AUTOMATED DEFENSE AGAINST TARGETED ATTACKS USING SUSPICIOUSNESS TRACKING

A Thesis presented to

the Faculty of the Graduate School

at the University of Missouri

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

---

by

TRAVIS NEELY

Dr. Prasad Calyam, Thesis Supervisor

MAY 2021

The undersigned, appointed by the Dean of the Graduate School, have examined the thesis entitled:

AUTOMATED DEFENSE AGAINST TARGETED ATTACKS
USING SUSPICIOUSNESS TRACKING

presented by Travis Neely,
a candidate for the degree of Master of Science and hereby certify that, in their opinion, it is worthy of acceptance.

Dr. Prasad Calyam

Dr. Khaza Anuarul Hoque

Dr. Yaw Adu-Gyamfi

ACKNOWLEDGMENTS

I would like to thank Dr. Prasad Calyam as my advisor. He has shared his excitement and enthusiasm for this field with me, and I am privileged to have been able to work under his guidance. He has helped me to grow as a developer, engineer, and researcher with his depth of knowledge and understanding of the field. I sincerely hope that I may always share the same level of excitement and elation for the field as he does.

Next, I would also like to thank Roshan Lal Neupane, Mark Vassell, and Nishant Chettri who all worked tirelessly with me into many late nights in order to advance our research and develop something new and unique. Our combined efforts truly pushed our research forward.

Additionally, I give many thanks to all the members of VIMAN lab, who have shared their time and resources with me selflessly. Their dedication to the lab and their research is beyond measure and each one of them brings so much to the advancement of their respective fields. Likewise, all the professors who have guided me and lent me their wisdom and knowledge; have lifted me up beyond where I would otherwise be.

Lastly, I truly grateful to all of my family, my mom, dad, and brothers for their support. Especially, my loving wife and my wonderful little boy Simon who fills my life with joy every single moment.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

ABSTRACT

Cloud ecosystems, technologies, and paradigms have transformed our world in recent years revolutionizing supply chains, healthcare, energy distribution, as well as our home functions. With everything in our lives so interconnected in these cloud systems, they are now prime targets for targeted attacks such as Advanced Persistent Threats (APTs). The targeted attacks lead to exposure of sensitive data (data exfiltration) as well as stolen computing resources (resource exfiltration).

In this thesis, we present a novel methodology, which we call ADAPTs (Automated Defense of Advanced Persistent Threats), developed to assist in defending cloud systems against APTs. We show how ADAPTs can be extended to defend against other targeted attacks such as DDoS and cryptojacking. Using an open cloud testbed, we mimic multiple cloud systems, monitor network traffic between them, and generate a suspiciousness score for devices connected to said cloud networks. Using the suspiciousness scores, we demonstrate how we determine what work that device on the network is participating in, be it data exfiltration, resource exfiltration, or some other unwanted practice. Using these suspiciousness scores, we block the attacks while they are taking place and using pretense, continue to allow the attackers to believe their attack is successful. Our experimental results show how ADAPTs tricks attackers to continue to waste their own resources on an attack which is fruitless, while also protecting the targeted system by keeping the related services working as expected for actual users.

Chapter 1

Introduction

1.1 Security in Cloud-hosted Applications and Services

Cloud-hosted applications and services have been distributed all across the globe. These services and application systems encompass many sectors from manufacturing systems to the financial markets, as well as to individual homes and personal devices. Such wide and broad facing systems have their advantages; such as allowing instant access and accessibility from just about anywhere in the modern world, provide instant access to information, and entertainment. However, they also open themselves up to mischievous and detrimental attacks from larger, well-funded hacker groups and adversary governments. For such groups there is much to be had: from reputation and monetary gains by having targeted customers lose faith in valuable applications and related services, to power and political influence to threaten governance. Attacks from such groups can cause any number of issues including Loss of Confidentiality (LoC), Loss of Integrity (LoI), and Loss of Availability (LoA). This culminates in the loss of income, undesired exposure of confidential, or personally identifiable information.

1.2 Targeted Attacks

1.2.1 Advanced Persistent Threats (APTs)

One of the main attack vectors for these groups is Advanced Persistent Threats (APTs). These form of attacks are characterized by computer viruses/trojans/worms, which hide on network devices (personal computers, servers, mobile devices) while waiting in attempts to exfiltrate data from within the network, to devices outside the network. These are long-term attacks, intended to go unnoticed for as long as possible so that maximum exfiltration may occur. Many APTs will attempt to exploit both Zero-Day attacks (faults in software which have not been discovered by the application developers or hardware vendors and can be exploited) as well as human error (e.g., the curiosity of finding a flash drive in a parking lot, taking it, and attempting to use it). A combination of these methods are also used for initially breaking into an application system as well as spreading through an enterprise infrastructure [1].

Since the APTs attempt to be stealthy and commonly use Zero-Day attacks, detection is difficult with existing technologies. Many of these attacks go unnoticed for years, such as Red October, which was active for over five years [2]. With such long lasting and subtle attacks, new methods and technologies are needed which can detect these attacks quickly and defend against them before any further long term damage or exfiltration can be accomplished.

1.2.2 Distributed Denial of Service (DDoS)

Compared to the new and advanced attack techniques, such as APTs, invented every year, Distributed Denial of Service (DDoS) attacks are ancient predecessors, the fist known attack occurring on September 12th, 1996 [3]. Despite there being many years of research on DDoS attacks and a variety of defenses against them, it still behooves us to experiment and

test new defensive designs and strategies as the attacks are still a common occurrence. In 2020 alone there were 10.8 million DDoS attacks with an estimated 15.4 million by 2023 [4]. As these attacks continue to grow in frequency, duration, and scale, perhaps becoming some sort of Massively Distributed Denial of Service attacks, we need continued advances in countermeasures in order to keep up with these attacks else we potentially experience Loss of Availability in services. Countermeasures which allow for faster, more accurate, and robust detection, as well as being affordable will continue to be in high demand. Novel approaches like our past Dolus work [5] are prime examples of how we can better tackle such attacks.

1.2.3 Advanced Persistent Mining (APM) and Cryptojacking

There has been much enthusiasm recently for cryptocurrencies. The volatile yet surprisingly expensive various crypto coins such as Bitcoin and Etherium means that someone who participates in the crypto market could potentially make a fortune. An estimate showed in 2017 that a popular torrent site could be earning upwards of $12,000 a month using a cyptominer which ran in their user's browser [6], and the value of many cryptocurrencies have vastly grown since then. The craze has spawned all manner of different means of mining or obtaining these coins, several such methods are nefarious in nature. Attackers will use attacks such as Advanced Persistent Mining (a variation of APT) and Cryptojacking to 'mine' coins on systems which they do not own, thus exfiltrating resources from said systems. This resource exfiltration can be highly profitable as there is little cost to the attacker and the burden of mining the coins is left at the expense of the owner of the attacked systems [7].

1.3 Novel Defense Against Targeted Attacks

In this paper, our goal for targeted attack defense is to detect which devices may be infected by or participating in a targeted attack by determining which devices are most suspicious while also tracking for data or resource exfiltration. Once a device is suspected of being a participant in an attack, the device's traffic can be rerouted so that it does not leave the enterprise network, but can instead be analyzed to determine what is being exfiltrated or what has been compromised.

1.3.1 Pretense Theory

The pretense is designed to create stimulus from the target side that matches the initial expectation of an attacker that a high-value target has not yet been compromised through an automated bot activity. Pretense theory concepts from [8] motivate us to address the issue of how a cognitive agent can present a pretense world, which is different from the real world using the following four steps:

(a) The basic assumption(s) or premise(s) that is used by a pretender on *what* is being pretended.

(b) Inferential elaboration which details of what goes into or what actually happens in the process of pretense.

(c) Appropriate behavior production which answers the question of whether the pretender was successful on the audience being tricked.

(d) Balancing and steering the effects of pretense.

1.3.2 Suspiciousness Scores

In order to detect possible targeted attacks and identify systems, which have been compromised we use a concept called *Suspiciousness Scores* [9]. A Suspiciousness Score is assigned to each device on or off the network. Each device will be assigned a score which is calculated based upon it's total number of network traffic destinations contacted, total number of packets sent outside the host network, and total number of bytes transmitted. Using these scores we are able to form a baseline for the entire network. Consequently, devices which are 'suspicious' will stand out with higher scores. Suspiciousness Scores do not only have to be calculated for internal devices, but can also be calculated for external devices and domains; therefore an external device or domain, which we find to be suspicious, can later be blocked from devices on the internal network. In addition, we use an allowlist which allows us to ignore scores for traffic which we deem as benign.

1.3.3 Policy Updating and Threat Sharing

Once an attack has been detected and a pretense initiated it becomes necessary to put measures into place which will properly defend against similar attacks in the future. Automated policy updating allows for suspicious devices which have been used for attacks to be blocklisted to prevent them from launching further attacks. These policy updates are then shared across the SDxI cloud network further preventing similar attacks on other systems which were not original targets of the attack.

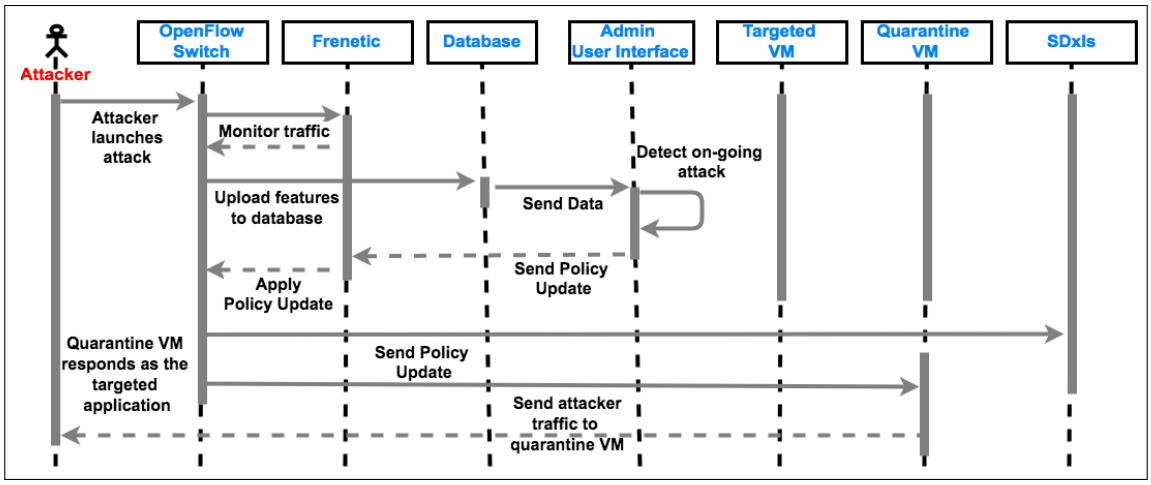

Figure 1.1: Sequence Diagram for system attack detection, quarantine, pretense initiation and mitigation.

1.4 Thesis Outline

The remainder of this thesis is organized as such: In Chapter 2, we describe the thesis background and review existing work and literature, which gives context and meaning to our proposed solution. In Chapter 3, we elaborate upon our solution's methodology and approach while providing details of each feature. Chapter 4 evaluates the effectiveness of the solution, findings, and performance evaluations. Chapter 5 discusses future work and provides an example of how the research has been extended. Chapter 6 concludes the thesis.

Chapter 2

Background and Related Work

2.1 Advanced Persistent Threats

APTs are long-term attacks [2, 10, 11, 12, 13, 14, 15]. They affect a target in four stages: *preparation, access, resident, and harvest* [16]. In the preparation stage, attackers apply a reconnaissance tactic through social engineering (e.g., via social networks) to bootstrap the attack [1]. Once the attack is bootstrapped, attackers identify a vulnerability, and/or a vulnerable target and send malwares either through email (e.g., spear phishing) or through third-party software/service (e.g., watering-hole attack) in the access stage. Subsequently, the malwares gain access to the control and command center, and spread across other targets in the resident stage; which is a slow and a stealthy phenomenon. Finally, in the harvest stage, attackers extract any vital information in an on-going fashion for extended periods of time.

Techniques to detect APTs have been of interest to the community [17, 18, 19, 19, 9, 20, 21, 22, 23, 24, 25, 26]. This includes finite angular state velocity machines and vector mathematics to model benign versus attack traffic, allowing a network operator to easily view the differences [23], assessing the outbound network traffic [9, 20], using honeypots [27]

and using distributed computing [26]. Another APT detection technique is based on a ranking system where all internal hosts are ranked based on number of bytes sent outside the network, number of data transfers initiated to an entity outside the network, and number of distinct destinations contacted outside the network per host [9]. Yet another APT detection technique is to monitor attack traffic using a detector in an enterprise network [24].

2.2 Prior Work on Countermeasures

Potential countermeasures against APTs are discussed in [1, 16, 28, 29, 30, 31, 32, 33]. Defense strategies include: (a) running routine software updates to avoid backdoors, bugs and vulnerabilities; (b) strengthening network access control and monitoring services; (c) enabling strict Internet access policies; and (d) dropping encrypted traffic from unknown hosts. Similarly, authors in [1] discuss a number of counter measures against APTs including training users about social engineering attacks, blocklisting hosts, dropping packets, etc. Furthermore, SDN-based defense [31] involves: (i) defining and maintaining a network baseline to identify any deviation from the baseline through analytical tools, and (ii) updation of flow policies for (re)directing and blocking traffic in any of the network segments. A framework that realizes such an SDN-based defense is discussed in [32]. In a similar vein, authors in [33] provide a sandbox environment using which a security professional can emulate the propagation of APTs across an enterprise network environment.

Chapter 3

ADAPTs Solution for Combating Targeted Attacks

3.1 Methodology and System Overview

Our novel ADAPTs (Automated Defense from Advanced Persistent Threats) system was originally designed to automatically defend against APTs and was later expanded to defend against DDoS and APM/Cryptojacking attacks. Building upon by our previous work, Dolus, ADAPTs uses pretense theory in child play to combat targeted attacks within an organization albeit with changes based upon the novel designed elaborated upon in sections 3.2, 3.4, and 3.5. ADAPTs consists of: (1) a Suspiciousness Score-based detection mechanism, which is robust against the threshold evasion problem; (2) internal quarantine VMs (iQVMs), which are a minimal version of honeypots to mimic hosts internal to an organization, along with performance/topology views to aid network administrators; (3) a coordination mechanism driven by enterprise defense policies to share threat intelligence about APTs among hosts; and (4) network policy update mechanism to mitigate attack spreading based on coordinated intelligence using iQVMs. We outline each one these mechanisms/components in the remainder of this chapter.

3.2 Suspiciousness Score-based Thresholds

Inspired by the work of authors in [9] to identify hosts exhibiting suspiciousness in a network, we propose a Suspiciousness Score (*SS*) in a similar vein. We calculate *SS* based on captured network traces (.pcap) using three main features: *destinations (dst)*, *flows*, and *bytes*.

Value	**Description**
switch_id	ID of the switch which received the frame
trace_id	ID for the trace under consideration
frame_number	Order in which the frame was received
frame_time	Unix timestamp at which the frame was received
frame_time_relative	Unix timestap at which the frame was received relative to last received frame
frame_protocols	Protocols used in the frame
frame_len	Size of the frame in bytes
ip_src	Source IP of the frame
ip_dst	Destination IP of the frame

Table 3.1: Features captured from a network trace.

Table 3.1 shows the list of values/features captured in network traces. For each packet, a trace_id *t* is assigned. For each *t*, we perform the following: the features are normalized and weighted based upon their targeted suspiciousness (further defined later in this subsection) and their combined Root Mean Square Error (RMSE) values are calculated. Using the RMSE values, we calculate the Suspiciousness Scores of each device as follows. The *Min* and *Max* values (below) are assumptions made per device type on what one may expect the minimum and maximum values to be on the type of network and traffic expectations.

Destination suspiciousness for trace t:

$$w_{dst} * dst_i = \frac{numDst_i - numDistMin_i}{numDstMax_i - numDstMin_i}; w_{dst} \in [0.0, 1.0] \tag{3.1}$$

Flow suspiciousness for trace t:

$$f_{dst} * flows_i = \frac{numFlows_i - numFlowsMin_i}{numFlowsMax_i - numFlowsMin_i} f_{dst} \in [0.0, 1.0] \tag{3.2}$$

Bytes suspiciousness for trace t:

$$b_{dst} * bytes_i = \frac{numBytes_i - numBytesMin_i}{numBytesMax_i - numBytesMin_i} b_{dst} \in [0.0, 1.0] \tag{3.3}$$

Device suspiciousness for trace t is based on equations 3.1, 3.2 and 3.3 as shown below.

$$ss_i = \sqrt{\frac{dst_i^2 + flows_i^2 + bytes_i^2}{3}} \tag{3.4}$$

Note that for each device on the network i we calculate a Suspiciousness Score and the overall network suspiciousness for trace t is calculated based on ss for each individual device (equation 3.4) that is connected. That is, the sum of all ss for each devices on the network n is the *overall network suspiciousness SS* for that particular t.

$$SS_t = \sqrt{\frac{(ss_1^2 + ss_2^2 + ss_3^2 + ... + ss_n^2)}{n}} \tag{3.5}$$

Relative change in device i's suspiciousness score on new traffic t is simply given by equation 3.6

$$\Delta ss_{i_t} = \frac{ss_{i_t} - \sqrt{\frac{ss_{i_1}^2 + ss_{i_2}^2 + ... + ss_{i_{t-1}}^2}{t-1}}}{\sqrt{\frac{ss_{i_1}^2 + ss_{i_2}^2 + ... + ss_{i_{t-1}}^2}{t-1}}} \tag{3.6}$$

In equations 3.1, 3.2 and 3.3, we assume a weight parameter i.e., w_{dst} w_{flows} w_{bytes} to be

equal to 1 in a general case of SS calculations. As shown later through experiment findings in 4.3.2, assigning suitable weights for a variety of suspicious traffic can maximize attack detection accuracy, as opposed to the general case. Consequently, we extend SS calculations as detailed in Algorithm 1 by introducing a novel concept of Targeted Suspiciousness Scores for specific traffic types that a system/network administrator would like to classify as suspicious. We remark that system/network administrators could allowlist certain traffic types, however none of the devices on the network will be entirely allowlisted. Thus, by using a targeted suspiciousness scoring for certain traffic types (e.g., for suspected APM-like traffic) can still be effective to detect malicious activities, even when allowlisting is performed for legitimate user traffic.

Input: $devices \in [1 \dots n]$ = array of all devices on the network,
$maxTrace$ = Maximum number of packet traces to be evaluated,
t = current trace being evaluated
Result: Targeted Suspiciousness Scores calculated for each network device after traffic analysis

```
1  function calcDst()
2  function calcFlows()
3  function calcBytes()
4  function calculateTargetedSuspiciousness(device_i)
5      calcDst(device_i, w_dst);
6      calcFlows(device_i, w_flows);
7      calcBytes(device_i, w_bytes);
8  end
9  function calculateNetworkSuspiciousness()
10     do
11         for each device_i;
12         calculateTargetedSuspiciousness(device_i);
13     while t <= maxTrace;
14 end
```

Algorithm 1: Targeted Suspiciousness Score Calculations

3.3 Internal Quarantine VMs (iQVMs)

VMs which are internal to an organization and which implement minimal versions of honeypot-like hosts are iQVMs. iQVMs are present in the network and their Suspiciousness Scores are monitored continuously. These are also the hosts that play the game of pretense i.e., they create a false notion of *high-value targets within an organization with sensitive data* to the external world. An attacker is lured to attack iQVMs first; they maintain pretense by sending data similar to what a host with sensitive data would send. Apart from monitoring the data sent out of iQVMs, they also add weights to the calculated Suspiciousness Scores, overcoming the threshold evasion problem.

To simplify the process of monitoring iQVMs and other hosts effectively, we develop a user interface as part of ADAPTs. It allows the administrator a more robust monitoring of the network with views separated based on the various requirements: devices connected to the network, blocklisted IPs, metrics, as well as any other requirements of the administrator. The user interface is developed using the traditional LAMP stack (Linux OS, Apache Web Server, MySQL, PHP), with views specifically built for ADAPTs including the following:

1. SS per device (see Figure 3.1) or for the overall network can be viewed in temporal fashion (see Figure 3.2). Moreover, when a suspiciousness score of a blocklisted device is shown to be above a certain threshold, an administrator can block all traffic from that device to the network or take an appropriate action.

2. Upload policy view: This view on the user interface enables administrators to push NetKAT-based policies [34] to a centralized database, which stores device configurations, thresholds, policies maintained by the organization. Interfaces are provided to select a specific device and a corresponding NetKAT policy to affect that device as shown in Figure 3.3.

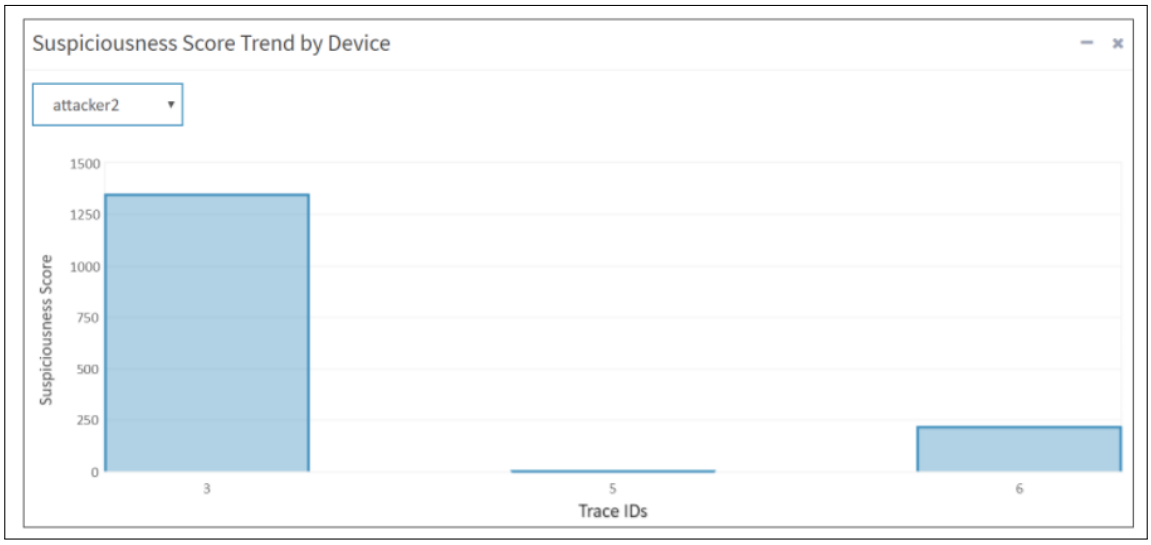

Figure 3.1: Suspiciousness score per device over time.

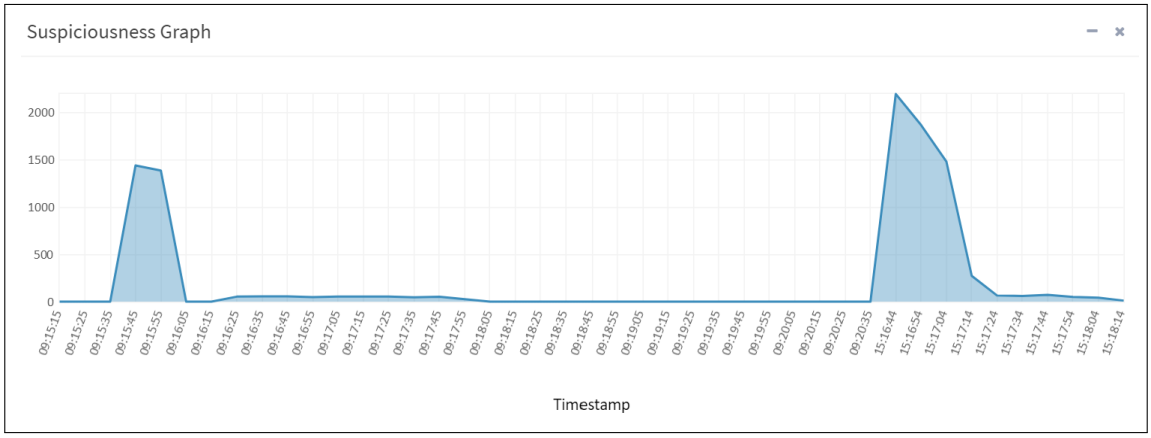

Figure 3.2: Overall network suspiciousness score over time.

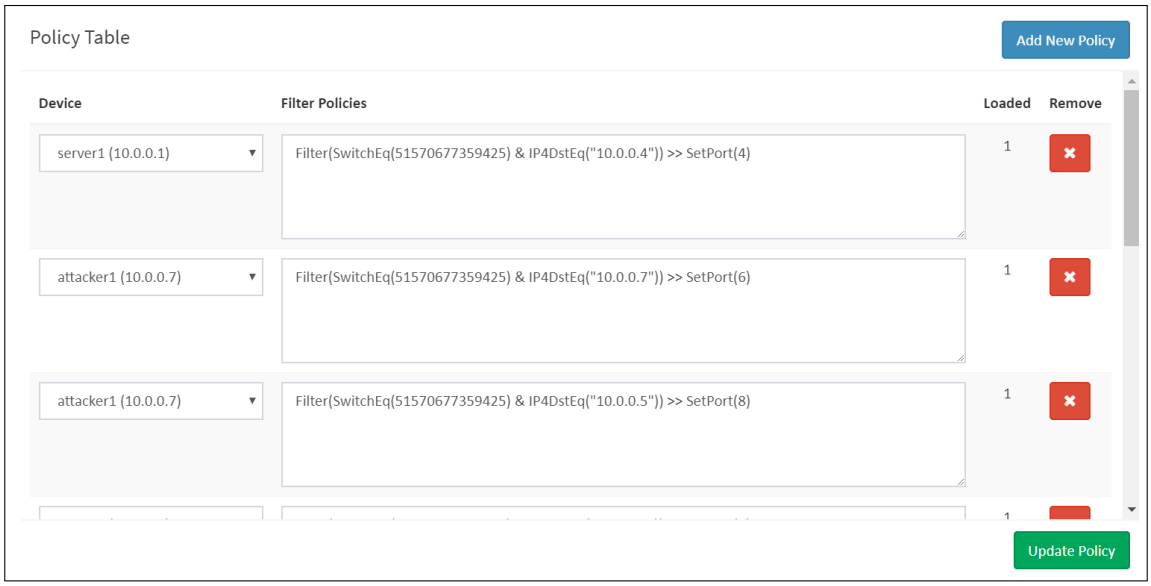

Figure 3.3: Policy table view.

3. Network view: a `vis.js`-based view to monitor the network as a graph of connected devices as depicted in Figure 3.4.

3.4 Policy Decision Making

In ADAPTs, each device has a corresponding access control policy to control/configure it remotely. We call this configuration policy, which determines the structure of the network and decides how traffic flows traverse through the network in normal versus attack conditions.

Similarly, ADAPTs also features a *defense policy* for the enterprise network. The defense policy is reactive i.e., it will take effect when the original configuration policy has failed to communicate erratic host behaviors such as *SS* threshold changes, jump in the number of external hosts contacted, etc., or if an attack has be detected and communication privileges need revocation. The interface can facilitate administrators to update policies directly in the event of an attack.

Both these policies and the revocation/enabling functionalities are instantiated based

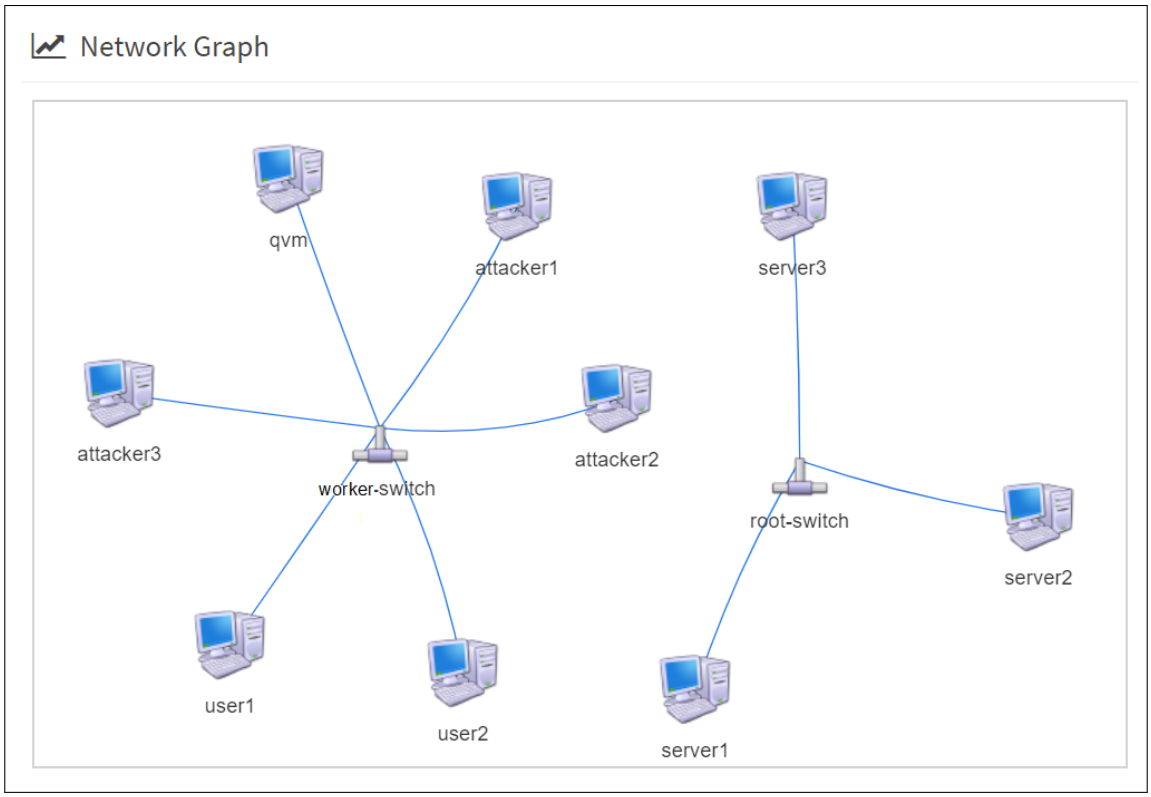

Figure 3.4: Network graph of all the connected devices.

on the policy updater mechanism, whose main objective is to simplify the learning curve for users/administrators to become proficient at writing policies (e.g., using network programming languages such as Frenetic [35])—a daunting and tedious task. With this in mind, the updater component can auto-generate policies based on simplified inputs that are provided via the user interface. For example, to minimize the process, the policy updater takes a generic command such as "user1 to server1" and all possible configuration policies would be generated by the updater. The updater works with the centralized database and is pre-programmed with the network architecture.

3.5 Threat Sharing Mechanism

Algorithm 2 runs in the monitor component and coordinates/shares intelligence with the switches deployed in the network and across different providers. This in turn enables

a collaborative environment among providers such that the targeted attacks can be detected closer to the source without affecting the cloud infrastructure. A natural question is why would a provider share the attack intelligence, especially in a business that is driven by competition? We posit that the coordination among different Autonomous Systems (ASes)/providers is mutually beneficial for all the entities involved. Of course, a particular AS/provider can decide not to share the attack intelligence to others. However, if an AS experiences an attack and if it shares the intelligence with other ASes, a global and unified hardening of infrastructure against such targeted attacks can be achieved. In addition, any downtime is money lost in a business; sharing the attack intelligence in turn provides a cheaper alternative to lost host/business downtime.

Our iQVM monitors also coordinate and share the targeted attack threat intelligence such as *SS* thresholds, policy updates, etc. with other hosts in the network. Apart from providing a collaborative environment amongst pertinent hosts to effectively counter said attacks, the mechanism also provides a way to drill down on specific segments of the network with suspicious hosts. Furthermore, we believe that the coordination mechanism will pave the way to achieve a global and unified hardening of the enterprise network against APT, APM, DDoS and future attacks.

input : *attacker_ID* ▷ Attacker ID
src_ip ▷ source IP
dst_ip ▷ destination IP
no_of_packets ▷ number of packets
spoof_dst_ip ▷ spoofed IP
block_ip ▷ blocklisted IP list
Result: Attack traffic will be redirected to the quarantine VM and Targeted Attack blocking policy will be generated

```
1  function initQuarantine()
2      createVM();
3      updatePolicy(src_ip);
4      do
5          redirectTraffic();
6          pretense_data = generatePretense();
7          vmResponse(spoof_dest_ip, src_ip, dst_ip, pretense_data);
8      while true;
9  End Function
10 function updatePolicy (src_ip)
11     logAttackTraffic();
12     new_policy = generateNewPolicy();
13     collaborate(new_policy);
14 end
15 function collaborate (new_policy)
16     advertisePoliciesToNeighbors(new_policy);
17     block_ip = updateList(src_ip);
18     redirectTraffic();
19 end
20 function redirectTraffic ()
21     sendTrafficToQuarantineVM();
22 end
23 function main ()
24     /* Receive incoming data from external machine */
25     data = monitorPackets(attacker_ID, src_ip, no_of_packets, start_time,
         end_time);
26     attack = calculateNetworkSuspiciousness(data);
27     /* Update policy in case of attack detected */
28     if attack.suspiciousnessDetected == true then
29         initQuarantine(src_ip);
30     end
31     decideToStopOrContinue();
32 end
```

Algorithm 2: Dolus/ADAPTs system phases for spoofing pretense

Chapter 4

Performance Evaluation

In this section, we describe our evaluation of ADAPTs in a GENI testbed. We start by describing our testbed, followed by the experiments and results from our evaluation. The instructions on how to replicate this experiment can be found on github [36].

4.1 Testbed Setup

Original Dolus Testbed Setup

Our original Dolus/ADAPTs testbed is comprised of two open vSwitches (a worker and a root), nine nodes (which are hosts), and a controller as shown in Figure 4.1. The worker switch connects all the user nodes, and the root switch connects all the servers hosting the application system and related services. A controller is a standalone node, running the monitor and policy updaters, calculating *SS* thresholds for nodes and the overall network, managing all the traffic and defense mechanisms of ADAPTs. All these components run Ubuntu 16.04 on a GENI testbed.

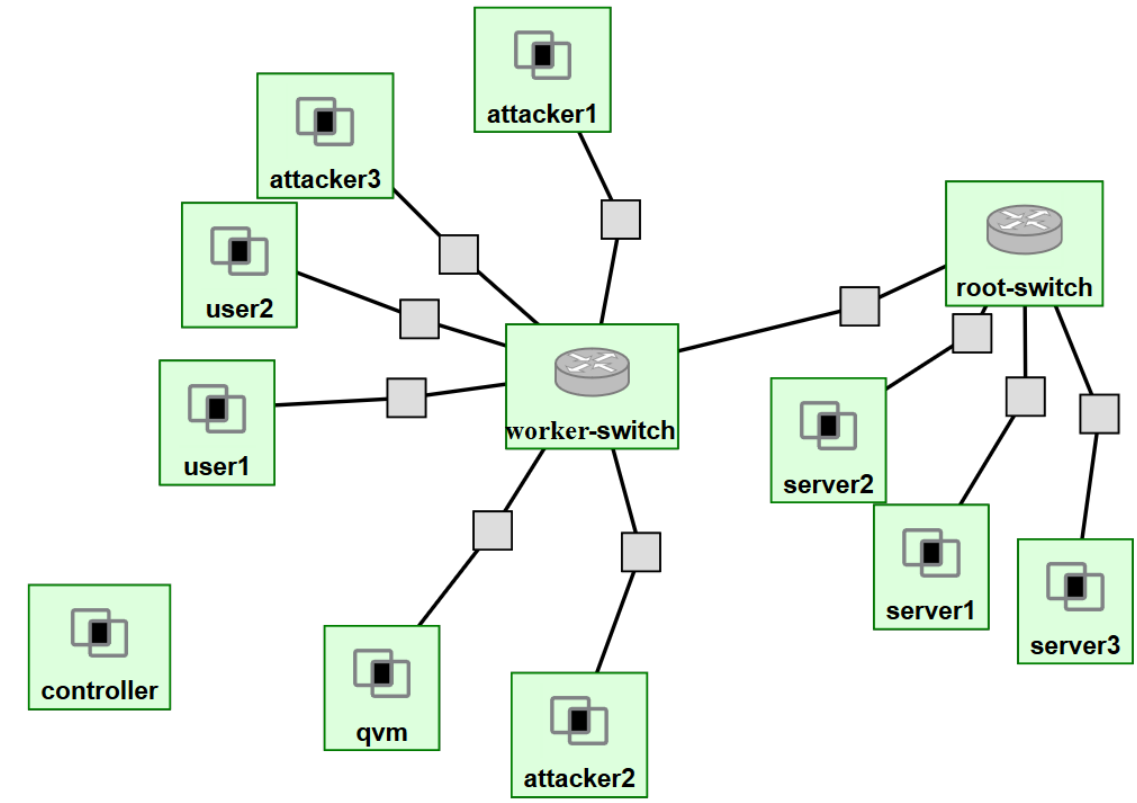

Figure 4.1: ADAPTs GENI slice for APT and DDoS experiments

Enhanced ADAPTs Testbed Setup

For the purposes of APT/APM attack detection and defense, a modified GENI Cloud testbed was setup as shown in Figure 4.2. The purpose of the enhanced GENI Cloud Testbed is to simulate a collaborative crossdomain SDxI architecture with the core servers and services located at the switch-root in the Clemson InstaGENI (blue) domain. Correspondingly, the user traffic originates from three other separate domains with two distinct paths to where the services are located. Since an APT is not a distributed attack, there was no need to consider multiple attack vectors from many directions. However, due to the nature of an APT attack being secretive and stealthy, we assume that an APT can be hiding anywhere in an SDxI.

Our testbed is comprised of multiple open vSwitches (multiple workers and a single root), numerous nodes (which are hosts), and a controller. The slave switches connect all the user nodes, and the root switch connects all the servers hosting the application system and related services to the worker switches. The controller in the setup is a standalone node, running the monitor and policy updaters, calculating *SS* thresholds for nodes and the overall network, managing all the traffic and defense by pretense mechanisms of the Dolus system.

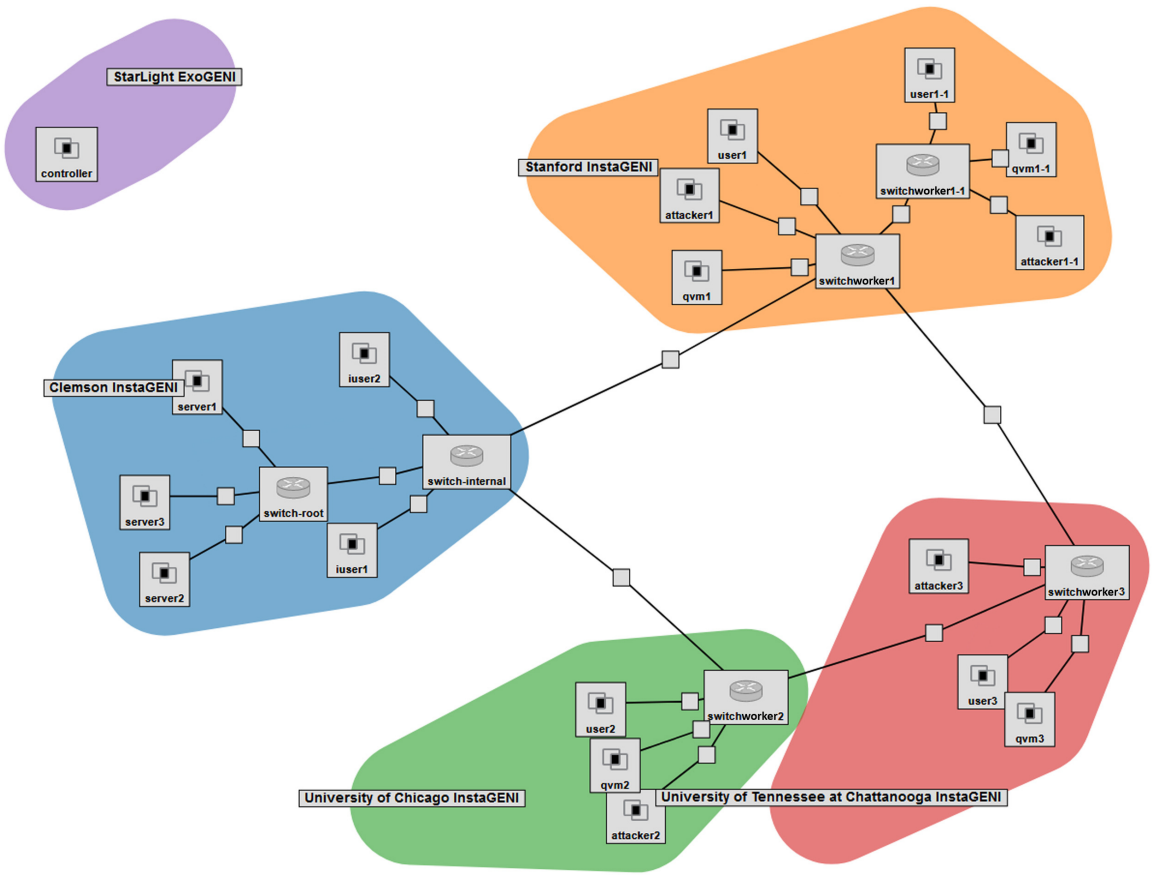

Figure 4.2: ADAPTs GENI slice for APM experiments

4.2 Experiments

4.2.1 Advanced Persistent Threat and Data Exfiltration

In the first experiment, we randomly selected three hosts, and compromised them by running `slowhttp (DDoS)` attacks from attacker 1 and attacker 3, and a secure copy, `scp (APT data exfiltration)`, from attacker 2. Before running the experiment, we specify minimum and maximum values for flows, connections, and bytes: the user and attacker nodes are each set to a minimum of 1 and a maximum of 10 connections, a minimum of 100 and a maximum of 1,000 flows, and a minimum of 10 and a maximum of 100,000 bytes. The servers had a minimum of 10 and a maximum of 1000 connections, a mini-

Node	Command	Score
Attacker1	slowhttp	8.8
Attacker2	scp	215.5
Attacker3	slowhttp	18.0
Server1	ping	17.3
Server2	Traffic Response	16.4
Server3	iperf -s	9.0
User1	iperf -c	5645.7
User2	wget	200.7

Table 4.1: Suspiciousness scores before allowlisting

mum of 1,000 and maximum of 10,000 flows, and minimum of 100,000 and a maximum of 100000000 bytes.

From the controller, we obtain the *SS* for these three attackers before (see Table 4.1) and after (see Table 4.2) allowlisting. Note that devices are not allowlisted on our network, which is why their suspiciousness scores are of focus. User1 initally displays the highest *SS* due to iperf being a measurement of maximum bandwidth available on a network. The results therefore, could look like a data exfiltration attack. This result is later allowlisted due to it being a network traffic test and expected benign traffic. Attacker 2 exhibited the highest *SS* out of three, due to actual data exfiltration [9]. The traffic that is being exfiltrated generates a much higher score than the regular traffic in the network.

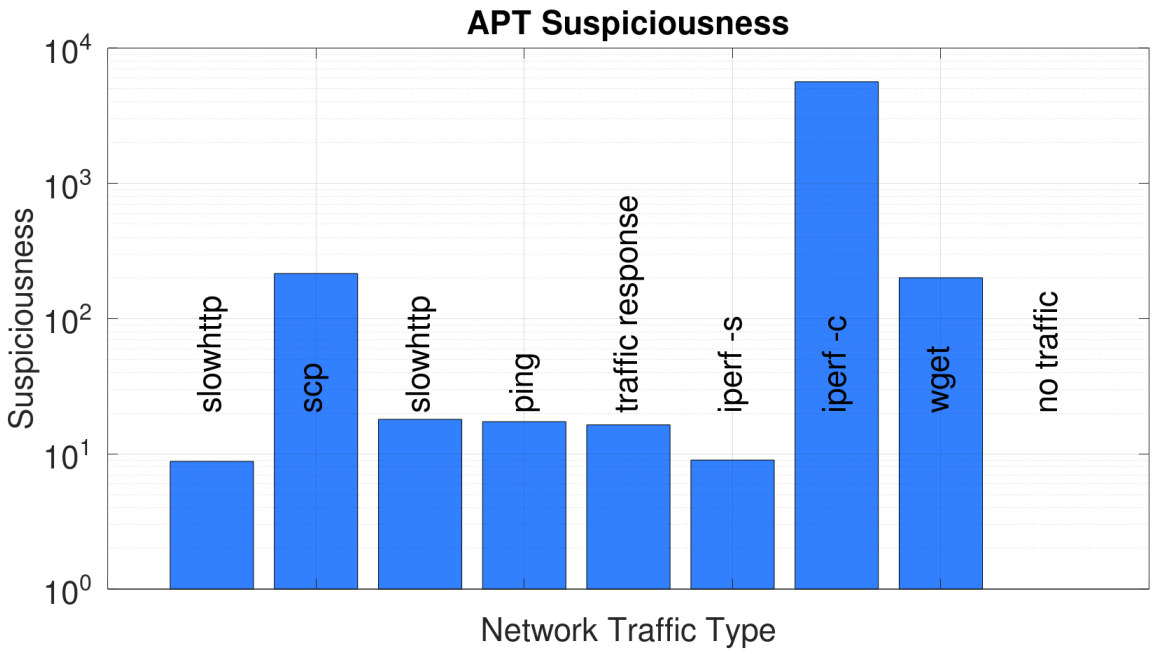

Figure 4.3: Overall suspiciousness for APT attack case

4.2.2 DDoS Attack and Loss of Service

For purposes of a more complete evaluation of our Suspicousness Scores, we decided to recreate our original Dolus DDoS attacks and determine how the same slowHTTPtest DDoS attack would be evaluated using the same Suspiciousness evaluation used for our above APT and Data Exfiltration experiments. The Suspicousness of the slowHTTPtest DDoS attack can be seen in figure 4.4.

As you can see from the results, it was somewhat surprising that the DDoS attack is actually far less suspicious than the wget and scp traffic. We needed to perform further evaluation and analysis to determine what needed to be done to successfully detect a DDoS attack as suspicious traffic. It was upon investigation of our next attack, Advanced Persistent Mining and Loss of Resources in 4.2.3, in which we determined how to better evaluate this type of attack. This evaluation for DDoS attacks is further discussed in 4.3.3.

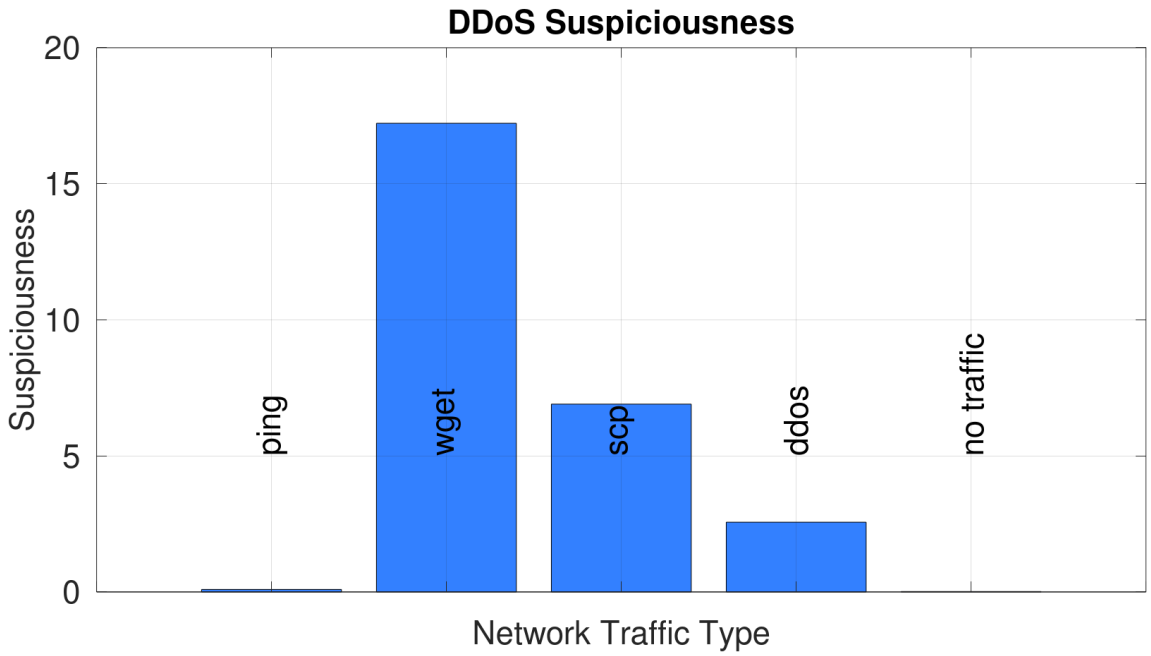

Figure 4.4: Overall suspiciousness for DDoS attack case

4.2.3 Advanced Persistent Mining and Loss of Resources

Advanced Persistent Mining attacks we expected to be evaluate quite differently from the previous APT attacks discussed in 4.2.1 as the APT attacks lie in wait and exfiltrate data at opportune times whereas APM attacks would tend to use resources immediately and quickly to generate crypto currencies for the attacker before administrators have an opportunity to respond to the attack.

For the purposes of our attack, we decided to use several variations of mining in an attempt to defeat the suspiciousness detection. We used the Etherium miner geth for CPU mining. We ran three variations of the geth mining. The initial geth mining was not limited in any way and thus used 100% of the available CPU and as much network traffic as it needed without limit. The second evaluated geth attack, geth_{t}, used a network limitation on geth which only allowed for a maximum of 10 kbps download and upload speed. The final of the three geth attacks, geth_{cpu} used a CPU limiter on the geth process, only allowing it to use a maximum of 10% of the CPU at any given time.

In addition, we also ran our previous ping, wget, scp, and DDoS traffic alongside the geth APM to evaluate it alongside the various traffic types. We also decided to evaluate torrent traffic, as we expected that torrent traffic may look quite similar to geth APM traffic.

As you can see in figure 4.5, the most suspicious traffic was actually the torrent traffic. This actually wasn't entirely surprising as the torrent traffic transfers over 1 Gigabyte of network traffic outside the network and could be seen as a form of data exfiltration if not accounted for. Geth CPU mining does not actually transfer a large number of bytes outside the network as most of the work is simply CPU calcuations, therefore we needed some additional analysis of the network traffic to determine what would actually make the geth traffic suspicious.

Upon further analysis, we found an interesting statistic of the geth traffic which made it stand out far and above all other traffic types we evaluated. The geth traffic actually establishes far more IP connections than any other form of traffic. This can be seen quite clearly in figure 4.6. Both the regular geth traffic as well as the network limited geth traffic established about three times and two and a half times IP connections than the torrent traffic respectfully. Even the geth attack limited to 10% CPU still produced nearly as many IP connections as the torrent traffic.

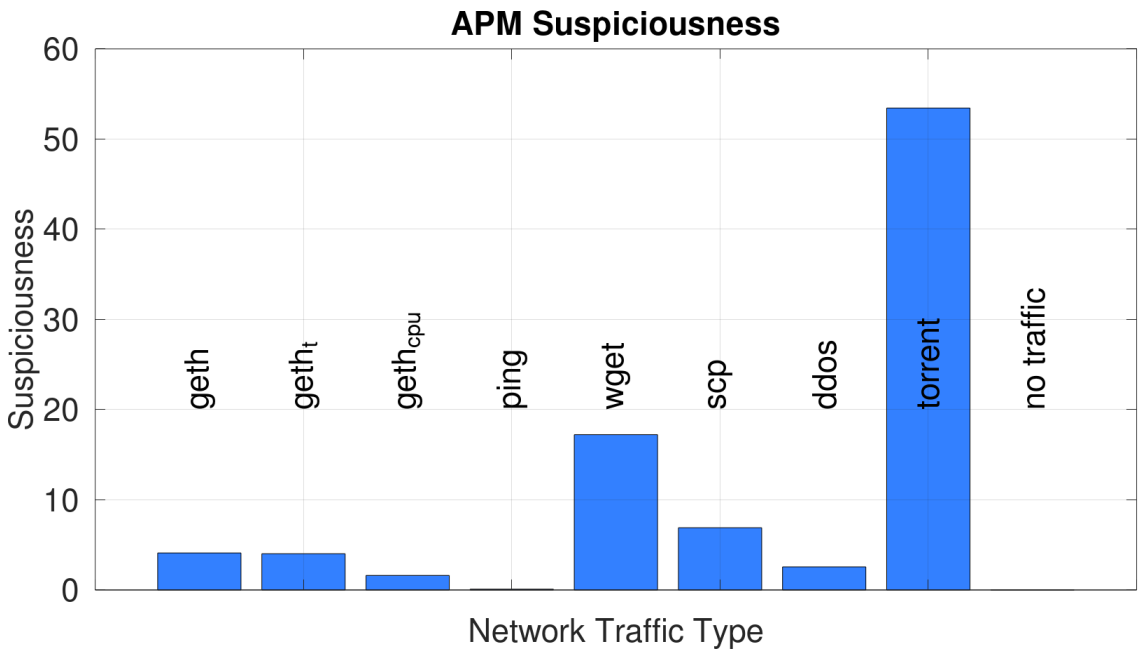

Figure 4.5: Overall suspiciousness for APM attack case

4.3 Findings

4.3.1 Advanced Persistent Threats

The purpose of allowlisting is to allow administrators to ignore traffic, which is not going outside of the network. For example, if we consider that both server 1 and user 1 are within our own network, then any data transmitted between those two machines would not be data being exfiltrated from the enterprise network. Therefore, we can consider such traffic as benign. However, whenever we consider attacker 1 and server 1, since attacker 1 exists outside our controlled network, we consider all traffic from attacker 1 to be possible data exfiltrated from the enterprise network.

As you can see in Table 4.2, we ignore the traffic between users and servers, even though there was data moving between them (Table 4.1). Moreover, by considering the allowlisting prior to the Suspiciousness Score calculations, we decrease the overall time spent on speed of the calculations since we will need to calculate scores for *only* a portion of the network.

Node	Command	Score
Attacker1	slowhttp	8.8
Attacker2	scp	215.5
Attacker3	slowhttp	18.0

Table 4.2: Suspiciousness scores after allowlisting

4.3.2 APM Results and Findings

As we saw in section 4.2.3, the geth mining created many more IP connections than torrent traffic. Another interesting finding was the distinct IP connections over time. Figure 4.7 shows both the three forms of geth traffic as well as the torrent traffic. All forms of geth traffic displayed quite a wild variation of distinct IP connections made very few seconds. Even the CPU limited geth attack while clearly making fewer connections, still maintained the same wide variation every few seconds. This seems to indicate that a wide variation of distinct IP connections could be used as a fingerprint of an APM and used to increase the suspiciousness score of the devices with an active APM. Likewise, the torrent traffic initially creates more distinct IP connections, it remains steady and then completely drops off once the file transfer is complete. This can be used in a similar manner to increase/decrease suspiciousness of a device depending on what type of attack one may be intersted in detecting.

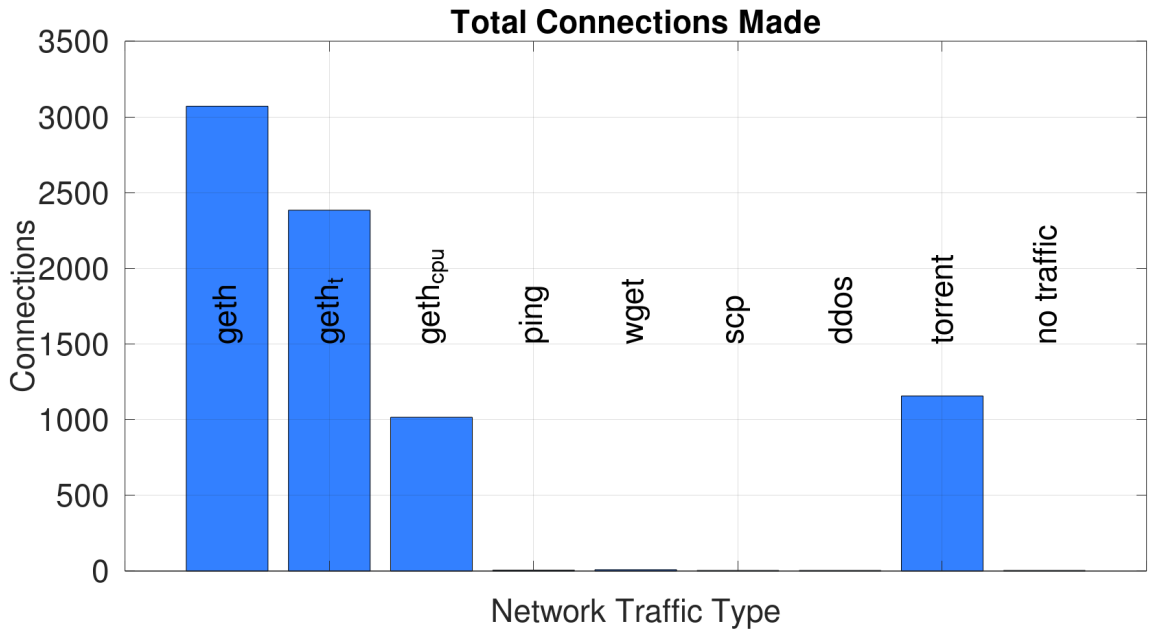

Figure 4.6: Total distinct IP connections made for APM attack case

Fading Pretense Against APM

The Fading Pretense policy can be implemented as an effective defense mechanism against APM attacks as described earlier in Section 3.1. Herein, we describe a demonstration of an experiment which illustrates how the fading pretense policy could deter an APM attacker in practice. Figure 4.8 shows the experiment conducted using a safe system zone, and an attacked system zone. The x-axis is an arbitrary unit of time progression that can be configured (on the order of several minutes, hours or even days) by the system/network administrator depending on the duration of the pretense policy being in effect. Behavioral psychology considerations also could be factored into determining the rate of progressive decline of the resource allocation on a compromised device. In any case, we can observe that three distinct phases of pretense should occur:

1. (Phase-1): Fading pretense begins. Assuming at time x = 1, a resource exfiltration attack is detected to be occurring in an attacked system zone, at which point the attacker will have 100% availability to the system resources, and the fading pretense policy is initiated at time x = 2.

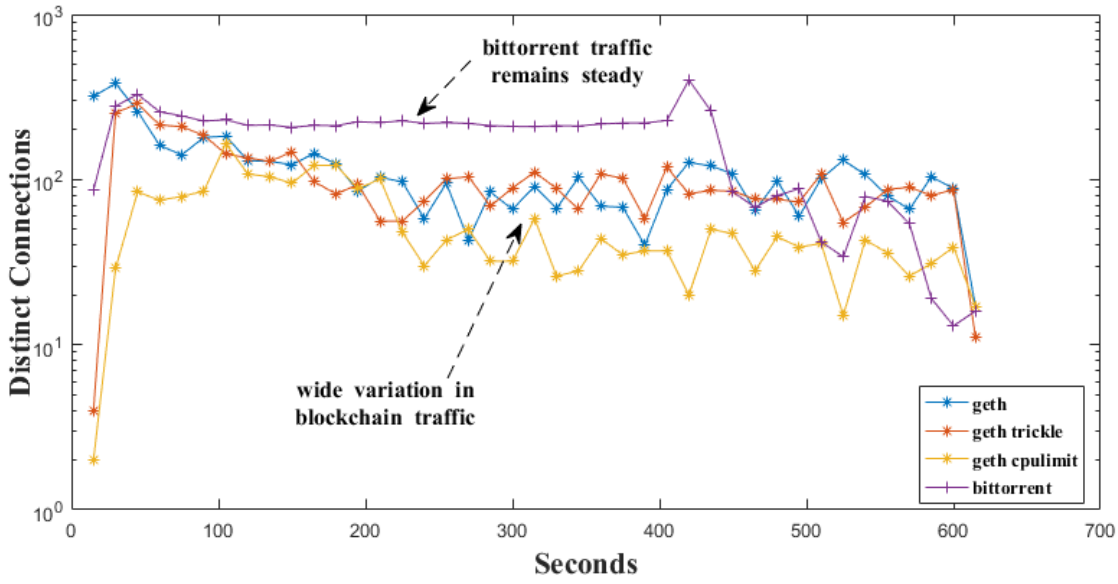

Figure 4.7: Distinct IP connections contacted for APM and Torrent traffic per every 15s.

2. (Phase-2): Attacker is deterred. In the time after the fading pretense policy is in effect, the availability of the system resources is reduced progressively to 90%, 60%, 30% and 10% to ultimately cause the attacker to consider a redirection of the APM attack to a morepotent device with higher resource allocations.

3. (Phase-3) Safe system zone restoration. Once the attacker is found to have been deterred at time x = 6, the previously compromised device can be added to into the safe system zone with 100% availability of system resources. We remark that the safe system zone restoration should be performed only after suitable patching or system re-imaging in order to ensure that there is no re-occurrence of the APM attack on that particular device in the future.

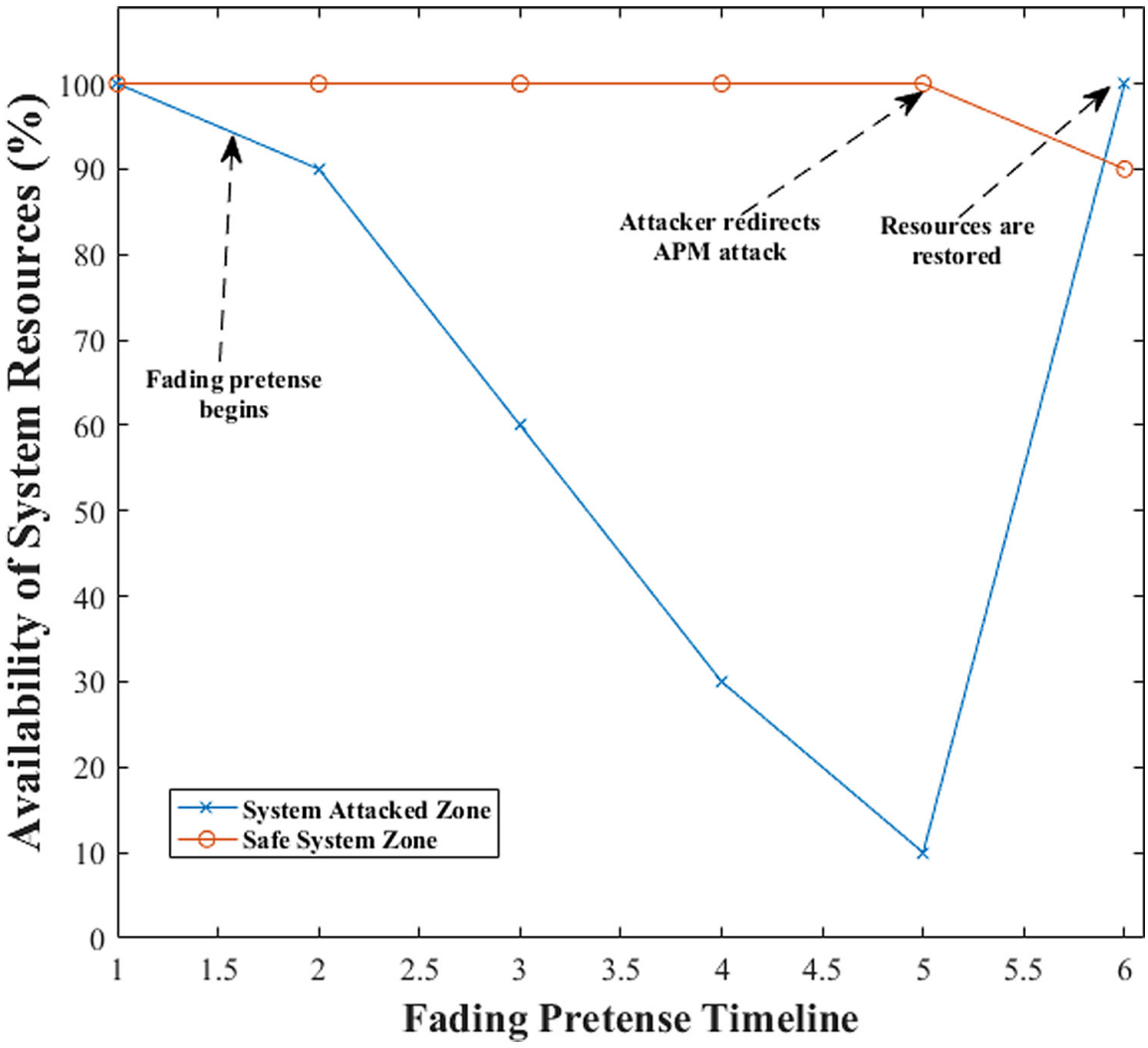

Figure 4.8: Demonstration of a fading pretense policy implementation to deter an APM attack after its detection on a compromised device.

4.3.3 DDoS Results and Findings

Coming back to our DDoS results after the APM evaluation in 4.3.2 we had new ideas for how we can similarly detect a DDoS attack and flag it as suspicious. Much like the APM results, we decided to look at various types of traffic over time to determine a better DDoS suspiciousness evaluation. This can be seen in figure 4.9, the DDoS attack starts by sending a large number of bytes which tapers of quickly within the first ten seconds, remains close to zero bytes transfered for another ten seconds before it quickly bursts a large number of bytes again. The initial burst of bytes was unique among the various types of traffic we

evaluated and as such, could also be a possible fingerprint to better the detection of a DDoS attack using suspiciousness scores.

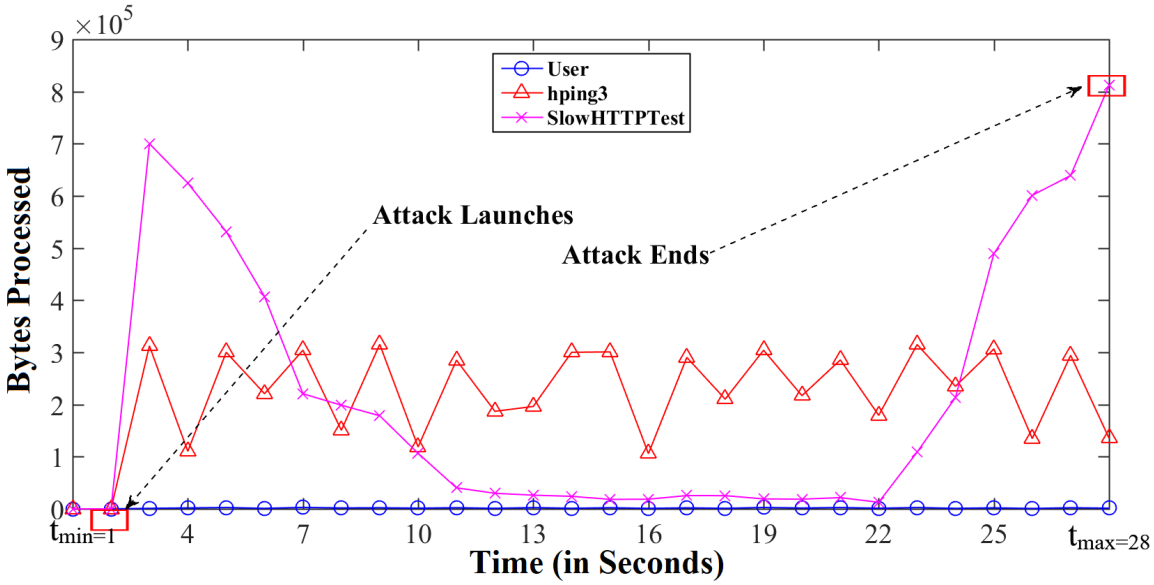

Figure 4.9: DDoS slowHTTPtest attack over time

Fading Pretense Against DDoS

Much the same as our previous Dolus experiments with Moving Target Defense (MTD) [5], figure 4.10 compares the time taken by our Dolus/ADAPTs system to stop a DDoS attack versus MTD-based and no defense strategies. These results are nearly exactly the same as our previous Dolus only defense. This is expected, as we implement the same defence mechanisms which Dolus uses, redirection of attack traffic and pretense, however with the attack detection based upon suspiciousness scores. After a warm-up period of 6 *seconds*, we start the SlowHTTPTest and hping3 at the 7^{th} *second* from the attackers. In a SDxI-based cloud network with no defense strategy, the services are immediately affected by the attack traffic. Similarly, the MTD-based defense strategy takes $\sim$6 *seconds* to mitigate the attack traffic impact. However, our Dolus/ADAPTs system supported service on the other hand, does not suffer from any loss of availability in comparison with the other two strategies. This is due to the sharing of attack intelligence between the slave switches and

redirection of attack traffic to quarantine VMs closer to the attackers, making the cloud network completely oblivious to the attackers. This shows how despite the different attack detection methodology, the ADAPTs implementation of suspiciousness scores with Dolus does not lose any capability but rather maintains the same defense capacity as before.

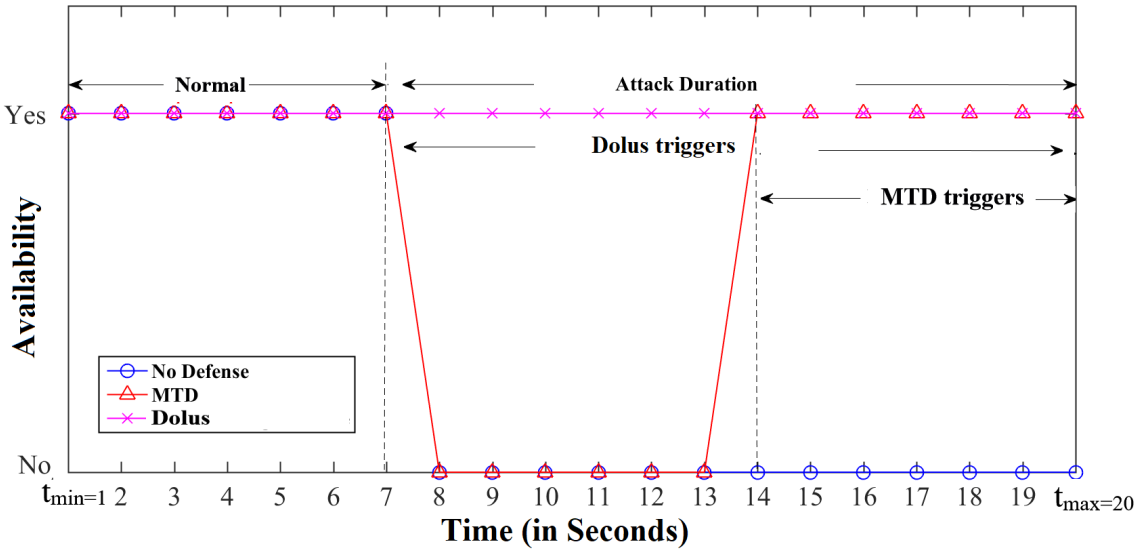

Figure 4.10: Comparison of cloud service restoration time metric with cases of: no Defense, MTD and Dolus/ADAPTs

4.3.4 ADAPTs Performance

Table 4.3 shows the time taken by ADAPTs to calculate the *SS* for devices, each running on a single core. It also shows the number of traces, and their corresponding processing times. As high as 1.8 Million traces for 8 devices can be processed under 100s, which demonstrates the efficacy of ADAPTs. However, there is a linear increase in time as the number of traces grow. To address this issue we plan to implement parallelization, and evaluate ADAPTs on a multi-core setting as part of our future work.

Single Threaded		
Devices	**Number of traces**	**Time (in seconds)**
3	590,492	50
6	1,249,490	77
8	1,839,982	94

Table 4.3: Processing time taken by ADAPTs on single core nodes.

Chapter 5

Future Work

There are a number of enhancements which I believe could greatly improve the overall impact of the Dolus and ADAPTs systems. Firstly some of the greatest benefits would come from parallelized calculations of suspiciousness scores. At the time of our experimentation, we were severely limited by the GENI system in which we could only use a single threaded MySQL database for calculating the suspiciousness scores. This in turn, increased the time required for detection of an APT.

Secondly, our system could also benefit from additional Big Data analytics. In our small test cases and scenarios, a single minute of traffic could easily produce several gigabytes of data. Perhaps both Hadoop and MapReduce could be leveraged to help process the network traffic in a more real time manner and allow for faster suspiciousness score calculation.

Lastly, a method in which we could more easily share the results of the network traffic analysis and suspiciousness calculations would be ideal for network policy sharing within a Confederation. A possible solution to this could be a blockchain implementation, where the the suspiciousness score and traffic analysis is mined and network policy updates are written to the blockchain allowing other members of the Confederation to then apply these network policies. In fact, there has been some future progress made on such efforts. DefenseChain has been shown to give organizations an incentive-based and trustworthy co-

operation to mitigate the impact of cyber attacks [37].

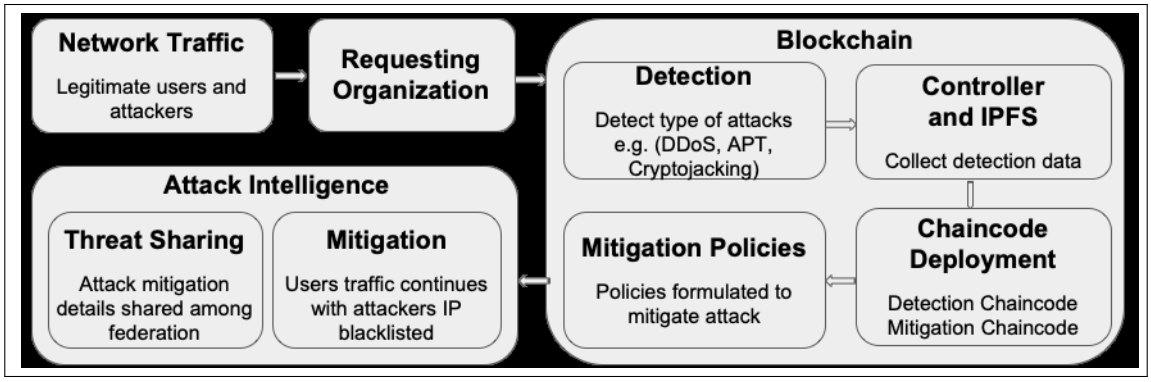

Figure 5.1: DefenseChain execution pipeline.

Chapter 6

Summary and Concluding Remarks

With the advent and advancement of cyber attacks such as APTs, the development of proper countermeasures is of the utmost importance. The nature of APTs is one of subtlety and trickery; therefore, proper systems must be in place to combat against these threats. We have found that with the use of Suspiciousness Scores, the difficulty in determining which devices on your network may be compromised can be greatly eased to initiate a pertinent defense strategy.

In the future, we would like to continue to improve upon the Suspiciousness Scores calculator, the policy updater, and the administrator user interface components. For the Suspiciousness Scores, one direction we intend to pursue is depending on the type of traffic, type of device, or the expected user of the device, real-time weight update mechanism can be added. The goal is to vary (i.e., increase or decrease) the Suspiciousness Score, in real-time, depending on the how suspicious a device or traffic is in a fine-grained fashion. For example, typical $http$ traffic may have a lesser weight associated with it as opposed to *SSH* traffic, which would be more closely examined and therefore carry a larger weight. Likewise, Suspiciousness Scores calculated on a device used by an expert user may carry a lesser weight, whereas new or unknown devices to the network with unknown users may carry a greater weight. With respect to the administrator user interface, we would like to

continue on the development of automated policy management using SDN controllers. In our prior work, the development of Dolus [5], we wanted to show how tricks and pretense can be used to combat against DDoS attacks. We would like to further these concepts and attempt pretense against Command and Control (C&C) servers, the remote machines which command the APTs and to which they exfiltrate data. For the policy updater, it is possible make the process even more simplified. We also plan to implement the ability to drag and drop network elements that will dynamically update an enterprise network's policy. This will in turn provide the enterprise network administrator with a graphical overview of the network and a more user-friendly way to interact with voluminous network flows.

Bibliography

[1] Ping Chen, Lieven Desmet, and Christophe Huygens. A study on advanced persistent threats. In *IFIP International Conference on Communications and Multimedia Security*, pages 63–72. Springer, 2014.

[2] Nikos Virvilis, Dimitris Gritzalis, and Theodoros Apostolopoulos. Trusted computing vs. advanced persistent threats: Can a defender win this game? In *Ubiquitous intelligence and computing, 2013 IEEE 10th international conference on and 10th international conference on autonomic and trusted computing (uic/atc)*, pages 396–403. IEEE, 2013.

[3] Charalampos Patrikakis, Michalis Masikos, and Olga Zouraraki. Distributed denial of service attacks. *The Internet Protocol Journal*, 7(4):13–35, 2004.

[4] Cisco. Cisco annual internet report (2018–2023), 2020.

[5] Roshan Neupane, Travis Neely, Nishant Chettri, Mark Vassell, and Prasad Calyam. Dolus: Cyber defense using pretense against ddos attacks in cloud platforms. In *Distributed Computing and Networking, 2018 ICDCN International Conference of*. ICDCN, 2018.

[6] Ernesto Van der Sar. How much money can pirate bay make from a cryptocoin miner?, 2017.

[7] Jérôme Segura. A look into the global 'drive-by cryptocurrency mining' phenomenon, 2017.

[8] S. Nichols and S. Stich. A cognitive theory of pretense. *Cognition*, 74(2):115–147, 2000.

[9] Mirco Marchetti, Fabio Pierazzi, Michele Colajanni, and Alessandro Guido. Analysis of high volumes of network traffic for advanced persistent threat detection. *Computer Networks*, 109:127–141, 2016.

[10] Michael K Daly. Advanced persistent threat. *Usenix, Nov*, 4(4):2013–2016, 2009.

[11] Greg Hoglund. Advanced persistent threat, what apt means to your enterprise, 2009.

[12] Eric Cole. *Advanced persistent threat: understanding the danger and how to protect your organization*. Newnes, 2012.

[13] Ibrahim Ghafir and Vaclav Prenosil. Advanced persistent threat attack detection: An overview. *International Journal of Advances in Computer Networks and Its Security (IJCNS)*, 4(I), 2014.

[14] Ari Juels and Ting-Fang Yen. Sherlock holmes and the case of the advanced persistent threat. In *5th USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET 12)*, San Jose, CA, April 2012. USENIX Association.

[15] Merete Ask, Petro Bondarenko, John Erik Rekdal, André Nordbø, Pieter Bloemerus, and Dmytro Piatkivskyi. Advanced persistent threat (apt) beyond the hype. *Project Report in IMT4582 Network Security at Gjøvik University College, Springer*, 2013.

[16] Nikos Virvilis and Dimitris Gritzalis. The big four-what we did wrong in advanced persistent threat detection? In *Availability, Reliability and Security (ARES), 2013 Eighth International Conference on*, pages 248–254. IEEE, 2013.

[17] Jisang Kim, Taejin Lee, Hyung-guen Kim, and Haeryong Park. Detection of advanced persistent threat by analyzing the big data log. *Cloud Security Alliance*, 13(4):101–107, 2013.

[18] Pradip Kumar Sharma, Seo Yeon Moon, Daesung Moon, and Jong Hyuk Park. Dfa-ad: a distributed framework architecture for the detection of advanced persistent threats. *Cluster Computing*, 20(1):597–609, 2017.

[19] Parth Bhatt, Edgar Toshiro Yano, and Per Gustavsson. Towards a framework to detect multi-stage advanced persistent threats attacks. In *Service Oriented System Engineering (SOSE), 2014 IEEE 8th International Symposium on*, pages 390–395. IEEE, 2014.

[20] Beth Binde, Russ McRee, and Terrence J O'Connor. Assessing outbound traffic to uncover advanced persistent threat. *SANS Institute. Whitepaper*, 2011.

[21] Johannes de Vries, Hans Hoogstraaten, Jan van den Berg, and Semir Daskapan. Systems for detecting advanced persistent threats: A development roadmap using intelligent data analysis. In *Cyber Security (CyberSecurity), 2012 International Conference on*, pages 54–61. IEEE, 2012.

[22] J Vukalović and D Delija. Advanced persistent threats-detection and defense. In *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2015 38th International Convention on*, pages 1324–1330. IEEE, 2015.

[23] Gregory Vert, Ann Leslie Claesson-Vert, Jesse Roberts, and Erica Bott. A technology for detection of advanced persistent threat in networks and systems using a finite angular state velocity machine and vector mathematics. In *Computer and Network Security Essentials*, pages 41–64. Springer, 2018.

[24] CIARA FIU. 25th geni engineering conference (gec25)-part iii, 2017.

[25] Martin Lee and Darren Lewis. Clustering disparate attacks: mapping the activities of the advanced persistent threat. *Last accessed June*, 26, 2013.

[26] Paul Giura and Wei Wang. Using large scale distributed computing to unveil advanced persistent threats. *Science J*, 1(3):93–105, 2012.

[27] Zainab Saud and M Hasan 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*, pages 154–157. ACM, 2015.

[28] Zakiah Zulkefli, Manmeet Mahinderjit Singh, and Nurul Hashimah Ahamed Hassain Malim. Advanced persistent threat mitigation using multi level security–access control framework. In *International Conference on Computational Science and Its Applications*, pages 90–105. Springer, 2015.

[29] Eric M Hutchins, Michael J Cloppert, and Rohan M Amin. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research*, 1(1):80, 2011.

[30] Pengfei Hu, Hongxing Li, Hao Fu, Derya Cansever, and Prasant Mohapatra. Dynamic defense strategy against advanced persistent threat with insiders. In *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pages 747–755. IEEE, 2015.

[31] Inc. Radware. Defenseflow - sdn based network, application ddos and apt protection, 2014.

[32] Moustafa Ammar, Mohamed Rizk, Ayman Abdel-Hamid, and Ahmed K Aboul-Seoud. A framework for security enhancement in sdn-based datacenters. In *New Technologies, Mobility and Security (NTMS), 2016 8th IFIP International Conference on*, pages 1–4. IEEE, 2016.

[33] Mohamed Saher and Jayendra Pathak. Malware and exploit campaign detection system and method, September 10 2014. US Patent App. 14/482,696.

[34] Carolyn Jane Anderson, Nate Foster, Arjun Guha, Jean-Baptiste Jeannin, Dexter Kozen, Cole Schlesinger, and David Walker. Netkat: Semantic foundations for networks. In *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '14, 2014.

[35] Nate Foster, Rob Harrison, Michael J Freedman, Christopher Monsanto, Jennifer Rexford, Alec Story, and David Walker. Frenetic: A network programming language. In *ACM Sigplan Notices*. ACM, 2011.

[36] Travis Neely, Mark Vassel, Nishant Chettri, Roshan Neupane, Prasad Calyam, and Ramakrishnan Durairajan. Automated defense against advanced persistent threats using suspiciousness tracking - github repository. `https://github.com/Travisivart/ADAPTS`, 2017.

[37] Soumya Purohit, Prasad Calyam, Songjie Wang, RajaniKanth Yempalla, and Justin Varghese. Defensechain: Consortium blockchain for cyber threat intelligence sharing and defense. In *Blockchain Research & Applications for Innovative Networks and Services (BRAINS), 2020 2nd Conference on*. IEEE, 2020.