chosen model for APST attacks. We will try to link each tool to the stages it will be most effective for detection of APST.

3.3   Federated Solution

All the tools selected for our FSM will be independent and autonomous components which will be used together in order to detect APST. Each component will be chosen to provide detection capabilities across different parts of the APST life cycle. No single tool is able to provide this capability across the entire life cycle, so multiple tools must be used. In order for the output of

these detection tools to be queried in an effective and efficient manner, we use log aggregation in Elastic Stack to bring together all the data they provide and make it accessible in a centralized, user friendly manner. This will allow queries to be sent from one unified interface (Elasticsearch or Kibana) to explore our data without the need of querying each tool individually. Because our tools work together to achieve the detection of APST we say our solution is *federated*. Since, our solution brings the logs together in a central system we say our solution uses *log aggregation*, or simply *aggregation*. In short, this leads us to conclude our solution is a *federated aggregation*.

3.4   Developing the FSM Kibana Plugin

After all our sensors are configured and logs are being sent to the Elastic Stack it will be beneficial to have a way of viewing the logs in a centralized console. A Kibana plugin was proposed for this, as it will be able to extend the normal Kibana interfaces and application programmer interfaces (APIs) to access the logs stored in Elasticsearch. These log files are contained in various indexes presented in Table 5.4. The Kibana plugin will not be exhaustive, but will have an easy to use and extensible user interface which allows incorporating pre-defined DSL queries which can be clicked on to return data. A screenshot of the user interface of the plugin can be seen in Figure 3.1. JavaScript is the language used in the plugin implementation. Our source code was made available in a GitHub repository [75].

Our plugin presents multiple queries which the user can click on to return pertinent data. The data is present in ElaticUI tables which offer basic pagination. This plugin is currently implemented to run on Kibana 6.7.1, which is the version of Kibana we decided to centre our implementation around. (With the quick release cycle of Kibana it was not possible to constantly update our plug-in to function with the latest version of Kibana.)

A brainstorm of possible queries was as follows:

1. Boundary Firewall (Snort / pfSense)

   (a) Blocked incoming packets from known bad sites

28

Figure 3.1: FSM Kibana Plugin

kibana

Discover
Visualize
Dashboard
Timelion
F   Fsm Plugin
H   Hello World
Wazuh
Dev Tools
Management

fsm_plugin Query Dashboard

Please select a query to display the corresponding table:

Blocked incoming packets from known bad sites in the last day
Firewall Login Attempts in the last thirty minutes
HoneyTrap Non-Heartbeat Events in the last three days
Snort Messages for the last day
Windows Defender Detected Malware in the last five days

Blocked incoming packets from known bad sites

Timestamp ↓	Source IP	Source Port	Destination IP	Destination Port	Message
2019-07-16-23:47:03	218.92.0.211	52455	192.168.3.151	22	53,,,1000000301,re0,match,block,in,4,0x0,,53,34707,0,DF,6,tcp,60,218.92.0.211,192.168.3.151,52455,22,0,S,2800093144,,29200,,mss;sackOK;TS;nop;wscale
2019-07-16-23:47:01	218.92.0.211	18307	192.168.3.151	22	53,,,1000000301,re0,match,block,in,4,0x0,,53,16690,0,DF,6,tcp,60,218.92.0.211,192.168.3.151,18307,22,0,S,598293080,,29200,,mss;sackOK;TS;nop;wscale

    (b) Amount of data coming from a particular host

    (c) Login attempts to the firewall

2. HoneyTrap

    (a) Lateral movement in the network (e.g. Network scans)

    (b) Login attempts on high value ports (SSH, Telnet, FTP)

    (c) Brute force attempts

    (d) Detect the source IP addresses which are trying to connect to the HoneyTrap as an indicator that the source host has been compromised

3. Sweet Security (Zeek)

    (a) High value open ports on the host which were not there before

    (b) DNS queries - Known bad DNS - Command and Control Servers

(c) Geolocation (somewhat) to known bad IP lists (uncommon IP ranges - Russia, China)

(d) Multiple protocols connects to a host IP or out of a host IP (DNS, FTP, etc.)

(e) Regular expression matching for malformed packets

(f) New unknown hosts entering the network

(g) HTTPS / HTTP requests to "strange" websites

4. Wazuh

(a) Event Log (System, Application, Security)

    i. Login attempts (failed or successful)

    ii. Login types (network share, RDP, console, etc)

    iii. Crashing applications / blue screens / bug logs - commonly compromised applications (E.g. Internet explorer crashing.)

    iv. Application installs

    v. Windows features that are enabled

    vi. Activation or creation of new login accounts

(b) Sysmon

    i. Processes that do not commonly run

    ii. Executable name is in the "correct folder"

    iii. Strange command line arguments (cmd.exe)

    iv. Uncommon running applications

    v. Common hacking tool names, based on the MD5 hash

vi. Running any type of command line interpreter (wscript, cmd, powershell, etc.)

(c) Windows Defender

i. Malware detected by Windows Defender

ii. Warnings from Windows Defender

(d) Other Stuff

i. Modified Windows registry values

ii. Monitor file shares or folders for access attempts

iii. Changes to host file

iv. Accesses to password hash files

We selected a few of these queries which can be found in the Appendix section under Appendix B.1 through to Appendix B.5.

3.5 Penetration Lab Network Setup

In order to test our tool we will setup a penetration testing lab as shown in Figure 4.1. We focus on the setup of a virtualized Windows Active Directory domain network as that will mimic the most common configuration of a corporate network.

We have drawn from many sources to get a sense of the best way the lab should be built and configured, with a distinct emphasis on easy setup, configuration, and restoration of machines to a pristine state. One source of information would be chapters 4 and 5 in [76] which talks about setting up a penetration testing lab. Another source identified on setup of the lab environment for malware analysis is chapter 2 in [77]. Both these references give some guidance for the setup of a lab environment. Our specific environment uses virtual machines created in VMware Workstation. (VMware was used because we are able to obtain free student licensing for it.)

3.6   Security Software Configuration

The security software, or "sensors" which were used to collect data required for our federate will run on various hosts and devices in our penetration testing lab. You can refer to Figure 5.2 for the proposed location of various pieces of security software in our network. All pieces of security software will be configured to send their logs to *fmsrv* Linux server which will be running the Elastic Stack. This server will also house the Wazuh Server and the Sweet Security Server. Our firewall will be using pfSense software and then have Snort installed as an add-in. All our Windows hosts will run the Wazuh client, Sysmon, and Windows Defender (if available). The honeypot will run on a separate virtualized Linux server. The network monitoring system will reside on a Raspberry Pi connected to our LAN switch which will allow it to sniff any traffic through the local network.

3.7   Testing the Federated System

Most testing will take place on *win81-client.corp.local* the Windows 8.1 VMware virtual machine. The use of virtual machines will allow us to restore test machines to a pristine state between each test we perform. We will also have to make sure that all sensors are active and working before we begin our testing.

In order to test the federated system, we will first conduct some baseline tests to make sure that our components are returning. Two available test suites we will use to mimic APST are APTSimulator [78] and FlightSim [79]. APTSimulator will simulate APST on our Windows hosts and FlightSim will simulate malicious traffic on our enterprise network. These tools will form the first level of testing we will do to evaluate our FSM solution. They will also give us information about what we should look for in the various log files to show malicious activity is happening within our network.

For each sensor tested, Appendix A will contain example JSON log files we have collected during various tests.

CHAPTER 4

THE NETWORK LAB FOR PENETRATION TESTING

4.1   Penetration Testing Lab Network Setup

In order to test our tool we setup a penetration testing lab as shown in Figure 4.1. This network has been setup to simulate a small-scale corporate (enterprise) LAN. The setup include an Active Directory domain (corp.local) containing two domain controllers and one client operating system joined to the domain.
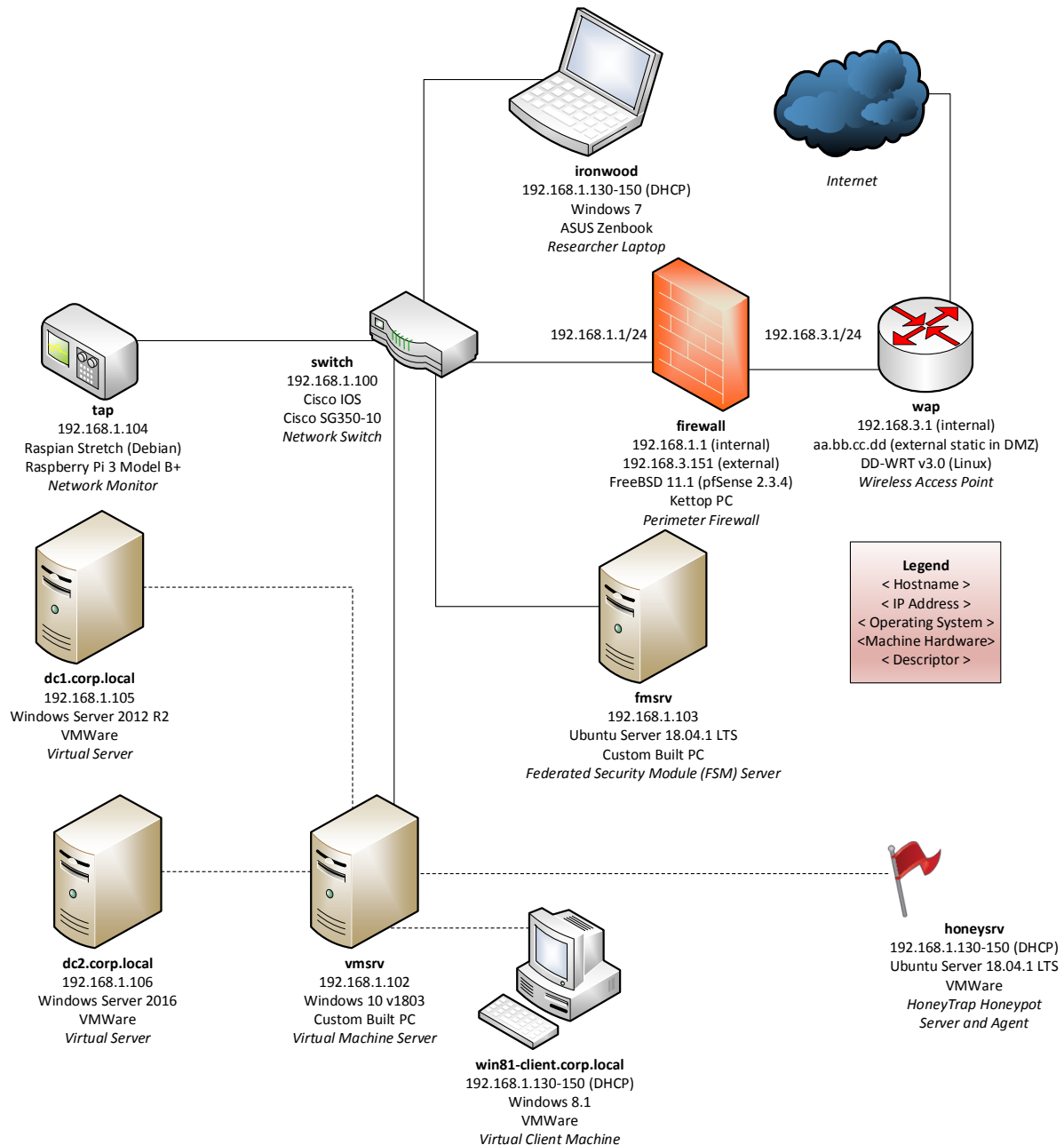
Figure 4.1 incorporates lots of information and should be studied carefully. Each device and host in our network setup is shown listing the hostname, IP address, operating system, machine hardware (e.g. device name), and a brief English description of what the device is on our network. Solid lines between devices represent physical Ethernet connections between devices, while dotted lines represent virtual network connections between devices.

We have drawn from many sources to get a sense of the best way the lab should be built and configured, with a distinct emphasis on easy setup, configuration, and restoration of machines to a pristine state. One source of information would be chapters 4 and 5 in [76] which talks about setting up a penetration testing lab. Another source identified on setup of the lab environment for malware analysis is chapter 2 in [77]. Both these references give some guidance for the setup of a lab environment. Our specific environment uses virtual machines created in VMware Workstation.

Within our setup there are many different components:

- Desktop (Virtual Client Machine) - one virtual workstation running Window 8.1 end user operating systems.

- FSM Server - the main analysis server where all data will be sent for processing as part of the FSM. This server is running Ubuntu Server 18.04.1 LTS on custom built

Figure 4.1: Penetration Testing Lab Configuration

ironwood
192.168.1.130-150 (DHCP)
Windows 7
ASUS Zenbook
Researcher Laptop

Internet

switch
192.168.1.100
Cisco IOS
Cisco SG350-10
Network Switch

192.168.1.1/24   192.168.3.1/24

tap
192.168.1.104
Raspian Stretch (Debian)
Raspberry Pi 3 Model B+
Network Monitor

firewall
192.168.1.1 (internal)
192.168.3.151 (external)
FreeBSD 11.1 (pfSense 2.3.4)
Kettop PC
Perimeter Firewall

wap
192.168.3.1 (internal)
aa.bb.cc.dd (external static in DMZ)
DD-WRT v3.0 (Linux)
Wireless Access Point

Legend
< Hostname >
< IP Address >
< Operating System >
<Machine Hardware>
< Descriptor >

dc1.corp.local
192.168.1.105
Windows Server 2012 R2
VMWare
Virtual Server

fmsrv
192.168.1.103
Ubuntu Server 18.04.1 LTS
Custom Built PC
Federated Security Module (FSM) Server

dc2.corp.local
192.168.1.106
Windows Server 2016
VMWare
Virtual Server

vmsrv
192.168.1.102
Windows 10 v1803
Custom Built PC
Virtual Machine Server

honeysrv
192.168.1.130-150 (DHCP)
Ubuntu Server 18.04.1 LTS
VMWare
*HoneyTrap Honeypot
Server and Agent*

win81-client.corp.local
192.168.1.130-150 (DHCP)
Windows 8.1
VMWare
Virtual Client Machine

hardware.

- Honeypot - a fake high value host running on the network. The honeypot is implementing using HoneyTrap running in an Ubuntu Server virtual machine.

- Laptop (Researcher Laptop) - one mobile workstations running Windows 7 end user operating system.

- Network Monitor - a Raspberry Pi 3+ system which is setup to use port mirroring on the switch.

- Perimeter Firewall - a pfSense firewall appliance which will separate the internal network from the external network (Internet).

- Router - The Wireless Access Point (WAP) functions as a router for our network and connects the network to the Internet.

- Switch - a Cisco switch which provides network connectivity for all the components in our internal network

- Servers - two Windows servers running in VMware Workstation on a single host computer.

In order to manage all these components remote access to all devices was setup either by web interface, SSH, RDP, or VNC, then a tool known as Remotix for Windows was used to remote into all systems for ease of access. A laptop wired on the internal network was used to connect with all services.

Network setup within the lab environment is carefully crafted to make sure all components function smoothly together. Most components are assigned static IP addresses, include all servers and devices. These local network addresses are drawn from the 192.168.1.0/24 class C subnet. Static addresses are assigned by pairing the MAC address of each device with an assigned IP address. There is no NAT used within the internal network, and all VMware machines are directly

connected to the network using a bridged network configuration. This means there is no NAT used for the VMware hosts and their IP addresses directly map within the LAN. Such a configuration was ideal because it allows the hosts to more naturally function as part of the network while simplifying the setup and flattening the hierarchy of the network. Our network offers a range of DHCP address in the range 192.168.1.130-192.168.1.150. These addresses are given out to our *win81-client.corp.local*, *honeysrv*, and the researcher laptop on the network. Due to the low number of DHCP address requests, the addresses assigned to these devices will remain long term. The static reservation of addresses and DHCP is a service handled by our *firewall*. Our lab network is exposed through the DMZ on *wap*. This means our lab environment does have exposure to the Internet, but it can also function independently if necessary.

The operating systems used for our research are far reaching and contain Windows, Linux, FreeBSD, and Cisco IOS. All our Windows operating systems were obtained from Microsoft at no cost using student licensing. It would not have been cost effective to perform this research without the minimal cost of these operating systems.

CHAPTER 5

SELECTED SOFTWARE TOOLS INSTALLATION AND CONFIGURATION

5.1 Tool Selection

The selection of tools for our project was important for the successful detection of APST. We wanted to find tools which met several criteria which included: open source driven, active development, supportive user base, and consisted of well known and proven technologies. Moreover it was imperative to make sure that we selected tools which were supported by our literature review to ensure they had the backing of our peers when dealing with APST. Please note that the four stage lifecycle model [2] was our chosen model for APST attacks. We will try to link each tool to the stages it will be most effective for detection of APST.

5.1.1 ELK Stack / Elastic Stack

The amalgamation of software tools was at the heart of our project. It was imperative to have a method of getting the tools to work together to solve the problems faced when hunting for threats within our enterprise network. In order for the federation to be successful, we needed a straightforward way for all our components to work together without having additional overhead or the need for extensive programming or modifications. Originally, we considered the use of Splunk [80], as the marketing material told us, "Splunk is uniquely suited to collect, index, correlate and analyze all data, and to monitor patterns of activity over the very long periods of time required to see a potential attack." However, Splunk often has a high cost associated with it [81] (unless using the limited free edition), so the search for alternatives lead us to the Elasticsearch, Logstash, and Kibana stack commonly referred to as ELK. The combination of these technologies offered the open source alternative to Splunk which satisfied our need for a system which has the capabil-

ities, but not the cost. A logging system is important [63] as it offers the ability to figure out what has happened on a system, which provides transparency and a record of past events which can be examined and audited should something go wrong. The ELK stack was a perfect example of the technology we were looking for to provide the heart of our FSM.

ELK has three main components:

- Elasticsearch [82] is a "real-time distributed search and analytics engine."

- Logstash [83] "provides an integrated framework for for log collection, centralization, parsing, storage, and search."

- Kibana offers a framework to create graphs, charts, and visualizations using the data collected in Elasticsearch.

The three tools within ELK provide a complete framework for gathering logs from many different sources, storing and parsing them so they can be searched in an efficient manner, and finally a way of creating visualizations and dashboards to view the logs in a user accessible and acceptable manner.

All selected tools must be able to log to the ELK stack for further processing and completeness. The method of logging may operate at slightly different layers within the stack using either Logstash or Elasticsearch.

5.1.2 Perimeter Firewall

Our literature review suggested that it was important to have a perimeter firewall when defending against attacks [7]. Finding a firewall which is both open source and has a solid reputation was difficult, but eventually we decided on the use of pfSense. pfSense is an open source firewall based on FreeBSD which claims to be the leading open source firewall [84]. The firewall is able to run on a standalone computer system which suits our purposes, as we acquired a computer with two networking ports for a reasonable price. Furthermore, pfSense comes with an extensive list of

add-ons which extend its existing functionality which is important when adding an intrusion detection system (IDS) to our system. A comparison of various open source firewalls concluded that pfSense [85] was, "...found to be a financially effective solution because of its easy upgradability, simple web configurator, and its wide range of extension and features." The perimeter firewall will be effective in protecting against attacks in the preparation and access stages. The attacker will need to penetrate the firewall protections before gaining access to the internal corporate network.

A summary of our findings for this tool can be found in Table 5.1.

5.1.3   Snort versus Suricata

The two competing technologies which can be used within pfSense as an Intrusion Detection System (IDS) were Snort and Suricata [50]. Snort is the mature technology which has been around for many years, while Suricata is a fairly new multi-thread technology. Due to the limitations of our environment, a single threaded application is more appropriate. However, it does appear that Suricata would be able to handle a much larger workload [86]. Our decision focused on which technology was more proven in the industry, although several sources felt that Suricata outperformed Snort in terms of dropped packets [87] and better performed on Linux [88]. Since we were using a FreeBSD firewall with low throughput it made more sense to use Snort even with its known limitations.

Snort will be important in defending in the preparation, access, and harvest stages. In the preparation stage, Snort will assist in blocking websites and various types of port scans. In the access stage Snort will be important as a NIDS system, to alert about possible attacks. Finally, in the harvest stage Snort can be used to identify personally identifiable information leaving the network [89].

A summary of our findings for this tool can be found in Table 5.1.

5.1.4 Network Monitoring

Our search for a network monitoring system was originally focused on Zeek [50] a packet capturing technology created in 1995 for passively monitoring a network line. This seemed to be the perfect, proven technology for monitoring our network [90]. This technology has been in development for over twenty years and is trusted within industry. Of course, it is also open source which met another one of our selection criteria. Some of the other reasons for choosing Zeek include [91]: grounding in research, the Zeek community, and powerful design.

Another difficulty within our setup was how to implement Zeek in our network. We attempted to find a low cost way of deployment, and found an open source project known as Sweet Security [92]. This project provided all the infrastructure to run Zeek on a Raspberry Pi and log what was detected to the ELK stack. This fit well within our overall project plan so we immediately decided to use this as part of our solution. We will discuss in later sections some of the difficulties this tool decision created during setup.

Sweet Security (Zeek) will be useful in the access, resident, and harvest stages on the internal network. Zeek has a large number of prebuilt protocol filters which allow traffic to be captured and interpreted in real-time. This deep packet inspection will allow us to detect suspicious activities on the network. This usefulness will continue into the resident and harvest stages because we will be able to filter for network traffic that suggests lateral movement and data exfiltration from the network.

A summary of our findings for this tool can be found in Table 5.2.

5.1.5 Host Monitoring

Open Source HIDS Security (OSSEC) has been around for a while and is used as a host-based intrusion detection system (HIDS) [93]. What this means is that it monitors the individual computers on the network through agents running on them, instead of monitoring the network traffic itself. This project was chosen because [62] it is open source, has been in development

for a long period of time, has the capability of monitoring many different host operating systems, and is still under active development. Additionally, we found there is an parallel project which incorporates OSSEC known as Wazuh. Wazuh takes OSSEC and combines it with the ELK stack [94] which is exactly our goal for this project. Wazuh also has an extensive developer community which is constantly updating their tool to work with the latest versions of the ELK stack.

Wazuh will be most beneficial in the access and resident stages as it monitors the hosts to see whether an intrusion might have taken place. Wazuh has many capabilities, but with the default ruleset certain indicators of compromise can be found in altered registry values and common files. Additional rulesets may need to be expanded for more targeted threats.

Since our enterprise environment focuses so closely on Windows operating systems, we will include Sysmon [95] to add some additional Windows process, network connection, and file operations monitoring. Sysmon will dump the additional information in the Windows logs and then we will use Wazuh to export the log files from the host machines [96]. Sysmon requires a configuration file, so we use one available from GitHub [97]. In order to bring information from Sysmon over to ELK we used Winlogbeat [98], which allows the transport of the Application, System, Security, and Sysmon Event Logs to Elasticsearch.

A summary of our findings for this tool can be found in Table 5.2.

5.1.6   Honeypot

We spent a large amount of time trying to find a honeypot which was under active development. Most projects are only active for a few years and then are abandoned as new techniques become available. Our main source [55] had an excellent table on pages 17-18 which summarized various historical honeypot projects, their protocols, and areas of focus. We needed to located a honeypot which would we usable within an internal network to simulate some of the services which might be available therein, we also had a strong preference for log files to be made available in the ELK stack we had setup. Another reference we located was [56] which had a section discussing honeypots as they apply to APST. Only serendipity lead us to the current HoneyTrap project,

but once we located it, it fulfilled all our requirements. It is under active development, has several output channels which includes the ELK stack, is able to pretend to be various services, has a fairly straightforward setup and configuration, and is open source. HoneyTrap met all our requirements and was a natural fit for our solution.

The honeypot will be useful in the resident stage in detecting the lateral movement of the APST as it attempts to explore and map the network. By pretending to be a high value target on the internal network we can monitor attempts to access specific ports like SSH, FTP, etc. These logs will provide an indication that certain hosts within the network have been compromised.

A summary of our findings for this tool can be found in Table 5.1.

5.1.7 Anti-virus Software

Open source anti-virus software is available through the ClamAV project [61]. ClamAV is the only open source anti-virus product available at this time [99]. While ClamAV is useful in detection of viruses and various forms of malware, modern Windows operating systems come installed with Windows Defender. One paper on spyware concluded, "Windows Defender is competitive enough to compete with the existing Anti-Spyware products [100]." We decided to leave Windows Defender as the default anti-virus on our Windows hosts as it was free to download and install on these operating systems. Furthermore, it regularly updates and has a better detection rate than ClamAV due to commercial backing by Microsoft. Windows Defender will form part of our APST protection suite, but will not be considered as one of the open source tools making up our ELK stack.

While an effective open source anti-virus was not forthcoming, we felt it was important to include an anti-virus within the overall toolkit. Anti-virus software is most important during the resident phase, as it can sometimes detect suspicious files or exploits that are being brought onto compromised hosts by the attackers.

5.1.8 Tool Summary

The tools selected for this project are summarized in Table 5.1 and 5.2 along with some of the criteria for why they were selected. All tools contain open source software which we believe will be useful in the detection of APST and meets our commitment to build our solution using open source tools as shown in Table 5.3. Furthermore, all tools can be setup and configured to use the ELK stack.

Figure 5.1 summarizes how the various tools will fit into detection of APST throughout the threat lifecycle. The four stage lifecycle model [2] was our chosen model for APST attacks.

5.2 Federated Solution

"Federation [is] the set of components (many of them being COTS tools) which together implement a software application [101]." In our system, we have bound together multiple autonomous software applications in order to implement our complete FSM. Each component has been carefully chosen to make sure all phases in Figure 5.1 will have multiple tools of detection. Participation in our federation, requires each program to give up some limited autonomy for the overall system to function properly. All the components are setup to feed their detection results into Elastic Stack which will aggregate these logs together to make them accessible and searchable. The FSM Kibana plugin will then take this data and perform queries for detection of APST on the data. Multiple indices search will allow us to query various sources at one time to produce results across the gathered data helping to bind the federation together. The resulting software system, creates a loose federation of components which work together towards the detection of APST.

5.3 Security Software Configuration

The configuration and the setup of the software tools which were selected as part of this project involved a large amount of time and effort. We have detailed a high level overview of the steps

Table 5.1: Summary of Criteria for Selection of Each Tool

Tool	Criteria	Result
HoneyTrap	Log to ELK?	Yes
	Open Source / Free Software?	Free Software
	Cost?	Free
	Proven Technology?	Still in development
	Active Developer Community?	Yes
	Special Considerations?	Configuration file defines available honeypots and ports
	Fit for purpose?	Yes
	Lifecycle Phases?	Resident
pfSense	Log to ELK?	Customized to send logs to ELK
	Open Source / Free Software?	Open Source Software
	Cost?	Free, paid devices available
	Proven Technology?	Yes
	Active Developer Community?	Yes
	Special Considerations?	Requires dedicated hardware appliance with dual NICs
	Fit for purpose?	Yes
	Lifecycle Phases?	Preparation, Access
Snort	Log to ELK?	pfSense logs sent to ELK (includes Snort)
	Open Source / Free Software?	Free Software
	Cost?	Free, Subscription for latest threats
	Proven Technology?	Yes
	Active Developer Community?	Yes
	Special Considerations?	Installed on pfSense
	Fit for purpose?	Makes more sense than Suricata for our implementation
	Lifecycle Phases?	Preparation, Access, Resident

Table 5.2: Summary of Criteria for Selection of Each Tool

Tool	Criteria	Result
Sysmon	Log to ELK?	Limited, through Wazuh alerts and Full, through Winlogbeat
	Open Source / Free Software?	No, Free Proprietary
	Cost?	Free
	Proven Technology?	Yes
	Active Developer Community?	Yes
	Special Considerations?	Needs a ruleset
	Fit for purpose?	Used for indepth investigation on Windows hosts
	Lifecycle Phases?	Resident
Sweet Security (Zeek and Elastic Stack)	Log to ELK?	Yes
	Open Source / Free Software?	Open Source Software
	Cost?	Free
	Proven Technology?	Zeek is proven, Sweet Security is a proof of concept
	Active Developer Community?	No
	Special Considerations?	Requires Raspberry Pi
	Fit for purpose?	Yes, but does not handle encrypted traffic
	Lifecycle Phases?	Access, Resident, Harvest
Wazuh (OSSEC and Elastic Stack)	Log to ELK?	Yes
	Open Source / Free Software?	Open Source
	Cost?	Free
	Proven Technology?	Yes
	Active Developer Community?	Yes
	Special Considerations?	Installed on each Windows host system
	Fit for purpose?	Yes
	Lifecycle Phases?	Resident, Harvest

Table 5.3: Tool License and Website Summary

Tool	License	Website
ELK Stack (Elastic Stack)	Apache License Version 2.0	https://www.elastic.co/
OSSEC	GNU GPL v2	https://www.ossec.net/
HoneyTrap	GNU Affero General Public License version 3	https://github.com/honeytrap/honeytrap
pfSense	Apache License 2.0	https://www.pfsense.org/
Snort	GNU GPL 2 and Commercial	https://snort.org/
Sysmon	Freeware	https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon
Sweet Security (Zeek and Elastic Stack)	Apache License 2.0	https://github.com/TravisFSmith/Sweet Security
Wazuh (OSSEC and Elastic Stack)	GNU GPL 2	https://wazuh.com/
Windows Defender	Commerical	https://www.microsoft.com/en-ca/windows/comprehensive-security
Zeek	BSD license	https://www.bro.org/

Figure 5.1: APST Lifecycle Model With Selected Tools for Identifying Threats derived from [2, Fig. 1]
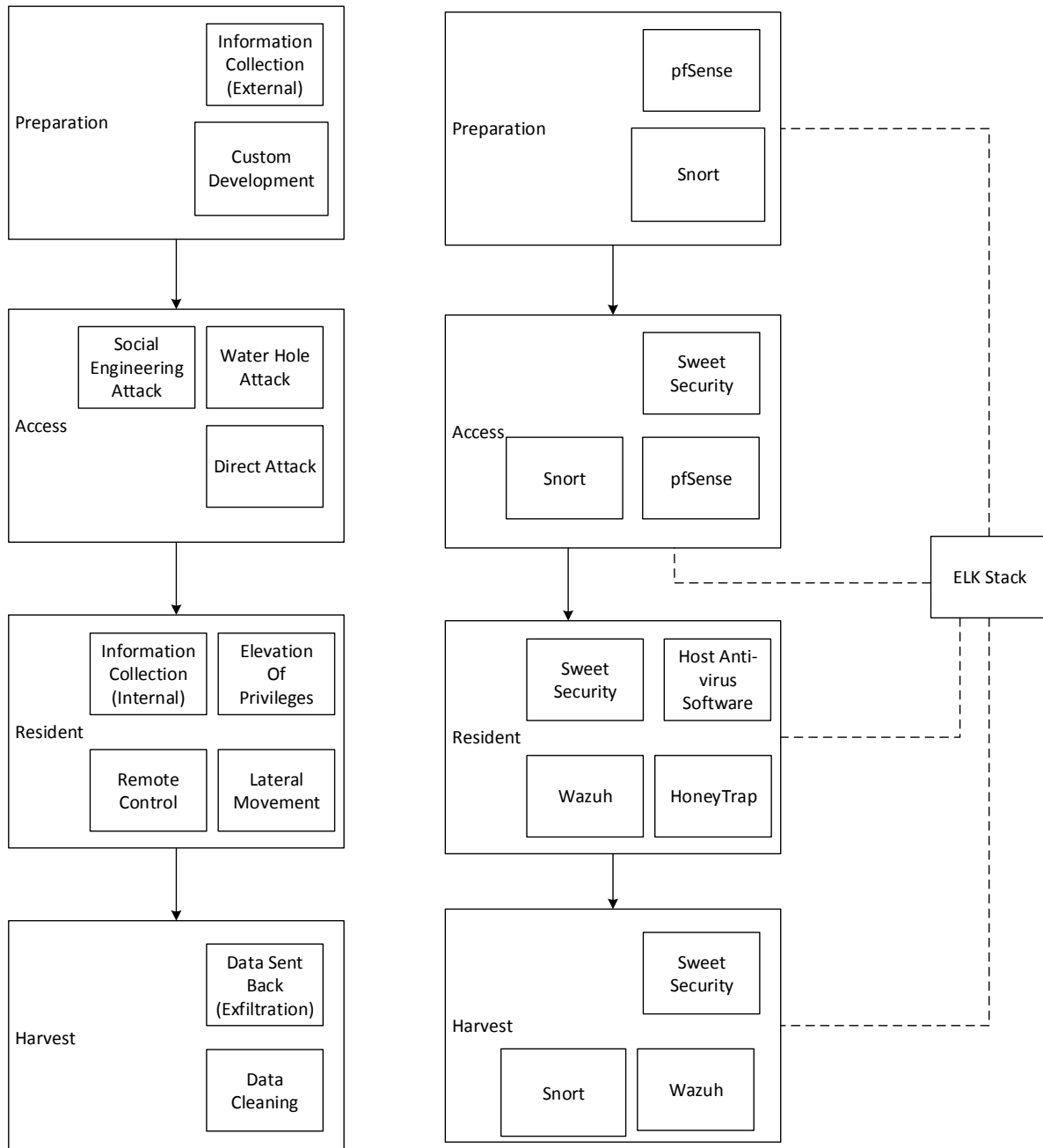
required for the setup of the lab and respective software environment. You can refer to Figure 5.2 for the installation location of various pieces of security software. All references to setup documentation in individual projects will be linked to, but for brevity only important steps are highlighted.

It is important to make sure that all systems within the research environment are synchronized to the correct time using a local network time protocol (NTP) server. This will make sure that all log files have correct date and time stamps.

1. Install pfSense on *firewall* [102].

2. Install Snort on the *firewall* [103].

3. Install Sweet Security on *fmsrv*.

    (a) Use the latest Ubuntu Server ISO to install the operating system [104].

    (b) Follow the normal setup steps to install and run Linux.

    (c) Make sure to configure the time server [105].

    (d) Download Sweet Security and install it using the Web Server Only configuration [106].

4. Do the upgrade and install of the latest version of ELK stack that Wazuh will support. This install will use the single-host architecture [107].

    (a) Install Wazuh server with DEB packages, but do not install FileBeats [108].

    (b) Install Elastic Stack with Debian packages [109].

5. Upgrade Sweet Security information by re-indexing to the latest version of the ELK stack [110].

6. Install the latest Wazuh Kibana plugin.

7. Restore the default logstash template. (Sweet Security messes with this a bit, and when we were trying to get the logs from the pfSense sent to the ELK stack this was causing problems.)

8. Perform the Sweet Security Client install on the Raspberry Pi 3 B+ and get it to report back to the ELK server.

   (a) Download NOOBS network install to the Pi SD card [111].

   (b) Install latest Raspian OS on the Pi.

   (c) Download Sweet Security from GitHub.

   (d) Do additional install steps for Debian 9 [112].

   (e) Patch the required files so the software can compile successfully, modify line 24, 26, 38, 40, 45, 47 in file Sweet Security/install/-packages.py and change libssl-dev to libssl1.0-dev [113].

   (f) Install Sweet Security on the Raspberry Pi and give it the settings to send logs to the *fmsrv* ELK stack.

9. Install Wazuh agents on all Windows virtual machines [114]. You will need to register each agent [115].

   (a) The Wazuh agent configuration was mainly left with the default. However, the capabilities of the agent are quite extensive and evolving [116]. Default settings are available for Windows agent clients [117]. The main configuration file *ossec.conf* is modified on each host to add additional capabilities required for our research, an example modified file is in [118].

   (b) Install Sysmon on all Windows hosts [95].

49

10. Setup pfSense to log to the ELK stack [119], [120].

11. Setup Snort to log directly to the pfSense system log, then both the Snort and pf-Sense logs will come over at the same time.

12. Create a new Linux virtual machine and setup the HoneyTrap Server and Agent [121].

13. We have provided a configuration file and a script which can assist with HoneyTrap setup and execution [122].

14. Setup port mirroring on the Cisco switch to send information to *tap* [123].

15. Fix the ELK stack so the cluster will report green instead of yellow. This is because we are using a simplified setup, so we need to reduce our number of replicas to zero [124].

16. Install Winlogbeat [98] on the Windows hosts to allow the transport of the Application, System, Security, and Sysmon Event Logs to the Elastic Stack.

17. Most of the tools are being rapidly updated, so periodic updates to all components is ideal to make sure all new features are available [125].

Now you should have a working installation of the ELK stack with all logs going to it! Table 5.4 gives a summary of the main indexes in Elasticsearch and which tools they collect logs files from. When you see a * at the end of log file name that means that it is actually a series of log files denoted with a timestamp of -YYYY.MM.DD, for example *wazuh-alerts-3.x-2018.08.19*.

5.4 Difficulties and Future Recommendations on Setup

Many concerns came to light during the setup, here are some of the high level difficulties encountered:

Table 5.4: Log Files Gathered by the ELK Stack

Elasticsearch Index	Source Tool	Type of Logs
honeytrap*	HoneyTrap	Status Messages, Honeypot Connection Attempts
logstash-*	Zeek	Zeek Network Traffic Monitoring
pfsense-*	pfSense Firewall, Snort	Firewall Status Messages, Snort Alerts
sweet_security	Sweet Security	Detected Device Information, Port Scans
sweet_security_alerts	Sweet Security	New(Unique) Sweet Security Log Events
tardis	Sweet Security	Historical Hosts, IP Addresses, and Websites
wazuh-alerts-3.x-*	Wazuh	Log Events Above the Alert Threshold
wazuh-monitoring-3.x-*	Wazuh	All Wazuh Monitoring Logs
winlogbeats-*	Windows Event Logs	Specific Windows Event Logs (Application, System, Security, Sysmon)

Figure 5.2: Security Software Setup Within the Network

ironwood
Researcher Laptop

Internet

switch
Network Switch

tap
SweetSecurity Client (Bro)
Network Monitor

firewall
pfSense
SNORT
Perimeter Firewall

wap
Wireless Access Point

Legend
< Hostname >
< Software 1 >
....
< Software N >
< Descriptor >

dc1.corp.local
Wazuh Agent
Virtual Server

fmsrv
Logstash
ElasticSearch
Kibana
SweetSecurity Server
Wazuh OSSEC Server
Federated Security Module (FSM) Server

dc2.corp.local
Wazuh Agent
Virtual Server

vmsrv
Virtual Machine Server

honeysrv
HoneyTrap Server
HoneyTrap Agent
Honeypot Server and Agent

win81-client.corp.local
Wazuh Agent
Virtual Client Machine

- Sweet Security uses ELK 5.5, and there was a major change from ELK 5 to 6 under which many of the mappings are broken. This means that the indexes which are created in ELK 5.x will not work in ELK 6.x unless ELK stack is upgraded [126].

- Sweet Security does not appear to be under active development, and many problems reported on their official bug forum went unanswered by the developer during the years we worked on the project.

- ELK stack and Wazuh both have a very quick development cycle and were releasing a new version every one or two months. This forced updating and upgrading these software applications on an ongoing basis which posed an issue with making sure project files were up-to-date.

- HoneyTrap is under active development, so we had to request the developers do several software updates to their documentation and code before it was able to be used in this project.

- A good guide on getting pfSense to log to ELK stack does not exist, so we had to use several unofficial guides found on the Internet. We were able to get pfSense and Snort to log to the ELK stack, but a tutorial from the developers would have been more ideal.

- The Apache HTTPS wrapper Sweet Security uses around the ELK stack does cause some troubles with logging.

- Zeek compilation time on the Raspberry Pi exceeds an hour, so rebuilding it from scratch can take a very long time.

- Doing a complete install of Sweet Security on the Pi took an extremely long time, so we separated the installs to the Client on the Pi and Server on *fsmsrv*. This had

the added advantage of allowing the Client to be rebuilt more easily without losing the Server configurations.

- The Raspberry Pi sometimes gets overloaded and fails to respond, it must then be hard booted.

- The Raspberry Pi does not have sufficient RAM to have adequate performance when running Sweet Security full configuration. We found that often the Pi would become unresponsive during testing, and thus we recommend having an hourly reboot to keep it operational for the purposes of testing.

- Our *fmsrv* did not have adequate RAM to process large data sets, and we encountered a problem where it would fail to start Kibana because it was waiting for the shards to be available.

- Sysmon logs are too verbose for storage in in the Elastic Stack and thus we keep a 65 MB circular log of them on the Windows hosts for further examination as required. The idea would be that an intrusion would result in further examination of these logs which store process specific events.

CHAPTER 6

KIBANA PLUGIN FOR FSM

6.1   System Overview

Kibana gives many ways of searching the Elasticsearch indexes to find data you are looking for, but it can be difficult to wade through the interfaces to find the data you are looking for. In our implementation, the various sensors which observe our network for malware report data back to the Elastic Stack server as shown in Figure 6.1. All the data is available in the server, and we created a Kibana plugin to demonstrate how this data can be queried to assist in finding indicators of APST.

6.2   Programming and Implementation

An initial plugin was created using the Kibana Plugin Generator [127] which allowed us to generate a skeleton plugin for a specific version of Kibana setup in a development environment. The development environment is ideal because it will monitor the plugin for code changes and will automatically recompile the code after changes are made. Our plugin was implemented for Kibana 6.7.1 and the source code is available on GitHub [75].

Our main design goal with our plugin was to have a fool-proof user interface made available through standardized Kibana user interfaces. In order to do this we used the Elastic UI Framework, specifically Elastic UI Tables [128]. All our data was returned using EuiBasicTable with some column customization for the browser to return the specific fields we believed were most relevant. An example of the table appearance in our user interface can be seen in Figure 3.1.

We implemented five queries that the user can run in our Kibana plugin. The first query shown in Appendix Appendix B.1 returns *Blocked incoming packets from known bad sites in the last*

Figure 6.1: FSM Plugin System Overview
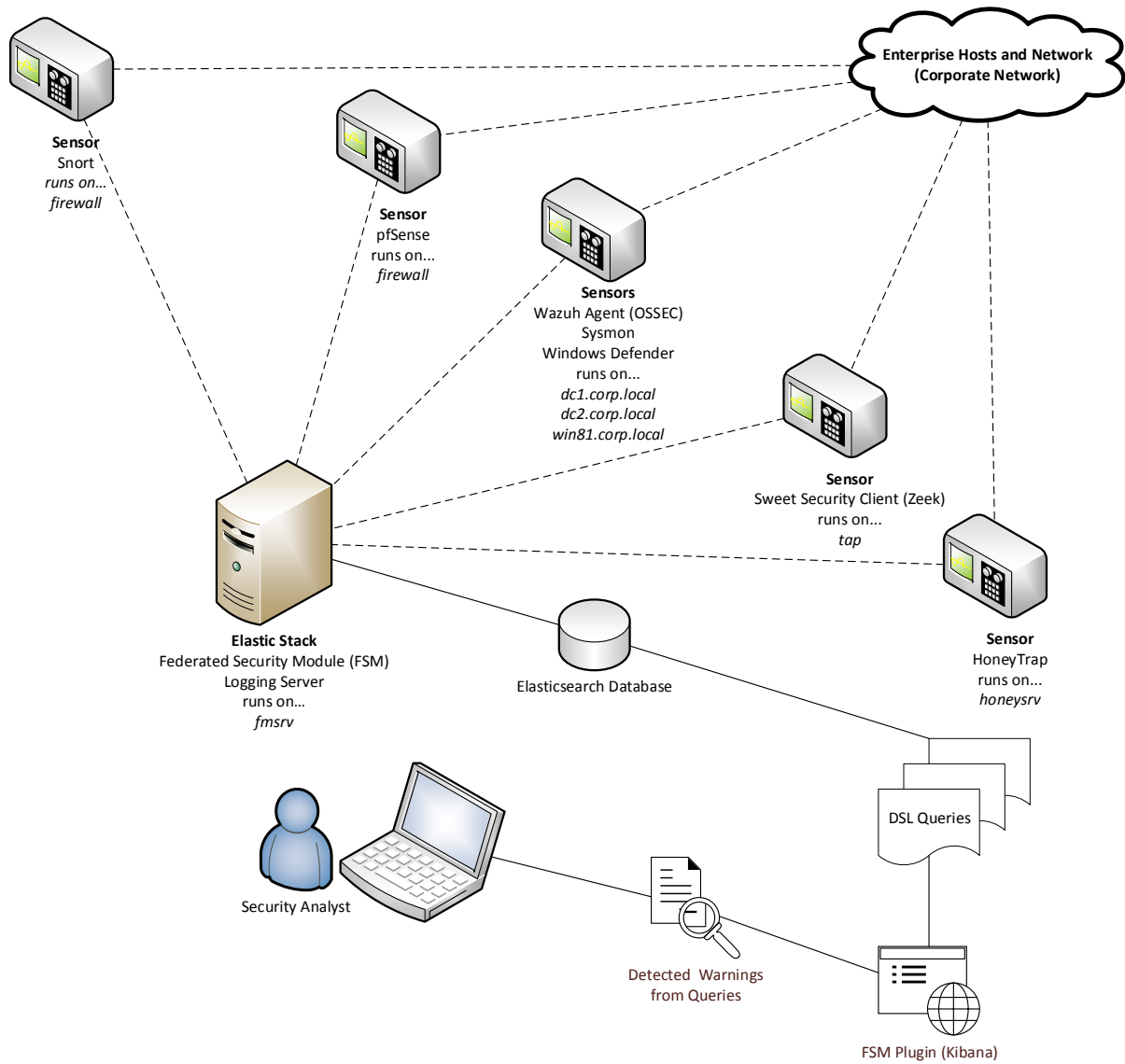
day. The second query shown in Appendix Appendix B.2 returns *Snort messages for the last day*. The third query shown in Appendix Appendix B.3 returns *Firewall login attempts in the last thirty minutes*. The fourth query shown in Appendix Appendix B.4 returns *HoneyTrap non-heartbeat events in the last three days*. And the final query shown in Appendix Appendix B.5 returns *Windows Defender detected malware in the last five days*.

All queries are implemented as asynchronous calls to Elasticsearch UI using Query DSL. The calls had to be asynchronous and will wait for Elasticsearch to return something, otherwise we could get invalid results. You can see these calls in action by sending a HTTP request to using special URLS. There URLS are registered in the primary index.js of the application. All query calls are done on the server-side.

Once calls have returned some data in Elasticsearch the formatting is done on the client side. As mentioned, the formatting is done by putting the returned query data into EuiBasicTables which are then rendered using ReactUI. In ReactUI, React Router is used to automate the display of individual tables when you click on the high level query link. Each table is interactive, and allows sorting on the column headings.

As mentioned, an example of the Query Dashboard which is available in our user interface can be seen in Figure 6.2. When a user clicks on one of the query links a background query will gather data from Elasticsearch. If the user clicks on *Blocked incoming packets from known bad sites* and some data is returned it would look similar to Figure 6.3. Sometimes, no data is available to be displayed, an example of what might occur should a user click on the *Windows Defender Detected Malware* query and no results are found is shown in Figure 6.4.

6.3   Why Tables?

Our basic FSM plugin provides results in a tabular format, which was useful to find important data without needing to create and run our own queries in Elasticsearch. We defined queries which we felt could provide a Security Analyst with relevant data in an easy to view manner and would

57

Figure 6.2: FSM Kibana Plugin Dashboard
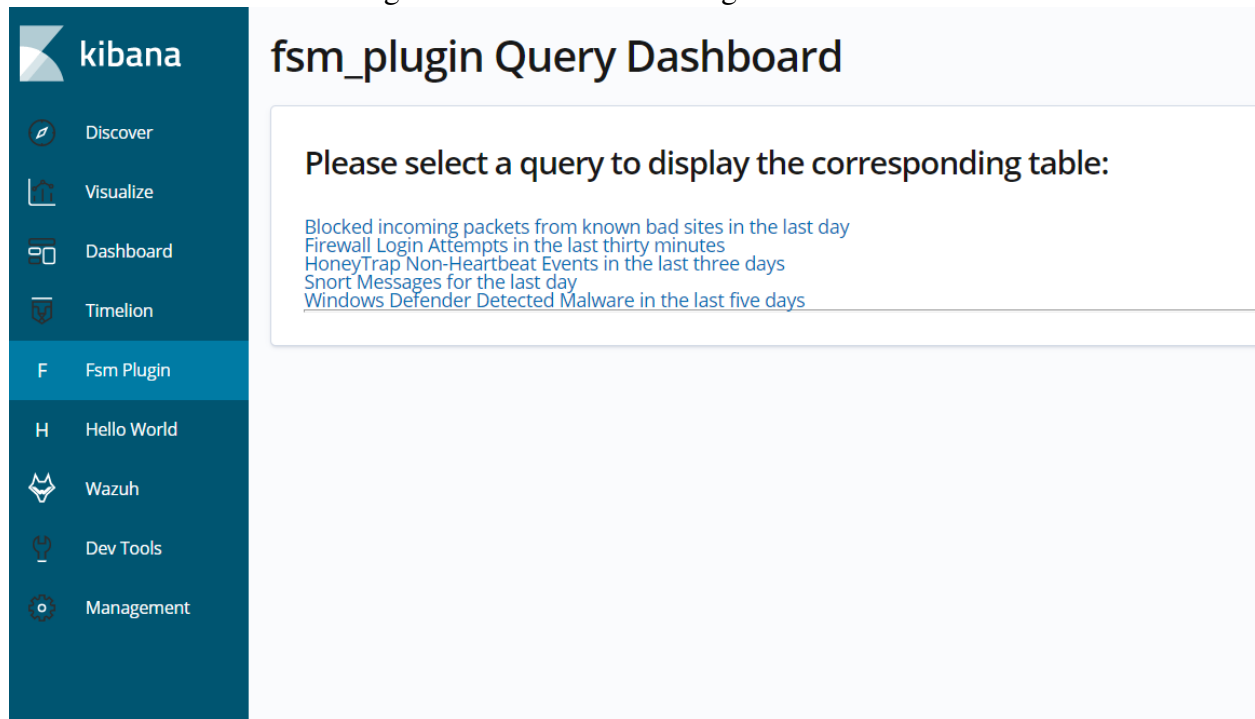
Figure 6.3: FSM Kibana Plugin Query with Results
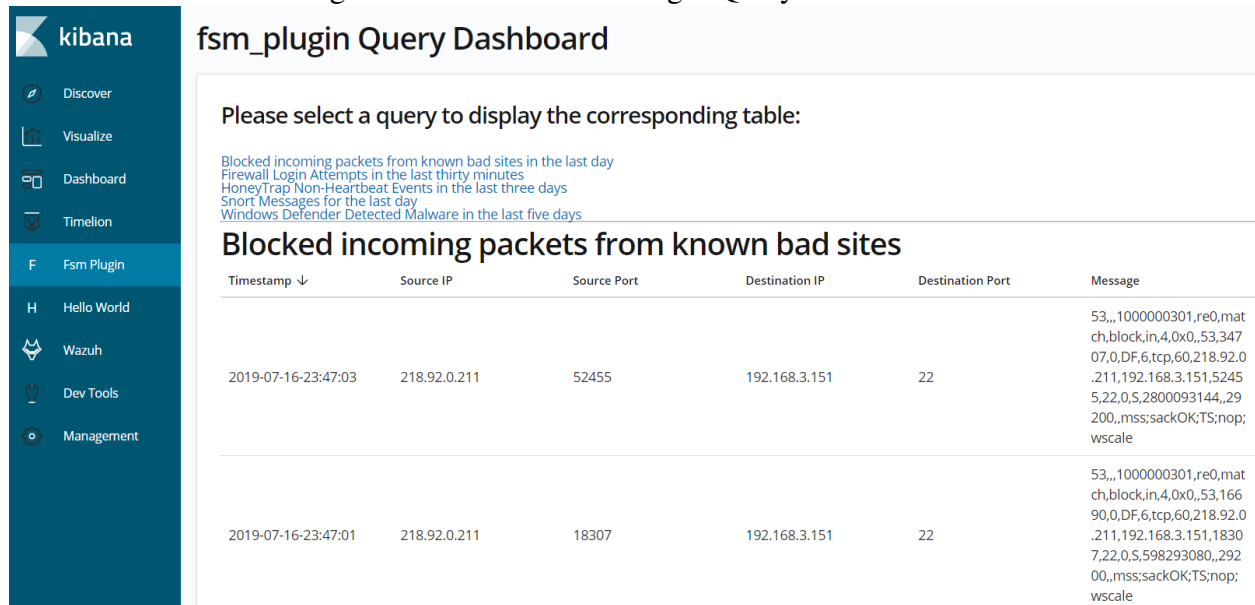
Figure 6.4: FSM Kibana Plugin Query Returning No Results
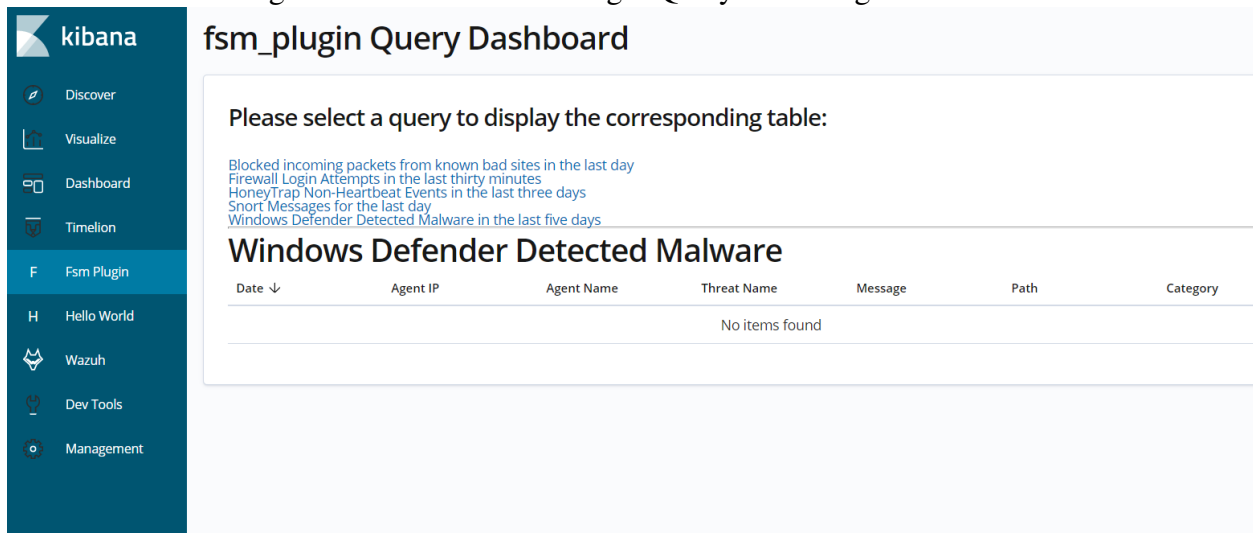
illustrate the plugin possibilities. As we can see from the Appendices, the data returned from Elasticsearch is often too unfiltered for someone to view and understand easily. By limiting the fields shown in the tables it is easier for someone to begin to understand the raw logs as irrelevant fields are filtered out. It is also possible to sort each column in ascending or descending order depending on preference. And finally, the tables are formatted in slices, so all the fields are not displayed overwhelming the user. For these reasons we felt basic tables were a good starting point for development of our plugin.

6.4 The Power of Visualization

Kibana has the ability to produce many different types of visualizations and time related charts. Two powerful ways we can use Kibana to search our indexes include the use of Visualize and Timelion within the Kibana dashboard. This will allow the Security Analyst to "see" trends and correlations between the various data sets, and can assist in the search for APST. The main way this can be done is through the showing trends from the various sensors we have deployed in our network. One simple example would be to present a "word cloud" for the honeypot to demonstration which protocols are most often being attacked on it, with the size of the protocol shown

correlated with the number of instances that protocol is seen. Our plugin does not currently use visualizations, but they would be helpful in future iterations.

6.5   Kibana Direct

Since Kibana has the ability to search the logs already, it may be required for the Security Analyst to directly interface with the logs in Kibana. They are able to use the full power of Kibana to search the log files using visualizations, charts, graphs, machine learning, and direct log searches. The Security Analyst must provide the search criteria into Kibana in order to search the logs. One way to do this is to use the X-Pack in Kibana which provides extended functionality. This of course requires the purchase of a license for more advanced features.

6.6   Evaluation of Plugin

Elasticsearch functions very similar to Google or other search engines, in that it is very quick and efficient at finding information in a stack of documents (or logs). It is also able to perform queries that span multiple search indexes to return results. However, it is important to point out that Elasticsearch does not perform queries that are relational in nature, and we cannot imagine it to have an exact mapping to SQL type queries. This means that JOIN type queries are not easily done in Elasticsearch and thus the cooperation between the tools is more easily done by querying each tool individually and then aggregating the results. This is demonstrated in our Kibana plugin in that each query returns data from a specific index in the database. We leave it up to the Security Analyst to interpret the results of each query and then take appropriate action.

Our module was limited to five queries as we wanted to have it available for demonstration purposes and not as a fully integrated tool. Many more queries could be added in an easy and straightforward manner. The queries we present do not span all available indexes, for example a query which searched Sweet Security for IRC traffic would have been an effective hunting tool for C&C malware. We have given a list of possible queries which could indicate malware in a previous

section, and it is left to the reader to expand upon the completed queries we have provided. In any case, the provided queries can give information about the presence or possible attack of APST within our simulated environment.

As a caveat, our plugin is not meant to replace Kibana itself which is the primary threat hunting suite. The Security Analyst will have all the log files available to Kibana and must make use of available search methods to find any threats, as the nature of APST makes them immune to any single detection method. Through the use of all tools, the hope would be that there is sufficient overlap to provide enough detection capability that indicators of compromise are found. Once those are found, the log files will allow backward searching to find the nature of the threat as it develops. In essence, logs allow you to travel back in time to get a representation of your network as it changed and is affected by various threats.

CHAPTER 7

APT HUNTER QUERY TOOL FOR FSM

"Proactively seeking unknown, malicious behaviours and looking for anomalies inside the network is the right approach..." when hunting for threats [129]. The likelihood of unknown threats occurring within the network is commonplace today, and is even more likely when dealing with APST. This type of threat landscape requires the threat hunting process shown in Figure 1 of [130]. Threat intelligence is gathered from various sources, a hunting process is done, and then substantiated intrusions are documented. This paper [130] continues with many possible solutions to find outliers in a data set, including statistical models, visualizations, machine learning, and manual searching. Our present approach is to use the power of Elasticsearch to parse large volumes of log data generated by our sensors to search for threats.

7.1   System Overview

APT Hunter [131] is a quick and dirty tool written in Python to query our data source indexes in Elasticsearch where our APST detection sensors store their log file data. Researchers have found that the use of Elasticsearch is an effective way of searching large data sets [132]. The Security Analyst will create Elasticsearch queries which specifically identify positive results for the threat they are hunting for, and then the tool will return results from the Elastic Stack. One the results are returned a detection algorithm will be used to determine whether the results are indicative of the presence of an APST. One simple detection algorithm has been implemented as an example in APT Hunter for detection of APST. This tool was created to show how the various sensors can work together to return a result on whether APST may be detected on the enterprise network. Our implementation uses a candid algorithm which reports on whether any of the sensors detected APST based on the queries given by the Security Analyst.

7.2 Programming and Implementation

The tool is implemented in Python 3 using several packages which allow it to parse command-line arguments and perform DSL Elasticsearch queries. The arguments tell the tool which queries should be run on which indexes. A connection to the Elasticsearch database is opened and all the queries are executed sequentially and the results are gathered into variables. After the queries have been run, a detection algorithm is used to decide whether there was any presence of APST detected.

7.3 Example Detection Algorithm

The detection of APST on any network is difficult and fraught with the potential for false positives. As mentioned in [133], "People make decisions in security - and detections of suspicious behaviour using complex mathematical models that cannot be explained to the analyst are not actionable." Often, when tools make determinations which are removed from the experience of the Security Analyst they will not trust that they can action the results without being able to defend their decision should issues arise. Therefore, it is imperative that a Security Analyst is able to understand why a threat report was generated. This demand for simplicity should underpin any search for APST. We present a simple, straightforward algorithm for the detection of APST. Basically, the number of results returned by all the queries are added together and returned as hits. We also present the average hits returned per sensor in a separate metric. If no hits are returned, we conclude no APST have been located with the given queries.

Algorithm 1: Naive Detection Algorithm

    Input: array *Results* of JSON Results from Elasticsearch Queries and integer of *sensors* used

    Result: Prints APST Detection Results

1 **if** *Results.length* > 0 **then**

2 | *hits* ⟵ 0;

3 | **for** *Result* ∈ *Results* **do**

4 | | *hits* = hits + *Result.hits.total*

5 | *Hits_Per_Sensor* ⟵ hits ÷ sensors;

6 | APST Query Hits: *hits*;

7 | Hits Per Sensor: *Hits_Per_Sensor*;

8 **else**

9 | No APST detected;

7.4 Evaluation of Query Tool

Our project has not used Machine Learning to replace the Security Analyst, who is integral in a successful APST hunt. The quality of the queries supplied by the Security Analyst will directly correlate with the success in identifying APST within the enterprise system. This means the APST hunter must become innovative when looking for behaviour that will indicate the presence of APST within the network. Papers we have examined during the research provide many different indicators that can be looked for [1], [59], [73]. The major work remaining will be to translate these indicators into queries that can be run in Elasticsearch. Although, we note that the use of Sigma [134] may make translation of detection rules easier by borrowing from existing rule sets. Finally, in our conclusion section we have mentioned the addition of Yara to detection [135], which we perceive will dramatically increase the detection rate on host systems. In all cases, the quality of the Security Analyst is the main determinate of the success in APST detection. Our project has provided the tools, concepts, and techniques which are required for a successful APST hunt.

CHAPTER 8

PENETRATION TESTS

8.1 Baseline

The baseline is an examination of normal traffic and reports generated by the various tools which form part of the FSM. To generate the traffic, we allowed the network to run normally for a number of weeks. During this time our Researcher Laptop was connected to DHCP on the network, and we completed normal tasks we would do like checking e-mail, surfing the web, watching videos, and downloading programs.

Our firewall had a basic ruleset, which used default Snort rules to filter out malicious traffic. Figure 8.1 demonstrates some hosts which Snort blocked going through the firewall. These blocked sites were identified using a free Oinkmaster account to download updated rulesets for Snort. Reasons for why certain hosts were blocked are identified by the Alert Descriptions. Snort will filter traffic traveling from the wide area network (WAN) to the local area network (LAN).

Sweet Security Server has a web interface which is hosted on *fmsrv* at https://192.168.1.103. This site has a simple password interface to limit access. Once you login, you are able to view devices which have been detected on the local network, an example is seen in Figure 8.2. When you click on a detected device, it will show you the expanded view similar to Figure 8.3. Finally, you can view new DNS / IP entries on the LAN as seen in Figure 8.4. These views show some of the information which is available in SweetSecurity including IP addresses, DNS queries, nmap scans of open ports, etc.

On the hosts, Wazuh and Windows Defender will provide endpoint protection and HIDS. Wazuh has been left with a slightly modified default ruleset, so it will generate alerts in a standardized manner. Windows Defender has also been left with default settings. It is expected that Windows Defender will be running up-to-date signature definitions.

Figure 8.1: Example Snort Alerts

Last 500 Hosts Blocked by Snort

#	IP	Alert Descriptions and Event Times
1	93.115.28.164 🔍	ET SCAN Sipvicious User-Agent Detected (friendly-scanner) -- 2018-08-27 01:54:45 ET SCAN Sipvicious Scan -- 2018-08-27 01:54:45
2	5.188.10.241 🔍	ET CINS Active Threat Intelligence Poor Reputation IP TCP group 6 -- 2018-08-27 12:36:38 ET DROP Spamhaus DROP Listed Traffic Inbound group 1 -- 2018-08-27 20:09:05 ET CINS Active Threat Intelligence Poor Reputation IP TCP group 5 -- 2018-08-27 20:09:05
3	82.221.105.6 🔍	ET CINS Active Threat Intelligence Poor Reputation IP TCP group 79 -- 2018-08-22 15:06:41 ET CINS Active Threat Intelligence Poor Reputation IP UDP group 79 -- 2018-08-22 12:25:01 ET CINS Active Threat Intelligence Poor Reputation IP TCP group 81 -- 2018-08-23 17:45:28 ET CINS Active Threat Intelligence Poor Reputation IP TCP group 88 -- 2018-08-27 12:41:20 ET CINS Active Threat Intelligence Poor Reputation IP UDP group 88 -- 2018-08-26 17:49:32 ET CINS Active Threat Intelligence Poor Reputation IP TCP group 89 -- 2018-08-27 22:31:49

Figure 8.2: Generic Device Information in Sweet Security

192.168.1.100
(CC8E715DDD1F)

IP: 192.168.1.100

Vendor: Unknown

Firewall: ACCEPT

Open Ports: 2

Monitored: Yes

More Info...

Figure 8.3: Expanded Device Information in Sweet Security

Figure 8.4: New DNS/IP Entries in Sweet Security

Baseliner	A new DNS query was added to the baseline: valve802.steamcontent.com	9CEBE8063EA1	2018-08-02 07:06:25
Baseliner	A new DNS query was added to the baseline: valve618.steamcontent.com	9CEBE8063EA1	2018-08-02 07:06:24
Baseliner	A new DNS query was added to the baseline: valve806.steamcontent.com	9CEBE8063EA1	2018-08-02 07:06:25

Within the internal network, HoneyTrap is active and presenting itself as a high value target. Besides the probes sent out by Sweet Security there should be no additional traffic attempting to connect to the HoneyTrap agent. Any other connection attempts from other hosts on the network mean they have been compromised, and should be investigated further.

8.2 EICAR Test File Download

The EICAR test file [136] is a unique signature which is used to provide rudimentary baseline testing of anti-virus software. We are going to download an EICAR executable on to the *win81-client* to see whether some useful log files are generated. This is a basic test to make sure that our ELK stack is running properly and appears to be gathering information through Wazuh for the Windows Defender Alert message regarding this software.

Wazuh agent on the Windows host will need to be be modified to correctly take the logs from Windows Defender. The local Wazuh configuration file C:\Program Files (x86)\ossec-agent\ossec.conf will be modified so collected logs will be sent to the ELK stack [137]. This will configure the Wazuh agent to monitor the specific Windows events for Windows Defender.

When the EICAR file was downloaded, an alert was displayed by Windows Defender that it is disinfecting the file and when we check the Windows logs it detects it as Virus:DOS/EICAR_Test_File. We verified that the Windows Event logs contain an alert for the EICAR test file [138].

Checking the Wazuh logs we found the results shown in Listing Appendix A.1 which clearly show that the Windows Defender scan result has been recorded by the ELK stack. This baseline test shows how the anti-virus will alert us about possible malware on the Windows hosts.

8.3 Basic HoneyTrap Connection Tests

HoneyTrap is configured to listen on ports SSH (8022), Telnet (23), and FTP (21). We briefly tested this capability by using PuTTy terminal emulator to connect to the SSH port and Telnet port to see what output is logged to the ELK stack. One result from the SSH test is shown in Appendix

A.2 and one result for the Telnet test is shown in Appendix A.3. These examples show that logging is working properly and that our connection attempts are being recorded.

8.4   APST Testing Suites

We accept that without any type of early warning system, the detection and removal of APST will be ineffective and virtually non-existent. We take this to be self-evident and do not proceed to prove it further.

One of the easiest ways to simulate APST on our network is to use open source projects which have been designed for this purpose. Two available test suites are APTSimulator [78] and Flight-Sim [79]. APTSimulator will simulate APST on our Windows hosts and FlightSim will simulate malicious traffic on our enterprise network. These tools will form the first level of testing we will do to evaluate our FSM solution. They will also give us information about what we should look for in the various log files to show malicious activity is happening within our network.

For each trial we run, we have gathered example log files which indicate that a compromise has occurred. These examples are gathered to provide documented proof of what our tools are reporting and allow analysis of the effectiveness of the alerts. Also, by testing our setup we will hopefully find additional improvements which will allow our tool to catch various intrusion cases. In some of the test cases, the environment may be slightly modified to narrow our search parameters and avoid the overwhelming wave of data which can come from Zeek. Only relevant hosts will be monitored during these tests.

8.5   APTSimulator - Command and Control

The first test which we undertook was to run the Command and Control (C2) test in the *win81-client.corp.local*. This is the test which is activated by selecting option "2" in the APTSimulator batch file in an Administrator Command Prompt as seen in Figure 8.5. The first phase of this test attempts to connect to sites which are commonly used for malicious C2 servers and then ping them.

69

Sysmon catches this malicious activity in its logs as shown in Figure 8.6. Another component of this test attempts to use a malicious user agent to access websites, for which it receives HTTP 200 status codes [139]. A sample Sysmon log for one of these operations is seen in Figure 8.7. A total of around seventy Sysmon log entries were generated during this test.

We will now examine the ELK stack to see what logs have been generated during this time period. The first area of interest is the logstash-* logs. These logs will have captured network packets. We can actually see the HTTP requests from the various agents which are returning success HTTP code 200. These results are contained in Appendix A.4, Appendix A.5 and Appendix A.6. This is only one simple example of how detailed the logs provided by Zeek are in terms of the data they can return.

Looking through other log files, we did not find much of interest. Wazuh failed to detect much during this testing phase. It appears that only Sysmon and Zeek detected usable log file intelligence but the other sources did not detect much which was actionable.

8.6   APTSimulator - Persistence

The second test which we undertook was to run the Persistence Test in the *win81-client.corp.local*. This is the test which is activated by selecting option "8" in the APTSimulator batch file in an Administrator Command Prompt as seen in Figure 8.8.

This test generated many alerts in Windows Defender, it was important to note there was a delay of a few minutes between when the items were registered in the logs. One example of a log file which was generated can be seen in Appendix A.7. Many alerts were generated in Wazuh for this specific test showing Wazuh was very effective at logging these type of attacks.

Sysmon also had high quality Windows logs which were generated. One example of "At Job Creation" is shown in Figure 8.9. Many other logs were generated, but we will not enumerate through them at this time.

Zeek and other tools were not very effective at detecting these attacks because no data was

Figure 8.5: Administrator Command Prompt Running Command and Control Tests

```
RUNNING SET: "command-and-control"

==============================================================================
C2 Access
Using curl to access well-known C2 addresses
C2: msupdater.com
Result: 000
C2: twitterdocs.com
Result: 000
C2: freenow.chickenkiller.com
Result: 000
==============================================================================
DNS CACHE
Creating DNS Cache entries for well-known malicious C2 servers
C2: msupdater.com
Non-authoritative answer:
C2: twitterdocs.com
Non-authoritative answer:
C2: freenow.chickenkiller.com
*** UnKnown can't find freenow.chickenkiller.com: Non-existent domain
C2: www.googleaccountsservices.com
Non-authoritative answer:
==============================================================================
CACTUSTORCH

Using certutil to drop a CactusTorch shellcode lanucher injecting bind shell (po
rt 1234/tcp) into rundll32.exe
Fixing possible problems with JavaScript on the system
Downloading the CactusTorch dropper (press Enter if it takes more than 20s)
****  Online  ****
  0000  ...
  372f
CertUtil: -URLCache command completed successfully.
Executing the CactusTorch dropper
==============================================================================
MALICIOUS UA
Using malicious user agents to access web sites
HttpBrowser RAT
Result: 200
Dyre / Upatre
Result: 200
Sality
Result: 200
NJRat
Result: 200
==============================================================================
MALICIOUS UA

Using malicious user agents to access web sites
HttpBrowser RAT
Result: 200
Dyre / Upatre
Result: 200
Sality
Result: 200
NJRat
Result: 200
==============================================================================
NETCAT ALTERNATIVE
Dropping a Powershell netcat alternative into the APT dir
==============================================================================
NETCAT ALTERNATIVE

Dropping a Powershell netcat alternative into the APT dir
==============================================================================
```

Figure 8.6: Sysmon Windows Log for NSLookup of Bad Sites

Figure 8.7: Sysmon Windows Log for a RAT

sent over the network external to the *win81-client.corp.local* host. In a real-world scenario we would expect the hacking tools to be downloaded from an external host which would hopefully be detected through Zeek.

8.7   APTSimulator - Discovery

The third test which we undertook was to run the Discovery Test in the *win81-client.corp.local*. This is the test which is activated by selecting option "5" in the APTSimulator batch file in an Administrator Command Prompt as seen in Figure 8.10.
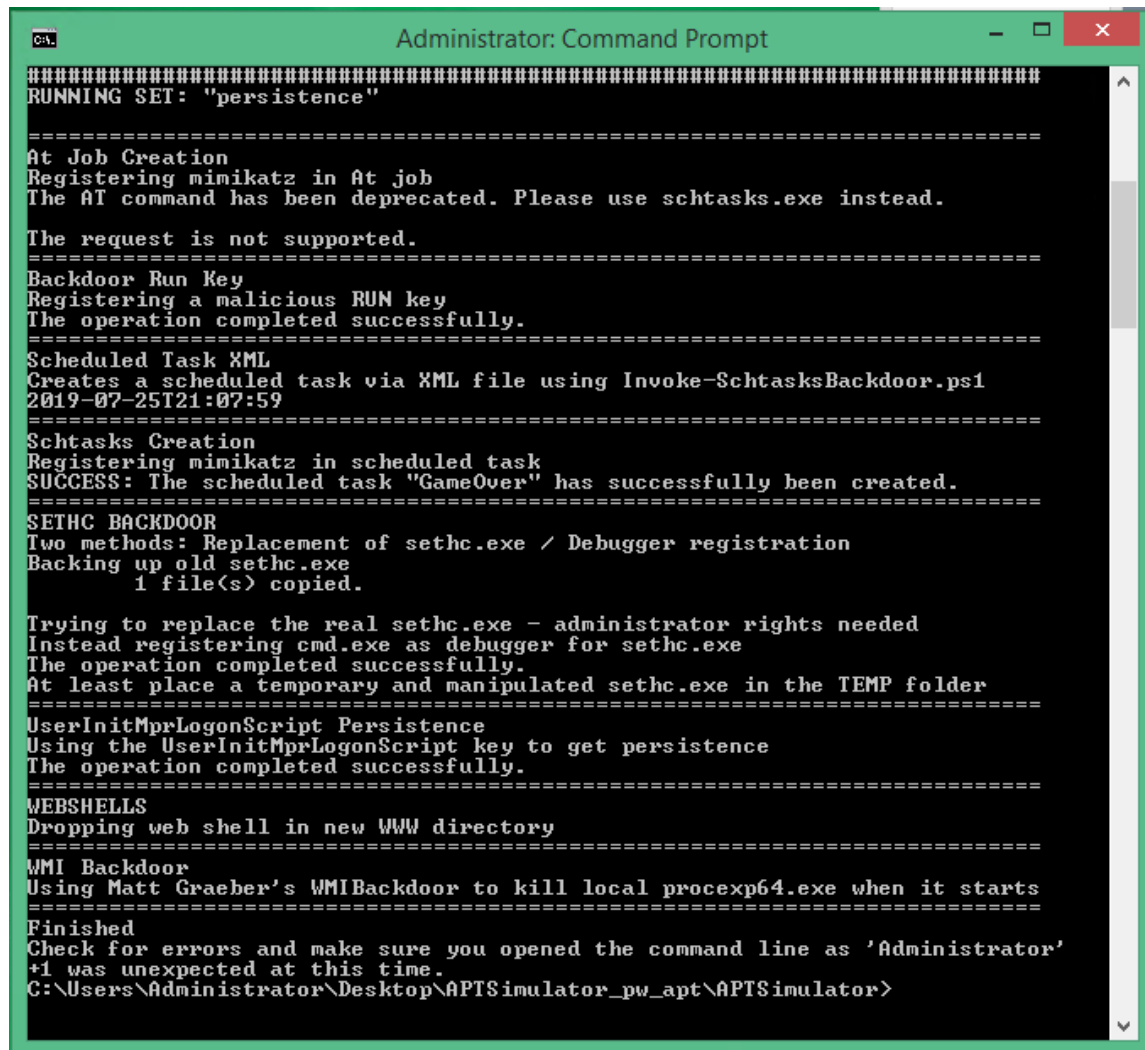
Sysmon was again very useful during this scan. It easily shows the nbtscan taking place and the ranges that are being scanned, in Figure 8.11 was can see the scanned range is 192.168.1.0/24. In Figure 8.12 we can see the use of the "systeminfo" command to gather intelligence about the host system.

Since this test includes a network scan, we would expect Zeek to have gathered some log files showing the network scan taking place, however we did not see any in the generated log files. Additionally, other components did not show log files for this test either.

8.8   FlightSim

We ran all the tests for FlightSim on *win81-client.corp.local* host. These tests are meant to simulate malicious network traffic originating from a host system. The specific tests which were run are shown in Appendix C.1. Wazuh did not detect much from this test, and from the logs we see that Wazuh only logged the failure of one of the DNS names *ompute.pl* to resolve in Appendix A.8. When we peer into Sysmon, we again find many high quality logs regarding the network connections which are being made. One example is the SMTP connection to an external website shown in Figure 8.13. Other logs provided by Zeek are extensive as expected, which shows the DNS queries and network connections being made. HoneyTrap, pfSense, and Snort do not show any useful logs.

Figure 8.8: Administrator Command Prompt Running Persistence Tests

Figure 8.9: Sysmon Windows Log for Persistence Test

Figure 8.10: Administrator Command Prompt Running Discovery Tests

8.9 Snort and pfSense Tests

Our testing methodology highlighted that most simulators provided attack vectors which came from inside the network. Snort and pfSense are aligned to attack vectors that occur from outside the network, for now we have assumed that the "in the wild" attacks which were coming at our firewall are sufficient proof that pfSense and Snort are working as expected. We have included an example pfSense blocked log in Appendix A.9 and Snort blocked log example in Appendix A.10 for completeness.

8.10 Assessment of Test Findings

All tests helped to illustrate the type of data which our sensors were able to gather for various simulated APST threats. The type of data returned varied widely depending on the test being performed, and ranged from no data being returned to large amounts of data returned. We found that Zeek often returned too much data, while other tools like pfSense and Snort did not return any

Figure 8.11: Sysmon Windows Log for nbtscan Test

Figure 8.12: Sysmon Windows Log for systeminfo Test

Figure 8.13: Sysmon Windows Log for Flightsim SMTP Test

data during our testing. Just because no data was returned does not mean no data was available, but rather that our simulated threats were inadequate especially for simulating threats which attack from the external network (Internet).

Another concern was that it was difficult to find general patterns in the data which was returned. We only performed one testing iteration to get a single sample of each data type, and it would have been beneficial to run several different trials to make sure the same type of data was being returned each time. However, even with our limited sample some general conclusions can be drawn.

Zeek is the most effective tool for peering at the network traffic within our network and was definitely well choosen. Sysmon was an excellent way of collecting data on the Windows host systems, and while Wazuh was able to gather limited data often the Sysmon logs were preferred to the Wazuh alerts. Windows Defender was able to detect known hacking tools, but we expect it would not work well against threats where it has to use heuristic analysis. HoneyTrap was an effective way of detecting network mapping attacks where APST are attempting to explore the network to gain a foothold on other machines.

Overall, given the amount of data returned by our solution we strongly believe an automated way of detection should be developed. While our Kibana dashboard will assist the security analyst, there may still be too much data returned by our sensors for a human to handle easily. This problem will only get worse as the size of the enterprise network being monitored grows.

CHAPTER 9

DISCUSSION

9.1 Assessment of the Federated Security Module and the System

There are many strengths to our current approach:

- Use of open source tools allows us to make available our full tool source code and capabilities, with the additive benefit of other researchers to expand upon our work. We have created a public repository to store all project documents and source code, so it will be possible for anyone with access to the Internet to continue and expand upon our ideas.

- Logstash allows easy log handling for a myriad of applications, which means we can easily expand the security tools which make up our FSM. While Logstash can ingest unstructured logs it can help to have some customization of the parsers to enforce structure on the logs.

- Wazuh (OSSEC) allows monitoring of a large number of host operating systems [140]. And the configuration file allows us to make many customizations depending on our requirements. Wazuh has a quick release cycle so new features are being added regularly.

- Kibana allows the inclusion of user defined visualizations and dashboards to make sense of the threat intelligence that is being gathered. The was data is represented still takes careful consideration and forethought to make sure dashboards are fit for the purpose they were designed.

- Our solution makes use of low-cost hardware like the Raspberry Pi 3 B+ and small

PC as the firewall appliance. The Raspberry Pi has been going through new releases since our project was launched several years ago. This means that newer, more robust versions are now available [141].

- Most of the tools are under active development so they are improving and adding new features all the time. Wazuh and Elastic Stack were under the most active development during our implementation period.

- Our solution keeps all data within the internal enterprise network and does not make use of cloud services for the storing of any data. This means that privacy can be maintained in regards to proprietary data which the tools may collect from within the enterprise network, and there are no security or privacy concerns in regards to data collected by the tools. During our project we did not develop a robust security architecture, but the Elastic Stack does allow for user authentication and encryption support [142].

- Our approach will scale to a large number of hosts and the tools we selected can be scaled to a large infrastructure.

- The ELK stack is very quick at making data available in real-time, so it can be searched and utilized when combating evolving threats.

There are some weaknesses in our current approach, which are limitations of the open source tools which were used:

- Encrypted traffic cannot be monitored effectively by Zeek. The initial traffic showing that an encrypted connection is setup will be captured, but the encrypted traffic will not be plain text when captured [143]. This will severely limit our ability to intercept encrypted communications within our internal network. Decrypting SSL/TLS traffic [144] is possible, but often requires corporate solutions which inject

an intermediate certificate to allow a man-in-the-middle attack (MITM) . These corporate solutions were not within the scope of our research.

- Critical Stack is not perfectly functional within Sweet Security, which will limit the ability to take note of certain types of traffic [145]. One example of traffic we may wish to flag within our network is Tor traffic as it indicates an attempt to send hidden traffic over our network to unknown destinations.

- Monitoring DNS requests for malicious sites is another important function which we did not select a tool for. While, Sweet Security will monitor DNS requests, it does not explicitly check for malicious DNS entries.

- A centralized logging solution within ELK stack would have been beneficial to have one unified dashboard for alerts. One example of this is to create a Kibana App which could contain these visualizations.

- The security of the FSM server is of course of paramount importance, and if our solution was deployed it would be prudent to spend additional time securing the server and associated logging devices.

- Our chosen default configurations may not be ideal in all situations. For example, in the case of the firewall it might be beneficial to log both blocked and allowed traffic, instead of only blocked traffic.

9.2   Statement of Problems Left Unsolved

As with any project there are many areas which could be improved with additional work. The main area of further work and development would be with the FSM Plugin [75] which was written as a proof of concept on one way data can be returned from our solution in an easy to view format. Currently, the data is displayed in a tabular format but the inclusion of graphs or figures may assist

a human analyst in finding connections between disparate data points. This plugin was developed for an older version of Kibana and it needs to be updated and kept inline with the latest version of Kibana. We did not attempt to make our plugin work with the latest version as this was not the direction our research followed.

The removal of APST was a goal of this report but was not investigated in detail. At this time removal of APST from a machine would require rebuilding the host from known good backups or doing a clean installation from vendor media. The removal of APST was unfortunately too broad of a topic for us to give appropriate consideration during our investigation.

CHAPTER 10

CONCLUSIONS AND RECOMMENDATIONS

10.1 Significance of the Results and Contributions

A overview of our project [146] and code repository [75] has been made available on GitHub. This has been used to document our project and made the general findings available to the public. GitHub was a good choice for this because it allows our project to be made publicly available and allows other users to easily fork our repository [147] to make their own changes and additions. Also, since most of the open and free software project we used are available through GitHub we were able to create forks of those projects to preserve the source code of the various tools used during our project.

Preliminary results suggest that some malware can be detected by our solution. The next investigation would be to perform a real-world test and monitor a production enterprise network to see whether threats are adequately flagged within the tool. Another opportunity for improvement is to greatly expand upon the FSM Plugin which we began to implement with the intent of expanding it into a fully functional dashboard which automatically highlights detected threats by unique network host.

Finding resources which allowed us to finish this research was often difficult. This was most apparent in documentation required to create a Kibana plugin. Due to the quick development cycle of Kibana any tutorials we were able to find were often out of date with the authors describing that there was no plans to update them at any time in the future [148]. We attempted to contact the developers through their online forum and had mixed success with the advice they provided. This could not be easily overcome except by gaining experience in Kibana plug-in creation through trial and error. In future work, we would like to find better resources to assist in the programming component of our project.

Using open source software was a large limitation on our project. Almost all software we used was free or open source software which drastically increased the complexity of what we had available to work with. Open source projects differ much in how well their caretakers look after them, in some cases like Sweet Security the project has not been cultivated in years. Another area this was very prevalent is in the anti-virus software space, since we found that ClamAV is one of the only open source anti-virus products and we believe it is inferior to the available commercial options. These two examples illustrate that a larger organization with the availability of commercial software applications would have greater success in looking for APST within their networks.

Sysmon was used to log process and registry changes on the Windows host systems. Sysmon was able to log a significant amount of information, but the connection between Wazuh and Sysmon was only returning some information and not the complete log files as we had expected. Further work can be done to bring Wazuh to a newer version and add additional processing of Sysmon logs to Wazuh.

Wazuh was one of the best tools we used and has a promising toolkit of features in development. One of the latest features we anticipate as being useful for APST mitigation is Yara support [135]. Yara [149] allows malware researchers to create signature definitions (indicators of compromise) for detecting APST as they are found in the wild. Bortniker [150] presents a good analysis of the efficacy of Yara.

10.2   Future Work

Adding new sensors to the network is a possibility for future exploration. We have observed through working with the five tools in our federated logging solution that adding more sensors increases the complexity of our overall solution. At the same time, there is a dramatic increase in the ability of our tools to accomplish the purpose we have set out for them. A few new sensors would have been welcome in our investigation, one example would have been a way of monitoring encrypted communication streams in real-time. Without having the ability to decode all data

streaming across the network to data streams which can be monitored causes us to lose out on a large amount of meaningful data which we could use to search for information. This could be fixed by using commercial software.

Automation within our system could be used to automatically update the dashboard with the latest intelligence. Having our system constantly monitoring for threats would reduce the overhead required to manually refresh the interface. This could be accomplish through cron jobs, polling, or simply running an infinite loop which updates everything once in a while.

Active response [151] would be another vein to mine for improvements. The limitations of a human security analyst means that an operator needs to maintain constant vigilance twenty-four hours a day, seven days a week. This is hardly an efficient use of time, and it would be great to have automated scripts which would trigger when certain events take place. A simple example would be to send an SMS or email to the security analyst when an event takes place. In a more extreme example, the system could actually perform an action like executing a script which blocks traffic from a particular host, shutdown an infected machine, or disables a new account which was just created. All these are examples of active response systems which would improve the overall effectiveness of detecting and responding to APSTs.

Elastic Stack offers a myriad of possibilities to expand our investigation into the depth of the data that our sensors have gathered on the network [152]. Machine Learning could be used to re-place the Security Analyst who has to monitor the network for anomalies by highlighting important findings. Alerting could be used to have a real-time system of letting Administrators know about particularly troubling problems. Graph could be used to show a diagrams of the hosts on the net-work and their interactions using one of the various security related networking models previously mentioned in the literature review. Reporting could be used to create summary reports of what has been found or to record the status of particular network details which are important to monitor for trend analysis. Another recent addition is Elastic SIEM which will bring SIEM capabilities to the Elastic Stack [153]. In short, there are potentially limitless possibilities within the Elastic Stack

which we could use to refine our never-ending search for understanding of APST.

Our research project has raised important questions about the detection of APST on our simulated enterprise network. Each component was able to perform in an adequate manner in detecting APST, but only with the help of the researcher. Some automation was available in Wazuh for automatic processing of alerts but in order to produced a unified solution we will definitely need the inclusion of Machine Learning for the automated processing of alerts. This is the future path we would expect this research to follow, and hopefully it can be investigated by other brilliant minds in the years ahead. The removal of APST is too advanced to be adequately covered in the time we had allotted, so we also look forward to future researchers taking on that additional challenge!

REFERENCES

[1] P. Chen, L. Desmet, and C. Huygens, "A Study on Advanced Persistent Threats," in *IFIP International Conference on Communications and Multimedia Security*. Springer, 2014, pp. 63–72. [Online]. Available: https://lirias.kuleuven.be/bitstream/123456789/461050/1/2014-apt-study.pdf

[2] M. Li, W. Huang, Y. Wang, W. Fan, and J. Li, "The study of APT attack stage model," in *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, June 2016, pp. 1–5. [Online]. Available: https://doi.org/10.1109/ICIS.2016.7550947

[3] J. Estublier, H. Verjus, and P. . Cunin, "Designing and building software federations," in *Proceedings 27th EUROMICRO Conference. 2001: A Net Odyssey*, Sep. 2001, pp. 121–129. [Online]. Available: https://doi.org/10.1109/EURMIC.2001.952446

[4] Y. Wang and J. Yang, "Ethical Hacking and Network Defense: Choose Your Best Network Vulnerability Scanning Tool," in *2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, March 2017, pp. 110–113. [Online]. Available: https://doi.org/10.1109/WAINA.2017.39

[5] C. Stoll, *The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage*. Gallery Books, 2005. [Online]. Available: https://books.google.ca/books?id=9B1RfCAar2cC

[6] B. I. D. Messaoud, K. Guennoun, M. Wahbi, and M. Sadik, "Advanced Persistent Threat: New analysis driven by life cycle phases and their challenges," in *2016 International Conference on Advanced Communication Systems and Information Security (ACOSIS)*, Oct 2016, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ACOSIS.2016.7843932

[7] C. Tankard, "Advanced Persistent threats and how to monitor and deter them," *Network Security*, vol. 2011, no. 8, pp. 16 – 19, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1353485811700861

[8] M. Courtney, "States of cyber warfare," *Engineering Technology*, vol. 12, no. 3, pp. 22–25, April 2017. [Online]. Available: https://doi.org/10.1049/et.2017.0300

[9] N. Shalev, I. Keidar, Y. Weinsberg, Y. Moatti, and E. Ben-Yehuda, "WatchIT: Who Watches Your IT Guy?" in *Proceedings of the 26th Symposium on Operating Systems Principles*, ser. SOSP '17. New York, NY, USA: ACM, 2017, pp. 515–530. [Online]. Available: https://doi.org/10.1145/3132747.3132752

[10] T. R. Jackson, J. G. Levine, J. B. Grizzard, and H. L. Owen, "An investigation of a compromised host on a honeynet being used to increase the security of a large enterprise network," in *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004.*, June 2004, pp. 9–14. [Online]. Available: https://doi.org/10.1109/IAW.2004.1437791

[11] R. Souza, C. Chavez, and R. A. Bittencourt, "Rapid Releases and Patch Backouts: A Software Analytics Approach," *IEEE Software*, vol. 32, no. 2, pp. 89–96, Mar 2015. [Online]. Available: https://doi.org/10.1109/MS.2015.30

[12] S. Burji, K. J. Liszka, and C. C. Chan, "Malware analysis using reverse engineering and data mining tools," in *2010 International Conference on System Science and Engineering*, July 2010, pp. 619–624. [Online]. Available: https://doi.org/10.1109/ICSSE.2010.5551719

[13] P. OKane, S. Sezer, and K. McLaughlin, "Obfuscation: The Hidden Malware," *IEEE Security Privacy*, vol. 9, no. 5, pp. 41–47, Sept 2011. [Online]. Available: https://doi.org/10.1109/MSP.2011.98

[14] M. Ussath, D. Jaeger, F. Cheng, and C. Meinel, "Advanced persistent threats: Behind the scenes," in *2016 Annual Conference on Information Science and Systems (CISS)*, March 2016, pp. 181–186. [Online]. Available: https://doi.org/10.1109/CISS.2016.7460498

[15] (2017) Vault 7: CIA Hacking Tools Revealed. WikiLeaks. Accessed 2017-12-06. [Online]. Available: https://wikileaks.org/ciav7p1/

[16] Network Operations Division Cryptographic Requirements. Central Intelligence Agency. Accessed 2017-12-06. [Online]. Available: https://wikileaks.org/ciav7p1/cms/files/NOD% 20Cryptographic%20Requirements%20v1.1%20TOP%20SECRET.pdf

[17] Network Operations Division CNE Operational Data Exchange Format (Codex) Specification. Central Intelligence Agency. Accessed 2017-12-06. [Online]. Available: https://wikileaks.org/ciav7p1/cms/files/Codex-Spec-v1-SECRET.pdf

[18] Network Operations Division Persisted DLL Specification. Central Intelligence Agency. Accessed 2017-12-06. [Online]. Available: https://wikileaks.org/ciav7p1/cms/files/ ICE-Spec-v3-final-SECRET.pdf

[19] Network Operations Division In-memory Code Execution Specification. Central Intelligence Agency. Accessed 2017-12-06. [Online]. Available: https: //wikileaks.org/ciav7p1/cms/files/ICE-Spec-v3-final-SECRET.pdf

[20] M. Hanspach and M. Goetz, "On covert acoustical mesh networks in air," *arXiv preprint arXiv:1406.1213*, 2014. [Online]. Available: https://arxiv.org/pdf/1406.1213.pdf

[21] L. Shing, J. Astacio, A. Figueroa, and C. C. Shing, "Vulnerabilities of radio frequencies," in *2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, Aug 2015, pp. 2682–2686. [Online]. Available: https://doi.org/10.1109/FSKD. 2015.7382381

[22] Y. Altshuler, N. Aharony, A. Pentland, Y. Elovici, and M. Cebrian, "Stealing Reality: When Criminals Become Data Scientists (or Vice Versa)," *IEEE Intelligent Systems*, vol. 26, no. 6, pp. 22–30, Nov 2011. [Online]. Available: https://doi.org/10.1109/MIS.2011.78

[23] F. Li, A. Lai, and D. Ddl, "Evidence of Advanced Persistent Threat: A case study of malware for political espionage," in *2011 6th International Conference on Malicious and Unwanted Software*, Oct 2011, pp. 102–109. [Online]. Available: https://doi.org/10.1109/MALWARE.2011.6112333

[24] A. A. Adams, "Report of a Debate on Snowden's Actions by ACM Members," *SIGCAS Comput. Soc.*, vol. 44, no. 3, pp. 5–7, Oct. 2014. [Online]. Available: https://doi.org/10.1145/2684097.2684099

[25] I. Koshiw. (2018, Aug) How an International Hacker Network Turned Stolen Press Releases into $100 Million. The Verge. Accessed 2018-08-23. [Online]. Available: https://www.theverge.com/2018/8/22/17716622/sec-business-wire-hack-stolen-press-release-fraud-ukraine

[26] G. Schryen and R. Kadura, "Open Source vs. Closed Source Software: Towards Measuring Security," in *Proceedings of the 2009 ACM Symposium on Applied Computing*, ser. SAC '09. New York, NY, USA: ACM, 2009, pp. 2016–2023. [Online]. Available: https://doi.org/10.1145/1529282.1529731

[27] R. Wojtczuk and A. Tereshkin, "Attacking intel bios," *BlackHat, Las Vegas, USA*, 2009. [Online]. Available: http://www.blackhat.com/presentations/bh-usa-09/WOJTCZUK/BHUSA09-Wojtczuk-AtkIntelBios-SLIDES.pdf

[28] R. Minnich. Replace Your Exploit-Ridden Firmware with Linux. YouTube. Accessed 2017-12-06. [Online]. Available: https://youtu.be/iffTJ1vPCSo

[29] A. Tanenbaum. An Open Letter to Intel. Accessed 2017-12-06. [Online]. Available: http://www.cs.vu.nl/~ast/intel/

[30] J. L. G. Dietz and J. A. P. Hoogervorst, "Enterprise Ontology in Enterprise Engineering," in *Proceedings of the 2008 ACM Symposium on Applied Computing*, ser. SAC '08. New York, NY, USA: ACM, 2008, pp. 572–579. [Online]. Available: https://doi.org/10.1145/1363686.1363824

[31] Y.-W. E. Sung, X. Sun, S. G. Rao, G. G. Xie, and D. A. Maltz, "Towards Systematic Design of Enterprise Networks," *IEEE/ACM Trans. Netw.*, vol. 19, no. 3, pp. 695–708, Jun. 2011. [Online]. Available: https://doi.org/10.1109/TNET.2010.2089640

[32] R. Khadka, B. V. Batlajery, A. M. Saeidi, S. Jansen, and J. Hage, "How Do Professionals Perceive Legacy Systems and Software Modernization?" in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014. New York, NY, USA: ACM, 2014, pp. 36–47. [Online]. Available: https://doi.org/10.1109/10.1145/2568225.2568318

[33] P. A. Sandborn and V. J. Prabhakar, "The Forecasting and Impact of the Loss of Critical Human Skills Necessary for Supporting Legacy Systems," *IEEE Transactions on Engineering Management*, vol. 62, no. 3, pp. 361–371, Aug 2015. [Online]. Available: https://doi.org/10.1109/TEM.2015.2438820

[34] J. Moubarak, M. Chamoun, and E. Filiol, "Comparative study of recent MEA malware phylogeny," in *2017 2nd International Conference on Computer and Communication Systems (ICCCS)*, July 2017, pp. 16–20. [Online]. Available: https://doi.org/10.1109/CCOMS.2017.8075178

[35] M. Lindorfer, A. Di Federico, F. Maggi, P. M. Comparetti, and S. Zanero, "Lines of Malicious Code: Insights into the Malicious Software Industry," in *Proceedings of the 28th Annual Computer Security Applications Conference*, ser. ACSAC '12. New York, NY, USA: ACM, 2012, pp. 349–358. [Online]. Available: https://doi.org/10.1145/2420950.2421001

[36] N. Kshetri, "Cyberwarfare: Western and Chinese Allegations," *IT Professional*, vol. 16, no. 1, pp. 16–19, Jan 2014. [Online]. Available: https://doi.org/10.1109/MITP.2014.4

[37] S. Bazan, "A New Way to Win the War," *IEEE Internet Computing*, vol. 21, no. 4, pp. 92–97, 2017. [Online]. Available: https://doi.org/10.1109/MIC.2017.2911419

[38] T. A. Berson and D. E. Denning, "Cyberwarfare," *IEEE Security Privacy*, vol. 9, no. 5, pp. 13–15, Sept 2011. [Online]. Available: https://doi.org/10.1109/MSP.2011.132

[39] J. Svoboda, I. Ghafir, and V. Prenosil, "Network Monitoring Approaches: An Overview,"

International Journal of Advances in Computer Networks and Its Security– IJCNS, vol. 5, pp. 88–93, 10 2015. [Online]. Available: https://www.researchgate.net/publication/ 305957483_Network_Monitoring_Approaches_An_Overview

[40] R. Scrimger, K. Adam, and Scrimaer, *MCSE TCP/IP with Cdrom*, 2nd ed. Thousand Oaks, CA, USA: New Riders Publishing, 1998.

[41] R. Brewer, "Advanced persistent threats: minimising the damage," *Network Security*, vol. 2014, no. 4, pp. 5 – 9, 2014. [Online]. Available: http://www.sciencedirect.com/science/ article/pii/S1353485814700406

[42] J. Chen, C. Su, K.-H. Yeh, and M. Yung, "Special issue on advanced persistent threat," *Future Generation Computer Systems*, vol. 79, no. Part 1, pp. 243 – 246, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X17324913

[43] B. I. D. Messaoud, K. Guennoun, M. Wahbi, and M. Sadik, "Advanced Persistent Threat: New analysis driven by life cycle phases and their challenges," in *2016 International Conference on Advanced Communication Systems and Information Security (ACOSIS)*, Oct 2016, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ACOSIS.2016.7843932

[44] M. Bere, F. Bhunu-Shava, A. Gamundani, and I. Nhamu, "How Advanced Persistent Threats Exploit Humans," *International Journal of Computer Science Issues (IJCSI)*, vol. 12, no. 6, p. 170, 2015. [Online]. Available: https://www.researchgate.net/profile/Attlee_Gamundani/publication/301689293_How_ Advanced_Persistent_Threats_Exploit_Humans/links/57223ec208ae586b21d3e6c6.pdf

[45] (2017) Advanced Persistent Threat Activity Targeting Energy and Other Critical Infrastructure Sectors. United States Computer Emergency Readiness Team. Accessed 2017-12-09. [Online]. Available: https://www.us-cert.gov/ncas/alerts/TA17-293A

[46] A. Lemay, J. Calvet, F. Menet, and J. M. Fernandez, "Survey of publicly available reports on advanced persistent threat actors," *Computers and Security*,

vol. 72, no. Supplement C, pp. 26 – 59, 2018. [Online]. Available: http: //www.sciencedirect.com/science/article/pii/S0167404817301608

[47] M. Li, W. Huang, Y. Wang, and W. Fan, "The optimized attribute attack graph based on APT attack stage model," in *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, Oct 2016, pp. 2781–2785. [Online]. Available: https://doi.org/10.1109/CompComm.2016.7925204

[48] R. Luh, S. Schrittwieser, and S. Marschalek, "TAON: An Ontology-based Approach to Mitigating Targeted Attacks," in *Proceedings of the 18th International Conference on Information Integration and Web-based Applications and Services*, ser. iiWAS '16. New York, NY, USA: ACM, 2016, pp. 303–312. [Online]. Available: https: //doi.org/10.1145/3011141.3011157

[49] M. Marchetti, F. Pierazzi, M. Colajanni, and A. Guido, "Analysis of high volumes of network traffic for Advanced Persistent Threat detection," *Computer Networks*, vol. 109, no. Part 2, pp. 127 – 141, 2016, traffic and Performance in the Big Data Era. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128616301633

[50] I. Ghafir, V. Prenosil, J. Svoboda, and M. Hammoudeh, "A Survey on Network Security Monitoring Systems," in *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, Aug 2016, pp. 77–82. [Online]. Available: https://doi.org/10.1109/W-FiCloud.2016.30

[51] M. Yamada, M. Morinaga, Y. Unno, S. Torii, and M. Takenaka, "RAT-based malicious activities detection on enterprise internal networks," in *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*, Dec 2015, pp. 321–325. [Online]. Available: https://doi.org/10.1109/ICITST.2015.7412113

[52] A. Greco, A. Caponi, and G. Bianchi, "Facing lateral movements using widespread behavioral probes," in *2016 11th International Conference for Internet Technology*

and Secured Transactions (ICITST), Dec 2016, pp. 159–160. [Online]. Available: https://doi.org/10.1109/ICITST.2016.7856688

[53] I. Ghafir and V. Prenosil, *Proposed Approach for Targeted Attacks Detection*. Cham: Springer International Publishing, 2016, pp. 73–80. [Online]. Available: https://doi.org/10.1007/978-3-319-24584-3_7

[54] Z. Saud and M. H. Islam, "Towards Proactive Detection of Advanced Persistent Threat (APT) Attacks Using Honeypots," in *Proceedings of the 8th International Conference on Security of Information and Networks*, ser. SIN '15. New York, NY, USA: ACM, 2015, pp. 154–157. [Online]. Available: https://doi.org/10.1145/2799979.2800042

[55] M. Nawrocki, M. Wählisch, T. C. Schmidt, C. Keil, and J. Schönfelder, "A Survey on Honeypot Software and Data Analysis," *CoRR*, vol. abs/1608.06249, 2016. [Online]. Available: http://arxiv.org/abs/1608.06249

[56] C. NG, L. Pan, and Y. Xiang, *Honeypot Frameworks and Their Applications: A New Framework*, ser. SpringerBriefs on Cyber Security Systems and Networks. Springer Singapore, 2018. [Online]. Available: https://books.google.ca/books?id=YydaDwAAQBAJ

[57] A. O. Olagunju and F. Samu, "In search of effective honeypot and honeynet systems for real-time intrusion detection and prevention," in *Proceedings of the 5th Annual Conference on Research in Information Technology*, ser. RIIT '16. New York, NY, USA: ACM, 2016, pp. 41–46. [Online]. Available: https://doi.org/10.1145/2978178.2978184

[58] M. Auty, "Anatomy of an advanced persistent threat," *Network Security*, vol. 2015, no. 4, pp. 13 – 16, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1353485815300283

[59] J. Vukalović and D. Delija, "Advanced Persistent Threats - detection and defense," in *2015 38th International Convention on Information and Communication Technology,*

Electronics and Microelectronics (MIPRO), May 2015, pp. 1324–1330. [Online]. Available: https://doi.org/10.1109/MIPRO.2015.7160480

[60] G. G. Granadillo, J. Garcia-Alfaro, H. Debar, C. Ponchel, and L. R. Martin, "Considering technical and financial impact in the selection of security countermeasures against Advanced Persistent Threats (APTs)," in *2015 7th International Conference on New Technologies, Mobility and Security (NTMS)*, July 2015, pp. 1–6. [Online]. Available: https://doi.org/10.1109/NTMS.2015.7266480

[61] L. E. S. Jaramillo, "Detecting malware capabilities with FOSS: Lessons learned through a real-life incident," in *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*, June 2018, pp. 1–6.

[62] J. Timofte, "Intrusion Detection using Open Source Tools," *Informatica Economica*, vol. 0, no. 2, pp. 75–79, 2008. [Online]. Available: https://ideas.repec.org/a/aes/infoec/vxiiy2008i2p75-79.html

[63] I. Y. M. Al-Mahbashi, M. B. Potdar, and P. Chauhan, "Network security enhancement through effective log analysis using ELK," in *2017 International Conference on Computing Methodologies and Communication (ICCMC)*, July 2017, pp. 566–570.

[64] R. Mehresh, "Schemes for surviving advanced persistent threats," Ph.D. dissertation, PhD thesis, Faculty of the Graduate School of the University at Buffalo, State University of New York, 2013. [Online]. Available: https://www.cse.buffalo.edu/caeiae/documents/pdf/ruchika-mehresh-2013-dissertation.pdf

[65] H. M. Deylami, R. C. Muniyandi, I. T. Ardekani, and A. Sarrafzadeh, "Taxonomy of malware detection techniques: A systematic literature review," in *2016 14th Annual Conference on Privacy, Security and Trust (PST)*, Dec 2016, pp. 629–636. [Online]. Available: https://doi.org/10.1109/PST.2016.7906998

[66] D. Durham, "Mitigating exploits, rootkits and advanced persistent threats," in *2014*

IEEE Hot Chips 26 Symposium (HCS), Aug 2014, pp. 1–39. [Online]. Available: https://doi.org/10.1109/HOTCHIPS.2014.7478798

[67] A. K. Sood and R. J. Enbody, "Targeted Cyberattacks: A Superset of Advanced Persistent Threats," *IEEE Security Privacy*, vol. 11, no. 1, pp. 54–61, Jan 2013. [Online]. Available: https://doi.org/10.1109/MSP.2012.90

[68] N. Virvilis, D. Gritzalis, and T. Apostolopoulos, "Trusted Computing vs. Advanced Persistent Threats: Can a Defender Win This Game?" in *2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing*, Dec 2013, pp. 396–403. [Online]. Available: https://doi.org/10.1109/UIC-ATC.2013.80

[69] S. Torii, M. Morinaga, T. Yoshioka, T. Terada, and Y. Unno, "Multi-layered defense against advanced persistent threats (apt)," *Fujitsu Sci. Tech. J*, vol. 50, no. 1, pp. 52–59, 2014. [Online]. Available: http://www.fujitsu.com/global/documents/about/resources/publications/fstj/archives/vol50-1/paper09.pdf

[70] B. Binde, R. McRee, and T. J. O'Connor, "Assessing outbound traffic to uncover advanced persistent threat," *SANS Institute. Whitepaper*, 2011. [Online]. Available: https://www.sans.edu/student-files/projects/JWP-Binde-McRee-OConnor.pdf

[71] AT&T Cybersecurity. (2019) Open Source SIEM: OSSIM The World's Most Widely Used Solution. Accessed 2019-07-16. [Online]. Available: https://www.alienvault.com/products/ossim

[72] Wikipedia contributors, "OSSIM — Wikipedia, the free encyclopedia," 2019, accessed 2019-07-16. [Online]. Available: https://en.wikipedia.org/wiki/OSSIM

[73] M. Ask, P. Bondarenko, J. E. Rekdal, A. Nordbø, P. Bloemerus, and D. Piatkivskyi, "Advanced persistent threat (apt) beyond the hype," *Project report in IMT4582 Network*

security at Gjøvik University College during spring 2013, 2013. [Online]. Available: https://andynor.net/static/fileupload/434/S2_NetwSec_Advanced_Persistent_Threat.pdf

[74] E. Cole, *Advanced persistent threat. [electronic resource] : understanding the danger and how to protect your organization.* Syngress, 2013. [Online]. Available: https://library.books24x7.com/toc.aspx?bookid=51023

[75] W. Wesolowsky. (2019) Federated Security Module Kibana Plugin. Accessed 2019-07-14. [Online]. Available: https://github.com/rndrev/fsm_plugin

[76] T. Wilhelm, "Professional Penetration Testing," in *Professional Penetration Testing*. Boston: Syngress, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9781597494250000014

[77] M. Sikorski and A. Honig, *Practical Malware Analysis: A Hands-On Guide to Dissecting Malicious Software*. No Starch Press, 2012. [Online]. Available: https://books.google.ca/books?id=FQC8EPYy834C

[78] F. Roth. (2018) APT Simulator. [Online]. Available: https://github.com/NextronSystems/APTSimulator

[79] AlphaSOC. (2018) AlphaSOC Network Flight Simulator. Accessed 2018-09-16. [Online]. Available: https://github.com/alphasoc/flightsim

[80] (2014) Advanced Threat Detection and Response: Using Splunk Software to Defend Against Advanced Threats. Splunk Inc. Accessed 2018-08-20. [Online]. Available: https://www.splunk.com/web_assets/pdfs/secure/Splunk_for_APT_Tech_Brief.pdf

[81] C.-j. LU, Z. Heng, J.-y. LIU, R. ZHANG, Y.-k. CHEN, and Y.-g. YAO, "Network Security Log Analysis System Based on ELK," *DEStech Transactions on Computer Science and Engineering*, no. cece, 2017. [Online]. Available: http://www.dpi-proceedings.com/index.php/dtcse/article/download/14597/14114

[82] C. Gormley and Z. Tong, *Elasticsearch: The Definitive Guide: A Distributed Real-Time Search and Analytics Engine*. O'Reilly Media, 2015. [Online]. Available: https://books.google.ca/books?id=d19aBgAAQBAJ

[83] J. Turnbull, *The Logstash Book*. James Turnbull, 2014. [Online]. Available: https://books.google.ca/books?id=5dIEBwAAQBAJ

[84] M. Arunwan, T. Laong, and K. Atthayuwat, "Defensive performance comparison of firewall systems," in *2016 Management and Innovation Technology International Conference (MITicon)*, Oct 2016, pp. MIT–221–MIT–224.

[85] A. Thakur. (2015) Open source firewall implementation: replacing traditional firewall with open source. [Online]. Available: https://www.theseus.fi/handle/10024/96447

[86] W. Park and S. Ahn, "Performance Comparison and Detection Analysis in Snort and Suricata Environment," *Wireless Personal Communications*, vol. 94, no. 2, pp. 241–252, May 2017. [Online]. Available: https://doi.org/10.1007/s11277-016-3209-9

[87] B. Brumen and J. Legvart, "Performance analysis of two open source intrusion detection systems," in *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, May 2016, pp. 1387–1392.

[88] A. Alhomoud, R. Munir, J. P. Disso, I. Awan, and A. Al-Dhelaan, "Performance Evaluation Study of Intrusion Detection Systems," *Procedia Computer Science*, vol. 5, pp. 173 – 180, 2011, the 2nd International Conference on Ambient Systems, Networks and Technologies (ANT-2011) / The 8th International Conference on Mobile Web Information Systems (MobiWIS 2011). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050911003498

[89] (2018) Snort FAQ - README.sensitive_data. Accessed 2018-09-03. [Online]. Available: https://www.snort.org/faq/readme-sensitive_data

[90] V. Paxson, "Bro: a system for detecting network intruders in real-time," *Computer Networks*, vol. 31, no. 23, pp. 2435 – 2463, 1999. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128699001127

[91] (2014, 11) Why Choose Bro? Bro Network Security Monitor. Accessed 2018-08-22. [Online]. Available: https://www.bro.org/why_choose_bro.pdf

[92] T. Smith. (2016, 01) Sweet Security Part 2 – Creating a Defensible Raspberry Pi. The State of Security Blog. Accessed 2018-08-22. [Online]. Available: https://www.tripwire.com/state-of-security/security-data-protection/sweet-security-part-2-creating-a-defensible-raspberry-pi/

[93] R. Bray, D. Cid, and A. Hay, *OSSEC Host-Based Intrusion Detection Guide*. Elsevier Science, 2008. [Online]. Available: https://books.google.ca/books?id=h37q2q3wvcUC

[94] (2018) User Manual. Wazuh Inc. Accessed 2018-08-23. [Online]. Available: https://documentation.wazuh.com/current/user-manual/index.html

[95] T. Mark Russinovich. (2018, July) Sysmon v8.0. Accessed 2018-09-29. [Online]. Available: https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon

[96] Alberto Rodriguez. (2017, May) Using Wazuh to monitor Sysmon events. Accessed 2018-09-29. [Online]. Available: https://blog.wazuh.com/using-wazuh-to-monitor-sysmon-events/

[97] @SwiftOnSecurity. (2018, Jan) sysmon-config — A Sysmon configuration file for everybody to fork. Accessed 2018-09-29. [Online]. Available: https://github.com/SwiftOnSecurity/sysmon-config#sysmon-config--a-sysmon-configuration-file-for-everybody-to-fork

[98] (2019, Sept) Download Winlogbeat. Accessed 2019-09-21. [Online]. Available: https://www.elastic.co/downloads/beats/winlogbeat/

[99] M. I. Al-Saleh, F. M. AbuHjeela, and Z. A. Al-Sharif, "Investigating the detection capabilities of antiviruses under concurrent attacks," *International Journal of Information Security*, vol. 14, no. 4, pp. 387–396, Aug 2015. [Online]. Available: https: //doi.org/10.1007/s10207-014-0261-x

[100] R. Shams, M. Farhan, S. A. Khan, and F. Hashmi, "Comparing Anti-Spyware products — A different approach," in *2011 6th IEEE Joint International Information Technology and Artificial Intelligence Conference*, vol. 1, Aug 2011, pp. 75–80.

[101] J. Estublier, H. Verjus, and P.-Y. Cunin, "Building Software Federation," in *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA '2001, Las Vegas, USA*, 2001. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.12.8993&rep=rep1&type=pdf

[102] (2018) Installing pfSense. Rubicon Communications LLC. Accessed 2018-08-23. [Online]. Available: https://www.netgate.com/docs/pfsense/install/installing-pfsense.html

[103] (2018) Configuring the Snort Package. Rubicon Communications LLC. Accessed 2018-08-23. [Online]. Available: https://docs.netgate.com/pfsense/en/latest/ids-ips/setup-snort-package.html

[104] (2018) Download Ubuntu Server. Canonical Ltd. Accessed 2018-08-23. [Online]. Available: https://www.ubuntu.com/download/server

[105] (2018) Time Synchronization. Canonical Ltd. Accessed 2018-08-23. [Online]. Available: https://help.ubuntu.com/lts/serverguide/NTP.htm

[106] T. Smith. (2018) Getting Sweet Security. Accessed 2018-08-23. [Online]. Available: https://github.com/TravisFSmith/SweetSecurity/wiki

[107] (2018) Installation Guide. Wazuh, Inc. Accessed 2018-08-23. [Online]. Available: https://documentation.wazuh.com/current/installation-guide/index.html

[108] (2018) Installing Wazuh Server. Wazuh Inc. Accessed 2018-08-23. [Online]. Available: https://documentation.wazuh.com/current/installation-guide/installing-wazuh-server/wazuh_server_deb.html#wazuh-server-deb

[109] (2018) Install Elastic Stack with Debian packages. Wazuh Inc. Accessed 2018-08-23. [Online]. Available: https://documentation.wazuh.com/current/installation-guide/installing-elastic-stack/elastic_server_deb.html#elastic-server-deb

[110] (2018) Upgrading the Elastic Stack. Elastic. Accessed 2018-08-23. [Online]. Available: https://www.elastic.co/guide/en/elastic-stack/6.4/upgrading-elastic-stack.html

[111] (2018) Cisco SG350-10 10-Port Gigabit Managed Switch. Raspberry Pi Foundation. Accessed 2018-08-26. [Online]. Available: https://www.raspberrypi.org/downloads/noobs/

[112] SeppBender. (2017, Aug) Can I Install on Debian 9? Accessed 2018-08-26. [Online]. Available: https://github.com/TravisFSmith/SweetSecurity/wiki/FAQ

[113] cloudstrifeedge. (2018, Aug) For those who want to install this project to one single Raspberry Pi. Accessed 2018-08-26. [Online]. Available: https://github.com/TravisFSmith/SweetSecurity/issues/48

[114] (2018) Install Wazuh agent on Windows. Wazuh Inc. Accessed 2018-08-23. [Online]. Available: https://documentation.wazuh.com/3.x/installation-guide/installing-wazuh-agent/wazuh_agent_windows.html

[115] (2018) Register Agent. Wazuh Inc. Accessed 2018-08-23. [Online]. Available: https://documentation.wazuh.com/3.x/user-manual/agents/command-line/register.html

[116] (2018) Capabilities. Wazuh Inc. Accessed 2018-08-23. [Online]. Available: https://documentation.wazuh.com/3.x/user-manual/capabilities/index.html

[117] R. Cenit. (2018, Aug) Monitoring Windows System. Accessed 2018-08-26. [Online]. Available: https://groups.google.com/forum/?utm_medium=email&utm_source=footer#!msg/wazuh/27-9odTV4xA/0nv_A5ApAAAJ

[118] W. Wesolowsky. (2019) OSSEC Modified Default Configuration File. Accessed 2019-07-25. [Online]. Available: https://github.com/rndrev/FederatedSecurityModule/blob/master/configs/ossec.conf

[119] K. Elatov. (2016, Nov) pfsense logging with elk. Accessed 2018-08-23. [Online]. Available: http://elatov.github.io/2016/11/pfsense-logging-with-elk/

[120] M. Walter. (2018, Apr) ELK + PFSENSE 2/2. Accessed 2018-08-23. [Online]. Available: http://www.hs-x.ch/index.php/elasticsearch-informationen/20-elk-pfsense-2-2

[121] R. Verhoef. (2018) HoneyTrap Documentation. Accessed 2018-08-23. [Online]. Available: http://docs.honeytrap.io/docs/home/

[122] W. Wesolowsky. (2019) HoneyTrap Config and Script. Accessed 2019-07-22. [Online]. Available: https://github.com/rndrev/FederatedSecurityModule/tree/master/configs/honeytrap

[123] (2018) Cisco SG350-10 10-Port Gigabit Managed Switch. Cisco. Accessed 2018-08-23. [Online]. Available: https://www.cisco.com/c/en/us/support/switches/sg350-10-10-port-gigabit-managed-switch/model.html

[124] J. Gonzalez. (2019) Wazuh icon missing on left in Kibana after new install version 3.8 (current). Accessed 2019-03-05. [Online]. Available: https://groups.google.com/d/msg/wazuh/vwzEwtHp31Q/6Sxc531cBwAJ

[125] Wazuh Inc. (2018) Upgrading Wazuh. Accessed 2019-01-20. [Online]. Available: https://documentation.wazuh.com/current/installation-guide/upgrading/index.html

[126] (2018) Removal of Mapping Types. Elastic. Accessed 2018-08-23. [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/current/removal-of-types.html

[127] (2019) Federated Security Module Kibana Plugin. Accessed 2019-07-29. [Online]. Available: https://github.com/rndrev/fsm_plugin

[128] Elasticsearch. (2019) Elastic UI Framework - Tables. Accessed 2019-07-29. [Online]. Available: https://elastic.github.io/eui/#/display/tables

[129] A. Bhardwaj and S. Goundar, "A framework for effective threat hunting," *Network Security*, vol. 2019, no. 6, pp. 15 – 19, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1353485819300741

[130] M. N. S. Miazi, M. M. A. Pritom, M. Shehab, B. Chu, and J. Wei, "The design of cyber threat hunting games: A case study," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, July 2017, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ICCCN.2017.8038527

[131] W. Wesolowsky. (2019) apthunter. Accessed 2019-09-22. [Online]. Available: https://github.com/rndrev/apthunter

[132] H. Almohannadi, I. Awan, J. Al Hamar, A. Cullen, J. P. Disso, and L. Armitage, "Cyber threat intelligence from honeypot data using elasticsearch," in *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*, May 2018, pp. 900–906. [Online]. Available: https://doi.org/10.1109/AINA.2018.00132

[133] D. Anstee, "The great threat intelligence debate," *Computer Fraud & Security*, vol. 2017, no. 9, pp. 14 – 16, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1361372317300994

[134] F. Roth and T. Patzke. (2019) Generic Signature Format for SIEM Systems. Accessed 2019-09-29. [Online]. Available: https://github.com/Neo23x0/sigma

[135] Jose R. Cenit. (2019) YARA module implementation. Accessed 2019-07-16. [Online]. Available: https://github.com/wazuh/wazuh/issues/3357

[136] (2018) EICAR Test File Intended Use. Accessed 2018-09-26. [Online]. Available: http://www.eicar.org/86-0-Intended-use.html

[137] M. Barbero. (2017, Aug) Get Windows Defender logs on Wazuh Manager. Accessed 2018-09-26. [Online]. Available: https://groups.google.com/forum/#!topic/wazuh/2_b1KRVKT6o

[138] J. Andrea Bichsel. (2018, Sept) Review event logs and error codes to troubleshoot issues with Windows Defender Antivirus. Accessed 2018-09-26. [Online]. Available: https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-defender-antivirus/troubleshoot-windows-defender-antivirus

[139] R. Fielding and J. Reschke. (2014, June) Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. Accessed 2019-07-25. [Online]. Available: https://www.rfc-editor.org/info/rfc7231

[140] Wazuh Inc. (2018) Compatibility matrix. Accessed 2018-09-29. [Online]. Available: https://documentation.wazuh.com/current/installation-guide/compatibility_matrix/index.html

[141] Wikipedia contributors, "Raspberry Pi — Wikipedia, the free encyclopedia," 2019, accessed 2019-07-17. [Online]. Available: https://en.wikipedia.org/wiki/Raspberry_Pi

[142] Elasticsearch B.V. (2019) Encrypting communications in Elasticsearch. Accessed 2019-07-17. [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/current/configuring-tls.html

[143] (2014) Using the Bro SSL analyzer. Accessed 2018-09-23. [Online]. Available: https://www.bro.org/brocon2014/ssl_exercise.pdf

[144] T. Radivilova, L. Kirichenko, D. Ageyev, M. Tawalbeh, and V. Bulakh, "Decrypting SSL/TLS Traffic for Hidden Threats Detection," in *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, May 2018, pp. 143–146.

[145] (2018) Critical Stack Client not Fetching Lists. Accessed 2018-09-23. [Online]. Available: https://groups.google.com/forum/#!topic/security-onion/axOCfBgjva4

[146] W. Wesolowsky. (2019) Athabasca University Integrated Project Repository. Accessed 2019-07-14. [Online]. Available: https://github.com/rndrev/FederatedSecurityModule

[147] GitHub Inc. (2019) Fork a repo. Accessed 2019-07-14. [Online]. Available: https://help.github.com/en/articles/fork-a-repo

[148] T. Roes. (2015, Dec) Writing Kibana 4 Plugins – Basics. Accessed 2019-07-14. [Online]. Available: https://www.timroes.de/writing-kibana-4-plugins-basics

[149] Victor M. Alvarez. (2019) YARA - The pattern matching swiss knife for malware researchers (and everyone else). Accessed 2019-07-16. [Online]. Available: http://virustotal.github.io/yara/

[150] J. Bortniker, "Malware analysis for cyber-threat intelligence," Master's thesis, Utica College, 2016. [Online]. Available: http://0-search.proquest.com.aupac.lib.athabascau.ca/docview/1800532902?accountid=8408

[151] K. E. Himma and D. Dittrich, "Active response to computer intrusions," *SSRN Electronic Journal*, 2005. [Online]. Available: https://staff.washington.edu/dittrich/misc/handbook-arc.pdf

[152] Elasticsearch. (2019) Elastic Stack Features. Accessed 2019-07-14. [Online]. Available: https://www.elastic.co/products/stack

[153] (2019) SIEM Guide (Beta) [7.3] Overview. Accessed 2019-09-21. [Online]. Available: https://www.elastic.co/guide/en/siem/guide/current/siem-overview.html

APPENDIX A

Raw JSON Log Files

This appendix contains all the raw JSON log files from the ELK stack which are cited as reference examples for our test cases. We have included each one as a separate listing framed with line numbers on the left margin for easy reference.

https://www.overleaf.com/project/5d4117ad7f0be967b2695701

Listing Appendix A.1: Wazuh-alerts-3.x Log File Results for EICAR Test File

```
1  {
2    "_index": "wazuh-alerts-3.x-2018.09.27",
3    "_type": "wazuh",
4    "_id": "KAq_GGYBI-9SBrwSyob1",
5    "_score": 1,
6    "_source": {
7      "path": "/var/ossec/logs/alerts/alerts.json",
8      "manager": {
9        "name": "fmsrv"
10     },
11     "predecoder": {
12       "program_name": "WinEvtLog",
13       "timestamp": "2018 Sep 26 19:58:34"
14     },
15     "rule": {
16       "description": "Windows Defender: error when taking action on
             potentially unwanted software",
17       "id": "83002",
18       "mail": false,
19       "groups": [
20         "windows",
21         "windows_defender"
22       ],
23       "gdpr": [
24         "IV_35.7.d"
25       ],
26       "level": 7,
27       "firedtimes": 1
28     },
29     "decoder": {
```

```
30        "parent": "windows",
31        "name": "windows"
32      },
33      "id": "1538013513.436444",
34      "full_log": "2018 Sep 26 19:58:34 WinEvtLog: Microsoft-Windows-Windows
            Defender/Operational: INFORMATION(1117): Microsoft-Windows-Windows
            Defender: SYSTEM: NT AUTHORITY: win81-client.corp.local: Windows
            Defender has taken action to protect this machine from malware or
            other potentially unwanted software. For more information please see
            the following: http://go.microsoft.com/fwlink/?linkid=37020&name=
            Virus:DOS/EICAR_Test_File&threatid=2147519003&enterprise=0 \tName:
            Virus:DOS/EICAR_Test_File \tID: 2147519003 \tSeverity: Severe \
            tCategory: Virus \tPath: file:_C:\\Users\\Administrator\\Downloads\\
            Unconfirmed 965242.crdownload \tDetection Origin: Local machine \
            tDetection Type: Concrete \tDetection Source: Real-Time Protection \
            tUser: NT AUTHORITY\\SYSTEM \tProcess Name: C:\\Program Files (x86)\\
            Google\\Chrome\\Application\\chrome.exe \tAction: Quarantine \tAction
             Status: No additional actions required \tError Code: 0x80508023 \
            tError description: The program could not find the malware and other
            potentially unwanted software on this computer. \tSignature Version:
            AV: 1.275.1349.0, AS: 1.275.1349.0, NIS: 119.0.0.0 \tEngine Version:
            AM: 1.1.15200.1, NIS: 2.1.14600.4",
35      "location": "WinEvtLog",
36      "@timestamp": "2018-09-27T01:58:33.776Z",
37      "data": {
38        "id": "1117",
39        "srcuser": "NT AUTHORITY\\SYSTEM"
40      },
41      "agent": {
42        "id": "001",
43        "name": "win81-client"
44      }
45    },
46    "fields": {
47      "@timestamp": [
48        "2018-09-27T01:58:33.776Z"
49      ]
50    }
51  }
```

Listing Appendix A.2: HoneyTrap Log File Result for SSH Connection

```
1  {
2    "_index": "honeytrap",
3    "_type": "event",
4    "_id": "ca6e5cf3-873e-4db2-bd5e-45dd340c7e71",
5    "_version": 1,
6    "_score": null,
7    "_source": {
8      "category": "ssh",
9      "date": "2019-03-13T04:36:37.379975444Z",
10     "destination-ip": "192.168.1.137",
11     "destination-port": 8022,
12     "sensor": "services",
13     "source-ip": "192.168.1.102",
14     "source-port": 61190,
15     "ssh.password": "",
16     "ssh.sessionid": "bi48hiijuo3g00etibf0",
17     "ssh.username": "test",
18     "token": "bi47gvijuo3g00etibdg",
19     "type": "password-authentication"
20   },
21   "fields": {
22     "date": [
23       "2019-03-13T04:36:37.379Z"
24     ]
25   },
26   "sort": [
27     1552451797379
28   ]
29 }
```

Listing Appendix A.3: HoneyTrap Log File Result for Telnet Connection

```json
1   {
2     "_index": "honeytrap",
3     "_type": "event",
4     "_id": "ea4badc4-5a70-4a8d-bca9-f5c317beb6a3",
5     "_version": 1,
6     "_score": null,
7     "_source": {
8       "category": "telnet",
9       "date": "2019-03-13T04:07:06.608871804Z",
10      "destination-ip": "192.168.1.137",
11      "destination-port": 23,
12      "sensor": "services",
13      "source-ip": "192.168.1.102",
14      "source-port": 61156,
15      "telnet.sessionid": "bi483qijuo3g00etibeg",
16      "token": "bi47gvijuo3g00etibdg",
17      "type": "connect"
18    },
19    "fields": {
20      "date": [
21        "2019-03-13T04:07:06.608Z"
22      ]
23    },
24    "sort": [
25      1552450026608
26    ]
27  }
```

Listing Appendix A.4: Logstash-* Log File Result for HTTP Browser Connection

```
1  {
2    "_index": "logstash-2019.07.26",
3    "_type": "logs",
4    "_id": "-TzIK2wB2FezEx5T04ZU",
5    "_version": 1,
6    "_score": null,
7    "_source": {
8      "server_name": "www.google.com",
9      "status_code": "200",
10     "geoip_dst": {
11       "timezone": "America/Los_Angeles",
12       "ip": "172.217.3.164",
13       "latitude": 37.419200000000004,
14       "continent_code": "NA",
15       "city_name": "Mountain View",
16       "country_name": "United States",
17       "country_code2": "US",
18       "dma_code": 807,
19       "country_code3": "US",
20       "region_name": "California",
21       "location": {
22         "lon": -122.0574,
23         "lat": 37.419200000000004
24       },
25       "postal_code": "94043",
26       "region_code": "CA",
27       "longitude": -122.0574
28     },
29     "orig_h": "192.168.1.132",
30     "resp_p": "80",
31     "info_msg": "-",
32     "path": "/opt/nsm/bro/logs/current/http.log",
33     "uid": "CLKjEQ3uvmSNYcsz51",
34     "resp_mime_types": "-",
35     "trans_depth": "1",
36     "password": "-",
37     "orig_p": "49348",
38     "@version": "1",
39     "host": "tap",
40     "user_agent": "HttpBrowser/1.0",
41     "resp_fuids": "-",
42     "method": "HEAD",
43     "request_body_len": "0",
44     "geoip_src": {},
```

```
45    "proxied": "-",
46    "message": "1564102675.012333\tCLKjEQ3uvmSNYcsz51\t192.168.1.132\t49348\
          t172.217.3.164\t80\t1\tHEAD\twww.google.com\t/\t-\t1.1\tHttpBrowser
          /1.0\t0\t0\t200\tOK\t-\t-\t(empty)\t-\t-\t-\t-\t-\t-\t-\t-\t-\t-",
47    "orig_mime_types": "-",
48    "uri": "/",
49    "version": "1.1",
50    "tags": [
51      "(empty)",
52      "_geoip_lookup_failure"
53    ],
54    "referrer": "-",
55    "@timestamp": "2019-07-26T00:57:55.012Z",
56    "resp_h": "172.217.3.164",
57    "orig_fuids": "-",
58    "orig_filenames": "-",
59    "status_msg": "OK",
60    "resp_filenames": "-",
61    "response_body_len": "0",
62    "ts": "1564102675.012333",
63    "info_code": "-",
64    "username": "-"
65  }
66 }
```

Listing Appendix A.5: Logstash-* Log File Result for wget Connection

```
1  {
2    "_index": "logstash-2019.07.26",
3    "_type": "logs",
4    "_id": "BTzIK2wB2FezEx5T04dU",
5    "_version": 1,
6    "_score": null,
7    "_source": {
8      "server_name": "www.google.com",
9      "status_code": "200",
10     "geoip_dst": {
11       "timezone": "America/Los_Angeles",
12       "ip": "172.217.3.164",
13       "latitude": 37.419200000000004,
14       "continent_code": "NA",
15       "city_name": "Mountain View",
16       "country_name": "United States",
17       "country_code2": "US",
18       "dma_code": 807,
19       "country_code3": "US",
20       "region_name": "California",
21       "location": {
22         "lon": -122.0574,
23         "lat": 37.419200000000004
24       },
25       "postal_code": "94043",
26       "region_code": "CA",
27       "longitude": -122.0574
28     },
29     "orig_h": "192.168.1.132",
30     "resp_p": "80",
31     "info_msg": "-",
32     "path": "/opt/nsm/bro/logs/current/http.log",
33     "uid": "CCcSYm2JdVCUDx83ub",
34     "resp_mime_types": "-",
35     "trans_depth": "1",
36     "password": "-",
37     "orig_p": "49349",
38     "@version": "1",
39     "host": "tap",
40     "user_agent": "Wget/1.9+cvs-stable (Red Hat modified)",
41     "resp_fuids": "-",
42     "method": "HEAD",
43     "request_body_len": "0",
44     "geoip_src": {},
```

```
45      "proxied": "-",
46      "message": "1564102675.492290\tCCcSYm2JdVCUDx83ub\t192.168.1.132\t49349\
           t172.217.3.164\t80\t1\tHEAD\twww.google.com\t/\t-\t1.1\tWget/1.9+cvs-
           stable (Red Hat modified)\t0\t0\t200\tOK\t-\t-\t(empty)\t-\t-\t-\t-\t
           -\t-\t-\t-\t-",
47      "orig_mime_types": "-",
48      "uri": "/",
49      "version": "1.1",
50      "tags": [
51        "(empty)",
52        "_geoip_lookup_failure"
53      ],
54      "referrer": "-",
55      "@timestamp": "2019-07-26T00:57:55.492Z",
56      "resp_h": "172.217.3.164",
57      "orig_fuids": "-",
58      "orig_filenames": "-",
59      "status_msg": "OK",
60      "resp_filenames": "-",
61      "response_body_len": "0",
62      "ts": "1564102675.492290",
63      "info_code": "-",
64      "username": "-"
65    },
66    "fields": {
67      "@timestamp": [
68        "2019-07-26T00:57:55.492Z"
69      ]
70    }
71  }
```

```
1  {
2    "_index": "logstash-2019.07.26",
3    "_type": "logs",
4    "_id": "EzzIK2wB2FezEx5TOoWH",
5    "_version": 1,
6    "_score": null,
7    "_source": {
8      "server_name": "www.google.com",
9      "status_code": "200",
10     "geoip_dst": {
11       "timezone": "America/Los_Angeles",
12       "ip": "172.217.3.164",
13       "latitude": 37.419200000000004,
14       "continent_code": "NA",
15       "city_name": "Mountain View",
16       "country_name": "United States",
17       "country_code2": "US",
18       "dma_code": 807,
19       "country_code3": "US",
20       "region_name": "California",
21       "location": {
22         "lon": -122.0574,
23         "lat": 37.419200000000004
24       },
25       "postal_code": "94043",
26       "region_code": "CA",
27       "longitude": -122.0574
28     },
29     "orig_h": "192.168.1.132",
30     "resp_p": "80",
31     "info_msg": "-",
32     "path": "/opt/nsm/bro/logs/current/http.log",
33     "uid": "CWOjPr7EUH4Bwqcs7",
34     "resp_mime_types": "-",
35     "trans_depth": "1",
36     "password": "-",
37     "orig_p": "49350",
38     "@version": "1",
39     "host": "tap",
40     "user_agent": "Opera/8.81 (Windows NT 6.0; U; en)",
41     "resp_fuids": "-",
42     "method": "HEAD",
43     "request_body_len": "0",
44     "geoip_src": {},
```

```
45    "proxied": "-",
46    "message": "1564102675.672318\tCWOjPr7EUH4Bwqcs7\t192.168.1.132\t49350\
         t172.217.3.164\t80\t1\tHEAD\twww.google.com\t/\t-\t1.1\tOpera/8.81 (
         Windows NT 6.0; U; en)\t0\t0\t200\tOK\t-\t-\t(empty)\t-\t-\t-\t-\t
         -\t-\t-\t-",
47    "orig_mime_types": "-",
48    "uri": "/",
49    "version": "1.1",
50    "tags": [
51      "(empty)",
52      "_geoip_lookup_failure"
53    ],
54    "referrer": "-",
55    "@timestamp": "2019-07-26T00:57:55.672Z",
56    "resp_h": "172.217.3.164",
57    "orig_fuids": "-",
58    "orig_filenames": "-",
59    "status_msg": "OK",
60    "resp_filenames": "-",
61    "response_body_len": "0",
62    "ts": "1564102675.672318",
63    "info_code": "-",
64    "username": "-"
65  }
66 }
```

Listing Appendix A.7: Wazuh-alerts-3.x Log File Result for Windows Defender

```
 1  {
 2    "_index": "wazuh-alerts-3.x-2019.07.26",
 3    "_type": "wazuh",
 4    "_id": "rkROLGwB2FezEx5TkYRc",
 5    "_version": 1,
 6    "_score": null,
 7    "_source": {
 8      "data": {
 9        "win": {
10          "system": {
11            "version": "0",
12            "channel": "Microsoft-Windows-Windows Defender/Operational",
13            "message": "Windows Defender has detected malware or other
                   potentially unwanted software.",
14            "threadID": "1628",
15            "task": "0",
16            "opcode": "0",
17            "processID": "1500",
18            "providerGuid": "{11CD958A-C507-4EF3-B3F2-5FD9DFBD2C78}",
19            "eventID": "1116",
20            "providerName": "Microsoft-Windows-Windows Defender",
21            "level": "3",
22            "keywords": "0x8000000000000000",
23            "eventRecordID": "866",
24            "computer": "win81-client.corp.local",
25            "systemTime": "2019-07-26T03:21:32.381661900Z",
26            "severityValue": "WARNING"
27          },
28          "eventdata": {
29            "action ID": "9",
30            "type Name": "%%822",
31            "product Name": "%%827",
32            "state": "1",
33            "detection ID": "{548EE24B-21CD-4037-BC8D-4BB526719B4E}",
34            "category Name": "Tool",
35            "type ID": "0",
36            "execution Name": "%%813",
37            "status Code": "1",
38            "process Name": "C:\\Users\\Administrator\\Desktop\\
                   APTSimulator_pw_apt\\APTSimulator\\helpers\\7z.exe",
39            "path": "file:_C:\\inetpub\\wwwroot\\b.jsp-&gt;(SCRIPT0065)",
40            "origin ID": "1",
41            "pre Execution Status": "0",
42            "product Version": "4.10.209.0",
```

```
43          "action Name": "%%887",
44          "source ID": "3",
45          "error Code": "0x00000000",
46          "detection User": "CORP\\Administrator",
47          "severity Name": "High",
48          "additional Actions ID": "0",
49          "additional Actions String": "No additional actions required",
50          "execution ID": "1",
51          "post Clean Status": "0",
52          "source Name": "%%818",
53          "threat Name": "HackTool:JS/Jsprat",
54          "error Description": "The operation completed successfully.",
55          "category ID": "34",
56          "threat ID": "2147708292",
57          "severity ID": "4",
58          "fWLink": "http://go.microsoft.com/fwlink/?linkid=37020&amp;name=
                HackTool:JS/Jsprat&amp;threatid=2147708292&amp;enterprise=0",
59          "origin Name": "%%845",
60          "detection Time": "2019-07-26T03:21:32.272Z",
61          "engine Version": "AM: 1.1.15700.9, NIS: 2.1.14600.4",
62          "signature Version": "AV: 1.289.1202.0, AS: 1.289.1202.0, NIS:
                119.0.0.0"
63        }
64      }
65    },
66    "rule": {
67      "gdpr": [
68        "IV_35.7.d"
69      ],
70      "id": "62103",
71      "level": 12,
72      "groups": [
73        "windows",
74        "windows_defender"
75      ],
76      "mail": true,
77      "description": "Windows Defender: detected potentially unwanted
            software ",
78      "firedtimes": 10
79    },
80    "@timestamp": "2019-07-26T03:24:01.519Z",
81    "decoder": {
82      "name": "windows_eventchannel"
83    },
84    "path": "/var/ossec/logs/alerts/alerts.json",
```

```
 85       "agent": {
 86         "name": "win81-client",
 87         "ip": "192.168.1.132",
 88         "id": "001"
 89       },
 90       "manager": {
 91         "name": "fmsrv"
 92       },
 93       "id": "1564111441.4145480",
 94       "location": "EventChannel"
 95     },
 96     "fields": {
 97       "data.win.eventdata.detection Time": [
 98         "2019-07-26T03:21:32.272Z"
 99       ],
100       "@timestamp": [
101         "2019-07-26T03:24:01.519Z"
102       ]
103     },
104     "highlight": {
105       "agent.ip": [
106         "@kibana-highlighted-field@192.168.1.132@/kibana-highlighted-field@"
107       ]
108     },
109     "sort": [
110       1564111441519
111     ]
112 }
```

121

```
 1  {
 2    "_index": "wazuh-alerts-3.x-2019.07.27",
 3    "_type": "wazuh",
 4    "_id": "HJqsMWwB2FezEx5TJBY6",
 5    "_version": 1,
 6    "_score": null,
 7    "_source": {
 8      "data": {
 9        "win": {
10          "system": {
11            "version": "0",
12            "channel": "System",
13            "message": "Name resolution for the name ompute.pl timed out after
                   none of the configured DNS servers responded.",
14            "threadID": "1336",
15            "task": "1014",
16            "opcode": "0",
17            "processID": "328",
18            "providerGuid": "{1C95126E-7EEA-49A9-A3FE-A378B03DDB4D}",
19            "eventID": "1014",
20            "providerName": "Microsoft-Windows-DNS-Client",
21            "level": "3",
22            "keywords": "0x4000000010000000",
23            "eventRecordID": "14984",
24            "computer": "win81-client.corp.local",
25            "systemTime": "2019-07-27T04:24:19.659986800Z",
26            "severityValue": "WARNING"
27          },
28          "eventdata": {
29            "addressLength": "128",
30            "queryName": "ompute.pl",
31            "address": "02000000
                   C0A80169000000000000000000000000000000000000000000000000000000000000000
                   "
32          }
33        }
34      },
35      "rule": {
36        "id": "61109",
37        "level": 5,
38        "groups": [
39          "windows",
40          " windows_system"
41        ],
```

```
42        "mail": false,
43        "description": "Name resolution for the name ompute.pl timed out",
44        "firedtimes": 1
45      },
46      "@timestamp": "2019-07-27T04:24:19.415Z",
47      "decoder": {
48        "name": "windows_eventchannel"
49      },
50      "path": "/var/ossec/logs/alerts/alerts.json",
51      "agent": {
52        "name": "win81-client",
53        "ip": "192.168.1.132",
54        "id": "001"
55      },
56      "manager": {
57        "name": "fmsrv"
58      },
59      "id": "1564201459.26395832",
60      "location": "EventChannel"
61    },
62    "fields": {
63      "@timestamp": [
64        "2019-07-27T04:24:19.415Z"
65      ]
66    }
67  }
```

Listing Appendix A.9: pfSense-.x Log File Example for pfsense

```
 1  {
 2    "_index": "pfsense-2019.07.26",
 3    "_type": "doc",
 4    "_id": "x1vfLGwB2FezEx5TCb5T",
 5    "_version": 1,
 6    "_score": 2,
 7    "_source": {
 8      "evtid": "134",
 9      "iface": "re0",
10      "offset": "0",
11      "src_ip": "185.176.27.118",
12      "type": "syslog",
13      "message": "51,,,1000000118,re0,match,block,in,4,0x0,,243,31274,0,none
               ,6,tcp,44,185.176.27.118,192.168.3.151,47199,33894,0,S
               ,2016230250,,1024,,mss",
14      "prog": "filterlog",
15      "action": "block",
16      "ttl": "243",
17      "@version": "1",
18      "length": "44",
19      "tags": [
20        "PFSense",
21        "firewall",
22        "GeoIP"
23      ],
24      "proto_id": "6",
25      "src_port": "47199",
26      "geoip": {
27        "ip": "185.176.27.118",
28        "country_name": "Russia",
29        "country_code2": "RU",
30        "latitude": 55.7386,
31        "location": {
32          "lon": 37.6068,
33          "lat": 55.7386
34        },
35        "continent_code": "EU",
36        "country_code3": "RU",
37        "longitude": 37.6068
38      },
39      "rule": "51",
40      "direction": "in",
41      "@timestamp": "2019-07-26T06:01:39.000Z",
42      "reason": "match",
```

```
43      "flags": "none",
44      "data_length": "0",
45      "dest_ip": "192.168.3.151",
46      "dest_port": "33894",
47      "host": "192.168.1.1",
48      "tracker": "1000000118",
49      "id": "31274",
50      "proto": "tcp",
51      "tos": "0x0",
52      "ip_ver": "4"
53    },
54    "fields": {
55      "@timestamp": [
56        "2019-07-26T06:01:39.000Z"
57      ]
58    }
59  }
```

Listing Appendix A.10: pfSense-x Log File Example for Snort

```
1  {
2    "_index": "pfsense-2019.07.26",
3    "_type": "doc",
4    "_id": "vlztLGwB2FezEx5Ts7Kx",
5    "_version": 1,
6    "_score": null,
7    "_source": {
8      "evtid": "33",
9      "@version": "1",
10     "host": "192.168.1.1",
11     "@timestamp": "2019-07-26T06:17:40.000Z",
12     "message": "[1:2400002:2716] ET DROP Spamhaus DROP Listed Traffic
          Inbound group 3 [Classification: Misc Attack] [Priority: 2] {TCP}
          46.3.96.66:58604 -> 192.168.3.151:6663",
13     "type": "syslog",
14     "tags": [
15       "PFSense",
16       "firewall"
17     ],
18     "prog": "snort[82834]"
19   },
20   "fields": {
21     "@timestamp": [
22       "2019-07-26T06:17:40.000Z"
23     ]
24   }
25 }
```

APPENDIX B

Query DSL JSON Examples

This appendix contains some Query DSL JSON examples to return various information from the different indexes in the ELK stack which are cited as reference examples for our test cases. We have included each one as a separate listing framed with line numbers on the left margin for easy reference.

Listing Appendix B.1: Pfsense-* JSON Query for Blocked Firewall Events

```
1  GET /pfsense-*/_search
2  {
3    "_source": ["@timestamp", "src_ip", "src_port", "proto", "dest_ip", "
         dest_port", "message"],
4    "query": {
5      "bool": {
6        "must": [
7          {"match": {"action" : "block"}}
8        ],
9        "filter" : [
10         {"range": {
11               "@timestamp": {
12                 "gte" : "now-1d/d"
13                 }
14             }
15         }
16       ]
17     }
18   }
19 }
```

Listing Appendix B.2: Pfsense-* JSON Query for Blocked Snort Events

```
1  GET /pfsense-*/_search
2  {
3    "_source": ["@timestamp", "message"],
4    "query": {
5      "bool": {
6        "must": [
7          {"regexp": {"prog" : "snort.*"}}
8        ],
9        "filter" : [
10         {"range": {
11               "@timestamp": {
12                 "gte" : "now-1d/d"
13               }
14           }
15        }
16       ]
17     }
18   }
19 }
```

Listing Appendix B.3: Pfsense-* JSON Query for Blocked Firewall Login Attempts

```
1  GET /pfsense-*/_search
2  {
3  "_source": ["src_ip", "src_port", "proto", "dest_ip", "dest_port", "message
      "],
4    "query": {
5      "bool": {
6        "must" : [
7          { "term" : { "action" : "block" } },
8          { "term" : { "dest_port" : 22 } },
9          { "term" : { "dest_ip" : "192.168.1.1"} }
10         ],
11       "filter" : [
12         {"range": {
13               "@timestamp": {
14                 "gt" : "now-5m"
15                 }
16             }
17         }
18       ]
19     }
20   }
21 }
```

Listing Appendix B.4: HoneyTrap JSON Query for Non-Heartbeat Events

```
1  GET /honeytrap/_search
2  {
3   "_source" : ["date", "source-ip", "source-port", "destination-ip", "
       destination-port", "category", "type"],
4    "query": {
5      "bool": {
6        "must_not": [
7          {"match": {"category" : "heartbeat"}}
8        ],
9        "filter" : [
10         {"range": {
11               "date": {
12                 "gte" : "now-3d/d"
13               }
14            }
15        }
16       ]
17     }
18   }
19 }
```

Listing Appendix B.5: Wazuh-alerts-3.x-* JSON Query for Windows Defender Malware Detected

```
1  GET /wazuh-alerts-3.x-*/_search
2  {
3    "_source" : ["@timestamp", "agent.ip", "agent.name", "data.win.eventdata.
         threat Name", "data.win.system.message", "data.win.eventdata.path", "
         data.win.eventdata.category Name"],
4    "query": {
5      "bool": {
6        "must" : [
7          { "term" : { "rule.groups" : "windows_defender" } },
8          { "term" : { "data.win.eventdata.severity Name" : "Severe" } }
9          ],
10       "filter" : [
11         {"range": {
12               "@timestamp": {
13                 "gt" : "now-5d/d"
14                 }
15             }
16         }
17       ]
18     }
19   }
```

APPENDIX C

Raw Output and Log Files

This appendix contains miscellaneous log files and output that did not fit elsewhere in the report. It will contain raw log files and output stored as plain text. We have included each one as a separate listing framed with line numbers on the left margin for easy reference (especially because some lines are long so they will wrap).

Listing Appendix C.1: FlightSim Output Files

```
 1  C:\Users\Administrator\Desktop>flightsim-windows-amd64.exe run
 2
 3  AlphaSOC Network Flight SimulatorT v1.1.1 (https://github.com/alphasoc/
       flightsim)
 4  The IP address of the network interface is 192.168.1.132
 5  The current time is 26-Jul-19 22:24:16
 6
 7  Time Module Description
 8  ------------------------------------------------------------------
 9
10  22:24:16 c2-dns Starting
11  22:24:16 c2-dns Preparing random sample of current C2 domains
12  22:24:19 c2-dns Resolving ompute.pl
13  22:24:20 c2-dns Resolving www.dreadtraders.tk
14  22:24:21 c2-dns Resolving fpbqrouphaiti.com
15  22:24:22 c2-dns Resolving hosting123123.net23.net
16  22:24:23 c2-dns Resolving vcv.no-ip.biz
17  22:24:24 c2-dns Resolving dallena.biz
18  22:24:25 c2-dns Resolving playstation2online.com
19  22:24:26 c2-dns Resolving trioeffisentec.com
20  22:24:27 c2-dns Resolving caprichomob.pt
21  22:24:28 c2-dns Resolving livingwaterphotography.com
22  22:24:29 c2-dns Finished
23  22:24:29 c2-ip Starting
24  22:24:29 c2-ip Preparing random sample of current C2 IP:port pairs
25  22:24:29 c2-ip Connecting to 185.225.139.63:443
26  22:24:30 c2-ip Connecting to 103.75.118.230:447
27  22:24:31 c2-ip Connecting to 188.64.170.198:1604
28  22:24:32 c2-ip Connecting to 188.19.102.8:1604
29  22:24:33 c2-ip Connecting to 86.108.66.41:82
```

```
30  22:24:34 c2-ip Connecting to 45.8.229.113:447
31  22:24:35 c2-ip Connecting to 193.37.212.106:447
32  22:24:36 c2-ip Connecting to 35.225.217.216:1604
33  22:24:37 c2-ip Connecting to 174.52.228.60:54984
34  22:24:38 c2-ip Connecting to 125.209.246.112:443
35  22:24:39 c2-ip Finished
36  22:24:39 dga Starting
37  22:24:39 dga Generating list of DGA domains
38  22:24:39 dga Resolving skphukp.com
39  22:24:40 dga Resolving skphukp.top
40  22:24:41 dga Resolving skphukp.biz
41  22:24:42 dga Resolving wnmpgyz.com
42  22:24:43 dga Resolving wnmpgyz.top
43  22:24:44 dga Resolving wnmpgyz.biz
44  22:24:45 dga Resolving azpxerl.com
45  22:24:46 dga Resolving azpxerl.top
46  22:24:47 dga Resolving azpxerl.biz
47  22:24:48 dga Resolving cerfstj.com
48  22:24:49 dga Resolving cerfstj.top
49  22:24:50 dga Resolving cerfstj.biz
50  22:24:51 dga Resolving obphjbp.com
51  22:24:52 dga Resolving obphjbp.top
52  22:24:53 dga Resolving obphjbp.biz
53  22:24:54 dga Resolving ynqsmph.com
54  22:24:55 dga Resolving ynqsmph.top
55  22:24:56 dga Resolving ynqsmph.biz
56  22:24:57 dga Resolving oeppibd.com
57  22:24:58 dga Resolving oeppibd.top
58  22:24:59 dga Resolving oeppibd.biz
59  22:25:00 dga Resolving cqwmbxd.com
60  22:25:01 dga Resolving cqwmbxd.top
61  22:25:02 dga Resolving cqwmbxd.biz
62  22:25:03 dga Resolving xsmanpp.com
63  22:25:04 dga Resolving xsmanpp.top
64  22:25:05 dga Resolving xsmanpp.biz
65  22:25:06 dga Resolving ykauobe.com
66  22:25:07 dga Resolving ykauobe.top
67  22:25:08 dga Resolving ykauobe.biz
68  22:25:09 dga Finished
69  22:25:09 hijack Starting
70  22:25:09 hijack Resolving alphasoc.com via ns1.sandbox.alphasoc.xyz
71  22:25:09 hijack Success! DNS hijacking is possible in this environment
72  22:25:10 hijack Finished
73  22:25:10 scan Starting
74  22:25:10 scan Preparing random sample of RFC 5737 destinations
```

```
75   22:25:10 scan Port scanning 192.0.2.137
76   22:25:11 scan Port scanning 192.0.2.64
77   22:25:11 scan Port scanning 192.0.2.0
78   22:25:12 scan Port scanning 192.0.2.177
79   22:25:13 scan Port scanning 192.0.2.88
80   22:25:13 scan Port scanning 192.0.2.144
81   22:25:14 scan Port scanning 192.0.2.31
82   22:25:15 scan Port scanning 192.0.2.209
83   22:25:15 scan Port scanning 192.0.2.115
84   22:25:16 scan Port scanning 192.0.2.75
85   22:25:17 scan Port scanning 198.51.100.100
86   22:25:17 scan Port scanning 198.51.100.72
87   22:25:18 scan Port scanning 198.51.100.137
88   22:25:19 scan Port scanning 198.51.100.159
89   22:25:19 scan Port scanning 198.51.100.70
90   22:25:20 scan Port scanning 198.51.100.75
91   22:25:20 scan Port scanning 198.51.100.177
92   22:25:21 scan Port scanning 198.51.100.142
93   22:25:22 scan Port scanning 198.51.100.99
94   22:25:22 scan Port scanning 203.0.113.29
95   22:25:23 scan Port scanning 203.0.113.141
96   22:25:24 scan Port scanning 203.0.113.45
97   22:25:24 scan Port scanning 203.0.113.168
98   22:25:25 scan Port scanning 203.0.113.70
99   22:25:26 scan Port scanning 203.0.113.0
100  22:25:26 scan Port scanning 203.0.113.153
101  22:25:27 scan Port scanning 203.0.113.208
102  22:25:28 scan Port scanning 203.0.113.67
103  22:25:28 scan Port scanning 203.0.113.41
104  22:25:29 scan Finished
105  22:25:29 sink Starting
106  22:25:29 sink Preparing random sample of current sinkhole IP:port pairs
107  22:25:29 sink Finished
108  22:25:29 spambot Starting
109  22:25:29 spambot Preparing random sample of Internet mail servers
110  22:25:32 spambot Connecting to mx.tb.ukmail.iss.as9143.net:25
111  22:25:33 spambot Connecting to etb-4.mail.tiscali.it:25
112  22:25:34 spambot Connecting to nam.olc.protection.outlook.com:25
113  22:25:35 spambot Connecting to mx3.hanmail.net:25
114  22:25:36 spambot Connecting to mxs.mail.ru:25
115  22:25:37 spambot Connecting to mx1.nate.com:25
116  22:25:38 spambot Connecting to mx-eu.mail.am0.yahoodns.net:25
117  22:25:39 spambot Connecting to mx3.bol.com.br:25
118  22:25:40 spambot Connecting to mx.yandex.ru:25
119  22:25:41 spambot Connecting to smtpz4.laposte.net:25
```

```
120  22:25:42 spambot Finished
121  22:25:42 tunnel Starting
122  22:25:42 tunnel Preparing DNS tunnel hostnames
123  22:25:42 tunnel Resolving qziuxryfgzwgkhzwnkwrbrniubavdb.sandbox.alphasoc.
         xyz
124  22:25:43 tunnel Resolving yemohlssirsxrdsyfxujnupzqyihdi.sandbox.alphasoc.
         xyz
125  22:25:44 tunnel Resolving ancjdxcqklnwwfvagkacilqgiqooxn.sandbox.alphasoc.
         xyz
126  22:25:45 tunnel Resolving gtjxinrzhvakwurxptqnlotuyofzlh.sandbox.alphasoc.
         xyz
127  22:25:46 tunnel Resolving isfgcritktpsmckmukiiiqjaenfdne.sandbox.alphasoc.
         xyz
128  22:25:47 tunnel Resolving okroozhaibigqbkbxwthkduwhrcbfx.sandbox.alphasoc.
         xyz
129  22:25:48 tunnel Resolving vvqwurldbvgcaocbapgxzqmgnzovat.sandbox.alphasoc.
         xyz
130  22:25:49 tunnel Resolving uyjeynxkwagofopydxlpztjgybsjbb.sandbox.alphasoc.
         xyz
131  22:25:50 tunnel Resolving tkkabqksmgdvqftiomrhvnpnlfemwx.sandbox.alphasoc.
         xyz
132  22:25:51 tunnel Resolving cvcnpeczpqhntrgppjnodymlokkoat.sandbox.alphasoc.
         xyz
133  22:25:52 tunnel Finished
134
135  All done! Check your SIEM for alerts using the timestamps and details above.
```