6-17-2020

Mitigating Stealthy Link Flooding DDoS Attacks Using SDN-Based Moving Target Defense

Abdullah Aydeger
Florida International University, aayde001@fiu.edu

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

MITIGATING STEALTHY LINK FLOODING DDOS ATTACKS USING SDN-BASED MOVING TARGET DEFENSE

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

ELECTRICAL AND COMPUTER ENGINEERING

by

Abdullah Aydeger

2020

To: Dean John Volakis
College of Engineering and Computing

This dissertation, written by Abdullah Aydeger, and entitled Mitigating Stealthy Link Flooding DDoS Attacks Using SDN-Based Moving Target Defense, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

A. Selcuk Uluagac

Alexander Perez-Pons

Deng Pan

Kemal Akkaya, Major Professor

Date of Defense: June 17, 2020

The dissertation of Abdullah Aydeger is approved.

Dean John Volakis
College of Engineering and Computing

Andres G. Gil
Vice-President for Research and Economic Development
and Dean of University of Graduate School

Florida International University, 2020

DEDICATION

To my parents.

ACKNOWLEDGMENTS

First of all, I would like to express my deepest and sincere gratitude to my advisor Dr. Kemal Akkaya, for his valuable guidance and extensive support from the beginning until the end of the completion of this research work. He has motivated me whenever I lost hope and helped me to go in the right direction whenever I needed it. I would not be able to complete this work without his assistance. I am grateful that he has taught me various fields of computer networking and cybersecurity with his unsurpassed knowledge. I am also beholden for the opportunity he provided me to work in his research group at his research lab, Advanced Wireless and Security Lab (ADWISE). This lab atmosphere helped me to develop my professional skills in a research environment. I am also thankful for the great colleagues that I had in this lab.

I would also like to thank Dr. A. Selcuk Uluagac for his comments during our collaboration for several projects. His great advice has been beneficial not only for my doctoral journey but also for my academic career search. I appreciate his support in many ways.

I also appreciate Dr. Alexander Perez-Pons and Dr. Deng Pan for serving on my dissertation committee and giving me insightful comments and constructive suggestions.

I would also like to thank Dr. Nico Saputro for so many works that we have been together and his many great ideas, which enabled me to think about new research problems.

I would also like to thank Dr. M. Ashiqur Rahman for the generous feedback he provided during our research collaboration and helped me to improve my academic writing skills significantly.

I would also like to express my gratitude to Dr. Samet Tonyali and Dr. Abbas Acar who have provided significant help and feedback in writing my dissertation.

I would like to thank the Electrical and Computer Engineering Department at Florida International University for their support throughout my doctoral journey with the teaching assistant position, which has covered most of my living costs.

Finally, I would also like to acknowledge the financial support by a seed grant from the Florida Cybersecurity Center and University Graduate School at Florida International University for conference travel expenses.

ABSTRACT OF THE DISSERTATION

MITIGATING STEALTHY LINK FLOODING DDOS ATTACKS USING SDN-BASED MOVING TARGET DEFENSE

by

Abdullah Aydeger

Florida International University, 2020

Miami, Florida

Professor Kemal Akkaya, Major Professor

With the increasing diversity and complication of Distributed Denial-of-Service (DDoS) attacks, it has become extremely challenging to design a fully protected network. For instance, recently, a new type of attack called Stealthy Link Flooding Attack (SLFA) has been shown to cause critical network disconnection problems, where the attacker targets the communication links in the surrounding area of a server. The existing defense mechanisms for this type of attack are based on the detection of some unusual traffic patterns; however, this might be too late as some severe damage might already be done. These mechanisms also do not consider countermeasures during the reconnaissance phase of these attacks. Over the last few years, moving target defense (MTD) has received increasing attention from the research community. The idea is based on frequently changing the network configurations to make it much more difficult for the attackers to attack the network.

In this dissertation, we investigate several novel frameworks based on MTD to defend against contemporary DDoS attacks. Specifically, we first introduce MTD against the data phase of SLFA, where the bots are sending data packets to target links. In this framework, we mitigate the traffic if the bandwidth of communication links exceeds the given threshold, and experimentally show that our method significantly alleviates the congestion. As a second work, we propose a framework that

considers the reconnaissance phase of SLFA, where the attacker strives to discover critical communication links. We create virtual networks to deceive the attacker and provide forensic features. In our third work, we consider the legitimate network reconnaissance requests while keeping the attacker confused. To this end, we integrate cloud technologies as overlay networks to our system. We demonstrate that the developed mechanism preserves the security of the network information with negligible delays. Finally, we address the problem of identifying and potentially engaging with the attacker. We model the interaction between attackers and defenders into a game and derive a defense mechanism based on the equilibria of the game. We show that game-based mechanisms could provide similar protection against SLFAs like the extensive periodic MTD solution with significantly reduced overhead.

The frameworks in this dissertation were verified with extensive experiments as well as with the theoretical analysis. The research in this dissertation has yielded several novel defense mechanisms that provide comprehensive protection against SLFA. Besides, we have shown that they can be integrated conveniently and efficiently to the current network infrastructure.

TABLE OF CONTENTS

LIST OF TABLES

TABLE PAGE

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

With the proliferation of online services, social media, and smart devices, everything became connected with each other more than ever, forming huge cyberspace. While such a connected world brings a lot of advantages to our lives in terms of convenience, cost, safety, and efficiency, it also turned the attention of attackers to cyberspace, creating new vulnerabilities and attack surfaces for them. Attackers can get control of a large number of devices by exploiting their vulnerabilities, while these devices may not be aware of the attacker's manipulation [ESA12]. The attacker can launch a Distributed Denial of Service (DDoS) attack by manipulating a large number of devices under his control, which may result in substantial financial and information loss.

It has been reported that cyberattacks, more specifically DDoS attack activities in terms of type and quantity, continue to increase in the year of 2014 and represent more than 20 percent of all attacks [Net14]. A recent report in 2018 by Netscout states that DDoS attacks will continue to grow [Net]. According to the report in the year 2018, 1.35 Terabits per second (Tbps) DDoS traffic hit Github, and just after five days of that incident, 1.7 Tbps DDoS traffic launched against an unnamed US-based service provider. These attacks may cause a lot of revenue loss for the companies [Sym]. Indeed, the DDoS attacks caused the downtime of Internet service outages, which cost $221,836 on average in 2018 [Net]. Each company is only able to protect itself to a certain degree since the attacks have become more sophisticated and massive, while the companies' abilities and resources to mitigate such attacks are limited [Rod15]. Furthermore, with improved sophistication employed in DDoS attacks, it is becoming very challenging to design a defense mechanism against these attacks.

Even though many DDoS defense mechanisms are available [ZJT13], they are not capable of competing with some recent types of attacks, and most of these mechanisms became obsolete due to the recent shift of the attacked targets. For instance, SLFA is one of them where the attacker does not attack the target network/servers directly [SP09, KLG13]. The Crossfire attack is an exemplary kind of SLFAs, and it is primarily performed by congesting the communication links surrounding the network of the target servers by sending low-volume traffic over them from many bots in distributed locations. Since the packets the bots send are legitimate, and it does not attack the servers directly, it is very challenging to detect such attacks using traditional mechanisms. The consequence of this attack is the blockage of external access to the servers while they are still active/running without any problem within the network. Such attacks often cause a more significant DDoS impact than direct attacks since, by flooding the targeted links, not only a certain target area can be isolated but also other time critical traffic to or from different regions that may need to pass through the targeted links, will be delayed. Hence, it is critical to propose proactive defense mechanisms against such attacks. Furthermore, the reconnaissance phase of DDoS attacks is underexplored in literature, which is the first step of the DDoS attack chain. Thus, it is important to investigate and develop defense mechanisms against the reconnaissance phase of DDoS attacks.

Over the last few years, Moving Target Defense (MTD), originated from the idea of dynamic networks, has received increasing attention from the research community [NIT09]. The concept of MTD, in which the defense is done dynamically, often proactively, by introducing agility to the network behavior. This agility brings protection to the system by providing resistance, as it complicates the tasks of an attacker by adding inconsistency or confusion in the knowledge of the system. These features can be implemented in various ways, including but not limited to

changing IP addresses of network devices, the operating system of servers, and routing information. Extensive research has been conducted on the implementation of this paradigm with minimum or no disruption introduced to the network services [JASD12, KPB14, JASD15, CSP15, MS15]. The main idea behind MTD is to dynamically change the configuration and behavior of the network in order to deceive adversaries and make it harder for them to launch successful attacks to the networks [CCFB14]. The changes are typically controlled by an automated but centralized system.

The Software Defined Networking (SDN) infrastructure has been shown as a great candidate that can be integrated with MTD for efficient and cost-effective operations to address the problem of network management. SDN decouples the network control and data forwarding functions and provides the ability to have a global overview of the network as well as the capability of zooming in specific network portions to collect further information [HHB14]. It also allows administrators to dynamically adjust network-wide traffic flow to meet changing needs. Additionally, SDN makes network/resource management very convenient and cost-efficient by providing remote configuration and software/security upgrades on the network devices. All these features provide significant motivation for the deployment of SDN in MTD operations for efficiency and convenience. Network Function Virtualization (NFV) proposes virtualization in network components. Thus, NFV enables us to have virtual network nodes that can be easily controlled by SDN [HGJL15], making the networks more agile and reducing the hardware costs. Consequently, MTD can highly benefit from both SDN and NFV technologies.

In this dissertation, we consider the SLFA as our threat model and develop several defense mechanisms against it, utilizing SDN and NFV technologies while employing MTD techniques. In Section 1.1, we present the current challenges of

designing defense mechanisms, and in Section 1.2, we explain our main solutions against given problems.

1.1 Research Problems

In this section, we list the specific research problems we considered in this dissertation and explain each of them as follows:

- Even though there are a few proposed solutions in literature for the SLFA defense mechanisms [WLJW16, WXZ+17, WWL+18], they are based on attack detection, and they mitigate the attack after some problems in the network communication occur. There is a lack of work that can efficiently and effectively mitigate the attack before some harm is made.
- The existing MTD approaches in the literature that strives to obfuscate the network attributes [CSP15][DASJ13] [FTSB15][WW16] typically operate on the real network topologies. This issue raises a problem with the limited available alternate attribute values. For instance, if route mutation is employed, attackers can get the network map since every hop information can be collected each time MTD changes the routes. With limited node size and path alternatives, this is inevitable. Therefore, there is a need for a cost-efficient solution to employ MTD for a higher level of deception.
- The attacker's impact on the service disruption critically depends on the reconnaissance phase, which is usually considered as the first step of DDoS attacks [ARZMT06]. In this phase, the attackers gather information about their target, find out the IP addresses of the target machines, the network topology, and most used links. In particular, the Crossfire attack utilizes this phase to find out commonly used links in network topology and then aims to congest

these links through DDoS attack. Unfortunately, defense against the reconnaissance of DDoS attacks is often underexplored compared to DDoS attack detection and mitigation. Hence, there is a gap in the literature to be filled.

- The forensic components within the current network infrastructures usually have a high cost, and they are not integrated with security features [PJN10a]. Therefore, there is a need for a framework that stores critical information for postmortem forensic investigation while still operating other functionalities timely.
- MTD papers in literature create a dynamic environment by changing some of the features in the network [CSA+20]. However, none of these works consider legitimate users' impact whenever they change the attack surface. Therefore, there is a need for a framework where legitimate users can still troubleshoot the network, while malicious users cannot collect beneficial information.
- MTD brings a high cost to the system in many ways [ASAR16]. There is not any work in literature that considers an optimum strategy to employ MTD for defense purposes. Thus, there is an open question to be investigated, that is how frequently changes should be applied in the network.

1.2 Major Contributions

In this section, we list our major contributions that are proposed within this dissertation.

- In Chapter 4, we designed an SDN-based MTD mechanism to defend against the data phase of the Crossfire attack. We analyze the Crossfire attack reconnaissance phase and utilize the analyzed results to develop the defense mechanism, which in turn reorganizes the routes in such a way that the congested

links are avoided during packet forwarding. The detection and mitigation techniques are implemented using the Mininet emulator [Tea] and the Floodlight SDN controller [Pro]. The evaluation results show that the route mutation can effectively reduce the congestion in the targeted links without making any major disruption on network services.

- In Chapter 5, various network forensics mechanisms are introduced to help in locating the source and types of attacks as a reactive defense mechanism. This chapter considers MTD in the context of an Internet Service Provider (ISP) network and proposes an architectural framework that will enable it even at the reconnaissance phase while facilitating forensics investigations. We propose various virtual shadow networks through NFV to be used when implementing MTD mechanisms via route mutation. NFV proposes virtualization in network components. Thus, NFV enables us to have virtual network nodes that can be easily controlled by SDN [HGJL15], making the networks more agile and reducing the hardware costs. The idea is to dynamically change the routes for specific reconnaissance packets so that attackers will not be able to easily identify the actual network topologies for the Crossfire attack while enabling the defender to store potential attacker's information through a forensics feature. We present an integrated framework that encompasses these features. The proposed framework is implemented in Mininet to test its effectiveness and overheads. The results demonstrated the effectiveness in terms of failing the attackers at the expense of slightly increased path lengths, end-to-end delay, and storage for forensic purposes.

- In Chapter 6, we propose a cloud-based deception framework which aims to confuse the attacker with reconnaissance replies while allowing legitimate uses such as the debugging of network elements. The deception is based on forward-

ing the reconnaissance packets to a cloud infrastructure through tunneling so that the returned IP addresses to the attacker will not be genuine. For handling legitimate requests, we create a reflected virtual topology in the cloud to match any changes in the original physical network to the cloud topology using SDN. Through realistic experimentation on GENI platform [DRS+12], we show that our framework can provide reconnaissance responses with negligible delays to the network clients while also reducing the management costs significantly.

- In Chapter 7, we address the challenge of obtaining the optimal MTD strategy that effectively mitigates SLFA, while incurs a minimal overhead. We design the problem as a signaling game considering the network defender and the attacker as players. A belief function is established throughout the engagement of the attacker and the defender during the Crossfire attack campaign, which is utilized to pick the best response/action for each player. We analyze the game model and derive a defense mechanism based on the equilibria of the game. We evaluate the technique on a Mininet-based network environment where an attacker is performing SLFA, and a defender applies MTD based on equilibria of the game. The results show that our signaling game-based dynamic defense mechanism can provide a similar level of protection against SLFA like the extensive MTD solution, however, causing a significantly reduced overhead.

1.3 Organization of the Dissertation

The dissertation is organized as follows: In Chapter 2, background information about the concepts used in this dissertation is given. The literature review is listed in Chapter 3. In Chapter 4, we investigate how we can utilize MTD in order to de-

fend against the Crossfire attack. In Chapter 5, we present a comprehensive framework for ISPs that provides deception against the Crossfire attackers and forensics capabilities. In Chapter 6, we propose a cloud-based deception framework where we move the management of network deception tasks to cloud providers. We consider an interaction between the attacker and the defender by utilizing the signaling game in Chapter 7. Finally, we conclude the dissertation and present some potential future work in Chapter 8.

CHAPTER 2

PRELIMINARIES

In this chapter, we give a background information related to the technologies used throughout the dissertation. We specifically explain what SDN and NFV are and why they were selected in our projects. Then, we present DDoS attacks and LFAs, and talk about their differences. Finally, we introduce MTD concept.

2.1 Software Defined Networking

Software Defined Networking (SDN) is an emerging technology that proposes a separation of data and control planes, and that is the main difference from the traditional network environment, as shown in Fig. 2.1. On the one hand, the data plane consists of switches that are not capable of any routing/blocking decisions by themselves. On the other hand, the control plane (also known as SDN Controller) is the brain of the network and is responsible for all critical operations in the network.

SDN provides a flexible and cost-effective solution for network management by enriching network devices with programmability feature, and a centralized controller can be used by the network admin in order to configure the network devices. Even though network programming has been investigated for a long time, SDN has been developed and named for a couple of years now [FRZ14]. With the introduction of SDN, it is made possible to do dynamic traffic engineering, drop packets, reconfigure the network routing paths in case of failures and enforce certain policies that make network management convenient and more flexible. Thus, we leverage SDN capabilities to implement our network defense frameworks.

Traditional network devices (switch, router, etc.) require all the software on the hardware in order to run protocols before it can be installed in the network. Mean-

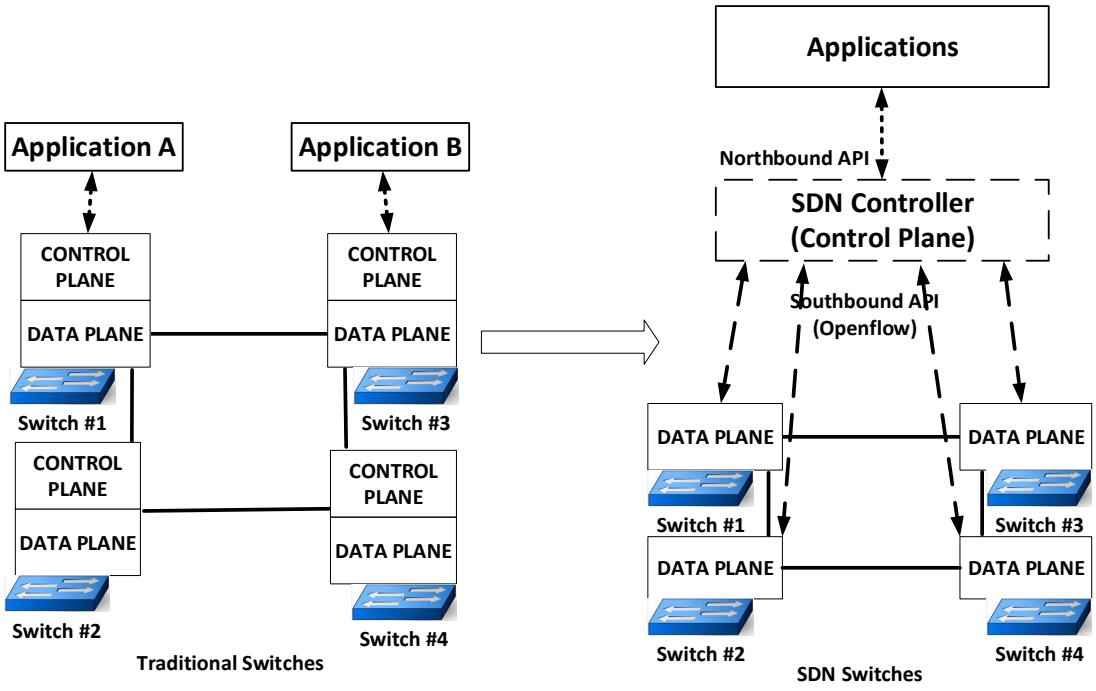


Figure 2.1: Traditional network switches vs SDN-based switches

while, SDN-based switches are basic devices and can be updated easily even after the installation. However, there is a need for a new communication protocol in order for SDN Controller to communicate to SDN switches. Even though there is not a standardized protocol yet, OpenFlow is the widely-used protocol by both researchers and industry [MAB+08]. SDN-based switches are also called OpenFlow switches, and these switches have flow tables that will provide the knowledge of the rules on what to do with each packet. *Flow table*, *OpenFlow Protocol* and *secure channel* between SDN Controller to switches are the main parts of OpenFlow switches. In addition to the mentioned OpenFlow switches and the protocol requirements, there is also a need to run an SDN Controller which network administrators can use to access the network devices remotely. For example, FloodLight [Pro], OpenDayLight [Ope] are publicly available SDN Controller.

The SDN Controller typically runs on a different machine in a centralized location, and only network administrator(s) should have permission to access it. The SDN Controller is able to do many operations, including but not limited to enforce security rules (block some packets, etc.) and decide forwarding tables. By exposing SDN-based solutions to networks, the high cost of network devices and complexities of maintenance of such devices can be minimized. The network administrator can configure and update some parts of the SDN Controller in order to manipulate the network or s/he can implement some applications on top of the SDN Controller to apply his/her own rules. The latter is more common and requires less knowledge of the SDN Controller's source code. Northbound API of SDN Controller is required to be used for this purpose and mostly used protocol is Representational State Transfer (REST) interfaces [Mas11]. The application layer can access network devices through these interfaces. These applications send REST inquiries to get information about the current situation or to update some flow tables. They can contain network functions, including but not limited to Intrusion Detection System (IDS), firewall. The detailed SDN network infrastructure is shown in Fig. 2.2.

2.2 Network Function Virtualization

Nowadays, computer networks consist of many different functionalities, such as firewalls and network load balancing [YWL+18]. These functionalities usually come with the hardware and software parts integrated, which are also called middle-boxes [BCAB15]. These hardware devices are usually costly and do not have much scalability. They are also vendor-specific and need professional labor who have expertise on these specific device set. Therefore, researchers proposed the idea of separation of software functionalities from the hardware devices, which establishes the main

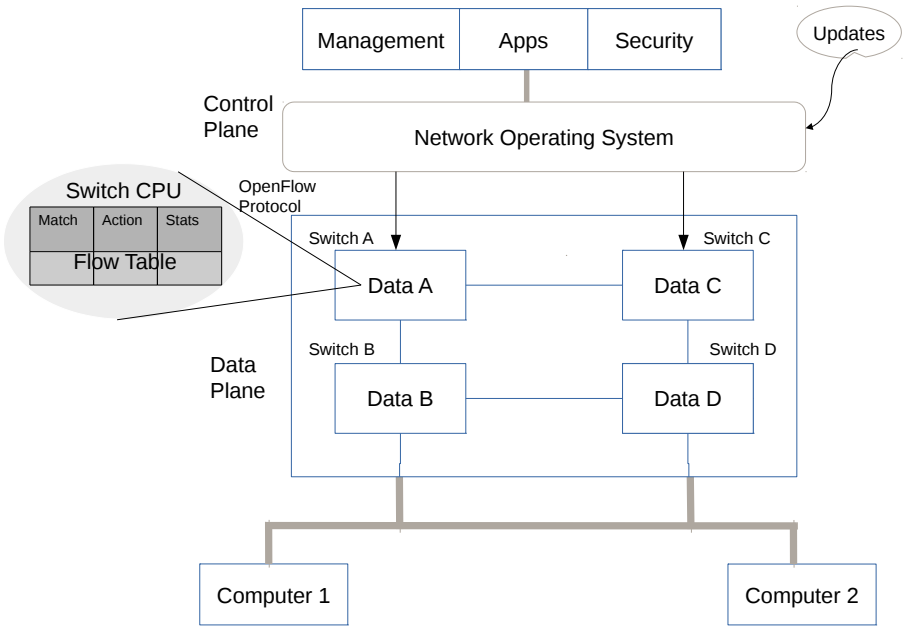


Figure 2.2: SDN Data and Control planes.

idea of Network Function Virtualization (NFV) [HB16]. The software can be implemented on standard servers, and the resources can be shared in these servers. The virtualization of network functions brings high scalability and low operating costs to the network infrastructures [YWL+18]. NFV and SDN are usually considered under the same umbrella since they enhance each other in many ways [MGT+15].

2.3 Denial-of-Service and Distributed Denial-of-Service Attacks

Denial-of-Service (DoS) attacks are a critical threat to current networks [DM04]. The aim of these attacks is to disrupt the services in the targeted server. DDoS attacks have the same goal as DoS, while a high number of coordinated Internet hosts are engaged in these attacks, differently [MR04]. DDoS attacks exploit the current network structure and communication protocols. These attacks can be in

many forms, including but not limited to flooding attacks [WZS02], amplification attacks [KHRH14], protocol exploit attacks [KK06] and malformed packet attacks [DM04]. Even though there are many DDoS attack types and mitigation techniques, our main focus in this dissertation is Link Flooding DDoS attacks.

2.4 Link Flooding DDoS Attacks

Direct DDoS attacks attempt to attack the target host specifically while indirect attacks attempt to isolate the area surrounding the target host and make the target host services unavailable. Link Flooding attacks (LFA) are one type of indirect DDoS attacks. Even though the concept of LFA has a long history, the two very recent LFAs, the Coremelt attack [SP09] and the Crossfire attack [KLG13], showed the significance and potential impact of these attacks. Hence, we proposed different mitigation techniques specifically against the Crossfire attack, which is explained in detail in Section 2.4.1.

2.4.1 The Crossfire Attack

The crossfire attack is an indirect attack where an attacker tries to isolate a specific area by performing link-flooding DDoS attacks to some targeted links so that the servers within that area are unable to provide their services. Fig. 2.3 illustrates an example of the Crossfire attack to a specific area. By flooding all targeted links, the targeted area becomes isolated from the rest of the network.

Basically, the Crossfire attack uses different pairs of the source and destination for reconnaissance and attacks. For reconnaissance, it picks a target server, while for the attack, it picks at least one decoy server that are located around the same target area. A sheer number of attack agents (i.e., bots), which are widely available

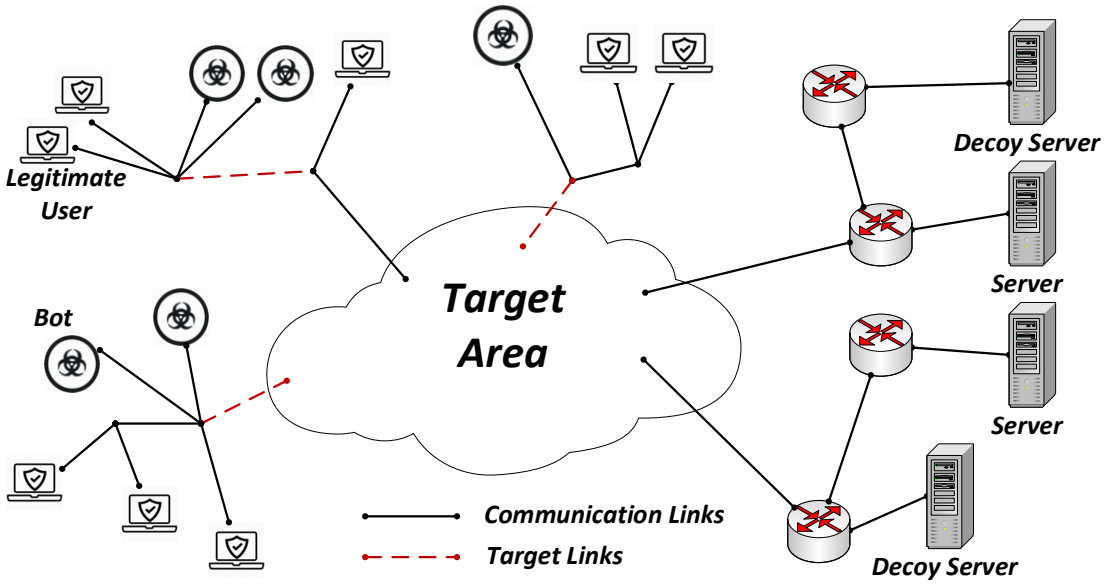


Figure 2.3: An example of Link-flooding DDoS (the Crossfire) attack target area

on the Internet, is used during the reconnaissance phase for creating the link-map of the network. These attack agents can be inside the ISP network or from outside the ISP network. All chosen attack agents will send *traceroute* messages to this target server to build the link-map of the network. After collecting adequate information to build the link-map, the attacker determines the *target links*. He/she will then try to find other pairs of attack-agents and decoy server(s) by ordering attack-agents to send *traceroutes* to the decoy server(s). The pairs of attack-agents and decoy server(s) whose path also passes through the selected target links will be used to launch the link-flooding attacks to the target links.

This type of attack is difficult to detect since the source-destination pairs that are used at the reconnaissance phase, and those are used at the attack phase can be different. Moreover, in order to avoid any detections at the attack phase, a plethora of coordinated bots send a low-rate flow to exhaust the available bandwidth on the selected links.

2.5 Moving Target Defense

In 2009, five new game-changing directions have been introduced to address some intractable problems in the digital infrastructures [NIT09]. These directions strive to solve these problems from a different perspective instead of the typical traditional approaches that tackle them directly. One of these directions called Moving Target Defense (MTD), was driven by the fact that the digital infrastructure settings are relatively static for a long period of time. For example, once the computer that we use every day has been installed with an operating system (OS) and assigned an IP address, these settings are barely changed. These conditions enable attackers to have unlimited time to perform any stage of the five-phase attack kill chain [OHBS14]. MTD enables defenders to build proactive self-defense mechanisms that dynamically change the digital system attributes while still ensuring the system accessibility for legitimate users [Man13]. These self-defense mechanisms introduce the redundancy and diversity in the system to make the attack surface unpredictable for attackers. For instance, instead of using a single OS platform all the time, a computer can dynamically rotate its OS to a different OS platform at a random time interval. This way, it will make it harder for attackers to perform their attacks since their insight of the digital system from their previous attack attempt may become obsolete since the OS platform has changed.

CHAPTER 3

LITERATURE REVIEW

In this chapter, we examine the related work of the studies presented in this dissertation. We first explain research papers proposing defense mechanisms against SLFAs, and later we discuss signaling games paper for cybersecurity.

3.1 Defense Mechanisms Against DDoS Attacks

DDoS attacks can come in various forms, and many defense mechanisms against these attacks have been proposed [ZJT13], including the use of emerging technologies such as NFV [JYR+16, FTSB15] and SDN [FTSB15]. For example, Jakaria et al. [JYR+16] proposed a defense mechanism against DDoS attack called VFence, which leverages the capability of dynamically allocated virtualized network functions (VNFs) on the commodity servers for defense when the system is potentially under SYN Flood attacks. Similarly, Bohatei [FTSB15], a flexible and elastic DDoS defense system, utilizes both NFV and SDN for defense against DDoS attacks such as UDP flood [TCB17], DNS amplification [AKK+13], SYN flood [BSR13], and Elephant flow [CY15]. However, these approaches are not effective for an indirect attack which is the main focus in the dissertation.

3.2 Defense Mechanisms Against Link Flooding Attacks

A number of defense mechanisms exist in the literature for the link-flooding DDoS attacks. They differ in a number of ways such as when they perform the detection and mitigation (e.g., proactive or reactive), how they detect the target links, and how they identify the attack agents. Most existing defense mechanisms are reactive

approaches that strive to detect and mitigate during the attacks (i.e., when a link congestion occurs) [LKG13, Gko14, GKLD16, XLCZ14, LKD16, XML+18].

The target links during the attacks can be identified by the anomalies of path performance metrics such as packet loss rate, round-trip time (RTT), jitter, connection failure rate, and available bandwidth. These anomalies are then correlated with the *traceroute* data to infer the target links or area [XLCZ14]. The defense mechanisms identify the attack agents by enforcing them to behave suspiciously by not complying to certain actions such as rerouting requests to follow the preferred paths or rate control requests [LKG13]. In [LKG13], even though collaboration between ASes is shown helpful whenever an SLFA hits, it is not clear how to manage different ASes to work together.

Meanwhile, authors in [GKLD16] propose Traffic Engineering (TE) as a solution to mitigate an SLFA. In their solution, the defender forces the attacker to use the improbable path (i.e., very unlikely to be used) so that the attacker ends up being identified. Similarly, observing traffic patterns while attack happens is applied to detect and defer an SLFA in [XML+18]. The main problem with these solutions is that they are reactive and some critical harm could have been done before the attack is mitigated. Traffic engineering based solution is also used by authors in [WLJW16]. The authors suggest upgrading switches to SDN-based switches in order to detect and mitigate an SLFA. However, it is not specified how to upgrade switches in run-time and how SDN switches are capable of detecting such attacks. [WXZ+17] observes link-probers by checking the packets at the ingress port. If a sender is found as link-prober, then Linkbait applies MTD to confuse its route. They suggest matrix-based feature extraction in order to detect which link-prober is a bot. In [WWL+18], authors propose utilizing SDN capabilities to detect the

attacker's activities and block the malicious activity. Yet, it may not be possible to detect which are the malicious activities.

While our solution is similar to [WXZ+17] and [WWL+18], we do not rely on identifying bots which could not be possible with an intelligent adversary model.

3.3 Defense Mechanisms Against Network Reconnaissance Attacks

While there are lots of proposed defense mechanisms against DDoS attacks [SGT+11], most of them do not consider defending at the first phase (i.e., the reconnaissance phase) of the DDoS attack kill chain [HKP13]. Typically, the DDoS defense mechanisms that take into account the first phase of attacks are based on MTD approaches such as port hopping [LWC14], address shuffling [CCFB14, JASD15], and path mutation [KPB14], which are used to obfuscate the attackers' views of the attack surface. Furthermore, the authors in [WW16] suggest applying MTD for network reconnaissance attacks using SDN. They have reflector machine in their framework and utilize Snort Intrusion Detection [R+99] system in order to detect possible network reconnaissance packets and forward them to the shadow network.

However, our focus within this dissertation is the LFA. Only a few of the proposed defense mechanisms consider defending against reconnaissance phase of the attack [HTS15] [MTL+18] which is the first phase of Link Flooding Attack kill chain [KLG13]. Authors in [HTS15] claim traceroute packets are increased before an SLFA occurs and they design a detection mechanism for the attacker based on that assumption. Even though we also have a similar belief that traceroute packets will increase before the attack, we do not rely on this feature individually and our attack mitigation technique is different. Meier et al. obfuscates the attacker's

reconnaissance by replying with some virtual hops that is consistent with the network topology [MTL+18]. While this solution aims to prevent the attacker to gain critical information, it gives wrong information to legitimate users. We argue that this approach is not the best practice since a legitimate user needs to get the right information in order to utilize the network facilities in the most efficient way. Otherwise, he might use longer path, and suffer higher transmission delays. In [AMR05] automated ICMP packet recognition by using SNORT Network Intrusion Detection System (NIDS) is introduced. We also use the similar idea but utilizing SDN capabilities to have NIDS feature integrated with forensics framework.

We utilize the idea of shadow networks but differently, in our work, we do this through NFV to eliminate any costs. In addition, we do not utilize a scan network. Instead, we assign rules to the network switches to forward scan traffic (ping, traceroute, etc.) to either shadow network (mirror or overlay) or just do the route mutation on those scan traffic packets. Therefore, our goals, targeted attacks and methods are different from these works.

In our earlier work [ASA18], we proposed to utilize VNFs in the commodity servers to increase route variability of the route mutation to obfuscate the attacker's view of the attack surface when they perform reconnaissance attempts before they launch any DDoS link flooding attacks. Our work differs from these studies in two major ways: First, we rely on the cloud as a service to launch our MTD defense, which was not used before. Second, we consider network troubleshooting capabilities and allow them to be pursued when MTD is applied.

In our latest work, we propose to use the NFV-based cloud instead of using the commodity servers in order to reduce the cost of the defender in terms of the commodity servers investment, increase the route variability, and increase flexibility due to the ease of deployment to various cloud locations. Furthermore, we also

provide network troubleshooting to work correctly, which has been missing in earlier works.

3.4 MTD Against Network Attacks

MTD research has been around for a while [CCFB14] [JSS13] and there are several MTD approaches that have been proposed against DDoS attacks [CSP15, DASJ13, RGAS$^+$16, JASD13a]. The main purpose of these approaches is usually to increase the attacker's costs of collecting the network information by dynamically updating the network attributes (e.g., IP addresses, port numbers, and routing information).

A number of network path mutation techniques to defend against network attacks (e.g., reconnaissance, eavesdropping, and DDoS attacks) can be found in the literature [DASJ13, JASD13a, CSP15, RGAS$^+$16]. These techniques are mainly differed based on the number of constraints and how they select a pool of multiple paths between two communicating nodes. A path is randomly selected if more than one eligible paths are available in the pool.

Duan et al. [DASJ13] proposed a proactive random route mutation (RRM) technique that considers the route selection as a constraint satisfaction problem and uses the Satisfiability Modulo Theory (SMT) [BSST09] solver to find all eligible paths that: (1) should not include any already overloaded nodes or links (i.e., capacity constraint), (2) minimize the reuse of any links or nodes from the recently used paths (i.e., overlap constraint), and (3) maintain the required quality (i.e., quality of service constraint). The proposed RRM technique can be implemented on both traditional and SDN-based networks as well as on an overlay network. Jafarian et al. [JASD13a] incorporated game theory in the RRM technique for RRM-aware attackers and added a security constraint, which takes into account any previously used

access control policies, in the selection of the new eligible paths. The interaction between an RRM-aware attacker that selects and attacks a route at each attacker mutation interval and an RRM defender is modeled as a static game of complete information. Rauf et al. [RGAS+16] proposed an efficient and decentralized End Point Route Mutation (EPRM) to protect against infrastructure level attacks without reconfiguring the infrastructure devices. EPRM operates on the presence of peers (i.e., end-hosts with virtual router capability), which are geographically distributed across the network, as the intermediate nodes in a virtual path between a source-destination pair. The aim of EPRM is to ensure the end-to-end resilient reachability of a source-destination pair by finding the mutation set (i.e., the potential set of virtual paths) and the mutation sequence that meet the QoS constraint (e.g., the number of intermediate hops, the available link bandwidth, and the intermediate peer load) and the resilience constraint (e.g., minimum link overlap). While our work also takes into account the infrastructure level attacks, it is different from this work since we proposed a mixed route mutation strategy that considers an overlay network as one of the route mutation strategies instead of only mutating virtual paths as in this work. Moreover, we also exploit virtualization and network simulator to provide a simulated network environment running on a virtual machine to increase the route variability.

Due to the importance of the network reconnaissance toward successfully staging attacks later, a number of MTD approaches that strive to harden this attempt has been proposed recently. The aim is to obfuscate an attacker's efforts to collect the states of an attacker's potential targets by dynamically changing the network attributes over time such as the network addresses, TCP/UDP port numbers, and network paths [CSP15]. Chaves et al. [CSP15] use a breadth first search algorithm to find all possible paths between two communicating devices and then select a

path randomly among these paths. While these works can be classified as a route mutation from a single source to a single destination, our work is different since we propose a single source to multiple destinations route mutation approach due to the use of multiple (virtual) shadow networks. Moreover, these works are intended for direct DDoS attacks towards enterprise network while our work focuses on indirect DDoS attacks to enterprise networks that strive to isolate the surrounding area of the target enterprise networks.

MTD's use with SDN has only started to receive attention recently [JASD12, KPB14, JASD15, CSP15, MS15]. These works focused on the change of different settings in the considered network.

For instance, [JASD12, JASD15] investigates changing the hosts IP address by involving the DNS interactions. A similar problem was studied in [MS15]. A host-based SDN approach is followed in [MS15] to allows defenders to distinguish between trustworthy and untrustworthy clients. Another recent work in [CSP15] considers the mutations on both the IP addresses and hosts and investigates their impact on the network operation. However, none of these works focus on defending against reconnaissance attempts.

3.5 SDN-based Forensics

For forensics analysis, there are only a few works in the literature that are related to SDN framework. For instance, the motivations of pursuing forensics by using SDN has been mentioned in [KGW+16]. Authors in this work propose an SDN Controller as a potential location for forensics operation in SDN. They also talk about using application or infrastructure level forensics alternatively. Our solution differs from this work by adding a middle-box level as forensics location into the

system while also comparing our solution to Controller-based forensics. Bates et al. [BBH+14], propose an SDN-based forensics system for byzantine attack for the data center network. This system employs secure network provenance that can operate in untrusted environments. The system uses Provenance Verification Point, an SDN middlebox, to collect forensics information from a group of switches. While our approach also uses a similar idea by using VCP for collecting network traffic, our work extends this collection point for traffic filtering and the suspected traffic detection that will subsequently trigger the MTD application.

3.6 Signaling Games for Network Defense

In many cybersecurity scenarios, the defender cannot clearly detect whether the received messages are coming from a benign user or they are a part of the attack scenario and are initiated by attackers. Signaling game is a two-player incomplete information game that has been used to model different cybersecurity problems, such as intrusion detection [SLXC11] or deception [ZBA10]. Furthermore, the authors model cybersecurity in terms of signaling games in [CMN+14] and W. Casey et al. model deception with signaling game in [CKM+18]. They present how signaling games provide a formal mathematical method to analyze the way of identity and deception coupling in cyber-social systems. The game-theoretic framework can be extended to reason about dynamical system properties and behavior traces. In [MMMZ16], the authors formulate a deception with signaling game in networks in which the defender deploys a fake avatar for identification of the compromised internal user. In [RMAS13], the authors propose a selective and dynamic mechanism for counter-fingerprinting. They model and analyze the interaction between a fingerprinter and a target as a signaling game. Following this work in [ZLG17], the

authors suggest changing attack surface (e.g., port numbers) depending on a belief that is observed in the signaling game. In [MMZ15], the authors investigate the interactions between a service provider and a client by signaling game, where the client does not have complete information about the security conditions of the service provider. In [AMMR17], the authors propose a moving target-based deceptive defense mechanism using a signaling game for the frequency of migrations of the virtual machines in clouds. While we also apply signaling game for network defense, we propose using RRM which has not been studied under a signaling game. Furthermore, our framework is applied not only to reconnaissance attacks but also real data phase of the attacks. The reason behind choosing and applying the signaling game approach specifically for SLFA is that we believe the dynamic and unnoticeable behavior of SLFA can be modeled into a signaling game accurately.

CHAPTER 4

SDN BASED MTD AGAINST THE DATA PHASE OF SLFA

In this chapter, we propose an SDN-based MTD for proactive (i.e., before the attacks) and reactive (i.e., during the attacks) defense against crossfire attacks to a certain area. Before the attack, our approach strives to obfuscate the link-map construction at the reconnaissance stage of the crossfire attacks. At this stage, an attacker tries to construct the link-map of the network using a route tracing utility for computer network diagnostic, such as the *traceroute* utility program. This program shows the route taken by packets from source to destination. This link map is then used to determine the targeted links. Typically, an attacker will try to find stable routes by issuing multiple *traceroutes* to the same destination to determine the route stability and multiplicity in order to select effective target link(s) [KLG13]. Obviously, when the routes are always changing in response to multiple *traceroutes*, it will be difficult for the attacker to find stable routes and determine the targeted links.

Our proposed approach will first perform dynamic *traceroute* profiling by collecting the time series of *traceroute* activities in the network. When excessive number of *traceroutes* are detected, based on the collected *traceroute* profiles, the potential targets of crossfire attack as well as source-destination pairs for *traceroute* generation can be identified. During the attack, when a link is heavily congested, the source and destination pairs from the existing traffic that pass through the congested link(s), will be compared with the collected *traceroute* profiles by the SDN controller. For the traffic that passes through the congested links and the sources have the history of sending *traceroute*, such traffic will be diverted to different routes. The new routes can easily be generated at the SDN controller by excluding the congested links.

We implemented the proposed approaches in a Mininet [LHM10] environment using Floodlight controller [Pro]. We generated ICMP packets for *traceroute* operations and then launched the attacks on a set of selected links. The evaluation results indicated that changing the routes as soon as the attack is detected provides significant relief for the congested links and enables service to the target server, particularly when TCP is used. The data delay for the transmitted packets increases slightly due to the use of different and perhaps non-shortest routes.

This chapter is organized as follows. In Section 4.1, describes the background information as well as the problem definition. In Section 4.2, we introduce the proposed MTD approaches and how we perform route mutation. Detailed performance evaluation of the proposed work is given in Section 4.3. We conclude the chapter in Section 4.4.

4.1 Preliminaries

In this section, we explain how *Traceroute* works and define our problem that is targeted within the chapter.

4.1.1 *Traceroute* Operation

Basically, *traceroute* uses *Time To Live (TTL)* information in the IP datagram to trace a route from source to destination [Koz05]. Multiple packets with increasing TTL value are sent to a legitimate destination but with a bogus port number. By default, three packets are sent for each TTL value, as illustrated in Fig. 4.1. These packets can be UDP packets, TCP packets, or ICMP packets. The number of packets also can be adjusted by users. Initially, three packets with a TTL value of 1 are sent to the destination. When these packets arrive at the first router, the

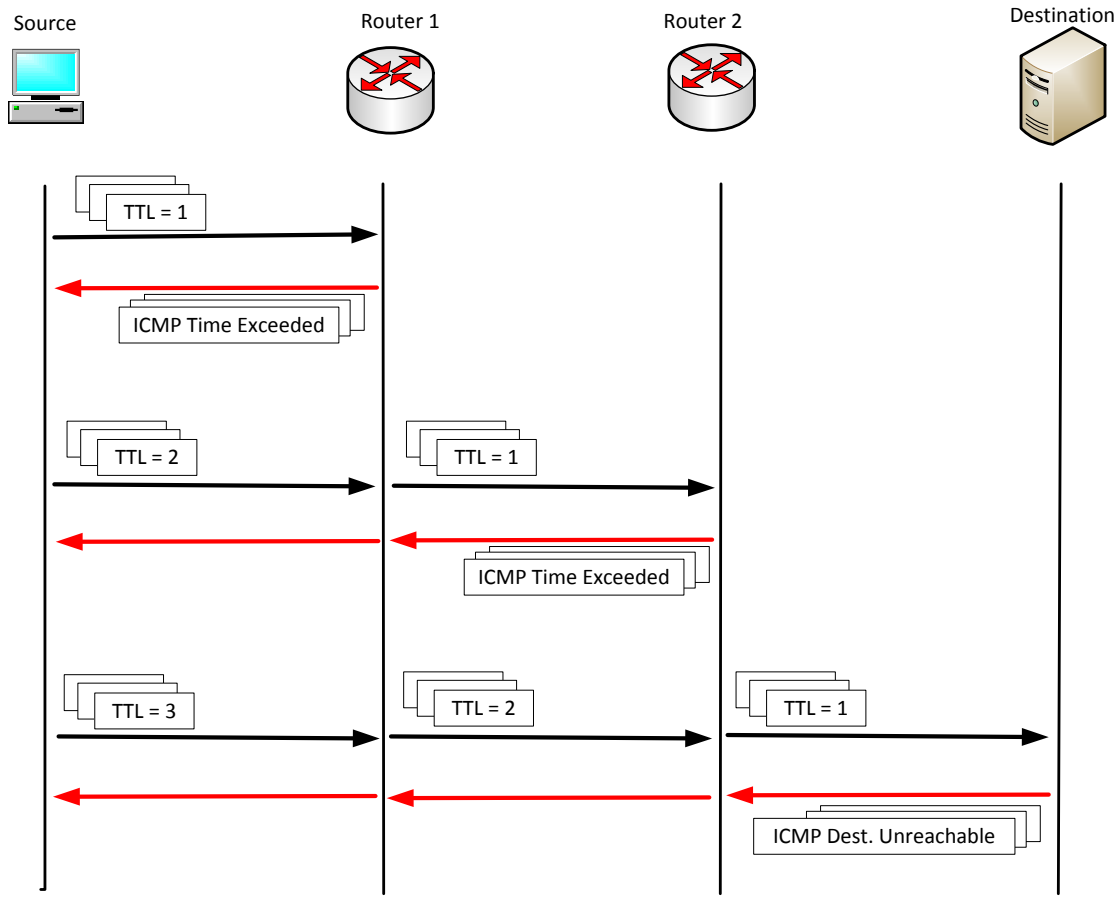


Figure 4.1: Example of traceroute operations when there are two intermediate routers between the source and destination.

router will decrement the TTL value. Since the new TTL value is now equal to zero, the packets are dropped, and three ICMP *Time Exceeded* reply messages are sent to the source. Besides knowing the router information, the source will also know three elapsed times from source to the first router. Subsequently, the source sends three more packets with a TTL value of 2. The first router will decrement the TTL value and forwards them to the next router. At the second router, the TTL value becomes zero when the router decrements it, and thus the packets are dropped, and three ICMP *Time Exceeded* reply messages are sent to source through the first router. The source will send three packets with a TTL value of 3 again, and so on until finally, they reach the destination. The destination will reply with ICMP *Destination Uncreachable* message since the port number of the destination in the UDP packet is a bogus port number. Thus the destination is unable to find this port. Note that one or more packets may be lost during the transmission, and thus, either the sender or receiver (intermediate router or final destination) will not receive all three packets.

4.1.2 Problem Definition

The problem definition can be divided into two parts: (1) the prevention problem for the link-map construction and (2) the detection and mitigation problem during the attack. During the link-map construction, an attacker strives to find the persistent links which become the candidate for the targeted link-flooding DDoS attacks. The persistent links of a route from a source to a destination are links that always present each time a pair of attack agents perform reconnaissance, while transient links of a route are links that may not always present. Typically, the transient links happen because of the implemented routing protocol operations, such as load-balancing

routing, which may cause a route from source to destination to change every time. Therefore, at the prevention stage, the problem that needs to be addressed is *how to identify and obfuscate the link map construction effort so that the cost of attacking the targeted links is high and makes it less attractive for an attacker.*

Link flooding DDoS attack is difficult to detect since an attacker uses a huge amount of bots that send low-rate persistent small traffic through the targeted links. This way, the targeted links will be flooded with a huge amount of low-rate small traffic from a plethora of bots. To identify the bots that are sending these traffic, typically, the defender will request sources to confront a certain request, and when the act is suspicious (i.e., not following the request), these sources can be considered as bots and further actions can be performed for these suspects. Instead of bots detection, our goal is to use the same traceroute profiles that are collected in the prevention stage, for the detection and mitigation during the link-flooding DDoS attack stage. Hence the problem definition would be *how to do detection and mitigation against the Crossfire attacks using SDN by utilizing the traceroute profiles collected from the network.*

4.2 Proposed SDN-based MTD Approaches

Our proposed approaches consist of two defense mechanisms: (1) obfuscating the links during the potential link-map creation of the attackers to make it harder to launch the attacks (i.e., proactive stage), and (2) detection and mitigation during the attacks (reactive stage). To perform these defense mechanisms, we relied on the abilities of SDN controller and OpenFlow protocol. Specifically, four inter-related SDN application modules are designed and deployed, as depicted in Fig. 4.2: (1) ICMP monitoring, (2) *traceroute* profiling, (3) route mutation, and (4) Congestion-

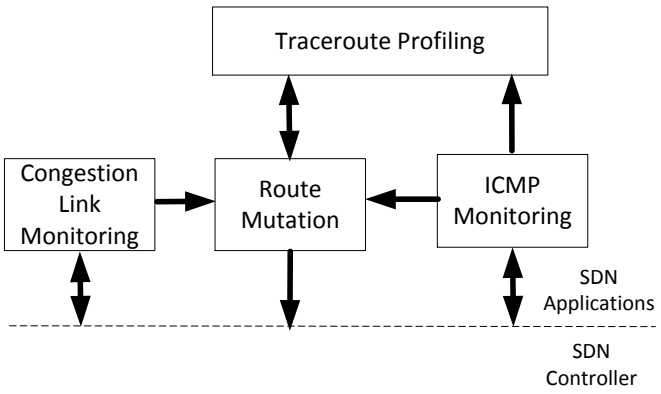


Figure 4.2: The proposed SDN-based MTD modules

Link monitoring. The first three modules are required for the proactive stage, while the last three modules are needed for the reactive stage. The detail of each module is explained next.

4.2.1 ICMP Monitoring Application

ICMP Monitoring continuously monitors the presence of ICMP packets by requesting every ICMP packet to be sent to the controller through *packet-in* messages of OpenFlow. On receiving these ICMP packets, the ICMP Monitoring application tries to determine whether the *traceroute* operations are present in the network by looking at the following patterns: (1) there are multiple ICMP echo packets with an increasing number of TTLs from a source to a certain destination, and (2) there are multiple ICMPs with TTL exceeded information packets to a specific destination followed by ICMP with destination unreachable to the same destination. When these patterns are detected, ICMP monitoring application sends the *traceroute* information to *traceroute* profiling application.

4.2.2 Traceroute Profiling Application

On receiving *traceroute* information, the proposed application builds the *traceroute* profile database that consists of the following information: source IP address, destination IP address, timestamp, the closest SDN switch to the source IP address, the closest SDN switch to the destination IP address, and all intermediate SDN switches between source and destination where this *traceroute* is detected. Note that the traffic may be generated from outside the AS, and thus the closest devices here can be switches at the boundary of the network.

Since the ICMP packets can continuously arrive at the SDN controller and the database can be updated anytime, this application will try to identify the potential target links within a T time interval when an excessive number of *traceroute* attempts are detected. *Traceroute* profiling module uses the *traceroute* profile database to identify the potential target links that can be attacked based on the number of *traceroute* that may pass through the common links within that T time interval. When such common links exist, the *traceroute* profiling module informs the route mutation module to find all alternate routes that will not pass through these potential target links and then sets rules of the SDN switches to randomize the routes for ICMP packets from suspected sources.

When the route mutation operations are active at the SDN switches, the SDN controller can decide to terminate the route mutation on the switches when the SDN controller detects that the number of *traceroute* attempts have dropped significantly within the T time interval.

4.2.3 Route Mutation Module

From the *traceroute* profiling module, the potential target links and suspected sources can be identified. Route mutation module strives to find all possible routes from suspected hosts to the destination by using the Algorithms 1 and 3. These algorithms are implemented at FloodLight SDN Controller. Algorithm 1 takes to start and end hosts, and the potential target links that should be avoided. By using these input parameters, it finds all possible routes available. Essentially, this is providing link-disjoint routes from a source to destination.

Algorithm 1 Find All Routes From *start* to *end*: GetAllRoutes(*start*, *end*, *dontUse*)

Require: *start*: Start of the route
Require: *end*: End of the route
Require: *dontUse*: List of routes not to be used
Ensure: *routes*: List of found routes
1: *rights* ← getLinks(*start*)
2: **if** *rights* != null **then**
3: **while** *right* in *rights* **do**
4: **if** *dontUse* not include right **then**
5: *route* ← *start* {add switch to route}
6: *route* ← *end*
7: Chain(*routes*, *route*, *right*, *end*, *dontUse*)
8: **end if**
9: **end while**
10: **end if**
11: **return** *routes*

As an example of these algorithms, by considering our experiment topology in Fig. 4.3, let us assume there is congestion at the link between switches S3 and S5. First, the most source-destination packet transmission of this link will be figured out. Let it be H1 to H4. Then Algorithm 1 will be called with H1 as start node, H4 as the end node, and *dontUse* list is just the link between S3 and S5. In this algorithm, all possible links that can be followed just after H1 will be considered

Algorithm 2 Recursive function to find all possible route from given $rightMost$ to end): Chain($routes, route, rightMost, end, noUse$)

Require: $routes$: List of all routes
Require: $route$: Current route
Require: $rightMost$: The last switch added to the list
Require: end: End of the route
Require: $noUse$: List of routes not to be used

```
1: if rightMost = end then
2:    routes ← route {add route to the list}
3:    return
4: end if
5: rights ← getLinks(rightMost)
6: if rights! = null then
7:    while right in rights do
8:       if !route.contains(right) AND dontUse not include right then
9:          newRoute ← right
10:         Chain(routes,newRoute,right,end,noUse)
11:      end if
12:   end while
13: end if
```

Algorithm 3 Find All Possible Links From Given Switch, Return all outport links given $switchToLook$ has: GetAllLinks($switchToLook$)

Require: $switchToLook$: Switch to find all links
Ensure: $allLinks$: List of all links from given switch

```
1: while there is switch not checked do
2:    if currentSwitch == switchToLook then
3:       while until there is a link do
4:          allLinks ← currentLink
5:       end while
6:    end if
7: end while
8: return allLinks {all possible links from switch}
```

as potential routes and recursive function Algorithm 2 will be called to see whether they can go until the destination host or not. In Algorithm 2, if a route is found, it will be added to the list by verifying that it does not include any links from *dontUse* list of links, link S3-S5 in our case. After running the Algorithm 1, it will return all possible routes, which are H1 - S1 - S2 - S4 - S6 - S9 - H4 and H1 - S7 - S8 - S9 - H4 in our example.

4.2.4 Congested-Link Monitoring

Concurrently with ICMP monitoring, Congested Link Monitoring is basically performed at the SDN Controller. Whenever the SDN Controller detects a congested link, the source-destination pair that is using this link will trigger route mutation for itself. The controller will consult the *traceroute* profiling and see whether there is the previous history of the pair of source-destination that also sent *traceroute*. If there are, then the controller will set new rules to all switches for the route picked randomly among all possible ones.

4.3 Experimental Evaluation

This section describes the experiment setup, performance metrics, and baselines and discusses the performance results.

4.3.1 Environmental Setup

We evaluated the effectiveness of our proposed MTD mechanisms through an emulator. As the SDN and OpenFlow emulation, we used Mininet [LHM10], which is widely used in SDN research. For the SDN Controller, we used FloodLight [Pro] con-

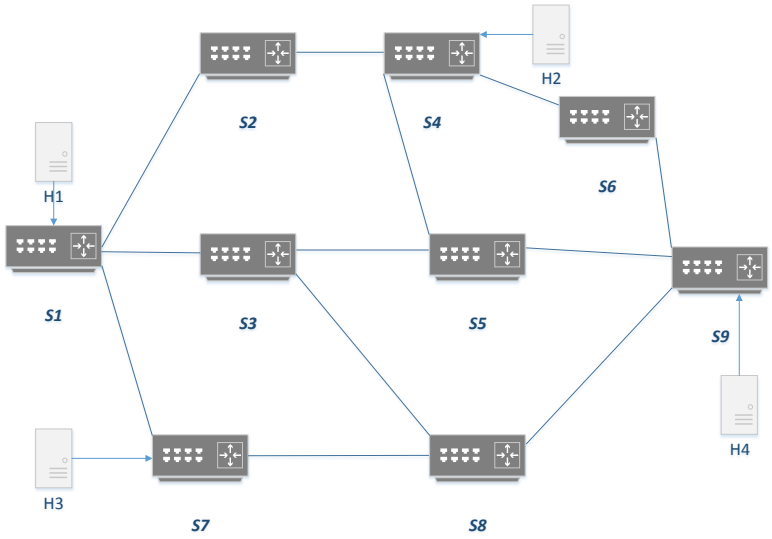


Figure 4.3: Our network topology

troller. For our experiments, a module is implemented at the FloodLight controller. The *packet-in* handling and route mitigation is performed by using this module. The *packet-in* feature of SDN enables directing the packets to the controller so that some processing can be done.

We created a small network topology to be used for our tests. The topology has 9 Mininet switches and 4 hosts where we considered 2 clients (H1 and H2) and 2 servers (H3 and H4) talking to each other, as shown in Fig. 4.3. That topology is chosen to show our alternative routes and route mitigation between clients and servers easily. We defined the routes before the transmission starts as follows:

- From H1 to H4: H1 - S1 - S3 - S5 - S9 - H4
- From H2 to H3: H2 - S4 - S5 - S3 - S8 - S7 - H3

In each experiment, this topology is created first on Mininet. For the traceroute profiling phase, all the switches are set the rule to drop all TCP/UDP packets so that we can initially only allow ICMP packets. The switches are set with the rule for

ICMP packets with output to FLoodlight Controller as OpenFlow *packet-in* message and to an output to port, which is defined manually, before actual data simulation starts. In this experiment, ICMP packets are generated and transferred between hosts for one minute. While ICMP transfer is going on, we set the time duration to 10 seconds for the Floodlight Controller to check the total number of ICMP packets each link is receiving.

After this phase, real data transfers are started and maintained for 5 minutes. At the beginning of this phase, the Floodlight Controller set a rule for each switch so that whenever it receives a TCP/UDP packet, it will forward it to both the Floodlight Controller as *packet-in* and to the next hop according to route that is defined at the beginning of the simulation. Two of the hosts, H1 and H2, generated data packets of size 16 bytes every 100ms. We implemented a client-server Python application for this end-to-end data transfer. Again FloodLight Controller checks the link usage every 10 seconds. Link usage will be calculated based on the number of *packet-in* messages received at the Controller. The threshold for route mitigation is set to 100 packets since this is the total bandwidth that can be provided for the links defined in Mininet.

4.3.2 Performance Metrics

As the baseline approach, we consider the case where we do not apply MTD and continue with the initial network configuration. In our case, routes between the hosts are set at the beginning of the experiment and are not being changed until the end of data transmissions. For the MTD case, as explained before, there will be route updates when there is a risk for crossfire attacks. We represent these cases as MTD and No-MTD in the graphs.

We consider the following metrics to assess the performance of the above approaches:

- *Average Link Usage*: This is defined as the average of all the link usages in the topology. The link usage is measured by the total number of packets passing on a link. This includes both the sent and received packets. Our goal is to minimize or balance the link usage so that certain links will not be congested.
- *Average end-to-end Delay*: This is the packet delay from a source to destination considered in our topology. The delay includes everything from packet generation at the source to packet arrival at the destination server. Our goal is to assess if applying MTD is dramatically affecting the packet delays in the network.

4.3.3 Experimental Results

We analyzed our experimental results in three different categories as TCP experiments, UDP experiments, and impact of the frequencies that we used to run MTD.

TCP Experiments

We first conducted experiments by using TCP. When MTD takes place, there is a significant reduction in average link usage in the network, as can be seen from Fig. 4.4. In some cases, this reduction is six times compared to that of the Non-MTD traffic. In all cases, there some reduction, which indicates the effectiveness of the route mitigation approach. In addition, the route mutation approach provides some load balancing among the link usages of different links. The usage in the first three cases are almost similar, and only in the case of S5.P2 and S5.P3, there is a slight increase, which is normal.

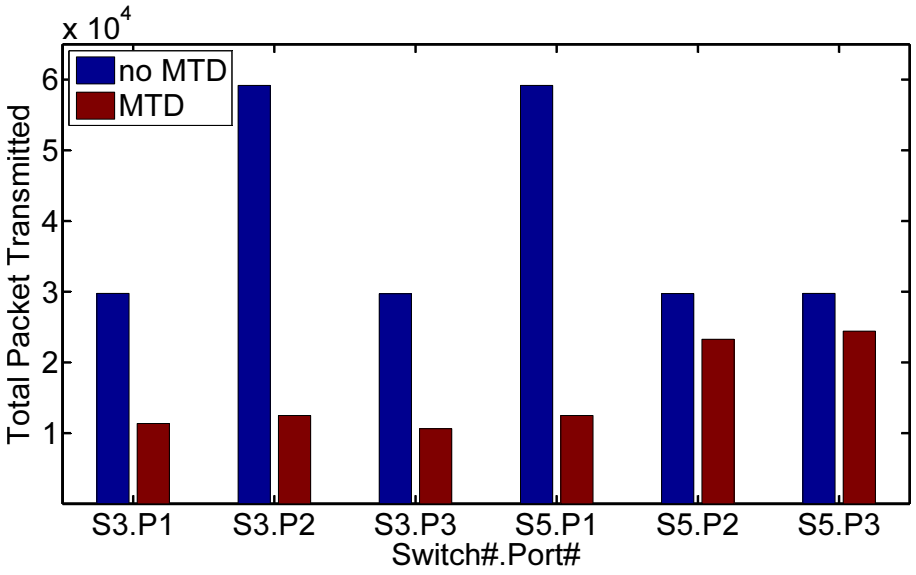


Figure 4.4: Average Link Usage with and without MTD for TCP Transmission. S3.P1 means port 1 at switch id 3, S5.P2 means port 2 at switch id 5 and so on.

We then looked at the average delay performance to see if the proposed MTD approach made any significant impact on this metric. The results are shown in Fig. 4.5. We can see that there is an increase in delay for the packets arriving at both servers. There are two reasons behind this: First, whenever SDN-based MTD is used, and route mitigation is done, the routes are changed. As TCP is a connection-oriented protocol, such route changes will force TCP to re-establish the connections with the servers. This will introduce an additional delay for some of the packets at the time of route changes. Specifically, we observed that some of the packets needed to wait for significantly increased times (e.g., the delay is about 900ms while the average is about 1.5ms). This actually causes the average delay of MTD to be more than that of the Non-MTD case. Second, since the routes are changed during the process, the new routes may not always be the shortest path routes in terms of packet delay. An increased number of hops, for instance, may

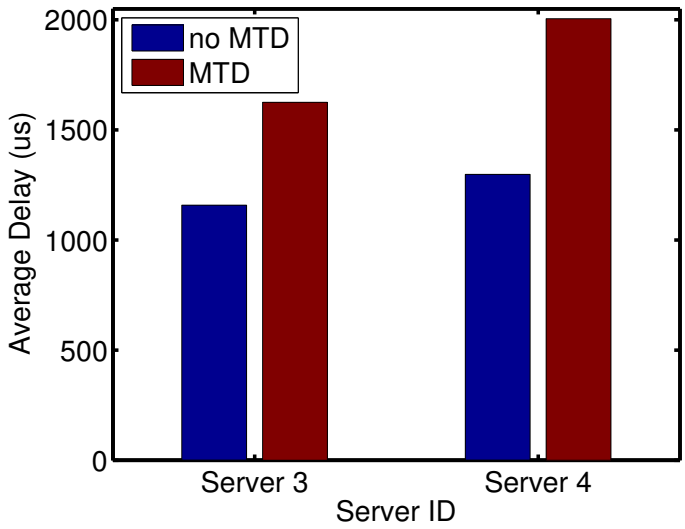


Figure 4.5: Average Delay with and without MTD for TCP Transmission for Server 3 (H3) and Server 4 (H4)

increase the end-to-end packet delay. Thus, if the packets do not have hard QoS constraints, such a minor increase in end-to-end delay would be tolerable.

UDP Experiments

We also run the same experiments by creating and transmitting UDP data at our network. Our goal was to investigate whether the MTD approach would have a similar impact as in the case of TCP.

The MTD approach also does a great job of balancing the load of links, as can be seen in Fig. 4.6, . However, it does not always provide a reduced load for each link compared to the No-MTD case. This is different from the TCP case, where we have seen significant reductions. The reason behind this can be explained as follows: In the case of UDP, there is less data transmission compared to TCP within the given amount of time in the network since there is no connection establishment and ACK messages received by the switches. Note that in the TCP experiments, we have considered both the transmitted and received packets, and thus ACK packets

were also part of the computations. Due to less amount of data, there will be fewer congestion cases (i.e., since the same threshold is used as in the case of TCP), and thus UDP case will trigger fewer route updates. As a result, the re-organization of the routes and re-distribution of the load is done at a slower pace, which causes to make less impact on the network. While the process is still helpful in load balancing compared to the No-MTD case, as seen in Fig. 4.6, it is not as significant as the TCP case.

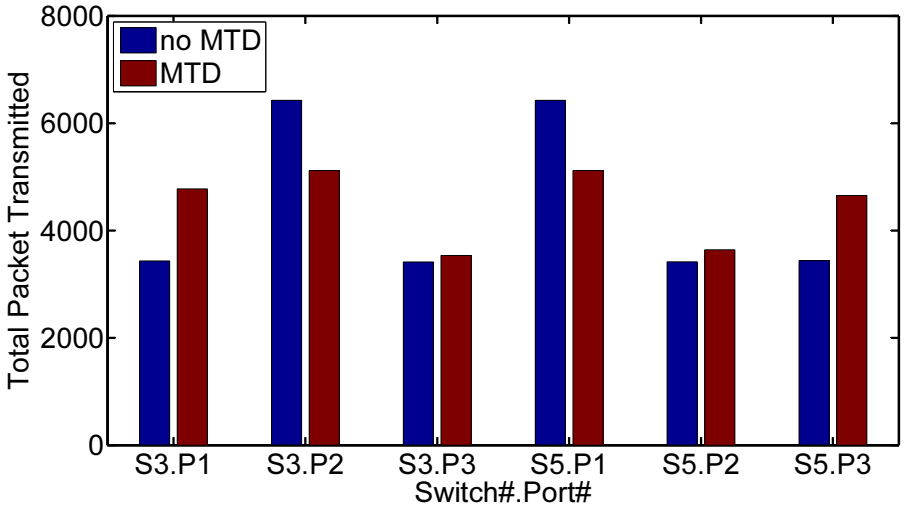


Figure 4.6: Average Link Usage with and without MTD for UDP Transmission

For the average delay between sources to servers, again whenever MTD is used, the delay is increased slightly, as seen in Fig. 4.7. However, this increase is not as much as the TCP case since UDP does not have a re-establishment phase. Whenever a route is changed, there is no waiting time due to connection re-establishment. Nevertheless, since the routes are changed to potentially to non-shortest ones, the packet delay still increases.

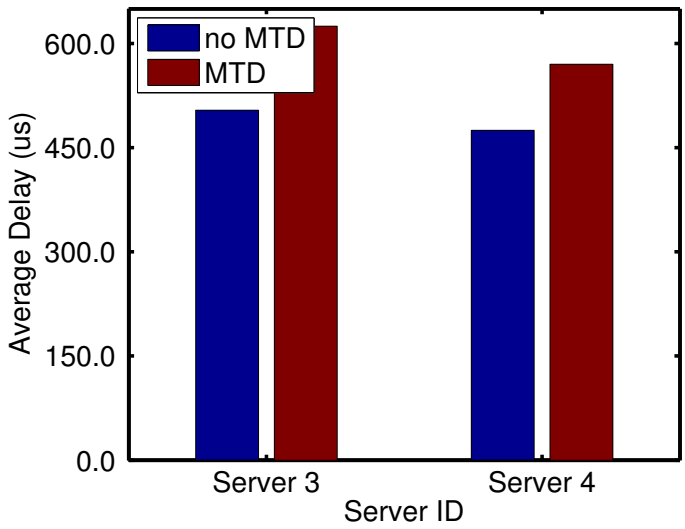


Figure 4.7: Average Delay with and without MTD for UDP Transmission for Server 3 (H3) and Server 4 (H4)

Impact of the Frequency of Checks

Finally, we investigated if there is an impact of the frequency of running the proposed algorithms at the Controller on the performance. Recall that at every 10 secs, we were checking at the Controller to see if there is any link congested. In this experiment, we changed it to 5 and 20 secs, respectively. We have used only TCP since the impact of UDP on the performance was not as dramatic as TCP. The results are depicted in Figures 4.8 and 4.9.

We observe that when the frequency of checking is increased, this helps to alleviate the congestion in most of the links. This is in line with the justification provided when comparing TCP and UDP. The more link congestion is checked, the more there is a chance to change the topology and thus have a better distribution of the load. However, this may increase the packet delay due to its overhead in connection establishment. Therefore, there is not always a pattern in delay distribution based on the quality of the selected routes.

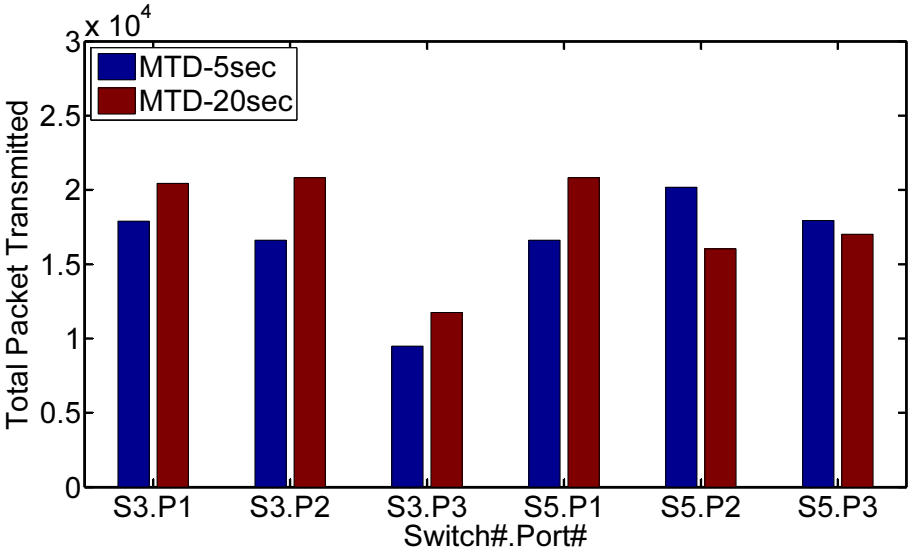


Figure 4.8: Average Link Usage of MTD when frequency is varied

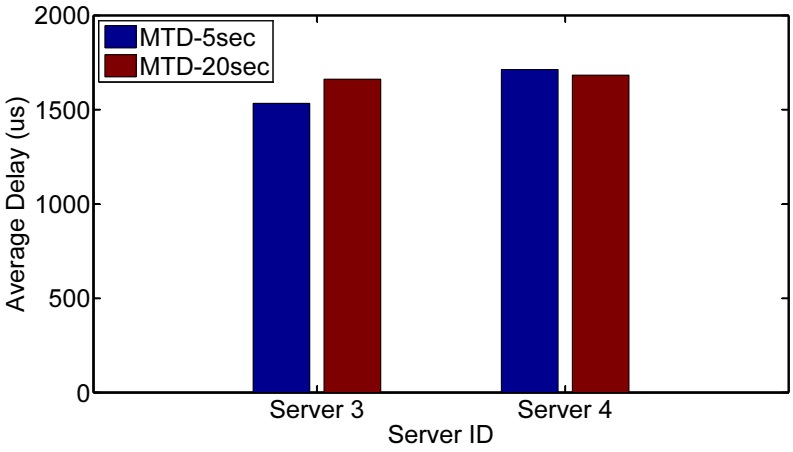


Figure 4.9: Average Delay of MTD for Server 3 (H3) and Server 4 (H4) when frequency is varied

4.4 Conclusion

In this chapter, we presented how the emerging SDN paradigm could be useful for MTD, especially to defend Crossfire attacks, an emerging indirect link flooding-based DDoS attacks. Specifically, we first presented an approach to detect attacks at the planning phase by considering the traceroute messages through the SDN controller. After this phase of traceroute profiling, we proposed an attack mitigation approach using route randomization by also utilizing the information from the profiling phase.

Experimental results indicated that our proposed SDN-based MTD methodology could effectively decrease the chance of link flooding by checking each link regularly and changing routes and thus lessening on congested links. On the other hand, because of the route updates, MTD will create a bit more delay, especially for those TCP packets flowing at the same time as route mutations. Also, we have seen the benefit of UDP from the proposed MTD approach is not as high as TCP.

CHAPTER 5

SDN BASED MTD AGAINST THE RECONNAISSANCE PHASE OF SLFA

In the previous Chapter 4, we considered the Crossfire attack and its consequences. Our main target in that work was the data phase of Crossfire attack and we strive to relieve the congested links load that attacker is trying to overburden. Our work in this chapter has many differences and additions: First, we consider reconnaissance phase of Crossfire attack. Second, we propose VCPs as a middleware solution by utilizing NFV that will help to decrease SDN Controller's load and store some network information in order to support forensic applications as an incident response. Finally, we consider NFV for deception.

That will also be costly in terms of updating all OF switches in the network. Our proposal of using VCPs as middleboxes is cost-effective compared to changing switch capabilities, especially considering the fact that VCPs can be implemented as virtual machines. In addition, firewalls do not come with forensics capabilities. Thus, they cannot, by themselves, be used as a sole defense mechanism to fully cover the nodes in the ISP network.

As the owner of the network that provides these companies with access to the Internet, Internet Service Providers (ISP) is the potential candidate to address any indirect attacks passing through its autonomous system (AS) since it has a strategic advantage of having the global view of the network. However, addressing these attacks pose many financial, legal, and technical challenges for the ISP. Specifically, an ISP may need to install new expensive dedicated security devices inside their networks (i.e., increase capital expenditure) and/or modify their existing rigid infrastructure (i.e., increase operational cost) that eventually increases the system

complexity [BAS11]. An ISP may also need to handle the legal aspect of capturing the customer's network traffic to detect such attacks.

However, even these efforts may not be enough. Due to the static nature of the existing network infrastructures, there is a favorable environment for attackers where they can take as much time as they need to obtain any information about the network states for launching successful attacks. While there are existing countermeasures through firewall and intrusion detection systems, the attackers may find ways to bypass these systems (e.g., through malware or internal attacks) and eventually obtain the network states inside the protected domain for launching further massive attacks. Therefore, there is a need to make it more difficult for attackers to obtain such information and thus launch attacks by enabling cooperation between ISPs and their subscribers. Besides this defensive effort, ISPs and companies need to have mechanisms to identify the source and type of attacks when the attacks cannot be prevented and occur successfully. Therefore, network forensics tools should also be employed as part of the network architecture to hold attackers responsible for their actions. In addition, the network forensics tools can also help ISPs to comply with regulations that expect ISPs to be more accountable when there is a security breach and prove the state of their security [PJN10b].

We argue, in this chapter that, MTD mechanisms can be used to address the aforementioned need. MTD is one of the effective defensive efforts to increase the attackers' costs [DASJ13, JASD13a, CSP15, RGAS+16]. It strives to break the static nature of a network by adding uncertainties and pursuing frequent changes of network states, which make it harder for attackers to obtain the network states for launching attacks.

The key enablers for the cost-effective deployment of these capabilities (i.e., MTD and forensics) in the ISP network are software defined networking (SDN) [KRV+15]

and network function virtualization (NFV) [LC15] that can provide a significant level of operational automation along with reduced capital and operational expenditures. SDN offers highly adaptable and manageable networks through the separation of the control and data plane, while NFV enables the implementation of network functions in software that operates on the general hardware platform and virtualization technologies with acceptable performance. While several SDN-based applications for ISPs have been proposed [BAS11, KAA+14, WRH15, HAB+15] in the recent years, the integration of SDN and NFV for ISP network security has just started to emerge [FTSB15][LC15]. Even though the efforts are still in the early stages, AT&T and Verizon, two major ISPs, have started their multi-year plan to leverage these architectures [ATT16] for significant operational and business transformation efficiencies.

Therefore, in this chapter, by envisioning a collaboration between ISP and its subscribers, we propose a novel framework to *thwart attacks at the reconnaissance phase*, by leveraging these two emerging technologies. We would like to emphasize that our work has three main novelties that do not exist in the current studies: 1) We tackle the Crossfire attack, which is a new type of attack with strong consequences [KLG13]. While there are a few works that targeted Crossfire attack handling, none of them focused on the reconnaissance phase of this attack. Our main focus is this phase, as it is the pre-phase before the attack and provides strong tools to the attackers. We strive to complicate this phase for the attackers through the proposed framework in cooperation with the ISPs; 2) We utilize deception based on NFV. Again, deception is not a new idea but as NFV is now very commonly used for various purposes, utilizing its benefits for deception in the reconnaissance phase is a novel idea that comes with almost zero costs; and 3) We supplement the package

with forensics capabilities that are convenient and efficient. Putting these under an integrative framework is our contribution different than the existing studies.

In our framework, MTD mechanisms are exploited to obfuscate attackers' attempts, while network forensics mechanisms are introduced for analyzing suspicious network packets. The proposed framework is collaborative and adaptive in the sense that the ISP's subscribers can voluntarily participate in the defense by submitting their security incidents or security policies to the ISP, which can then use these policies to decide which packets to apply MTD or store for further forensic analysis, etc. As part of this proposed framework, we introduce two new components that will utilize the SDN and NFV features in the most efficient manner: 1) a virtual collection point (VCP) that will be used to store the network traffic for network forensics purposes, to host NFV software, and calculate possible routes and select one from the list randomly each time route mutation is triggered; and 2) multi-purpose virtual *shadow* nodes and networks that will be used for MTD approaches. Specifically, we propose exploiting NFV to increase the diversity of network topologies by attaching various virtual shadow networks to them. Under a comprehensive framework, we investigate both overlay and mirror networks as shadow networks that will apply to a typical ISP network. In this way, MTD strategies will have more options (e.g., more routes) to switch to at the reconnaissance phase. Consequently, the attackers will be deceived, which will further complicate the process of actual network topology extraction. We propose various algorithms for the creation and selection of the shadow nodes and networks.

We implemented our prototype framework in Mininet emulator [Tea] using Floodlight SDN controller [Pro] that gives an ISP provider the ability to communicate with the system. We considered indirect DDoS such as the recent Crossfire attacks [KLG13], and focused on the network topology reconnaissance attacks, which is the

first phase of the Crossfire attack. In this attack, the goal of attackers is to identify permanent common links for a particular region. The experimental results indicated that even though 100% success rate is not achievable, the MTD approaches significantly extend the permanent links identification time at the expense of a 25% increase on the hop count for packet and some additional processing and storing times.

This chapter is organized as follows. Section 5.1 describes the threat model and problem definition. In Section 5.2, we describe the proposed framework and discuss the proposed MTD approaches. Detailed performance evaluation of the proposed work is given in Section 5.3. Finally, we conclude the chapter in Section 5.4.

5.1 Preliminaries

5.1.1 Threat Model

Addressing the reconnaissance phase for attacks is very important since it will have a significant impact on the attacker's effort to launch successful attacks. Therefore, we first considered the threat from the reconnaissance phase for both direct and indirect attacks in the framework. In this work, we particularly focused on the reconnaissance phase of the Crossfire attack [KLG13] that attempts to obtain the permanent links information that always appears when an attacker launches reconnaissance attempts. When an attacker obtains this information, he/she can try to isolate a specific area within an ISP network by flooding some of these links, as illustrated in Fig. 5.1.

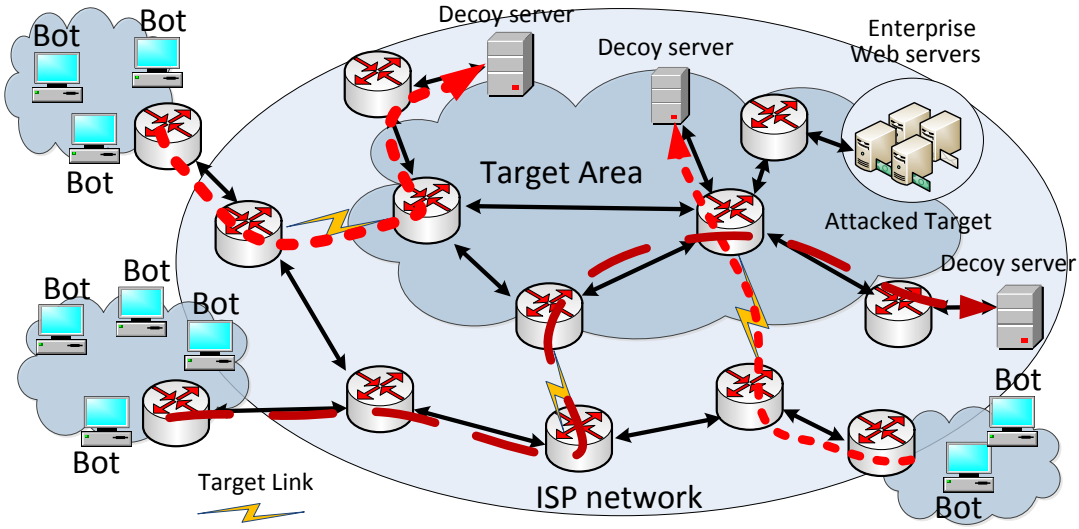


Figure 5.1: An example of Link-flooding (crossfire) DDoS attack target area

5.1.2 Problem Definition

A variety of challenges must be addressed in order to make the proposed framework feasible. Thus, our research questions are as follows: 1) How to harden the attacker's job of finding permanent links without increasing the overhead for the defender? 2) How NFV can help with the first problem and how much we can confuse the attacker by utilizing it? 3) Can we store network activities under a forensic framework as a possible defense mechanism? If so, what is the cost that the defender needs to pay for it?

5.2 SDN-NFV-based Framework for ISPs

5.2.1 Overview of the Framework

The proposed ISP network infrastructure utilizes SDN and NFV technologies to manage the traffic flows within its autonomous system. As depicted in Fig. 5.2,

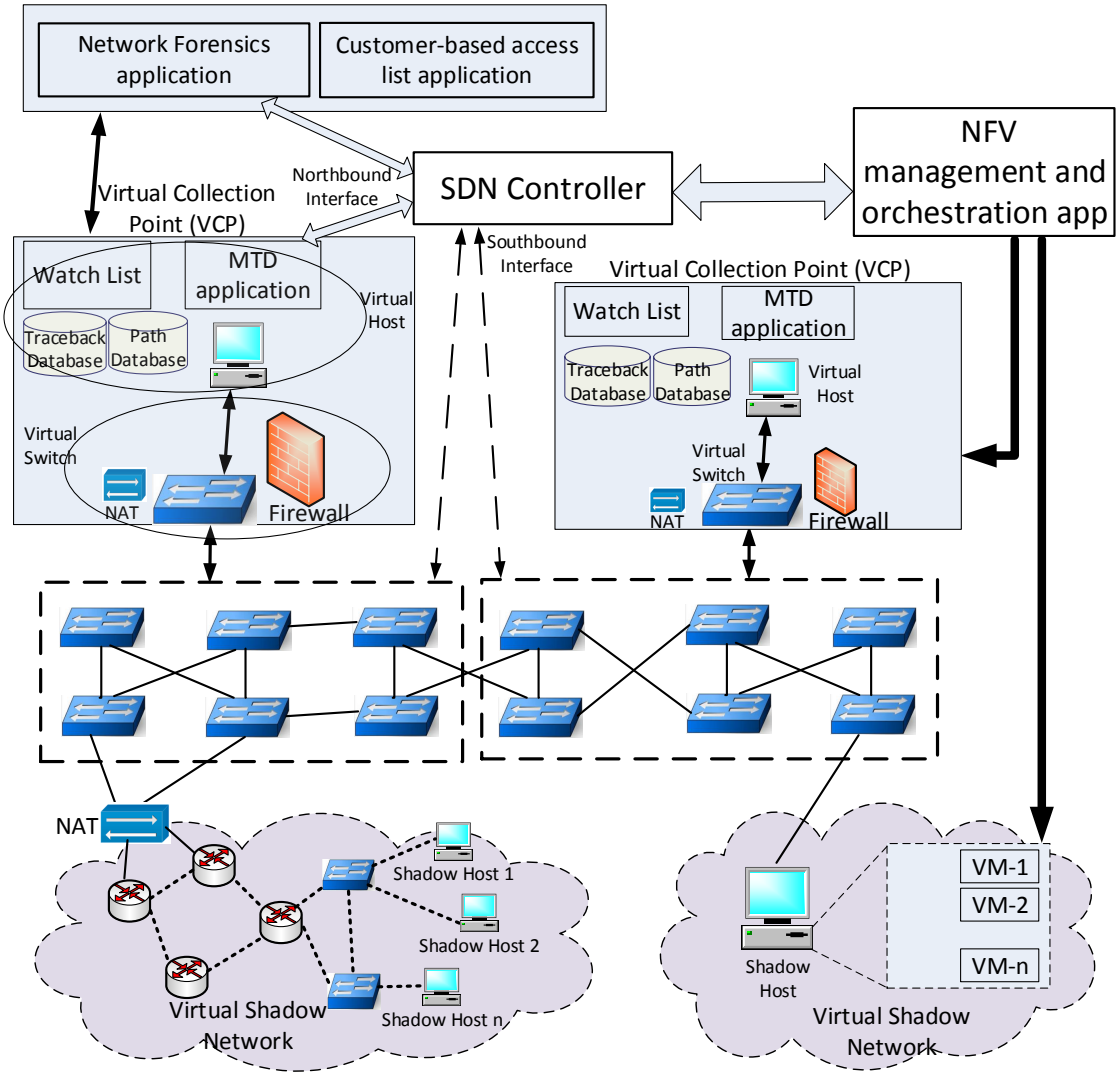


Figure 5.2: The proposed SDN-based framework for ISP Networks

the infrastructure consists of four major components: (1) VCPs; (2) VSNs; (3) SDN-based switches; and (4) SDN controller.

A VCP is basically a general server that can host various software virtualization applications. As detailed below, in our work, VCP is envisioned for pushing some of the tasks of an SDN controller in a general SDN network to itself (i.e., serving as a local middlebox) for scalability, load balancing and forensic purposes. This is one of our main contributions in terms of design. To this end, we incorporate the required applications in our proposed framework into two generic virtual devices, a virtual switch, and a virtual host that will sit within the VCP. The network applications such as Network Address Translation (NAT) and firewall software are hosted in the virtual switch. A Virtual host is used as the data storages (*watch list*, *traceback database*, *path database*), hosts the MTD applications, and handles the communication with other northbound applications (e.g., network forensics and customer-based access list), SDN controller, NFV orchestration, and the virtual switch. Multiple VCPs are deployed across the ISP network for load balancing. Each VCP handles a group of SDN switches.

A VSN is a general term used for a configurable, multi-purpose computer network that can be used to provide a variety of forged responses for security purposes. For instance, it can be used to represent a honeypot/honeynet, an overlay network, or a replica of a certain network inside a protected domain as in [WW16]. A VSN can be formed with at least a single virtual shadow host (VSH) that may have multiple virtual machines (VMs) installed in it for a variety of purposes. For example, a virtual honeypot or a replica of a web server can be built in a VM. Additionally, a network simulation software (e.g., ns-3, mininet, OPNET, etc.) can also be installed in each VM for running a simulated network environment that will be used as part of the MTD mechanisms. An ISP can have multiple VSNs distributed across its

AS. The placement of VSNs is crucial to prevent attackers from finding permanent links, as described in Sec. 5.2.5.

SDN switches are logically separated by their location in the ISP network and can be clustered into a number of clusters using an appropriate clustering algorithm (e.g., [MA07]). Each cluster is then assigned to a specific VCP, and the member of each cluster will be assigned a default flow rule to forward all traffic to the assigned VCP. The SDN controller will be responsible for setting this default flow rule for each member of the cluster.

Four major applications that are defined in our proposed framework: **Customer-based access list application**, **NFV management and orchestration application**, **Network forensics application**, and **MTD application**; are discussed next.

5.2.2 Customer-based access list Application

We envisioned that our framework can be integrated into an ISP managed security services, where instead of managing its own network security, customers can outsource it to their ISP. In this case, ISP uses the **customer-based access-list application** to register their applied security policies to those customer networks. For customers who do not use these services (e.g., they manage their own network security or outsource it to the other third party), they need to register it through this application when they want to participate voluntarily. For instance, a registered customer may submit a security preference that allocates a specific period for allowing any ICMP packets to be sent to the customer network. Outside that time period, ISP is allowed to drop these packets before they reach the customers' network or use them as part of the ISP defense (e.g., by redirecting those packets

to a honeypot/honeynet). Since sharing any customer incident report (e.g., alert from the customer's intrusion detection system) and/or the security policy to ISP may reveal other customer's sensitive network information (e.g., network topology, security infrastructure, network services, etc.) that can be used to pose a threat to the customer network, the application must be secure and privacy-preserving so that customers can feel confident and secure to participate in the ISP-level global defense.

The proposed application consists of four major components: (1) **participant registration** that handles the customer's registration for joining the ISP-level global defense; (2) **security incident and policy submission** that is used by both ISP and customers to submit any security incident and/or their security policies. In particular, for the ISP, this component is used to submit the applied security policies of the ISP's customers who are using the ISP's manage security services; (3) **incident and policy evaluation** that handles the evaluation of each submitted security incident and policy with respect to the ISP policies and the previously defined security policies in the *watch list* to identify any potential conflicts; and (4) ***watch list* management** that handles the creation, modification, and distribution of a *watch list* to the corresponding VCPs. A *watch list* is used to store the ISP and customers' security policy preferences as well as the suggested security from the network forensics application. The *watch list* is distributed to all VCPs and used to decide the action for any incoming traffic and set flow rules accordingly.

Each time there is an approved update to the *watch list*, the application notifies the SDN controller to set a new flow rule or delete an old rule based on the status of the update. When a new *watch list* is added, the SDN controller will set a *default flow rule* to forward all received traffic in the SDN switch that matches with the *watch list* description to the corresponding VCP. On the contrary, when an

old *watch list* is deleted, the SDN controller removes any relevant flow rules from the SDN switches accordingly. Additionally, the application also interacts with the **network forensics application** for the investigation of a verified security incident by sending an *investigation request* that contains any relevant information (e.g., the incident time, the attack packet header, etc.), to uncover any potential network attack and perform the appropriate action to thwart the attack.

5.2.3 NFV management and orchestration application

This application is responsible for the management and operation of NFVs in VCPs and VMs in the virtual shadow hosts. Based on the VSN usage (e.g., as a honeypot/honeynet, an overlay network, or a replica of a certain network), specific NFV management and orchestration applications may be required. Besides dynamically configuring the appropriate security software (e.g., Firewall, Network Address Translation, etc.) in VCPs, the application is also responsible for installing a VM with the appropriate network services such as a virtual honeypot software or a predefined simulated network environment that runs on a network simulator. In the case of the simulated network environment, the application's responsibility includes the creation, distribution to the appropriate VSH, operations, and removal of the simulated network environments.

In the framework, a VM in each VSH can be mutated by migrating a VM from one VSH to another VSH, either within the same VSN or to a different VSN to support the fast network reconfigurations as well as to further obfuscate attackers' reconnaissance attempts. In the case of VM migration, while the **MTD application** will decide on when and where to move the VM, the **NFV management and orchestration application** will coordinate the moving activities. A variety of VM

placement and migration techniques from the cloud data center can be investigated and used for these activities. For example, the VM migration can be performed by first turning off the source VM (i.e., VM Cold Migration) and then sending the configuration files to the destination VM to minimize the network traffic. Alternatively, it can be migrated when the VM is still running to minimize the downtime (i.e., VM Live/Hot Migration) by transferring the current working state and memory across the network [CFH+05].

5.2.4 Network forensics application

This application handles the *postmortem* (i.e., reactive) investigation upon request from authorities or as an initiative from ISP to strengthen its security policies (e.g., enhancing the MTD policy) or in response to the customer's security incident report. There are three major components in the application: (1) **forensics server**; (2) **forensics agent**; and (3) **forensics investigator**. The forensics server is located in a centralized server and acts as the forensic coordinator with the following tasks: (1) receives an *investigation request* from the **customer-based access-list application** or the SDN controller; (2) coordinates forensics agents for data collection from the *traceback database* located in each VCP by sending a *forensics collection request*; (3) provides secure communications between all entities in the **network forensics application**; and (4) provides all the necessary data from multiple sources to the forensics investigator. A forensics agent is placed in every VCP to collect data from *traceback database* when the forensics server requests it and then securely transport the collected data to the forensics server. The forensics investigator is responsible for the investigation and can be equipped with a number

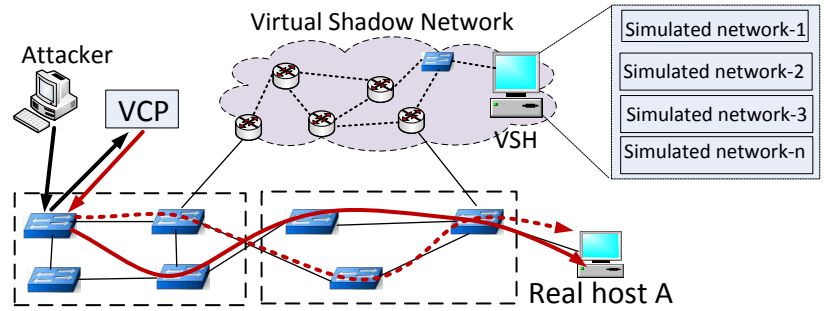
of algorithms such as machine learning or data mining techniques [JL14] to support the investigation.

5.2.5 MTD application

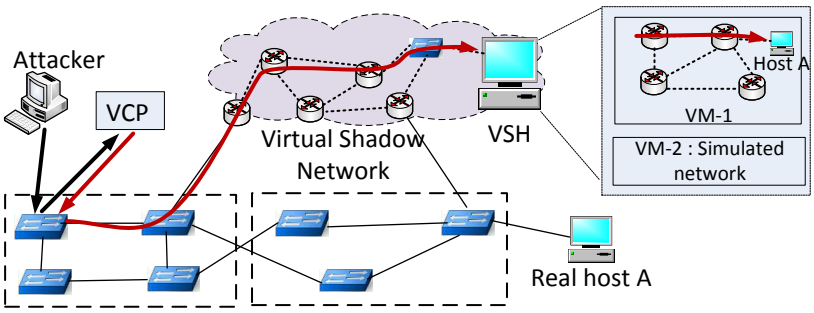
The application handles the dynamic defense of the ISP networks by dynamically changing routes in order to deceive attackers using the following MTD strategies, as depicted in Fig. 5.3: (1) the **direct route mutation strategy** that performs route mutation on the real network to the real host destination; (2) the **mirror network strategy** that utilizes a VSN as the mirror network and one of the VSHs in the VSN is used as the destination of the route mutation; and (3) the **overlay network strategy** that utilizes a VSN as an overlay network while the destination of the route mutation is still the real host. Additionally, a VSH in a VSN may have multiple VMs, where each VM is running a different simulated network environment, which will be used to provide forged routes for attackers. By exploiting virtualization, an ISP can create a variety of alternate forged routes while minimizing the unwanted traffic passing through the AS and reducing the deployment cost of the real network.

Fig. 5.4 shows the dynamic interaction between the infrastructure components (i.e., SDN switch, VCP, and SDN controller) to handle the traffic flow in the ISP network. Two databases, *traceback database* and *path database*, are used to support the MTD operations and forensics purposes. The *traceback database* is used to store the *first-time* traffic and *interesting traffic* that are being transmitted in the ISP network. The *path database* is used to store the route information whenever new flow rules related to that route are installed in SDN switches.

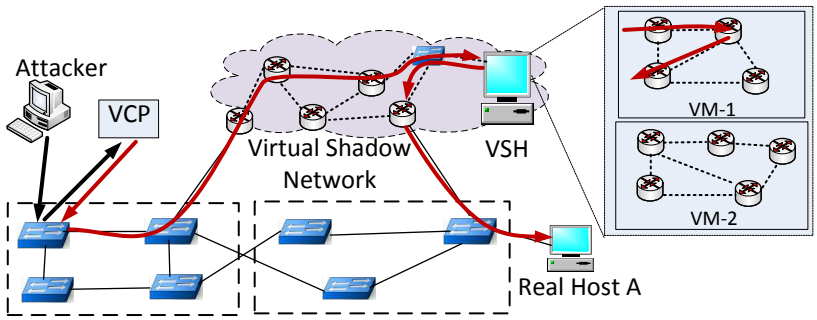
All SDN switches have a *default flow rule* that forwards all received *first-time* traffic (i.e., traffic where the received switch can not forward it to the next hop



(i) Direct Route Mutation on the network



(ii) Shadow network as the mirror network



(iii) Shadow network as the overlay network

Figure 5.3: MTD strategies

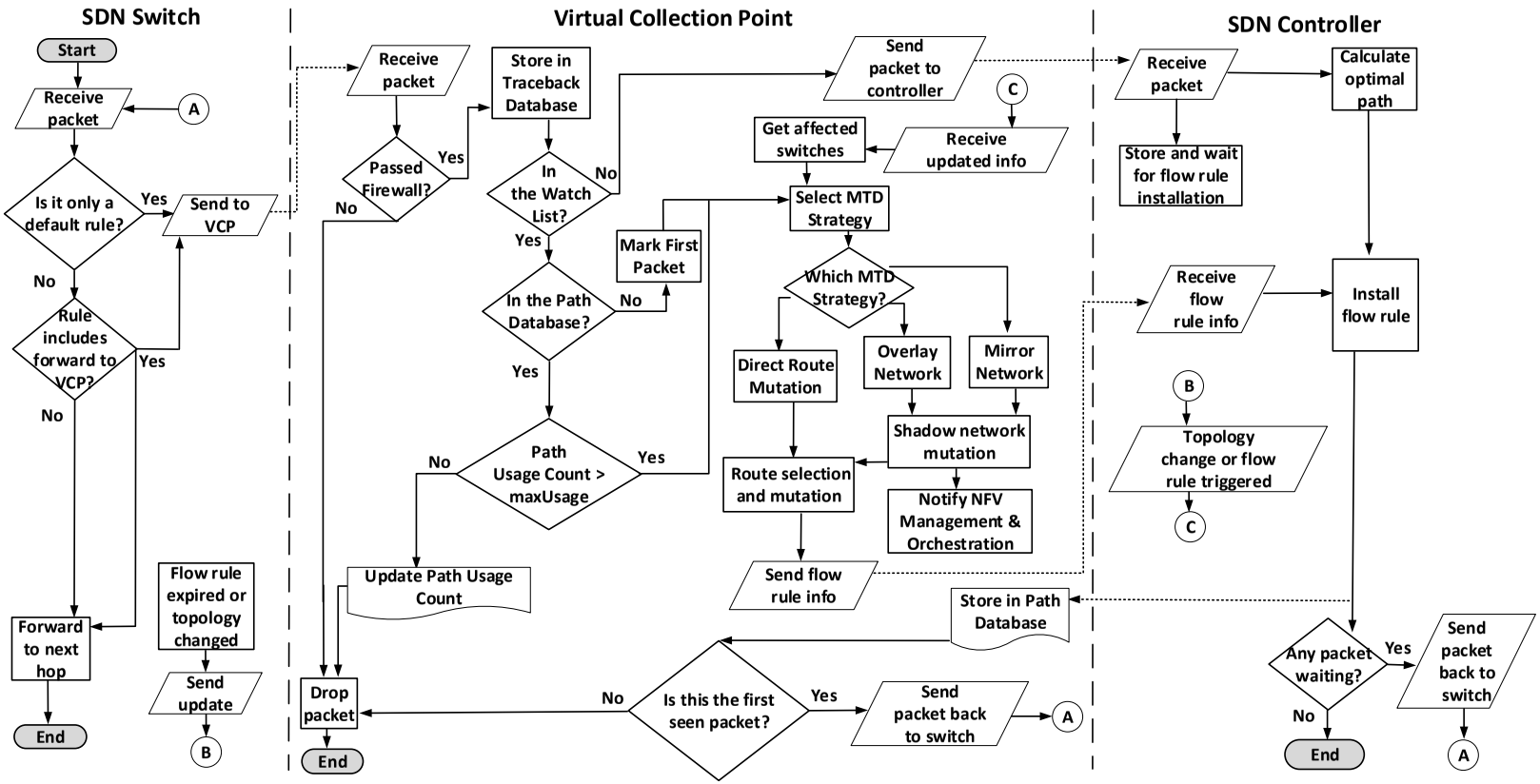


Figure 5.4: System flowchart for handling packets

since it does not have any specific assigned flow rule for this traffic). At the VCP, if this *first-time* traffic passes the firewall, the traffic will be stored in the *traceback database*. Then, this traffic will be classified further into either as a *legitimate traffic* that can be sent to the SDN controller because the traffic is not in the *watch list*, or as an *interesting traffic* that needs to be handled carefully for security reasons because the traffic is in the *watch list*. On receiving the *legitimate traffic*, the SDN controller will find an optimal path for it, installs the flow rules to SDN switches accordingly, and returns this traffic back to the switch. This way, when the same *legitimate traffic* continuously arrives at the switch, it just forwards them to the next hop according to the installed flow rules. There is no need to send this traffic to VCP/SDN controller again.

The *interesting traffic*, on the other hand, has a longer process than the *legitimate traffic*. The *path database* will be searched to see whether flow rules have been installed for this *interesting traffic* recently. When this is the *first-time* traffic, and thus there is no record in the *path database*, the next step will be the selection of the MTD strategy. A *shadow network mutation* and then **NFV management and orchestration application** may be called when the selected MTD strategy (i.e., **overlay network strategy** or **mirror network strategy**) requires a shadow network, before the **route selection and mutation** is executed. The selected path from the **route selection and mutation** is sent to the SDN controller that then installs the flow rules to SDN switches accordingly. For this *interesting traffic*, the assigned flow rule is always set to forward the traffic to both the next hop and to the corresponding VCP. This way, when the same *interesting traffic* repeatedly arrives at the switch, the **MTD application** can track the number of times the traffic has passed the switch by updating the corresponding record in the *path database*. When it has exceeded a predefined *maximum usage*, a new *MTD cycle* is kicked-in.

Besides based on the *maximum usage* limit, a new *MTD cycle* can kick-in in a number of ways. An SDN switch can trigger a new *MTD cycle* when the *mutation interval timeout* for a flow rule is expired or when there is a network topology change (e.g., new SDN switch is installed, or a link goes down). For these cases, the SDN switch will notify the SDN controller. Subsequently, the SDN controller notifies **MTD application** in all VCPs to check whether such situations may affect any of the active flow rules that are stored in their *path database*. A new *MTD cycle* will be conducted for the affected flow rule(s).

The algorithm that handles a sample *interesting traffic* is shown in Algorithm 4. The *maxUsage* indicates the maximum number of packets that are allowed to use this path before a new *MTD cycle* is initiated, while *triggeredUpdate* indicates that a new *MTD cycle* is initiated due to the *mutation interval timeout* or topology change. The complexity of Algorithm 4 depends on the function of finding alternative routes. If a shortest-path algorithm is used, then the complexity is $O(n * logn)$, where n is the number of nodes in the network. Since some of the links will be omitted because of the MTD rules (edge-disjoint, node-disjoint), the running time will be even smaller than this cost.

MTD Selection Strategy

In each *MTD cycle*, the first step is to choose the MTD strategy. The selection is determined by three factors: (1) the predefined *mutation probability* for each MTD strategy. Typically the **direct route mutation strategy** has a higher *mutation probability* than the **overlay network strategy** and **shadow network strategy** due to the high setup overhead of these two strategies; (2) the *duration time limit* of the current MTD strategy that has been used for this *interesting traffic*; and (3)

Algorithm 4 VCP Interesting Packet Received: receivePacket(SwitchID, srcIP, dstIP, protocol, maxUsage, triggeredUpdate)

Require: SwitchID: Identification of last Switch
Require: srcIP: Source IP address of the packet
Require: dstIP: Destination IP address of the packet
Require: protocol: Protocol information of the packet
Require: maxUsage: Maximum number of packets before running MTD
Require: triggeredUpdate: Boolean true if time is up for running MTD

```
victimSamples ← getSamples(victim);
RoutingEntry ← getFromPathDB(srcIP);
isExpired ← false
if RoutingEntry != null then
    usageCounter ← getEntryInfo(RoutingEntry)
    if (usageCounter ≥ maxUsage) OR (triggeredUpdate == true) then
        isExpired ← true
    end if
    usageCounter ←usageCounter+1
    Update usageCounter of RoutingEntry in PathDB
end if
if (RoutingEntry == null) OR (isExpired == true) then
    MTDStrategy ← SelectMTDStrategy(getEntryID(RoutingEntry))
    route ← findRoutes(srcIP, dstIP, protocol, MTDStrategy)
    RoutingEntry ←currentTime, srcIP,dstIP, protocol, route
    Add RoutingEntry to PathDB
end if
return
```

the predefined *alternate probability* that will be used to change to a different MTD strategy from the current one to increase the route variability.

For the *first-time* traffic (i.e., traffic that does not have any specific assigned flow rule in the received switch), the MTD strategy selection is only based on the *mutation probability*. We proposed to use a **roullete wheel selection mechanism**. In this approach, a roulette wheel is divided into three sectors proportional to the *mutation probability* of each MTD strategy and then a spin in the roulette wheel, which is represented by picking a random integer number between 0 and 100, is used to determine which strategy will be used. For example, let the *mutation probability*

for the direct mutation, overlay, and mirror be 60%, 25%, and 15%, respectively. In the roulette wheel, the direct mutation strategy occupies the first 60% sector of the wheel, overlay strategy occupies the next 25% sector of the wheel, and mirror strategy occupies the rest of the sector in the wheel. When the selected random number is 75, it indicates that the overlay strategy is chosen since this number points to the second sector of the roulette wheel.

When the *interesting traffic* is not the *first-time* traffic, the *duration time limit* and the *alternate probability* will determine whether we should try a different MTD strategy to increase the route variability. When the existing strategy has been used more than the *duration time limit* and the generated random number is greater than the *alternate probability*, then we first reduce the *mutation probability* of the currently used strategy by $n\%$, and proportionally increase the other two *mutation probabilities*, each by $(n/2)\%$. Based on the new *mutation probability* of each strategy, we use the **roullete wheel selection mechanism** again to select the MTD strategy. The strategy is provided in Algorithm 5. The complexity of this algorithm is O(1).

Virtual Shadow Network Placement

The VSN placement plays a significant role in the prevention of the reconnaissance attempts to find permanent links in the ISP network, in particular when VSNs are used in the overlay network strategy. An overlay network topology in the framework can be built in three ways: (1) by inter-connecting some of the VSNs, where each VSN is assumed to be a big overlay node; (2) by selecting some nodes in the ISP network as the overlay nodes and then building an overlay network over these overlay nodes; and (3) by combining overlay nodes and VSNs. The created overlay network should ensure end-to-end reachability for any source-destination pairs across an

Algorithm 5 MTD Strategy Selection: receivePacket($SwitchID, packet$)

Require: $IDofPathDatabaseEntry$: Identification of the Path Entry in Database
Ensure: $SelectedStrategy$: Selection of the MTD strategy

```
timeStamp, mutationProbabilityList ← getFromPathDB(IDofPathDatabaseEntry)
isMTDStrategyExpired ← false
if this is NOT the first-time packet then
    currentTime ← getSystemTime()
    if (currentTime ≥ (timeStamp + durationTime) then
        random1 ← randNumberGenerator(0,100)
        if random1 ≥ alternativeProbability then
            n ← smallestOf(mutationProbabilityList/2)
            updateMutationProbabilityList(n)
            isMTDStrategyExpired ← true
        end if
    end if
end if
if this is the first-time packet OR isMTDStrategyExpired ≡ true then
    Direct,Overlay, Mirror ← getProbabilities(mutationProbabilityList)
    random2 ← randNumberGenerator(0,100)
    if random2 ≥ (Direct + Overlay) then
        SelectedStrategy ← MirrorStrategy
        NFVManagement()
    end if
    if random2 ≥ Direct then
        SelectedStrategy ← OverlayStrategy
        NFVManagement()
    end if
else
    SelectedStrategy ← DirectMutationStrategy
    updatePathDB(IDofPathDatabaseEntry, currentTime, selectedStrategy)
end if
return SelectedStrategy
```

ISP network while taking into account the upper bound of the path length and maximizing the path independence in the overlay networks.

Randomly selected overlay nodes for forming overlay network even if they are topologically diverse, is not a good option to form overlay networks since paths from these overlay nodes to the same destination are very likely to traverse the same links at the IP layer, and thus, it cannot guarantee disjoint paths to destinations [HWJ06]. Therefore, the heuristic topology-aware algorithms such as the topology-aware K minimum joint spanning-tree [LM07], the topology-aware K connection [LM07], and the topology-aware node placement [HWJ05] can be used for the VSN/overlay node placement that maximizes the path independence in the IP-layer (i.e., the overlay paths passing through disjoint IP layer paths).

The first two topology-aware algorithms in [LM07] are based on the assumption that the location of the overlay nodes is pre-determined. Since our goal is to find the locations for the VSN, they cannot be directly used. Thus, we proposed to adopt a heuristic mechanism as part of the topology-aware node placement framework [HWJ05]. This framework is originally proposed for overlay networks across multiple ISPs, but part of the framework also considers the selection of a subset of overlay nodes inside the same ISP that maximizes the topological diversity between the nodes by using a clustered-based heuristic. We proposed to adopt this clustered-based heuristic for our VSN placement.

In our framework, we only consider where to place virtual networks without paying attention to limited resources. Our number of VM requirements in a machine is not so high, so we assume in our design that we are not limited in terms of resources. But in case of such limitation, we need to consider how to utilize network resources in an efficient way to create virtual nodes/links out of them. Mapping virtual network nodes to substrate network resources in an efficient way has been

studied under the name of the Virtual Network Embedding (VNE) problem. There are already a few works [YYRC08] [CRB12] that have investigated this problem. They can complement our solution for virtual shadow network placement.

Shadow Network Mutation

The shadow network mutation module is called when either overlay or mirror network strategy is chosen. The module consists of two sub-modules, the VSN selection, and the VSN mutation. The first sub-module is used to first pick k VSNs from N available VSNs based on certain criteria. In this chapter, we propose to select it based on the maximum edge-disjoint path criterion. Running time of our Algorithm 6 of finding and selecting VSNs is O(n*n). Note that since the VSN placements are done in advance (i.e., static placement), the selected k VSNs can be pre-computed in advance as well for every border source node (i.e., for every SDN switch that acts as the gateway to the ISP network for a customer premise or other ISP's network). Re-selection or re-computation of paths to these selected VSNs can be performed when there is a topology change. Based on the assigned VSNs and strategy (e.g., overlay or mirror network strategy), the VSN mutation sub-module will randomly select an active VSN from the assigned k VSNs and inform the route selection and mutation module about all possible edge-disjoint paths of the currently selected VSN.

The proposed heuristic approach to select k VSNs from N available VSNs, $k \geq 2$, for every border source node as represented in Algorithm 6, can be summarized as follows: Given a directed graph G that represents the ISP network topology including the VSNs, a border source node s, and N VSNs t_1, t_2, ..., t_N, we first find the maximum number of edge-disjoint paths for each (s,t_i), $\forall i \in [1, N]$ and store them in the LP list, $LP = \{(p_{11},p_{12},...,p_{1s_1}),(p_{21},p_{22},...,p_{2s_2}),\ ...,(p_{N1},p_{N2},...,p_{Ns_N})\}$.

Algorithm 6 Virtual Shadow Network Selection

1: **for** $i \leftarrow 1, N$ **do**
2: $LP_i \leftarrow$ maximum number of edge-disjoint paths for (s,t_i) *(e.g., using maximum flow algorithm [KT05])*
3: **end for**
4: $L \leftarrow \{\}$ *(an empty list L)*
5: **for** $x \leftarrow 1$, (N-1) **do**
6: **for** $y \leftarrow (x+1), N$ **do**
7: $tmp \leftarrow LP_y$ *(Store all paths of (s,t_y) to tmp)*
8: $tmpLength \leftarrow \|LP_y\|$
9: $count \leftarrow 0$
10: **for** $i \leftarrow 1, \|LP_x\|$ **do**
11: $tmp \leftarrow$ DeletePaths($LP_x(i)$,tmp,b) *(delete path(s) in tmp with common link $\geq b$ with $LP_X(i)$)*
12: **if** $\|tmp\| < tmpLength$ **then**
13: $tmpLength \leftarrow \|tmp\|$
14: **else**
15: $count++$ *(no path deletion. count $LP_x(i)$)*
16: **end if**
17: **end for**
18: $L \leftarrow (count + \|tmp\|, x, y)$ *(stores the total remaining paths and VSNs indexes in a list L)*
19: **end for**
20: **end for**
21: $sList \leftarrow$ DescendingSort(L) *sort based on the total remaining paths*
22: $VSNs \leftarrow sList[0].x$ *save the first VSN index*
23: $VSNs \leftarrow sList[0].y$ *save the second VSN index*
24: $i \leftarrow 1$
25: **while** $\|VSNs\| < k$ **do**
26: **if** NotInVSNs(sList[i].x) **then**
27: $VSNs \leftarrow sList[i].x$
28: **end if**
29: **if** NotInVSN(sList[i].y) and $\|VSNs\| < k$ **then**
30: $VSNs \leftarrow sList[i].y$
31: **end if**
32: $i++$
33: **end while**

(line 1-3). We then eliminate the edge-disjoint paths that have more than b common links for every pair (s, t_x) and (s,t_y), $\forall x, y \in [1, N]$, $x \neq y$. To this end, we can have some paths that are either edge-disjoint or have common links whose count is less than b for every (s,t_x) and (s,t_y) pair. The number of the remaining paths for every (s,t_x) and (s,t_y) pair along with the destination identities t_x and t_y form 3-tuples (r, x, y). This 3-tuples is stored in a list L, $L = \{(r_1, 1, 2)$,$(r_2, 1, 3)$,...,$(r_d, (N-1), N)\}$ (line 4-20). To select the k VSNs, we first sort L descending based on r and then select k VSNs based on the largest number of the remaining paths (line 21-33).

Route Mutation

Since the goal of route mutation for indirect attacks is to obfuscate attacker's efforts to find any permanent link(s) (i.e., a link that consistently appear during attacker's reconnaissance phase), the route mutation must select a non-overlapping path for each mutation round so that an attacker will not be able to find any shared link. This route mutation problem can be considered as an instance of k *edge-disjoint paths problem*. The ISP network can be represented as a directed graph G, while the SDN switch where network traffic entering the ISP network is the source s, and the target host is the destination t. Given these two nodes and the directed graph, we find the k paths from s to t such that no two paths have a common edge.

To solve this problem, a maximum flow algorithm [KT05] is used. The route mutation module will then randomly select one path from these k edge-disjoint paths for each mutation round. This means a malicious network reconnaissance packet from the same source will travel a different path next time it is sent. This way, the attacker will see a different view of the network at each attempt.

We also consider cases when an edge-disjoint path cannot be found. In this case, the problem becomes finding k paths with minimum edge vulnerability [YYWZ05],

where a minimum number of common links is still acceptable. The idea is to pick any path from the k paths for the next mutation round that has the least common links with some of the previously selected paths.

5.3 Experimental Evaluation

In this section, we first considered the experiments related to the usage of VCP vs. SDN Controller. Second, we observe the usage of different MTD strategies.

5.3.1 Experimental Setup

The proposed MTD framework is implemented in Mininet [Tea] emulator. Flood-Light [Pro] is used as the SDN Controller. Since, by default, an SDN switch in Mininet does not support network-layer operations, we added IP address and Time-to-live (TTL) operations to the SDN switches. The network topology in Fig. 5.5 is used for the experiments, and a number of bots are used to attack the servers.

Additionally, since our proposed framework suggests to utilize two databases to support the MTD application, we design two simplified data structure files in the experiments as follows: a record in the *path database* consists of the *ID of each entry* (2 bytes), *source IP field* (4 bytes) and *destination IP field* (4 bytes), *protocol information field* (1 byte), *creation timestamp field* (7 bytes) that stores the date and time, *path usage field* (1 byte) to store how many times this path has been used, and the *path field* that contains an array of the visited switch identities from the source to the destination. The size of this *path field* is varied depending on the path-length of the selected path from the route mutation. For the *traceback database*, on the other hand, a record consists of some part of an *interesting* packet's header information with some additional information. The size of a record is 39 bytes, which

includes timestamp (date and time) of the entry (7 bytes), source and destination port numbers (2 bytes each), sequence number (4 bytes), time to leave (1 byte), IP source and destination (4 bytes each), IP protocol (1 byte) if any, ether type (2 bytes) that is used in the data-link layer and corresponds to the protocol of network layer, and the source and destination MAC addresses (6 bytes each).

In addition to the given fields of *path database*, we also added several fields to store some MTD strategy information, as discussed in Section 5.2.5 in addition to the aforementioned previous fields in the *path database*. That information includes three mutation probability fields, each field (1 byte) for the *mutation probability* of each MTD strategy, the currently used MTD strategy field (1 byte), and the timestamp field of the used MTD strategy (7 bytes). We assumed that the maximum number of switches is less than 256 by considering some example ISP networks' size mentioned in [SMW02], and thus we used 1 byte to store the switch ID. Therefore, the minimum record size is around 31 bytes + the variable *path field* size that depends on the path-length.

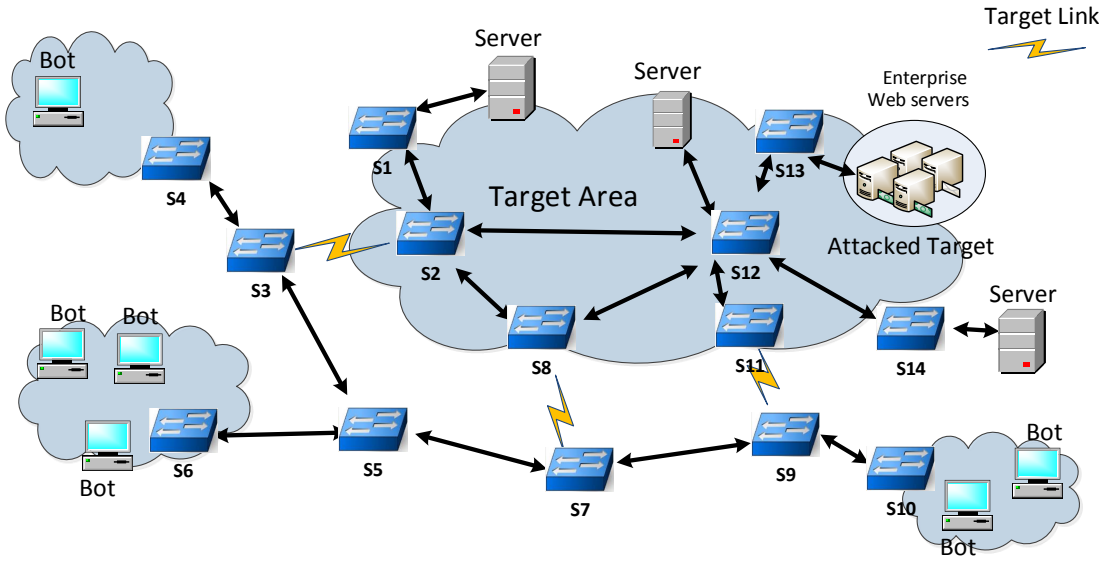


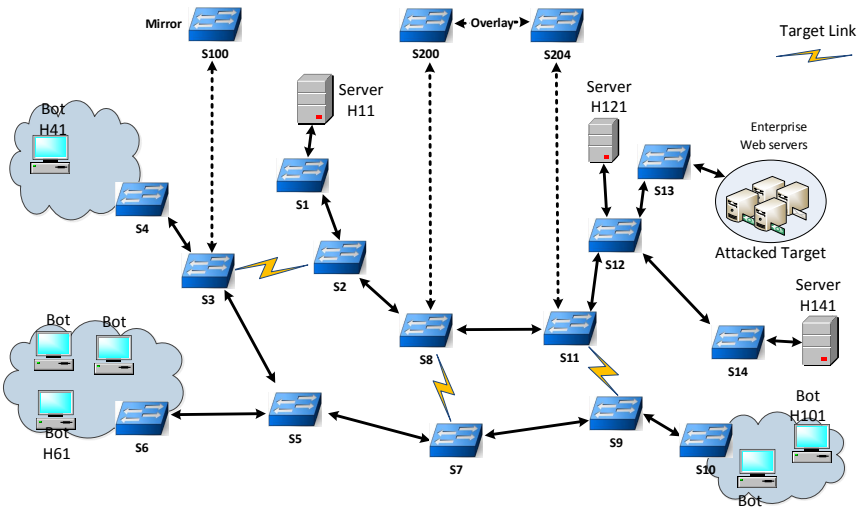
Figure 5.5: Network topology used in experiments.

In our experiments for comparing different MTD strategies (such as overlay, mirror networks), we used multiple VMs where each VM represents a different framework component, as explained in Section 5.3.2. We show these network topologies in Fig. 5.6. In order to make the Mininet-based network topology in multiple VMs talking to each other, we used GRE tunneling. GRE Tunnel is a virtual point-to-point tunnel that can transmit IP packets from one host to another [FLH+00]. Even though Open vSwitch (OVS) supports the GRE tunnel, we used the Linux GRE Tunnels since its wide support. Additionally, we also enable root access to each VM when it makes the connection through the GRE tunnel since a Mininet node requires root-privileges. Enabling the GRE Tunnel in such manner is achieved by changing some configuration parameters (permitting tunnels, etc.) of Secure Shell (SSH), which is used by the GRE Tunnel.

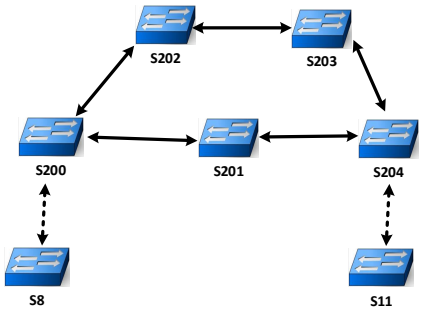
5.3.2 A Mininet-based Virtual Testbed Setup

In our experiments, we used four VMs, namely the *SDN controller* VM, the *simulated ISP Network* VM, the *Overlay Network* VM, and the *Mirror Network* VM. In the *SDN controller* VM, we installed Floodlight SDN controller and implemented the proposed MTD mechanisms. In the *simulated ISP network* VM, we created an experimental Mininet-based network topology, as shown in Fig. 5.6. In this topology, six bots are used to generate the reconnaissance traffic, and three servers are the target of this traffic. In the *overlay network* VM and the *mirror network* VM, we created the overlay network as in Fig. 5.6b and the mirror network as depicted in Fig. 5.6c respectively.

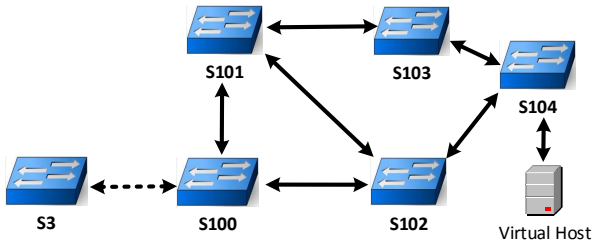
The communications between VMs are shown in Fig. 5.7. The *SDN controller* VM is connected to all network routers in the other VMs while the Linux GRE



(a) Experimental mininet-based Network Topology



(b) Mininet-based Overlay Network



(c) Mininet-based Mirror Network

Figure 5.6: Network Topologies used in the experiments

tunneling is used from the *simulated ISP network* VM to the *mirror network* VM and the *overlay network* VM. The NFV management component is responsible for the creation and deletion of these GRE tunnels. Two GRE tunneling connections that are required between the *simulated ISP network* and the *Overlay network* VM creates an identification problem for the GRE tunnels since the source and destination addresses are same. In order to solve this issue, we needed to assign a different key to each tunnel.

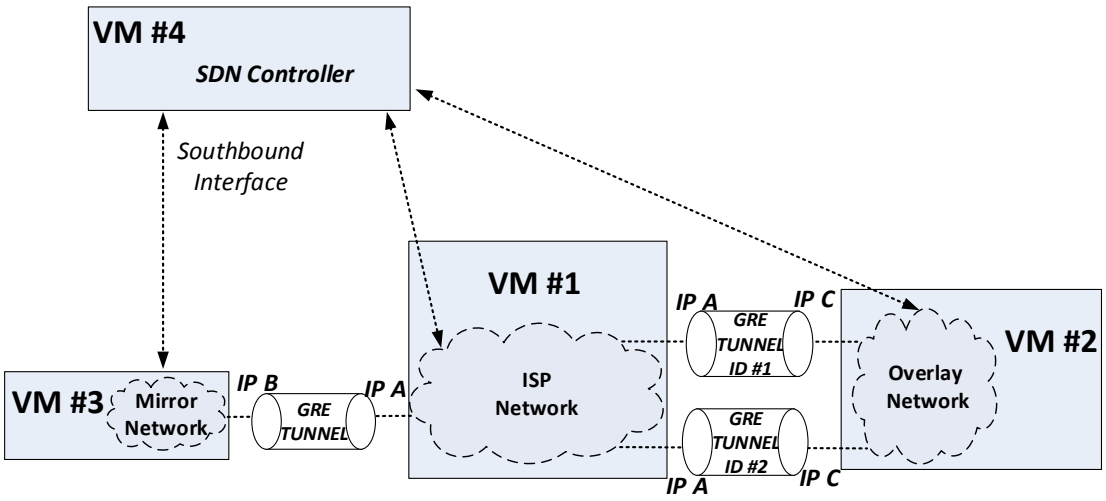


Figure 5.7: Virtual Machines Connection Through GRE Tunnels

Attacker Setup

For a realistic assessment, we developed an attack script and installed it on each bot. This script is used to run *traceroute* commands for a configurable number of times (e.g, 4, 10, 50, 250, and 1000 traceroute attempts) on each bot-server pair and store the *traceroute* results to a file. The result files from all bots are then analyzed by an attacker to determine the possible permanent links in the network.

Additionally, while the analysis can be done manually, typically, attackers would not do a manual evaluation since it is very time consuming. Instead, they develop a script to perform the analysis based on certain criteria. Therefore for the experiments, we also develop another script that performs the analysis and decides the permanent-links based on the most seen IP addresses in the *traceroute* replies. The details of this script is discussed in Section 5.3.4.

Defender Setup

On the defender side, we configured the MTD application in terms of parameter settings and the default route setting for each SDN switch. We used the same parameter values for both VCP-based and Controller-based experiments: *maxUsage* $\in$ (4, 10, 100) and *mutation Interval* $\in$ (30secs, 120secs). The default route for each SDN switch, however, is different between the VCP-based and the controller-based. The default route in each SDN switch for the former is assigned to forward packets to the assigned VCP for that switch, while the default route for the latter is assigned to forward packets to the SDN controller.

5.3.3 Baseline and Performance Metrics

We considered a baseline where all the operations would be performed by the SDN controller. Thus, the MTD applications and the required storages are handled in the centralized SDN controller, as depicted in Fig. 5.8. We labeled this baseline as the *Controller-based* method, while for our proposed framework, using VCP is labeled as the *VCP-based* method in the Table and discussion.

Two sets of metrics, namely the *attacker cost* and the *defender cost* metrics, are used to show the proposed framework's performance. Note that the choice where the MTD applications are installed, either *Controller-based* or *VCP-based*, does not have any impact on the attacker cost. However, it does have an impact on some of the defender cost metrics, as we discussed in Section 5.3.4.

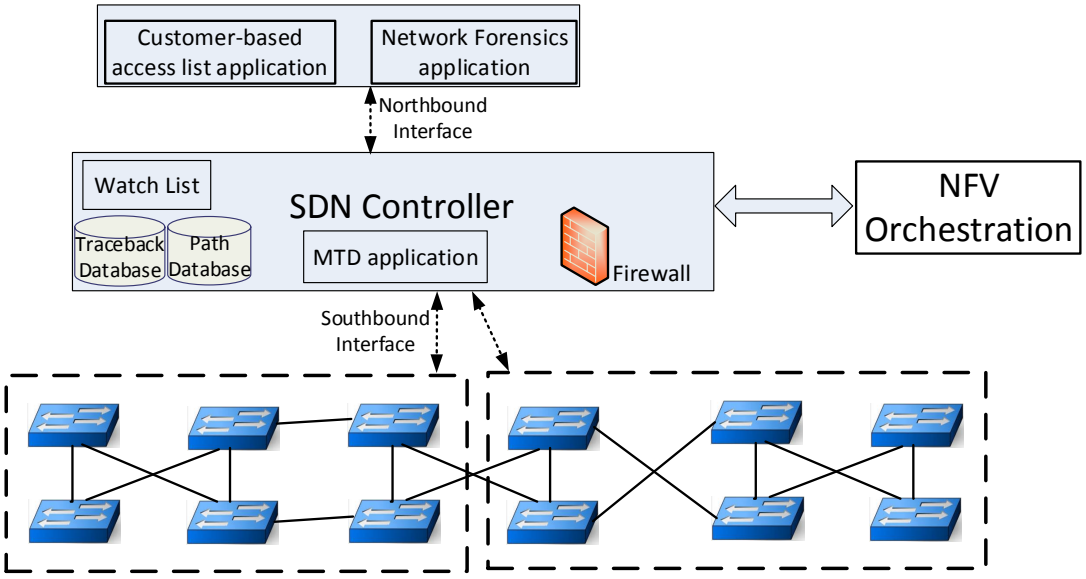


Figure 5.8: Controller-based approach where the MTD application is installed in the SDN controller

Attacker Cost

The proposed MTD application strives to increase the attacker's cost during the reconnaissance phase. Therefore, the performance metrics that we use to measure the effectiveness of the MTD application is as follows:

- *Attacker Success Rate*: This is the ratio of the number of correctly identified common links by an attacker with respect to the actual number of common links in the network.

Defender Cost

The proposed framework incurs additional cost that can be measured using the following metrics:

- *Route Calculation Time*: This is the total time it takes to find a new route from given source-destination pairs in a given scenario.

- *Storage*: This metric shows the storage overhead that can be occurred during the MTD operations and can be used for the forensics purposes as well.
- *Hop Count*: This metric shows the average percentage of increase in the number of hops (in routes) generated from the route mutation, which might be using longer paths to the destination and consuming more system resources.
- *End-to-end Transmission Delay (ETE Delay) Increase*: This metric shows the ETE delay experience by the legitimate packets due to the use of the proposed framework.

In addition to the metrics used in comparison VCP vs. SDN Controller, we used following metrics to observe different MTD strategies.

MTD Strategies

These metrics measure the cost of the comparison of Shadow Network vs. Direct Route Mutation. Specifically, we looked at the following metrics.

- *Route Mitigation Time*: This metric measures the computation time in the SDN controller to find and select a disjoint path for a given source-destination pair.
- *Time for Calculation of Routes*: This is the computational time to find out a new route when MTD is performed. This metric also relates to *traceroute* requirements. Note that the total delays should not exceed the default 5.0 seconds response time for the *traceroute* probe since the *traceroute*'s sender will consider that there is no response.
- *Resource Cost*: This metric displays the storage requirements of VMs to be operated as SNs.

Metrics for Traceroute Requirements

traceroute needs to provide a response within 5 sec. Thus, in addition to route computation time mentioned, the following delays also need to be assessed during the framework operations since they may have an impact on the maximum waiting time of the *traceroute* packet: (1) the MTD processing time that can be measured when a *traceroute* packet with the time-to-live (TTL) value of 2 in its header (which indicates that this packet is seeking for the second hop node response) is waiting at the first hop node due to the MTD operations until it can be released to the next hop node after the new flow rule is successfully installed; (2) the NAT processing time; and (3) the dynamic Linux GRE tunneling processing time that measures the time it takes to establish a GRE tunnel between two VMs.

5.3.4 Experimental Results

Attacker Cost

We first look at the attacker's difficulty, as this was our main objective in this project.

Success Rate. As the attacker heuristic strategy to determine a *permanent-link*, we consider that attacker first sorts the most used links in the network according to its *traceroute* reply results and select first k of the links from that sorted list. An attacker uses the following rules to selects the *permanent-link* in the network, which is based on the crossfire attack's characteristics, as follows: (1) the links should not be adjacent to each other since they are supposed to attack the surrounding area and cannot be linearly connected to each other, and (2) they are also not pointing to the same specific target server directly.

In our experimental topology, we have three common links. We run the attacker script five times for each case. The defender uses the *mutation interval* of 30 secs and the *maximum usage* of 10 traceroute packets. As can be seen in Tab. 5.1, the *success rate* never reaches 100% even though the number of *traceroutes* is increased significantly (e.g., up to 1000). The increase in the number of *traceroutes* attempts, obviously, increases the *collection time*, the time it takes to collect all the *traceroute* results from all bots as shown in the Table. Thus, in general, the attacker not only spends a lot of time but also is not able to find the permanent links thanks to the MTD approach.

Table 5.1: Attacker Cost

Number of TC	***Collection Time (min)***	***Attacker Success Rate***
4	0.26	33%
10	0.76	33%
50	3.8	33%
100	7.55	66%
250	17.96	66%
1000	68.35	66%

Defender Cost

The success of the VCP-based approach comes with some cost to the defender. Next, we look at the defender's cost in various aspects.

Storage Cost. First, we provide the storage cost analysis of our proposed framework that utilizes two databases for the baseline (i.e., Controller-based). Additionally, we also provide the analysis of the *packet_ins* information, which is stored at the controller for forensics purposes. Each entry in the path database, as previously described in Section 5.3.1, is expected around 34-38 bytes depending on the

Table 5.2: Defender Storage Cost

Threshold to Trigger	***Database Size***
4 maxUsage - 30 sec	239 Kb
4 maxUsage - 120 sec	238 Kb
10 maxUsage - 30 sec	104 Kb
10 maxUsage - 120 sec	95 Kb
100 maxUsage - 30 sec	43 Kb
100 maxUsage - 120 sec	17 Kb

hop count (e.g., 3-7 hop). The number of entries of the path database depends on two parameters, the *maxUsage* and *mutation interval*. Table 5.2 shows the path database storage cost for a variety of combinations of these two parameters.

As can be seen from the Table 5.2, the defender needs more storage when the MTD defense strategy takes an aggressive effort to any *traceroute* traffic by using a small value of *mutation Interval* or *maxUsage*. By relaxing these parameters by increasing the limit, either *mutation Interval* or *maxUsage*, the required storage is also reduced. However, the main driving force to reduce the storage size would be the maximum usage limit since this limit can be met faster than the mutation interval when an attacker also performs an aggressive attempt by sending multiple *traceroute* messages within a short amount of time. For instance, when an attacker wants to collect a more fine-grained *traceroute* information by increasing the number of traceroute attempts (e.g., from 4 up to 1000), their success rate was around 66%, while the storage overhead is much lower. So we can conclude that the overhead is much less and there is still a reasonable failure rate for the attacker.

As the size of each entry in the *traceback database* is 39 bytes, in the worst case for 1000 packets, the size is approximately 273 Kb considering the max number of hops (7 hops) in the selected random routes. The size of storage in the Controller

is the same as the one in VCP since the same information is being stored into the *traceback database*s after processing packet-ins. The total size of *packet_ins*, on the other hand, is around 440-450 Mb, depending on the selected random routing path (number of hops) that has been used. This is the size of the communication overhead, which is required to store packets' information to the *traceback database* for both of the approaches. In our proposed framework, we divide a load of Controller's forensic databases to VCPs. Thus, instead of bringing an extra 450 Mb to the Controller for such a simple network, we handle these databases in VCPs. We observed the CPU usage of a machine that is running the SDN Controller in this manner. Our results show that in our proposed framework, SDN Controller is using about 20% to 30% of CPU while that number goes up to 80% in case the Controller receives all the traffic to be stored. Thus, the Controller's CPU will be overloaded if we enable forensics investigations through the Controller machine. In addition, there is a single point of failure if we store all the data in one machine (SDN Controller). We eliminate that problem by disseminating the data to different VCP hosts.

The total storage cost for our VCP-based approach for two storages (path and traceback) would be the same as in the Controller-based, as well as communication overhead of *packet_ins*. The benefit of the VCP-based approach is that the total storage cost for both the *path* and *traceback database*s can be distributed into multiple VCPs instead of storing them at the SDN Controller (i.e., Load Balancing).

Route Calculation Time. In addition to storage constraints, we also compared the defender cost in terms of the time to find a new path each time an MTD is triggered for the Controller-based and our VCP-based approach. Thus, we recorded the time cost for finding a new path for both cases, as shown in Table 5.3. We can see that number of route changes is increasing with a decreased threshold as expected. The behavior is similar to Table 5.2 when Threshold to Trigger increases

Table 5.3: Defender Route Calculation Time

Threshold to Trigger	***Time Spent (ms) VCP/Controller***	***Number of Route Updates***
4 maxUsage - 30 sec	735.3 / 65.1	6128
4 maxUsage - 120 sec	730.8 / 59.8	6090
10 maxUsage - 30 sec	215.9 / 97.2	2669
10 maxUsage - 120 sec	194.6 / 86.4	2433
100 maxUsage - 30 sec	88.4 / 18.7	1106
100 maxUsage - 120 sec	33.9 / 9.0	424

from 4 to 100. That is the number of route updates and database storage size decreases. However, we observed that the total time it takes to calculate new routes does not correlate with the number of total route updates for the Controller-based approach. While we still get the least time cost with 100 traceroute count threshold, the time for 4 maxUsage (usage of traceroute count in our experiments) threshold decreases compared to the case of 10. We speculate that this could be happening because of controller software. Each time we do route updates, we access controller modules, and controller might store topology information in the cache, while for some other cases, it may need to run Link Layer Discovery Protocol (LLDP) packets to refresh its knowledge about the network. This is not the case for the VCP-based approach since it is supposed to call the Controller's Northbound Application program interface (APIs) to install the flow rules.

Furthermore, we can also see a significant increase in time for the VCP case. The main reason behind this is that VCP will call the Controller's API to install flow rules, which requires additional communication between VCP and Controller.

Note that each flow rule installation for a switch takes around 20 milliseconds. However, that rule installation can be done simultaneously with the packet transmission, since for the VCP-based approach instead of letting the packet wait, we

forward it to next hop and VCP at the same time. Thus, that delay does not affect the average delay of overall packets.

Hop Count. Since the defender is doing random route mutation, longer paths and more network resources may be used without depending on VCP or the Controller-based approach. To assess the impact on the path lengths, we collected the average hop count between each bot and server. The results in Table 5.4 indicate that, on average, we used 23% more hops in order to confuse the attacker.

Table 5.4: Hop Count Cost

Number of TC	***Avg Number of Hops***	***Min Hop***
S6-S1	6.8	5
S10-S1	6.9	6
S4-S12	7	6
S4-S1	6.4	4
S10-S12	6.3	4
S10-S13	7.4	5
S4-S13	7.9	7
S10-S14	7.2	5
S4-S14	8	7
S6-S12	7	6
S6-S13	7.9	7
S6-S14	7.9	7

ETE Delay. Furthermore, we observed ETE Delay for a regular packet that is being forwarded to either VCP or Controller for MTD purposes. The average ETE delay of a UDP packet from a bot that is connected to S4 to a server that is connected to S14 in Fig. 5.5 is about 562 microseconds for VCP-based and 643 microseconds for Controller-based approaches, which are quite close as expected since packets are forwarded to VCP or Controller at the same time they move to next hop. We believe that the reason for the Controller-based approach having slightly higher ETE delay

is because the Controller is busier with the processing the data, route calculation, storing information, and receiving lots of data to one centralized machine (SDN Controller). That might cause a longer delay overall.

Note that the ETE delay is increased 3201 microseconds for the VCP-based approach and 892 microseconds for Controller-based, on average, for a single packet being transmitted at the time when MTD is applied. Time for installing a flow to switch will be added to this delay for both cases which is 20ms in average for a switch. In other words, whenever MTD is triggered, the first packet that is exposed to new flow rules in the network will experience longer ETE delay compared to the next packet that will use the same flow rules. Having slightly longer ETE Delay at the VCP-based approach is expected since it will require extra time for the connection to Controller, which is not the issue in the Controller-based approach. Even though the Controller-based approach seems to be a better solution, we argue that it would not be a good solution if the network gets larger to collect all data to the centralized SDN-Controller, which would overload it as we have already displayed CPU usage for that case. In summary, while the performance of VCP and Controller-based approaches are quite similar to each other, the VCP-based approach is favorable since it can provide the features of network provenance and forensics in a load-balanced manner. Furthermore, our MTD operations are able to meet the delay requirement of a videophone application, which was given as 150ms in [CGC+07] even when considering the worst-case whenever MTD kicks in. It should also be noted that this ETE delay increase only for the suspicious reconnaissance packets that are passing through the network switches. We do not bring any additional delay to the regular data packets.

Scalability Experiments. In order to show the scalability of our framework, we extended the topology given in Fig. 5.5 by doubling ISP network size while still

keeping the target area the same as in the previous experiments We observed that the attacker's collection time increased to 2.5 times the ones given in Table 5.1 while the attacker's success rate decreased to 33% for 100 and 250 number of traceroutes and stays same in the other cases. These results are encouraging as they further complicate things for the attacker. Meanwhile, database size for defender given in Table 5.2 increases with the extended network topology since its size is related to the path size. The size of each entry in the new database would change between 34-46 that would bring an extra 21% increase as the worst case in the total database size. In addition to the given results, we also calculated the time to find a route in SDN Controller. Our results show that increasing the number of nodes has a negligible delay (e.g., in a few milliseconds level), which is in line with our algorithm's running time complexity, O(n*logn).

Cost Analysis of Shadow Networks

Comparison of DRM vs. SN Strategies. Table 5.5 shows the time it takes to find all disjoint edge paths, selecting one from the list, and updating the flow rules of switches when an attacker is using the strategy of sending 100 traceroute packets. The average route update time increases when the (virtual) SN is used. The reason is that by utilizing SN, more alternative routes need to be calculated and selected.

Table 5.5: Route Calculation Time

Strategy	***Average Time of Route Updates (ms)***
DRM	42
DRM + SN	76

Note that the network size used in our experiments was fixed. Scaling this network would increase the processing costs for route computation. Thus, in addition to the given experiments, we also changed the network topology to show how much

time it takes for our algorithm to find paths each time. In order to create randomly connected networks, we utilize the project given at [Wes13]. We show the average run time of the algorithm with respect to network sizes and the number of total links in Table 5.6. As can be seen from this table, the running time of our algorithm increases significantly by the increased number of edges, while the effect of the number of nodes is not major. If the ratio of links to nodes is closer to 2, our system is not able to support reasonable responses to the MTD module, especially after 30 nodes (i.e., total delay exceeds 5sec). If the ratio is closer to 1.5, we can support up to 50 nodes in the network and still can meet the *traceroute* requirements. The results suggest that adding virtual nodes, and more importantly the links, to the network after some threshold is not helpful in terms of feasibility of *traceroute* although it may be good for confusing the attackers.

Table 5.6: Runtime of Algorithm to Find Edge-Disjoint Paths

Network Topology		***Our Approach***
# of Nodes	***# of Links***	***Run Time (sec)***
14	21	0.001
14	28	0.002
24	36	0.003
24	48	0.107
30	45	0.01
30	60	4.499
35	53	0.045
35	70	239.016
40	60	0.305
50	75	1.267

Finally, we assessed the storage requirements of running VM on the host machines. In our experiments, we created Linux based VMs. These VMs are created

with 5 GB storage sizes, and we never exceeded more than half of the total storage of each VMs individually.

5.3.5 Meeting Traceroute Operational Requirements

To assess the delays mentioned, we performed several experiments. Each experiment is repeated ten times, and the average of the results is presented.

The first experiment is used to evaluate the MTD processing time. In the experiment, we sent two *traceroute* packets; the first packet is for a single hop node (i.e., *traceroute* with TTL=1) and the second packet is for the two-hop node (i.e., *traceroute* with TTL=2) and then find the time difference between the first and the second reply packets. This gives the MTD processing time. The experiment results show the average discrepancy of approximately 50 ms, which indicates that the MTD processing time is much less than the 5s limit. Thus the attackers will not notice it.

To measure the NAT processing time, we performed two experiments: the first experiment uses a virtual NAT, while the second experiment does not use the virtual NAT. On average, the discrepancy of these two experiments is around 7 ms, which is quite small and thus does not have a significant impact on the overall delay.

To measure the Linux GRE tunneling processing time, we ran an experiment using a UDP client and server application at each end of the GRE Tunnels that connect two VMs. By adding a timestamp in the UDP packet, the Linux GRE creation time can be measured by finding the difference between the UDP packet reception time at the other end and the timestamp on it. On average, the GRE tunneling processing time is approximately 53 ms. Hence, the GRE tunneling processing time is also very low when compared to the 5s limit.

Thus, overall, even the additive delays would not make an impact on the operation of the *traceroute*.

5.4 Conclusion and Future Work

In this chapter, we applied MTD and network forensics techniques to introduce agility to ISP networks by using an architectural framework that exploits SDN and NFV. A VCP is introduced in the framework as the data storage of the interesting traffic, some of which could be forwarded to the SDN controller for further investigation and starting the MTD process if needed. The framework offers three MTD strategies related to route mutation to obfuscate the network topology information for thwarting indirect DDoS attacks at the reconnaissance phase. These strategies either use direct mutation on the path from a source to a destination or utilizes the shadow networks and hosts as overlay or mirror networks for such mutation. By employing the virtualized network environments, the proposed framework offers a low-cost solution instead of deploying expensive dedicated hardware.

We implemented an emulated environment utilizing Mininet to be able to perform reconnaissance attacks and identify permanent links in the created topology. The implementation demonstrated that with the proposed MTD mechanisms, it is impossible for the attacker to reach 100% success in identifying the links. The process is also very time consuming for the attackers while it only slightly increases hop counts and storage complexity. In addition, the VCPs were successfully forwarding the *legitimate traffic* to the SDN controller while acting as a filtering point to filter some of the other traffic before it reaches the SDN controller as well as providing network provenance capabilities if needed. Moreover, the proposed framework through

NFV does not bring any significant overhead to the ISP network and execution of genuine *traceroute* operations.

CHAPTER 6

CLOUD-BASED EXTENSION OF SDN BASED MTD

In our earlier work, we considered reconnaissance attacks, and we applied MTD techniques against these attacks. However, we did not consider the legitimate reconnaissance attempts while deceiving the requester. Therefore, in this chapter, we propose a deception-based defense framework against reconnaissance attacks where we, as network defenders, strive to deceive and manipulate the attacker's vision of the network while aiming to keep network debugging tools useful. We specifically consider *traceroute* and *ping* packets as examples of network reconnaissance packets, which include useful information from source to destination in the response packets. In this manner, we employ virtual networks in order to enable fake paths between hosts by applying the MTD deception paradigm on routes. However, in order to provide scalability and elasticity of the system, we consider running virtual networks on cloud servers as part of the Network Function Virtualization (NFV) framework, which allows network administrators to utilize them whenever they want. The virtual network topology on the cloud not only provides a very cost-efficient way of executing MTD actions but also eliminates most of the management burden in the physical networks to run MTD functions.

In order to realize our proposed defense framework, we need to address the challenges of running virtual networks in cloud servers. First, *traceroute* responses include all the hops in response packets, which means that routers on the way to the cloud server will be seen in the response. However, this produces a much longer path than the authentic routing paths and disrupts the network debugging by providing non-realistic path responses. To overcome this concern, we propose creating Generic Routing Encapsulation (GRE) Tunnels [FLH+00] from the physical network nodes to the virtual cloud nodes. The second challenge in the system is to provide useful

information to the network debugger while confusing the attacker without knowing the identity of the requester. In this respect, we propose utilizing IP hopping for different *traceroute* responses (i.e., changing the network switches' IP addresses) as an element of MTD concept in order to create a complex puzzle for the attacker to solve to find out target nodes/links. In our framework, we achieve this by creating multiple cloud-based virtual networks and moving from one to another periodically. In this design, SDN facilitates IP hopping and routing updates within the local and cloud-based networks, which also simplifies to reflect network behaviors in cloud-based responses. Useful information for the network debugger is provided through creating a *reflection* network on the cloud that will exactly match the topology of the physical network.

We implemented our aforementioned MTD deception framework on the GENI platform [DRS+12], which is a well-known testbed that allows researchers to allocate virtual hosts, switches within different university campuses, and enables SDN capabilities. Our results show that cloud-based reflection network brings negligible delay and is able to reflect different network events in the reconnaissance response packets properly. This is achieved with much less cost in terms of resources and labor.

This chapter is organized as follows: In Section 6.1 we list assumptions and the attack model. Section 6.2 explains the problem motivation and the proposed system in detail, and Section 6.4 presents the experimental evaluations of the proposed

framework. In Section 6.5, a summary of the project's findings and some future extensions are discussed.

6.1 Assumptions and Attack Model

In this section, we explain our assumptions and attack model. We assume an Internet Service Provider (ISP) Network or a campus network that has network switches with SDN capabilities. The network owner is the potential client for our proposed framework, which would provide them network troubleshooting features without being exposed to malicious intentions.

We consider network reconnaissance attacks as our attack model. We specifically focus on the attacks where an attacker strives to collect network link information in order to exploit them for distributed Link Flooding Attacks such as Crossfire [KLG13] and Coremelt [SP09] attacks. These types of attacks are indirect attacks that strive to find any common links from multiple source-destination pairs such that when these links are attacked, the target machines will be isolated from the rest of the network. We assume that the attacker has access to a certain number of bots located in various locations for sending reconnaissance packets. The attacker pretends as a legitimate user while making reconnaissance attacks. The attacker is not recognizable from his/her activities, since there is not any particular way of telling who is sending reconnaissance packets to the network, and for what reason.

6.2 Cloud-based MTD deception Framework

In this section, our problem motivation and the cloud-based deception framework are described in detail.

6.2.1 Motivation

MTD approaches dynamically change the network parameters, while they enable to obfuscate the attacker's view of the network. However, they must ensure that legitimate users are still able to access the network without any disruptions. This issue was missing in the current works. In addition, the proposed approaches suffer from limited resources (such as routes, IP addresses, nodes, etc.) to be able to scale with affordable costs. Even though utilizing NFV [ASA18] can increase the route variability, the use of servers to host many VNFs to extend the real network paths introduces additional investment and management costs for servers. To address these issues, we propose to employ a cloud-based MTD deception framework, as explained next.

6.2.2 Framework Overview

In our framework, multiple reflection networks are formed in the cloud based on our physical network. A reflection network is an exact virtual implementation of an existing SDN-based physical network. For instance, instead of using a traditional SDN switch as in the physical network, a cloud-based virtual switch is employed in the reflection network. When multiple reflection networks are available, this means each of the switches in these networks has a different IP address assignment from each other. Furthermore, there will be multiple GRE tunnel connections from each SDN-based switch in the physical network to its corresponding cloud-based switch in each reflection network. The overall framework is shown in Fig. 6.1.

6.2.3 Components of Framework

As shown in Fig. 6.1, our framework has five components: (1) physical network switches, (2) cloud-based virtual switches, (3) SDN Controller for physical network (i.e., Local SDN controller), (4) SDN Controller for cloud-based virtual switches (i.e., Cloud SDN Controller), and (5) NFV management host. Cloud-based virtual switches are implemented in the cloud servers by network owners. These cloud-based virtual switches have IP address ranges as much as the cloud service provider has. A cloud SDN controller manages these cloud-based virtual switches, which organizes the packet flow within these switches and from/to physical network switches. The NFV management host acts as the interface between these two types of SDN controllers to ensure a smooth collaboration between these two types of switches for providing a connected route from a source to a final destination.

6.2.4 Procedure to Handle Reconnaissance Packets

In our proposed system, any reconnaissance packet received by a regular network switch is forwarded to a cloud-based virtual switch, follows the paths within cloud-based virtual switches, and then is forwarded back to another regular network switch to reach the destination. The response packet to this reconnaissance packet follows the same path but in reverse order.

As a first step of the framework, a reflection network is created by NFV Management. After these cloud-based virtual switches are ready, flow table rules are assigned for each switch specifically for handling reconnaissance packets. Then, the reconnaissance packet is routed along the reflection network path until the last hop of the packet before the server. In the last hop, the packet is forwarded back to

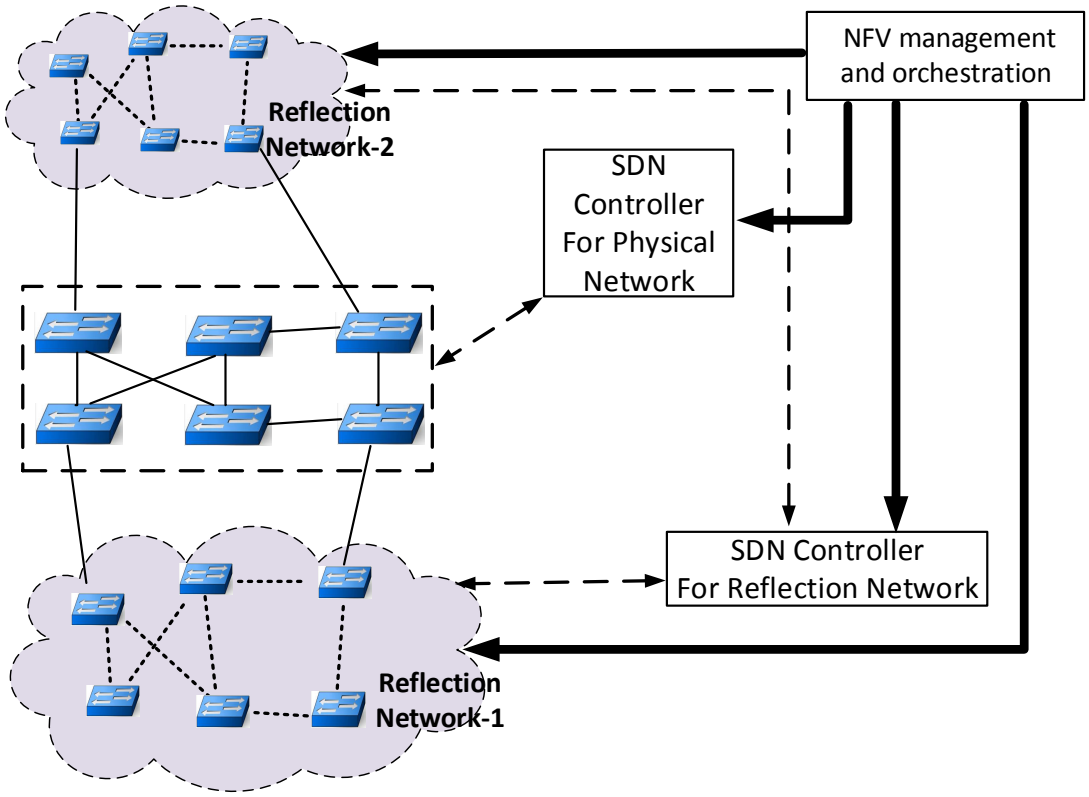


Figure 6.1: Cloud Based Defense Framework

the physical environment, and the real server generates the response. The response packet follows the same path in reverse order.

Even though it looks like a simple procedure, there are multiple issues and design problems to be considered. First of all, the response packet includes all the hops until it reaches the destination. However, this is not desirable as it will render network debugging infeasible. Therefore, GRE tunnels are created between a physical network switch to its corresponding cloud-based virtual switch in the cloud in order to solve this problem, as shown in Fig. 6.2. GRE is a packet encapsulation method, which is used to create virtual GRE Tunnels where the packets are transmitted through. Packets are encapsulated at the tunnel source, and they are decapsulated

only at the tunnel destination. We opted to use GRE Tunnels in our framework due to their ease of deployment and use.

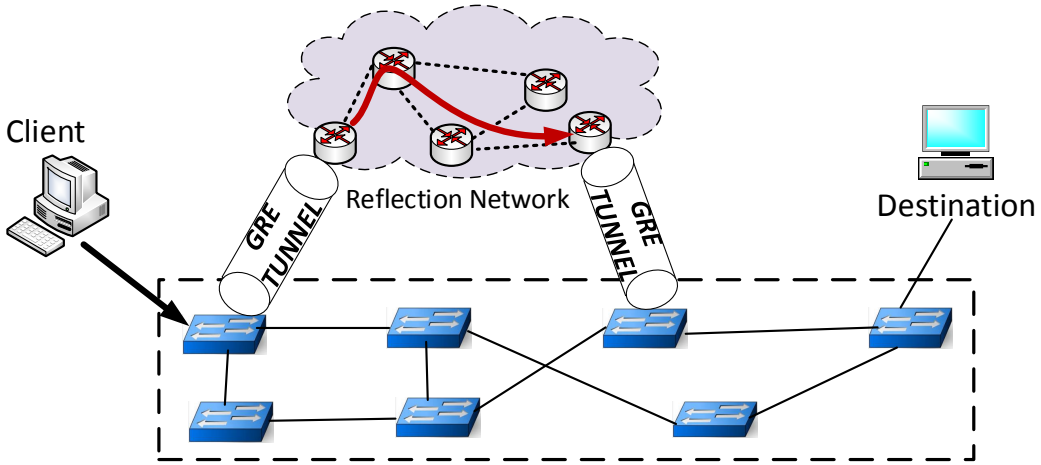


Figure 6.2: Connection between physical network and reflection network using GRE Tunnels

6.2.5 Reflecting Network Changes

The proposed framework should be capable of instantly reflecting any changes in the physical network to the reflection network. As network changes, we consider 1) re-routing in the network paths; and 2) insertion/deletion of network switches/links or their failures.

To this end, the NFV Management host is implemented as an interface, as shown in Fig. 6.1 to enable communication between SDN Controller in the physical network and the reflection network. In the first case, SDN Controller in the physical network informs the SDN Controller in the reflection network, and the routes are updated in both networks. In the second case, Link Layer Discovery Protocol (LLDP) packets that are generated and received by SDN Controller, are used to discover if a link goes down. After the SDN Controller in the physical network learns about it, the

NFV management is notified. Then, the NFV management turns the corresponding link interface to off. The same is valid for the addition of a new link.

6.3 Security Analysis of the System

In this section, we analyze the security features of the proposed framework based on our attack model.

In our framework, we provide forged responses to the reconnaissance requests. Our main target is not to let any malicious outsider collect routing paths/topology properly, which s/he can easily utilize in launching DDoS attacks such as Crossfire [KLG13] or Coremelt [SP09]. In the following example, we explain how the proposed system brings the deception so the security: Let us assume that our routing path in regular response from source to destination is IP-1, IP-5, IP-9, and with the cloud-based reflection network the response is IP-190, IP-195, IP-199, IP-9. Notice that only the last IP address is the same since it is the last hop is connected to the destination host, which a client sends requests to. However, the rest of the IP addresses are cloud service IP addresses. After responding to a few reconnaissance packets, the IP addresses of cloud-based network switches are changed. Hence, whoever sends additional requests to observe some patterns will receive new IP addresses (i.e., IP-200, IP-203, IP-205, IP-9). Thus, a malicious user, who is not aware of the deception mechanism, will not be able to properly understand network topology and the routing paths. Meanwhile, the network tester who is aware of such a defense mechanism and can transform those forged IP addresses to real ones can still debug the network.

6.4 Experimental Evaluation

6.4.1 Experimental Setup

Our proposed deception framework is implemented in the GENI environment, which is a nationwide network testbed. GENI supports Openflow switches and SDN Controller connection to these virtual switches. In this testbed, we created two networks, one of which is considered as the physical network and the other one as the reflection of this physical network in the cloud environment. The first (i.e., physical) network is created within the University of Kansas, and the second one (i.e., reflection) is within the University of Kentucky, as shown in Fig. 6.3. SDN Controller for the physical network is a virtual server located at Northwestern University, while the SDN Controller server for the reflection network is located at the University of California, Los Angeles. We connected network switches in different locations with GRE tunnels (shown by the links with yellow squares in the figure), which would allow network debugger tools to work properly, as discussed in Section 6.2.4. These tunnels are treated as interfaces of Openflow switches, and reconnaissance packets are forwarded to these interfaces as flow table rules. We considered *traceroute* and *ping* packets as reconnaissance packet examples in our experiments. As a proof-of-concept, we used five switches in both the physical network and the reflection network. In addition, we added two virtual hosts (i.e.,node-0 and node-1) in the physical network, as shown in Fig. 6.3. We use SSH to connect to virtual hosts and switches for traffic generation and run the simulations.

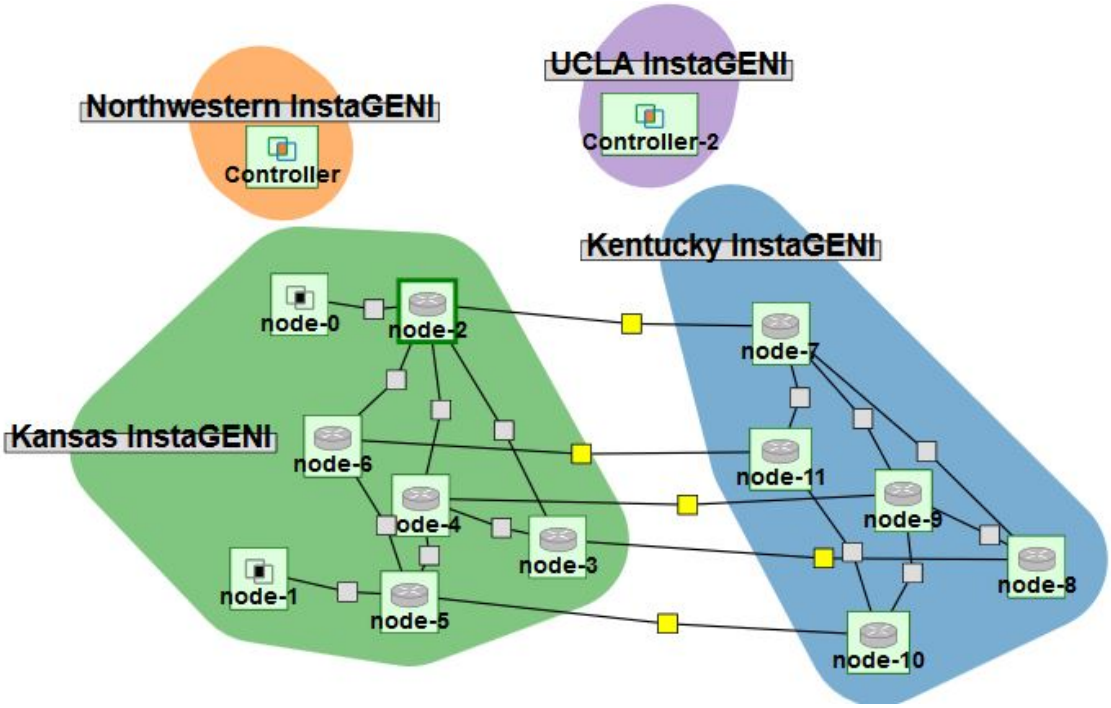


Figure 6.3: Network Topology on GENI

6.4.2 Benchmarks and Metrics

In our experiments, we considered the physical network (Kansas) with no reflection as our baseline network for normal reconnaissance operations. Then, we formed the reflection network (Kentucky) and connect it with Kansas to show cloud-based deception operations during a reconnaissance. We specifically consider the following metrics for our evaluations:

- *Response Time*: This metric represents the server's response time to reconnaissance packets.
- *Packet Loss Percentage*: Packet loss happens when the client does not receive a response to its request from the server.

Table 6.1: Response Times and Numbers of Packet Losses

Network Load	***Packet Loss Percentage***		***Response Time (in milliseconds)***	
	Regular Network	***Cloud Based***	***Regular Network***	***Cloud Based***
0%	0%	0%	2.1	47.8
10%	0 %	0%	2.2	47.5
20%	0 %	0%	2.3	47.6
30%	0 %	0%	2.3	47.4
40%	0 %	0%	3.3	47.4
50%	0 %	0%	4.2	47.4
60%	12%	0%	12.9	47.4
70%	16%	0%	13.1	47.4
80%	31%	0%	15.5	47.5
90%	32%	0%	18.6	47.5
100%	34%	0%	18.7	49.9
110%	34%	5%	19.2	50.9
130%	35%	10%	20.6	51.3
150%	36%	11%	19.5	51.6
180%	39%	12%	20.5	51.1
200%	40%	20%	19.6	51.9

- *Delay of Reflecting Network Behavior*: This metric represents the time it takes to reflect any network events such as route change in the physical network, network link failure, etc. on the cloud-based responses.
- *Cost*: This is the monetary cost to run our framework on the cloud with respect to owning it.

6.4.3 Experimental Results

We conducted several experiments based on the mentioned metrics as detailed below:

Response Time and Packet Loss Results

First, we looked at the response times and packet loss percentages for reconnaissance packets with and without a cloud-based approach. In these experiments, we changed the traffic load in the physical network from zero bandwidth usage (e.g., no other traffic in the network) to overloaded communication links by doubling the packet transmission rate with respect to the links' bandwidth. We show these results in Table 6.1.

In case of no traffic, there is not any packet loss in both cases as expected since network switches are only dealing with reconnaissance packets. Average delays in this case, however, have a big difference: from 2.1ms to 47.8ms. The difference is caused by sending and receiving the packets through the reflection network. This means having more number of hops due to GRE tunnels where the packet might be passing through multiple hops on the Internet, which eventually results in increased delay. However, the difference is smaller in a lightly loaded network environment. For example, we observe that in the case of 60% of network load, the delay jumps to 12.9ms for the physical network with a 12% of packet loss. Meanwhile, there is not any noticeable impact on the cloud-based response results. Each increase in the network load after 60% increases packet loss percentage for physical network responses. The cloud-based response is affected in the scenario when the network links' bandwidths are fully loaded. Overall, our results show that the cloud-based reflection network brings approximately 25-35ms additional delay while having less packet loss.

We also observed two interesting results in this table. The first is the response times in the physical network. More traffic load causes increased response times but only after 60% network load until it is 100%. After this point, the response time did not increase much; however, more packet loss is seen. We believe that this

Table 6.2: Average Ping Responses Times from Different Server

Google Server in Cloud	***Hawaii***	***Korea***
10 ms	100 ms	168 ms

happens because of the filled queues of the switches. By default, ping packets are not re-sent [Lin]. Thus, the packet loss keeps increasing if the switches are not able to catch up with forwarding requests. Due to dropped packets, the delay also keeps almost flat. The second interesting result is related to cloud-based response times. We notice that there might be faster responses even though the traffic is increased in the physical network (e.g., 20% network load vs. 30%). We speculate that this happens because of the inconsistency of Internet traffic. The packets are sent to the cloud-based reflection network by traveling through several routers. We do not have control over these routers, and at each minute or even second, there might be a huge difference in the number of packets transmitted from other networks.

Delay of Reconnaissance in Public Servers

In addition to our reconnaissance response times within the experimental setup, we also run reconnaissance requests to public servers in different locations. Specifically, we sent *ping* requests to *google.com*, *hawaii.gov*, and *kaist.edu* (a university in Korea). We observed that the Google server is able to respond within an average time of 10ms. We believe this happens because the Google servers are located in different cloud locations and possibly one within the same city or state of our node, which sends the requests. However, the response from *hawaii.gov* and *kaist.edu* has over 100ms response times as displayed in Table 6.2. By showing these results, we claim that our cloud-based response brings a reasonable and acceptable average delay of 25-35ms to the reconnaissance requests.

Cost of Reflecting the Network Events

The other metric we evaluate is the delay in reflecting network behavior changes in the cloud-based responses. These changes can be a new switch addition/deletion, a link fault, or rerouting in the communication paths. The processes for such events are explained in Section 6.2.4. Specifically, the total time it takes for reflecting the link failure is the sum of the time for failure detection of the network link, time to inform NFV Management, and time to turn the link off from NFV Management. These values are shown in Table 6.3 as 50ms, 22ms, and 2ms on average. The total time is less than 75ms. As another experiment, we considered rerouting networking paths. In this case, SDN Controller was responsible for this update. Whenever SDN Controller in the physical network changes any networking path, this updated information is forwarded to the SDN Controller in the cloud through NFV Management. After that, the cloud SDN Controller updates the flow tables of cloud switches in order to follow the same path in the reconnaissance packets as in the physical network. In this case, the total time is the sum of time it takes to inform NFV Management, and the time it takes to update network switches in the cloud. This is measured as less than 70ms (22ms + 45ms) on average. As it is shown in these two experiments, our reflection network is able to reflect the events of the physical network in less than 75ms.

We now discuss if 75ms time overhead would be acceptable in actual settings. The reconnaissance requests are typically repeated with a few requests. For example, *ping* is sent every 1-second after it receives the response [Lin]. It is also sent multiple times by default. Hence, we claim that our additional delay is negligible compared to 1 second waiting time, and the correct response will be received from the client. As another example, *traceroute* requests are typically sent three times consecutively. Even if one of the *traceroute* response represents the network switches wrongly, the

Table 6.3: Time for different Network Events/Updates

Network Link Failure Detection Detection	***Time to inform NFV Management***	***Time to Turn off the Link***	***Update Routes in the Cloud Network Switches***
50ms	22ms	2ms	45ms

other two responses from the same switch will be corrected. Thus, our reflection network is able to provide timely updates and responses to the clients correctly in case of any network events.

Cost Analysis of Cloud-based Reflection Network

In this part of the analysis, we provide some cost considerations comparing the cases of purchasing or renting these reflection network switches. If the reflection network is purchased and installed within the physical network infrastructure, these are the factors to consider: the cost of buying these switches and the cost of maintaining these switches (electricity, management, providing the IP addresses, and the physical location to store them). If the reflection network is rather rented from a cloud provider, then the network owner does not need to invest any money for setting up this system and maintaining it. The only cost would be cloud service cost, but this will be very minimal as these virtual switches will be used only for reconnaissance packets.

As a concrete example, we considered Microsoft Azure Virtual Network services as a potential cloud-based network provider. According to the pricing calculator on the Microsoft website, 100 Gb of data traffic per month costs only $1, and the network infrastructure is free to use [Mic]. We should note that reconnaissance packets are usually very tiny, and 100Gb of data traffic in a month is more than what is required. However, if the other option is pursued, buying the simplest

network switch would cost at least about $100, and installing full network topology with tens of switches would end up thousands of dollars to spend. It would also require labor work to install and maintain them. Hence, we claim that renting it as a cloud service makes it a more affordable and logical choice instead of owning these overlay network switches.

6.5 Conclusion and Future Work

In this chapter, we presented a cloud-based deception framework against network reconnaissance attacks by utilizing a virtual reflection network on the cloud. Specifically, we forward reconnaissance traffic to this virtual reflection network by creating GRE tunnels from the physical network. We demonstrated that our solution brings a negligible additional delay while letting the network tester to do the troubleshooting properly. We also provided a cost analysis of the system and justified that it is more economical to rent the reflection networks as a cloud service instead of purchasing and installing them locally.

CHAPTER 7

SDN BASED MTD AGAINST SLFA BY ENGAGING WITH THE ATTACKER: SIGNALING GAME APPROACH

MTD, more specifically, Random Route Mutation (RRM) [JASD13b], is found useful in defending a network system against SLFAs [ASA19]. However, RRM brings significant overhead to the network since it takes time to update the system characteristic (i.e., routing information), and this process is usually applied not only to malicious users but also to legitimate clients. This is because the defender is not aware of the type of client. Therefore, the concern of when and for whom to change system parameters in order to minimize the cost of the defender and impact of the attacker becomes a critical issue, and it should be investigated thoroughly. Hence, in this chapter, we consider the interaction between the attacker and the defender in order to precisely apply MTD techniques.

Even though there are many research works proposing defense mechanisms against SLFAs in the literature, they are mostly reactive solutions in which the attacker did some harm before the attack is mitigated [XML+18, GKLD16]. In addition to these works, there are other kinds of proactive defense mechanisms that apply detection of bots, sending forged responses to the hosts, or similar approaches [WXZ+17, MTL+18]. We argue that bot detection might not be possible considering that bots often behave similarly to legitimate hosts, and serving fake replies to hosts is not acceptable since that may cause letting the legitimate clients use the longer paths and having more delays. Therefore, there is a need for proactive, dynamic solutions that can harden the system from the attacks while bringing minimal service degradation to the hosts. To the best of our knowledge, this work is the first to introduce strategic defense against SLFAs in a proactive manner.

In order to model such a system that achieves the best efficiency of RRM, we need to design an intelligent defense mechanism that can take into account the above concerns for running RRM. To design such a strategic defense mechanism, in this chapter, we apply a signaling game [Noe88] to model and analyze the attack and defense actions together. The results from this analysis assist in applying RRM selectively and appropriately such that the target network can remain sufficiently protected against Crossfire attacks, and the legitimate clients experience a minimal cost. In this game, one player constructs a belief about the type of opponent. This belief is always updated by the opponent's actions. Then, the player can better decide about its optimal strategy, given its belief about the type of player from which it receives messages. Essentially, we consider the attacker and the defender as the players of this signaling game. We analyze the game and compute all potential Bayesian Nash equilibria. The game results are then used to decide when and for whom to perform RRM. We name the corresponding defense mechanism as *Strategic RRM*.

To evaluate our proposed solution, we first implement a network using Mininet, a virtual SDN testbed [Tea], and the defense mechanism on FloodLight Controller [Pro]. Then, we run experiments with extensive RRM, in which the routing paths are periodically changed, to observe the overhead to the legitimate clients when there is not an attack in place. Next, we run the SLFA and realize the reasonable frequency for periodic RRM, which is used later to compare with *Strategic RRM*. Moreover, we consider two kinds of attackers based on attack approach and capability. In each case, we report the number of packet losses when the defender has either no protection, periodic RRM, or *Strategic RRM*. We show that while periodic RRM provides a significant improvement to the network defense, it introduces packet delay as well as packet losses even when there is no attacker in the system. We also show

that *Strategic RRM* offers a similar defense performance as periodic RRM while it causes much less overhead.

In summary, our contribution in this chapter is threefold: First, we model the attacker and the defender as players of a signaling game and define their actions and payoff models. Second, we solve the game and derive all possible Nash equilibria of this game. According to the game results, we also design a strategic mechanism to defend against SLFAs. Finally, we evaluate the proposed mechanism by conducting extensive experiments considering different attack approaches and capabilities for different defense strategies.

This chapter is organized as follows. In Section 7.1, we discuss the preliminaries. In Section 7.2, we present the game model. In the following section, we analyze the game and design a defense mechanism. Detailed performance evaluation of the proposed work is presented in Section 7.4. Finally, we conclude the chapter in Section 7.5.

7.1 Preliminaries

In this section, we briefly discuss the attack model and proposed defense mechanism that we applied in this research project.

7.1.1 Attack Model

Stealthy link flooding attacks are indirect attacks such that the attacker targets the communication links instead of targeting the server directly [RKWM08], as explained in Section 2.4.1. The attack model of this research considers Crossfire attacks as the threat and the Autonomous System (AS) as the target domain.

Two different approaches are considered for an attacker:

- *Stealthy Attacker*: In this model, the attacker is not in a rush to conduct the attack. S/he runs reconnaissance, and accordingly creates a target link set. Sometimes s/he stays idle in order to behave like legitimate users. After that, s/he starts the attack phase to exhaust the links' bandwidth.
- *Aggressive Attacker*: In this approach, the attacker behaves more greedy and transmits more quickly to congest the communication links. Reconnaissance attack is performed, the information is collected and the attack pattern is designed within a given short time for the aggressive attacker.

Furthermore, an attacker often has a limited number of bots. Thus, two qualitative capabilities are considered:

- *Decent Attacker*: The attacker has a small number of bots to launch a Crossfire attack.
- *Strong Attacker*: The attacker has an extended capability (i.e., a significantly increased number of bots) compared to a decent attacker.

7.1.2 Proposed Defense Mechanism

Our proposed defense mechanism is based on a *signaling game*. Note that the signaling game is a dynamic game, in which the first move is made by the bot or legitimate user, and the defender is the second player. The game is incomplete information because the defender does not know with whom it is playing. This uncertainty is modeled with Nature. Moreover, we update the knowledge of the defender in repeated interactions with other nodes. Both players decide their actions according to their expected outcome. We explain the details of the game model in Section 7.2. The high-level architecture of our defense procedure is shown in Fig. 7.1. As can be seen in the Fig. 7.1, our system first initializes the parameters depending

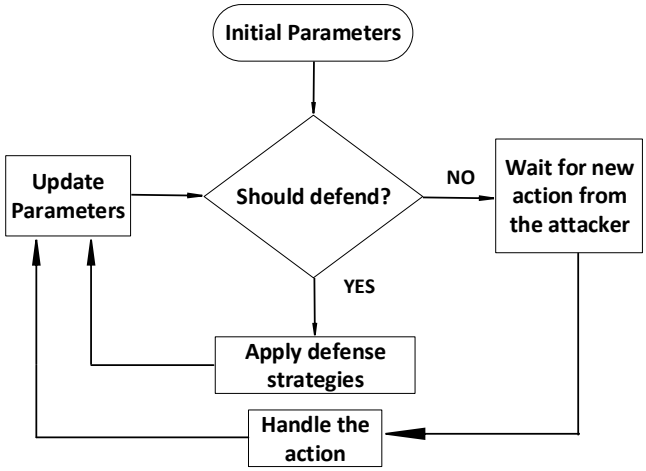


Figure 7.1: The procedure of defense decision

on the network environment and topology. Then, it checks if the conditions to trigger defense strategy is satisfied. If the decision is made as to defend, RRM is employed, and system parameters are updated. If not, the defender stays idle until it receives an action from the attacker. Each time an action received from the attacker, the defender updates his/her parameters and checks if it is time to defend.

7.2 Crossfire Attack and Defense Game Model

In this section, we first give an overview of the proposed game. Then, we define the players' action sets and model the belief function and payoffs.

7.2.1 Overview

In our Crossfire Attack and Defense Game model, a client can be malicious (a bot corresponding to a Crossfire attacker) or benign. Clients send various requests to the servers in the target network. A bot makes ping and traceroute requests so that the attacker can do the reconnaissance (i.e., link map construction) to launch a Crossfire

attack. The security administrator of the target network (simply the defender) can apply RRM to thwart this reconnaissance, so the data phase of the Crossfire attack. In the case of applying RRM, the defender randomly changes/mutates the routing paths, thus changing the link map. We model the interaction between the Crossfire attacker (in fact, each bot individually) and the defender using a *signaling game* $\mathcal{G}^{cf}$. Signaling game is a two-player incomplete information game, in which Nature has a unique randomizing strategy (i.e., θ) that is commonly known to both the defender and the attacker. With this randomizing strategy, the type of sender would be defined.

The first player, a.k.a. the sender (here, the attacker/bot), is informed of Nature's choice and chooses an action. The second player, a.k.a. the receiver (here, the defender), then chooses an action without knowing Nature's choice but observing the first player's action. Our choice of the signaling game is based on the dynamic and incomplete information characteristic of the Crossfire attack, where the action of one player is conditioned over its belief about the type of the opponent. The action flow of the game is shown in Fig. 7.2.

The game $\mathcal{G}^{cf}$ is played individually with each client/sender who can be a bot or a legitimate user. A bot is considered as a type t_1 sender, while a legitimate user as a type t_2 sender. We represent this set of sender types as $\mathbb{T}$, i.e., $\mathbb{T} = \{t_1, t_2\}$. The second player of game $\mathcal{G}^{cf}$ is the defender. The game is played in the following steps, as shown in Fig. 7.2. The defender receives two different types of messages, i.e., *Reconnaissance* or *Regular Traffic*. Nature draws type t_1 or t_2 with a probability of θ or $1-\theta$, respectively. The defender does not know exactly whether the observed message is coming from a bot or a legitimate user, but s/he can only form/follow a belief. According to this belief, the defender must decide whether to defend by applying RRM or not. In the following subsections, we define the actions of the

players and accordingly model the belief and payoff functions. In the game, we often use the term "attacker" to represent a bot.

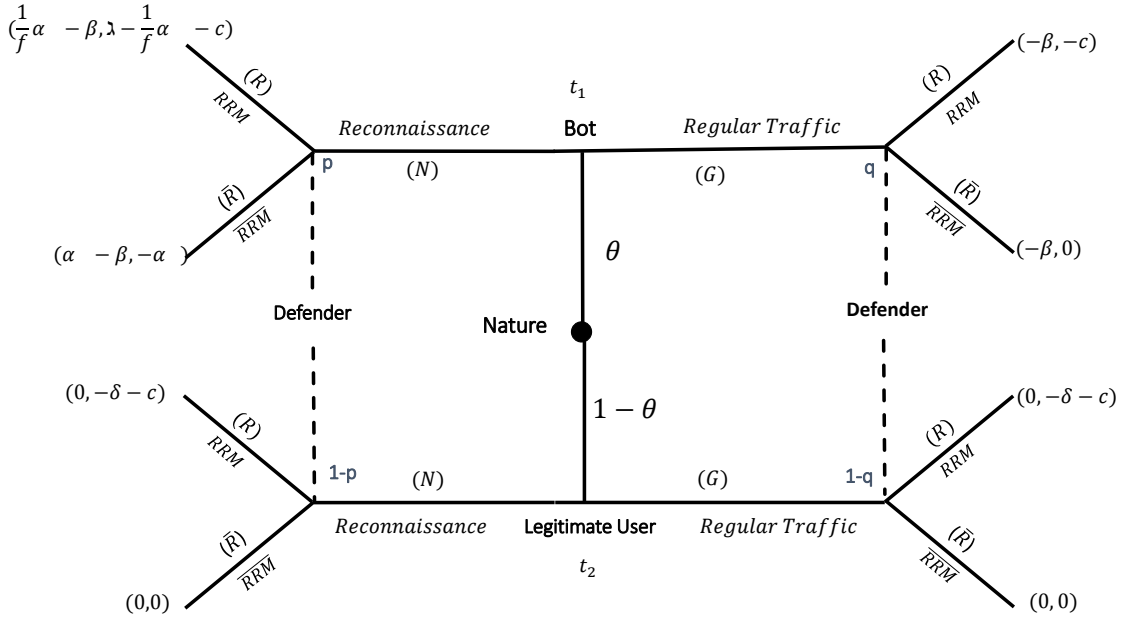


Figure 7.2: $\mathcal{G}^{cf}$ signaling game model representation.

7.2.2 Action Sets

In game $\mathcal{G}^{cf}$, we define the action set $\mathbb{A}$ of the first player by an ordered pair $(m(t_1), m(t_2))$, where $m(t_1)$ is the action of type t_1 (i.e., a bot) and $m(t_2)$ is the action of type t_2 (i.e., a legitimate user). We also assume that each sender, irrespective of his/her type, can select his/her action from the same action set. Let $\mathbb{A}$ be $\{N, G\}$, where N represents sending reconnaissance packets while G is about sending the usual/regular traffic. It is worth noting that if the attacker sends reconnaissance packets (i.e., plays N), then the defender may become suspicious. On the other hand, when the attacker sends regular traffic (i.e., plays G), s/he will not gain the necessary information.

Table 7.1: Symbols used in manuscript

Symbol	Definition
N	Reconnaissance packet sender
G	Regular packet sender
$\overline{R}$	Defender's action as no RRM
R	Defender's action as RRM
α	Attacker's gain from single reconnaissance packet
θ	Belief value of whether the client is legitimate or bot
β	Cost for renting bots
δ	Cost of not providing information to legitimate user
λ	Gain of defending through reconnaissance
c	Cost of applying RRM
f	Frequency of Random Route Mutation (RRM)
N_c	Number of hosts in the network

By using these actions, the attacker can make a strategic plan with a combination of actions. This strategy could either be slow and stealthy, or quick and greedy. If the attacker's strategy is stealthy, as defined in Section 2.4.1, a longer time will be required to do the reconnaissance. In this long period, there is a higher possibility of having changes in the link map due to existing defense strategies (or even usual network events). Hence, the attack will have less chance to be successful. On the other hand, if the attacker is so greedy about the attack, s/he might immediately do harm yet possibly end up being caught quickly. Thus, the attacker needs to do a trade-off between the time spent for reconnaissance and the risk of being detected by the defender. We will elaborate on this issue in more detail in Section 7.4.

Similar to the sender, the receiver/defender has action set $\mathbb{B}$. It can respond/reply with a route mutation to deceive the attacker (i.e., play R). Otherwise, it replies with no route change (i.e., play $\overline{R}$). Hence, $\mathbb{B} = \{R, \overline{R}\}$. If the defender applies route mutation (i.e., R), then there will be an extra cost depending on the time it takes to install flow rules on network devices. The cost for the regular replies (i.e.,

$\overline{R}$) is considered as zero since there are not any additional efforts required. We will later elaborate on the defense cost when we obtain the defender payoffs.

7.2.3 Belief Model

In game $\mathcal{G}^{cf}$, the defender does not know whether it receives messages from an attacker or a legitimate user. Yet, it observes these messages and builds the belief function (i.e., θ) about the sender's type based on this observation. Remember that this θ is common knowledge between the sender and the receiver. The defender has a different belief for each client/sender: θ_y, where $0 < y \leq N_c$. Furthermore, to represent the belief at a time instance x, we use the notation $\theta_y(x)$. The belief ranges from 0 to 1, while 0 showing no sign of suspicion of the client's legitimacy. The belief has two parts: the initial belief that is statically calculated at the beginning of the game and the dynamic belief that is updated according to the sender's actions.

Initial Belief

In our environment, we assume there is no information available about any client, and all clients are at the same level of suspicion. Thus, we assign 0 to each initial belief (i.e., $\theta_y(0)$). However, the defender can assign different values to the initial belief for each client if any prior information is available.

Dynamic Belief

The dynamic belief for a specific sender will be calculated each time the game is played. The belief is defined based on three different weighted factors, as shown in Equation (7.1). While these factors are normalized values, the summation of their weights is 1 (100%). The first factor is the previous belief (i.e., $\theta_y(x-1)$) that is weighted by F_1. The reason is that if a client has been behaving suspicious or

benign for most of the time, his/her current action, which may be different from this characteristic, cannot eliminate (or changes) his/her performance previously observed/learned. That is why we consider assigning a high weight (e.g., 90%) to F_1.

$$\theta_y(x) = \theta_y(x-1) \times F_1 + \frac{\sum_{k=1}^{N_c} \theta_k(x-1)}{N_c} \times F_2 + A(x) \times F_3 \tag{7.1}$$

The second factor is the average of the beliefs for all the network nodes (i.e., $\frac{\sum_{k=1}^{Nc} \theta_k(x-1)}{Nc}$). This factor reflects the Crossfire attack characteristic in which a number of bots are used to perform the attack. F_2 is the weight representing its impact on the belief function, and we consider it notable (e.g., 9%). The last metric we use to update our belief is the current/latest packet that we have received from the sender (i.e., $A(x)$). Its weight is F_3, which we consider as small (e.g., 1). Even though its weight is considered small, its impact on the belief adds up rapidly as the sender keeps sending packets to the receiver, and the game is continuously played for each packet receipt.

7.2.4 Payoff Model

In this section, we calculate the benefit and cost of each of the players considering all possible strategies. Let us first consider the case where the game is played between the attacker/bot (i.e., the sender type is t_1) and the defender. This case is represented by the upper part of the signaling game, as shown in Fig. 7.2.

If the defender applies RRM (i.e., plays R) in response to a reconnaissance message (i.e., the sender plays N) from the bot (i.e., given that Nature has chosen the bot to play with a probability of θ), the defender can successfully defend the targeted server(s) against the attack. Let us assume that α represents the number of packet losses that the attacker can cause by leveraging the information that is

collected through the reconnaissance process. In other words, α is the numerical gain of the attacker. If we assume that the frequency of running RRM is f, then the benefit of the attacker will be degraded for the higher frequency of RRM. Hence, in this case, we model the benefit of the attacker by $\frac{1}{f}\alpha$. We designate the attacker's cost by β, which is an increasing function on the number of bots that are deployed by the attacker in the network. This parameter also represents the capability of the attacker in our experiments: a decent or strong capability. While the decent attacker can afford a small number of bots, the strong attacker can deploy more bots. The number of packets sent by the attacker can be considered as another metric of the attacker's cost, yet this does not bring a direct cost to the attacker. Finally, considering the calculated benefit and cost for the attacker, we conclude that the attacker's payoff is $\frac{1}{f}\alpha - \beta$.

We represent the benefit of the defender (gain of defending against reconnaissance) by λ. This benefit is made by giving the attacker the wrong information about reconnaissance, which prevents the data phase of the attack. This action brings protection to the network in terms of decreased packet loss. Moreover, we represent the total cost of applying RRM by c, which is measured in terms of additional delay and packet loss that might be caused due to the flow table update. Given the calculated cost and benefit for the defender, the payoff of the defender running RRM against the bot is: $\lambda - \frac{1}{f}\alpha - c$.

Following the above discussion for the payoffs of the attacker and defender, if the defender does not react to the reconnaissance messages of the bot, the payoff of the attacker and defender will be $-\alpha$ and $\alpha - \beta$, respectively. With a similar analysis, we can show that the payoff of the defender is $-c$. If it does not defend, the payoff would be 0. The payoff of the attacker when s/he does not send reconnaissance packets will always be equal to $-\beta$.

Now we consider the second game, where the players are the legitimate user and the defender (i.e., the lower part of the signaling game presented in Fig. 7.2). Since RRM does not have a significant impact on the legitimate users' traffic, we consider its payoff as zero for all possible actions of the defender. However, running RRM against legitimate users generates wrong information to the legitimate user, and it may cause a problem with troubleshooting. We model this effect by parameter δ. Hence, the payoff of the defender when it runs RRM will be equal to $-\delta - c$.

7.3 Game Analysis and Protocol Design

In the following, we first examine game $\mathcal{G}^{cf}$ for the existence and properties of pure strategy Perfect Bayesian Nash Equilibria (PBNE). We then use our analysis to design a defensive protocol to optimize defender strategies against Crossfire attacks.

7.3.1 Game Analysis: Perfect Bayesian Nash Equilibrium

In complete information or non-Bayesian games, a strategy profile is a Nash equilibrium (NE) if every strategy in that profile is a best response to every other strategy. However, players in Bayesian games would like to maximize their expected payoffs, given their beliefs about the other players [SLB08]. A PBNE is characterized as a strategy profile and belief that satisfy the following four requirements [G$^+$92]:

Requirement 1: After observing any message m_j from sender j, the defender must have a belief about which types could have sent m_j. Denote this belief by the probability distribution $\mu(t_i|m_j)$, where $\mu(t_i|m_j) \geq 0$ for each type t_i, and $\sum_{t_i \in T} \mu(t_i|m_j) = 1$

Requirement 2: For each message m_j, the defender's action $a^*(m_j)$ must maximize his expected utility u_d, given the belief $\mu(t_i|m_j)$ about which type could have sent m_j. That is, $a^*(m_j)$ satisfies:

$$\max_{m_j \in M} \sum_{t_i \in T} \mu(t_i|m_j) u_d(t_i, m_j, a(m_j))$$

Requirement 3: For each type t_i, the sender's (whether a bot or a legitimate user) message $m^*(t_i)$ must maximize his utility (u_d), given the defender's strategy $a^*(m_j)$. That is, $m^*(t_i)$ satisfies:

$$\max_{m_j \in M} u_d(t_i, m_j, a^*(m_j))$$

Requirement 4: For each $m_j \in M$, if there exists type t_i such that $m^*(t_i) = m_j$, then the defender's belief at the information set corresponding to m_j must follow from Bayes' rule and the sender's strategy:

$$\mu(t_i|m_j) = \frac{p(t_i)}{\sum_{t_i \in T_j} p(t_i)}$$

where T_j denotes the set of types that send the message m_j. Considering the above requirements, we can now define the PBNE.

Definition 1. *A pure-strategy PBNE in a signaling game is a pair of strategy $m^*(t_i)$ and $a^*(m_j)$ and a belief $\mu(t_i|m_j)$ satisfying Requirements 1 to 4.*

In the following, we use $(p, 1-p)$ and $(q, 1-q)$ to denote the second player's (the defender) beliefs at its two information sets. Recall that for the defined signaling game in Figure 7.2, the sender's pure strategy determined by an ordered pair $(m(t_1), m(t_2))$ where $m(t_1)$ and $m(t_2)$ are the chosen strategies by user types t_1 and t_2, respectively. Note that in our model t_1 and t_2 are bot and legitimate types. Similarly, the defender's strategy is determined by an ordered pair $(a(N), a(G))$, in which $a(N)$ and $a(G)$ demonstrate the defender strategy following the sender's reconnaissance and regular traffic signals, respectively.

Furthermore, a pure strategy PBNE profile is determined as a tuple $\{\mathcal{S}_1, \mathcal{S}_2, p, q\}$, in which $\mathcal{S}_1$ is the pair of the sender strategy chosen by each type (whether bot or legitimate user), $\mathcal{S}_2$ is the pair of defender strategy in response to each signal, and p and q are attacker belief concerning the type of sender for reconnaissance (N) or regular (G) signal, respectively. According to the sender's pure strategy, two kinds of PBNE could exist in a signaling game, called *pooling* and *separating*.

A PBNE is called *pooling equilibrium* if $m(t_1) = m(t_2)$. In other words, the bot and legitimate user send the same signal, regardless of their types. In contrast, a PBNE is called *separating equilibrium* if $m(t_1) \neq m(t_2)$, i.e., the bot and legitimate users send a different signal, depending on their types. We now examine $\mathcal{G}^{cf}$ for (pure) PBNE. We first probe the existence of pooling equilibria.

Given the definition of pooling and separating equilibrium, in the following, we derive all conditions under which there exist these Nash equilibrium profiles in our defined game. In other words, since the values of payoffs vary given the topology of the network and the period of sending different packets and probes in the reconnaissance phase, there would be different conditions to be checked for the existence of these Nash equilibrium points.

Theorem 1. *For any values of θ, there exists a pooling equilibrium on N, in $\mathcal{G}^{cf}$ signaling game.*

Proof. Let us suppose that there exists a pooling NE with (N, N) strategy for the sender. Then the defender's information set corresponding to N is on the equilibrium path, so the defender's beliefs $(p, 1-p)$ at this information set is determined by Bayes' rule and sender's strategy: $p = \theta$. We first compute the expected payoff of the defender, given its belief. The defender's expected payoff for playing R is:

$$\theta \times (\lambda - \frac{1}{f}\alpha - c) + (1 - \theta) \times (-\delta - c) \tag{7.2}$$

And defender's expected payoff for playing $\overline{R}$ is:

$$\theta \times (-\alpha) + (1-\theta) \times (0) \tag{7.3}$$

Comparing the above payoffs for the defender we can define a threshold for belief, called θ^*, which is equal to $\frac{\delta+c}{\delta+\alpha(1+\lambda-\frac{1}{f})}$. We first assume that $\theta^* \leq 0$, then the following two cases could take place for the dominant strategy of the defender:

- $\theta \geq \theta^* := \frac{\delta+c}{\delta+\alpha(1+\lambda-\frac{1}{f})}$: Therefore playing R dominates $\overline{R}$, following N signal, which can be easily verified by comparing the expected payoffs presented in Equations (7.2) and (7.3).

  Now we should check whether the senders have incentives to deviate from the N strategies, given the defender strategy, which is R in this case. If the defender chooses strategy R for responding to the message G, there is no incentive for the senders to deviate from their strategies. In fact, in that case, the sender of type 1 (t_1 or Bot) achieves $-\beta$ instead of $\frac{1}{f}\alpha - \beta$. The sender of type 2 ($t_2$ or legitimate user) obtains 0 in both cases. Hence, there is no incentive for the senders to deviate from strategy N.

  It remains to consider the defender's belief at the information set corresponding to G (i.e., off the equilibrium path). We need to show if the strategy of playing R is optimal, given this belief. For this purpose and given that the R is the best response when $\theta \geq \theta^*$, we should calculate the expected payoffs of the defender when it plays R and $\overline{R}$ following strategy G. These payoffs are $q \times -c + (1-q) \times (-\delta - c)$ and $q \times 0 + (1-q) \times 0$. Since there are no values for q that makes the payoff of defender greater for playing R, there is no pooling on (G, G) when the defender plays (R, R).

  Similarly, to verify if there exists a NE where the defender plays $(R, \overline{R})$, we should first show that there are no incentives for the sender to deviate from the

pooling (N, N) strategy. This time considering the defender strategy $(R, \overline{R})$, both types of senders do not have any incentive to deviate from N and play G as their payoff would be decreased from $\frac{1}{f}\alpha - \beta$ to $-\beta$ for the bot player and the legitimate user, its payoff remains 0. Now, we should again calculate the payoff of the defender given its belief q. In other words, this time we need to compute the values of q, where $q \times -c + (1-q) \times (-\delta - c) \leq 0$. Hence for all values of $q \leq 0$, the payoff of the defender would be greater when it plays $\overline{R}$. Then, we can conclude that there exists one pooling equilibrium $\{(N, N), (R, \overline{R}), p = \theta, q\}$ for any q in $\mathbb{G}_{cf}$ when $\theta \geq \theta^*$.

- $\theta \leq \theta^* := \frac{\delta + c}{\delta + \alpha(1 + \lambda - \frac{1}{f})}$: In this case the best response of the defender following N signal is $\overline{R}$. Similar to the previous case, we should first verify if there is an incentive for the senders to deviate from N, if the defender plays $\overline{R}$ and $\overline{R}$ off the equilibrium path. Since both types will not gain any extra benefits (for bots, the payoff decreases from $\alpha - \beta$ to β and for the legitimate users, there are no differences), there are no incentives for them to deviate from playing N. Similar calculations can be done for the payoff of the defender, off the equilibrium path to find the possible values for q. Similar to the previous case, there are no values for q, where the expected payoff of playing R would be bigger than $\overline{R}$. Considering all possible deviations for the case that the defender plays $\overline{R}$ and the values of the belief for q there exists another pooling equilibrium, where $\{(N, N), (\overline{R}, \overline{R}), p = \theta, q\}$ for any q in $\mathcal{G}^{cf}$ when $\theta \leq \theta^*$.

□

Theorem 1 represents that if the selected strategies of both types of the senders are N, there is an equilibrium considering the belief of the defender for the possible attacker. In this case, depending on the value of θ, the defender should select one

Table 7.2: Equilibria and Their Conditions

Theorem	***Conditions***	***Range of*** θ	$\#$	***PBNE***	***Condition on Beliefs***	
					On-equilibrium	***Off-equilibrium***
Theorem 1	–	$\theta \geq \theta^*$	$\mathcal{PBNE}_1$	$\{(N,N),(R,\overline{R}),p,q\}$	$p=\theta$	$\forall q$
Theorem 1	–	$\theta \leq \theta^*$	$\mathcal{PBNE}_2$	$\{(N,N),(\overline{R},\overline{R},),p,q\}$	$p=\theta$	$\forall q$
Theorem 4	$\lambda \geq \lambda^*$	$\forall\theta^*$	$\mathcal{PBNE}_3$	$\{(N,G),(R,\overline{R}),p,q\}$	$p=1$	$q=0$
Theorem 4	$\lambda \leq \lambda^*$	$\forall\theta^*$	$\mathcal{PBNE}_4$	$\{(N,G),(\overline{R},\overline{R}),p,q\}$	$p=1$	$q=0$

of the strategies R or $\overline{R}$ upon receiving signal N. In other words, if the probability of the sender being a bot (i.e., θ) is greater than θ^*, the defender should run RRM. Otherwise, the best response for the defender is playing $\overline{R}$ strategy. In the case of $\theta \geq \theta^*$ and $\theta \leq \theta^*$, if users send reconnaissance packets, the defender's response will be R and $\overline{R}$, respectively.

Theorem 2. *For any values of* θ*, there does not exist any pooling equilibrium on* G*, in* $\mathcal{G}^{cf}$ *signaling game.*

Proof. In the game $\mathcal{G}^{cf}$ for any values of belief θ, the defender's best respond to pooling strategy of (G,G) is always $\overline{R}$. In other words, the defender does not perform RRM in this case. Consequently, we should see if the senders have any incentive to deviate from G strategy. Let us consider two possible strategies of the defender off the equilibrium path (when it believes in playing with Bot with probability p). Since by deviating from G to N, the sender of type t_1 (Bot) can always increase his/her payoff from $-\beta$ to $\frac{1}{f}\alpha - \beta$ (when the defender plays R off the equilibrium path) or from $-\beta$ to $\alpha - \beta$ (when the defender plays $\overline{R}$ off the equilibrium path), (G,G) cannot be at any pooling equilibrium. □

Theorem 3. *There is no separating equilibrium on* (G,N) *in the* $\mathcal{G}^{cf}$ *signaling game.*

Proof. Suppose (G,N) is a pair of senders' strategy, then both of the defender's information sets are on the equilibrium path, so both beliefs are determined by

Bayes' rule and sender strategy: $q = 1$ and $p = 0$. Defender's best response following these beliefs is always $\overline{R}$, for both types of the sender. Hence, we should check if the sender's strategy is optimal, given the defender strategy. If the sender of type 1 deviates by playing N signal instead of G, the defender responds with $\overline{R}$, giving t_1 (i.e., the Bot) a payoff of $\alpha - \beta$, which exceeds t_1's payoff of $-\beta$ from playing G. Thus, (G, N) cannot establish any separating equilibrium. □

Theorem 4. *There are two classes of separating equilibrium* $\{(N, G), (\overline{R}, \overline{R}), p = 1, q = 0\}$ *if* $\lambda \leq \alpha(1/f - 1) + c$ *and* $\{(N, G), (R, \overline{R}), p = 1, q = 0\}$ $\lambda \geq \alpha(1/f - 1) + c$ *in the* $\mathcal{G}^{cf}$ *signaling game.*

Proof. Similar to Theorem 3, we first assume that (N, G) is a pair of senders' strategy. Then both of the defender's information sets are on the equilibrium path: $p = 1$ and $q = 0$. Considering (N, G) strategy of the senders, the defender's best response following these beliefs can be calculated by comparing the payoffs of the defenders. For sender with type t_1 (i.e., bot), the best response of the defender is calculated by comparing the following two payoffs: $\lambda - \frac{1}{f}\alpha - c$ and $-\alpha$. Two cases can be identified given that $\lambda^* = \alpha(\frac{1}{f} - 1) + c$:

- $\lambda \geq \lambda^*$: The best response to play N by type t_1 is R and the best response to type G by type t_2 is $\overline{R}$. We should check, if the sender's strategy is optimal given the defender strategy. Since the bot payoff would be decreased from $\frac{1}{f}\alpha - \beta$ to $-\beta$ and there is no difference for the legitimate payoff, we can conclude that the $\{(N, G), (\overline{R}, \overline{R}), p = 1, q = 0\}$ is a PBNE. We name it as $\mathcal{PBNE}_3$ for later references.
- $\lambda \leq \lambda^*$: In this case, the best response to play N by type t_1 is $\overline{R}$ and the best response to type G by type t_2 is $\overline{R}$. Similar to the previous case there is no incentive for the senders to deviate from (N, G) strategy.

□

The above results present all possible PBNEs of game $\mathcal{G}^{cf}$. Considering these equilibria, we can now provide the best plans of actions for the defender, given its belief about the sender's type and its payoff at different strategies.

7.3.2 Protocol Design

In this section, we design a strategic Crossfire defense mechanism to optimize the strategy of the defender. Table 7.2 summarizes the results presented in Theorems 1, 2, 3, and 4. All possible separating and pooling PBNEs are displayed. Leveraging these results, we design our proposed strategic Crossfire defense mechanism, namely *Strategic RRM*. The pseudocode of this proposed mechanism is given in Algorithm 7.

In the Algorithm 7, we first pick a value for the frequency (f) of RRM that represents how often the defender can perform RRM. It is decided through our preliminary experiments. The minimum period of running RRM is shown as P, which is assigned to $1/f$. Then the belief for each client is initialized to zero at the beginning of the experiments. The defender always updates its belief each time it receives a packet, and it computes the potential gain/payoff. Later, if the belief is greater than θ^* and the received packet is reconnaissance, the defender chooses to run RRM at the end of P period. In addition to that, if the value of λ is greater than λ^*, the defender should also run the RRM. Hence, the optimal decision for the defender is obtained according to equilibria $\mathcal{PBNE}_1$ and $\mathcal{PBNE}_3$. The same processes after the initialization will be performed every P seconds.

Algorithm 7 Strategic Crossfire Defense Mechanism

1: h: Host ID (i.e., its IP address) communicating with the server
2: t^h_{alive}: The time that host h is still transmitting packets
3: f: The frequency at which we can run RRM
4: $P \Leftarrow \frac{1}{f}$
5: $n := 1$
6: $\theta_h \Leftarrow$ Initialize the belief for h
7: **while** $((n-1) \times T) \leq t^h_{alive}$ **do**
8: Estimate/calculate α, β, λ, δ, and c.
9: **for** Each packet received from h between $(n-1)T$ and nT **do**
10: Update θ_h according to the packet type (signal)
11: Compute θ^* (Theorem 1)
12: Compute λ^* (Theorem 4)
13: **end for**
14: **if** $\lambda \geq \lambda^*$ **then**
15: Perform RRM
16: **else if** $\theta \geq \theta^*$ **then**
17: Perform RRM
18: **end if**
19: $n := n + 1$
20: **end while**

7.4 Evaluation

In this section, we briefly discuss our experimental setup and evaluation metrics. Then, we present the findings from the experiments.

7.4.1 Experiment Setup

The network environment is implemented using Mininet [Tea] and FloodLight [Pro] is used as an SDN Controller. The proposed signaling game-based defense strategy is developed as an application on top of the FloodLight Controller. In the experimental setup, we consider the network topology with 20 switches, 3 decoy servers, 1 target server, and several clients, as shown in Fig. 7.3. The number of clients differs for each experiment based on the attack model and the purpose of the experiment. The target server is encircled by red and decoy servers are placed interior part of

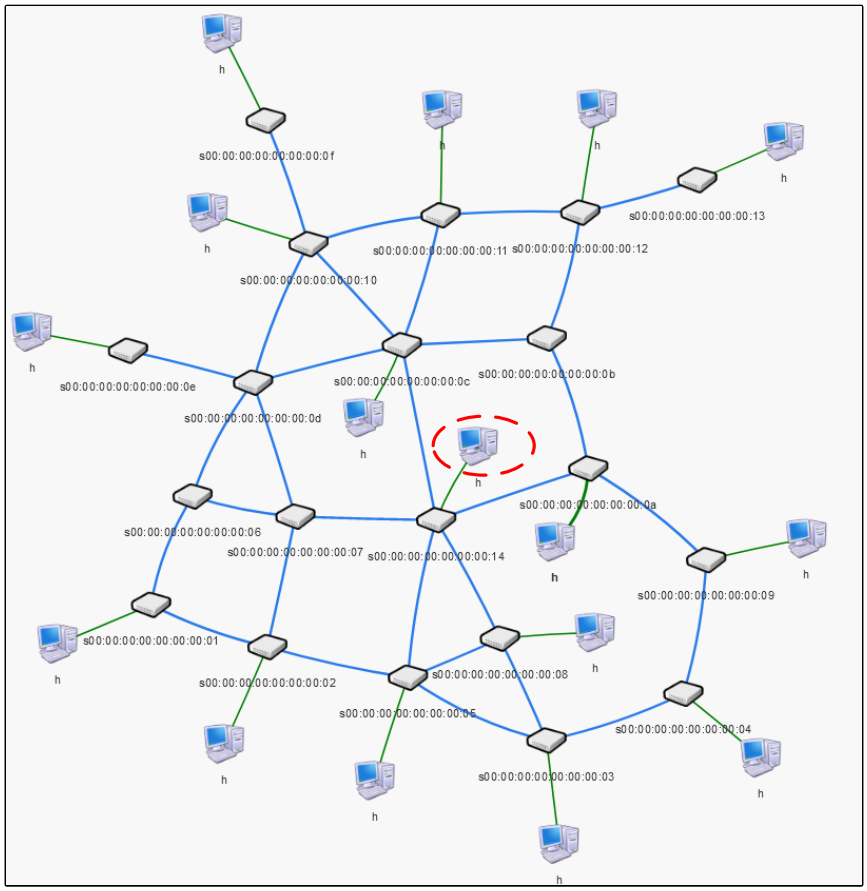


Figure 7.3: Experimental Network Topology

the network topology in Fig. 7.3. The bandwidth of each link in the network is configured as 100 Mbps. A python-based client/server application is implemented for the communication between host-target servers. We implement different attacker models and apply them in experiments to evaluate them. The configuration of the network is further explained below.

Defender's Setup

In our experiments, we consider 3 different configurations for the defender. We name them as *No Defense*, *Periodic RRM*, and *Strategic RRM*.

- *No Defense*: To observe the worst-case attack scenario the network environment is implemented without any defense mechanism. In this case, the attacker can easily gain the (static) routing information by doing reconnaissance attacks. Next, s/he attacks the target links which are found by using that static information.
- *Periodic RRM*: The second defense setup is based on performing RRM periodically. The routes are changed periodically to the alternative ones that are pre-calculated. We choose different values of frequencies to run RRM in order to observe the impact.
- *Strategic RRM*: It is our proposed signaling game-based defense mechanism. In this case, the defender takes actions based on the attacker's actions, as explained in Section 7.2.2. The mechanism is given in Algorithm 7 and is implemented accordingly.

Attacker's Setup

In a Crossfire attack setup, a decoy server does not receive a large amount of traffic from one or multiple bots simultaneously so that the activity is not considered as suspicious at the network administrator's side. Thus, we limit each bot's communication to only 1 decoy server with a regular amount of traffic. Each bot sends/receives approximately 5 Mbit of data per second in order to behave as legitimate.

We implement two different attacker behaviors, namely *aggressive* and *stealthy*, with two different capabilities, such as *decent* and *strong*, as mentioned in Section 2.4.1. The aggressive attacker's time for reconnaissance is fixed to 1 minute in experiments. The decent attacker has a limited number of bots, which is as much as the number of legitimate clients in the network. Meanwhile, the strong attacker is equipped with a doubled number of bots. Even though the attack impact is ex-

pected to increase, it should also be noted that the more bots the attacker employs, the higher cost is incurred on him/her. We run our experiments on each case to show how effective our solution would be with different attack models.

Table 7.3: Delay Caused by RRM (in microsecond)

Client ID	***Delay Without RRM***	***Delay with RRM***					
		Optimum size Path			***Any size Path***		
		60-seconds	***30-seconds***	***10-seconds***	***60-seconds***	***30-seconds***	***10-seconds***
1	2853	2892	3040	3398	3781	3717	4053
2	2854	2898	3050	3406	3761	3616	4145
3	1497	1533	1629	1882	1973	2075	2351
4	1499	2161	2322	2698	2986	3205	3578

7.4.2 Evaluation Metrics

The application of RRM introduces some overhead costs on the defender since it creates extra packets in the network. This overhead can cause Quality of Service (QoS) problems in the form of increased communication latency/packet delay or a higher number of packet losses for legitimate users. We specifically consider the following metrics to be measured in our experiments:

- *Delay for using longer path:* Using a randomized alternative path can cause longer end-to-end delay since the route may not be the optimal/shortest anymore. This metric shows the increase in delay compared to the optimal path.
- *Delay for flow table updates:* Flow tables are updated whenever RRM is triggered. This process of updating flow table entries of switches takes time. This time is measured to show additional delays that legitimate clients are exposed to. Related to this issue, some of the packets may drop if the queue of a switch becomes full, and it cannot handle any more packets whenever the flow table is updated even though no attack is being taken place.

- *The number of packet losses:* This metric represents the number of packets lost for legitimate users due to attacks or the execution of RRM.

7.4.3 Experimental Results

We run experiments on different network configurations to observe each evaluation metric separately. We first show the overhead of performing RRM on the network (i.e., legitimate users) varying the RRM frequency, i.e., the interval period between two subsequent route mutations. We find the optimal period among them, and use this *Periodic RRM* to compare with the proposed *Strategic RRM* by running experiments with different adversary models.

Observing RRM Costs without any Attacker

We run our experiments without considering any malicious activities in the network in order to measure the defender's cost (i.e., the cost imposed on its clients). In the experiments, we measure the additional delay that is caused due to updating flow tables as well as using longer paths when RRM is applied. We first change the RRM frequency to observe the flow table update cost. Here, the routing paths are kept the same. Then, to assess the delay for using longer paths, we run experiments selecting optimal (shortest length) paths as well as non-optimal (alternative) paths.

Table 7.3 presents the average packet delay for 4 randomly selected legitimate clients. We can easily see that a higher frequency of RRM (i.e., when the RRM interval period is 10 seconds) brings additional delay to the clients if alternative path sizes are the same as the optimum path. The slight difference between each column is caused by flow table updating. As an example, we can take a look at the delays for Client 1: during the 60-seconds period RRM, the average delay is 2892 milliseconds, while the delay becomes 3040 milliseconds and 3398 milliseconds

Table 7.4: Packet Drop Caused by Flow Table Update

10-seconds RRM	*30-seconds RRM*	*60-seconds RRM*
10.1%	8.6 %	1.6%

when the periods are 30-seconds and 10-seconds, respectively. Even though the difference between them looks negligible (in milliseconds), it should be noted that the average delay is computed for approximately 100 thousands of packets in a 10-minute long experiment. Thus, the sum of additional delays that are caused by flow table updates is a significant cost considering the delay increase for 100 thousands of packets where only a few times flow table updates are performed. We can also observe in Table 7.3 that if we use alternative (i.e., non-optimal) paths for clients, the average delay increases significantly. This is because the increased number of hops adds further delay to the packet communication.

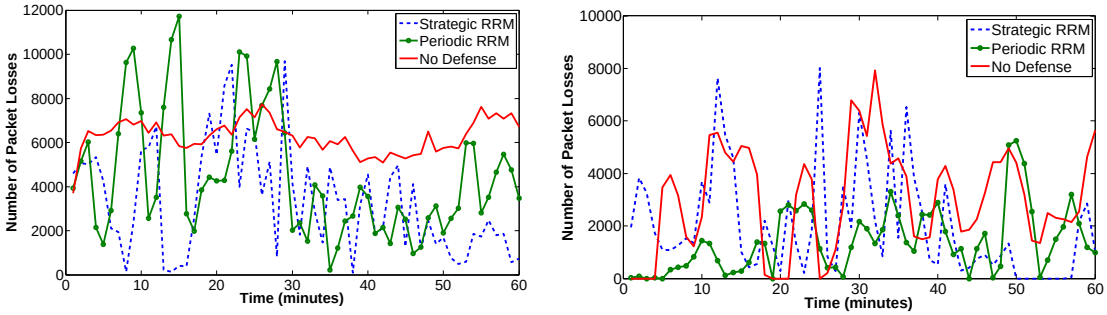
Next, we assess if the route updates can cause packet drops. The experimental setup remains the same as the previous ones except we consider a higher number of clients to send more simultaneous packets in order to occupy the queues/buffers at the switches and simulate an environment that would more likely occur during Crossfire attacks. We use small-sized packets (100 bytes each) so that the link bandwidth cannot be a cause of packet dropping. The results from the experiments are presented in Table 7.4. We observe that a higher frequency of RRM causes a larger number of packet drops. This is reasonable considering switches are busier updating their flow tables in the cases where more often route changes occurred. In other words, some of the packets cannot be handled on time and are dropped because of the overflowing of the queue.

The 60-seconds RRM case causes fewer delays and less packet loss compared to 30-seconds and 10-seconds RRM cases if there is no attack, as shown in Tables 7.3

Table 7.5: Cost Metric for Different Attacker Model

Attacker Model	*Packet Loss*			*Average Delay (in milliseconds)*		
	No Defense	*Periodic RRM*	*Strategic RRM*	*No Defense*	*Periodic RRM*	*Strategic RRM*
Strong-Aggressive	31%	22%	16%	1172	989	438
Strong-Stealthy	15%	7%	10%	579	380	278
Decent-Aggressive	7%	2%	2%	464	251	200
Decent-Stealthy	3%	0.7%	1%	263	168	141

and 7.4. In other words, the 10-seconds RRM case causes lots of overhead compared to 60-seconds RRM. Thus, we opt to run 60-seconds RRM in the following experiments to compare with *Strategic RRM*.

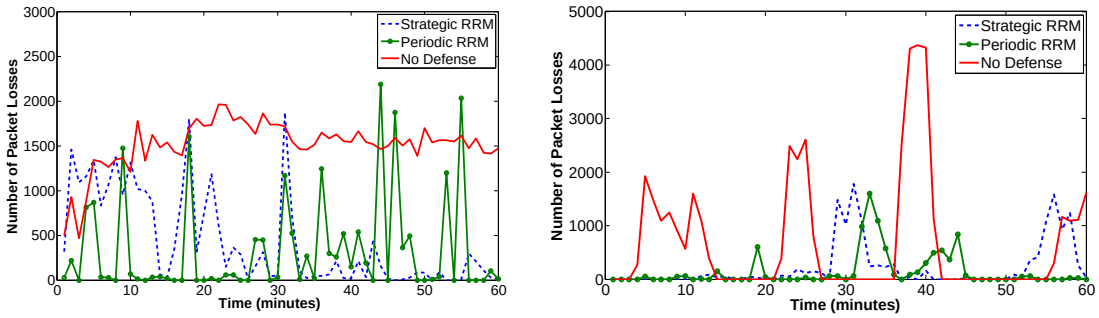


(a) Aggressive Behavior Model (b) Stealthy Behavior Model

Figure 7.4: Packet Loss with Strong Attacker

Comparing Strategic RRM with Periodic RRM

The values for different game parameters of *Strategic RRM* are chosen as shown in Table 7.6. These parameters are observed and tuned based on our preliminary experiments on the RRM cost and the impact of attacks. More specifically, α is the attacker's gain, and it is considered as the percentage of packet losses due to the attack (when there is *No Defense* mechanism), which is 3% where the attacker is stealthy and decent, as shown in Table 7.5. This value increases if the attacker's strategy is more intense or network size is smaller. The parameter λ is the gain of the defender that s/he earns by defending against the reconnaissance attack. It



(a) Aggressive Behavior Model (b) Stealthy Behavior Model

Figure 7.5: Packet Loss with Decent Attacker

Table 7.6: Parameters Used in Simulation

Parameter Name	α	λ	δ	c	f
Value	3	2.3	9	1.6	1

is calculated by deducting 0.7% packet loss when the 60-seconds *Periodic RRM* is applied from 3% packet loss when there is *No Defense* mechanism. Thus, λ is set at 2.3. λ increases if RRM is applied more often than 60-seconds. The parameter δ is the cost of the defender with respect to legitimate users. We derive this parameter as the percentage of increase in the average packet delay from the case of *No Defense* (a delay of 2175 microseconds) to the case of 60-seconds RRM (a delay of 2371 microseconds). Hence, we calculate δ as 9. The value of δ will increase if RRM is applied more often. In addition, we assign 1.6 to c since it is the packet loss (as shown in Table 7.4) in the case of 60-seconds RRM. The cost variable, c, increases if RRM is more frequent. Finally, we consider the minimum possible interval of running RRM as 1 second. Hence, the frequency (f) is set to 1.

In *Strategic RRM*, the defender builds its belief about each of its clients of being a bot and performs RRM accordingly, as it is mentioned in Section 7.2.3. In other words, the attacker is able to run a stealthy attack initially until the belief increases to some extent. Therefore, it is expected to see more packet losses for *Strategic*

RRM at the beginning of the experiments. Meanwhile, it is also expected that *Periodic RRM* causes a similar number of packet losses since it has the same strategy throughout the experiments.

In the next experiments, we implement different attack models for each defender type and run them separately. In order to observe the performance in different attack models, we plot the graphs individually in Fig. 7.4 and Fig. 7.5. Packet loss percentage and average delay for all network nodes are given in Table 7.5. The attacker model demonstrated in Fig. 7.4a considers a strong one with aggressive behavior. As Fig. 7.4a shows, in the case of *No Defense*, the number of packet losses are high, and it stays almost similar throughout the experiments. However, in the cases of *Periodic RRM* and *Strategic RRM* mechanisms, the scenario changes dramatically. *Strategic RRM* has a much less number of packet losses after the first few minutes of the experiment. This initial period takes the belief to a state at which RRM starts to perform properly against the possible bots. An important observation is that *Periodic RRM* usually has a higher number of packet losses continuously since it changes the routes at every time interval of the period, which may not overlap with the attacker's reconnaissance phase. However, this is not the case for *Strategic RRM*. It defends against the attack only when there are highly suspicious activities, which is reflected by the increased belief and the equilibrium conditions. The other interesting observation in Fig. 7.4a is that the peak number of packet losses appears at the beginning of the RRM cases, not in the *No Defense* case. This can be explained as the nature of RRM. Since it is based on random mutation, it may cause more routes to the same link, which can end up congesting that specific link intensively. However, in the *No Defense* case, the attacker can only target the same link set unless new bots are added, and new routing paths are identified. In this attack model, the overall network's packet loss percentages

are 31% for *No Defense*, 22% for *Periodic RRM*, and 16% for *Strategic RRM* as presented in Table 7.5. Average transmission delays are parallel to the packet loss results and decrease from 1172 milliseconds at *No Defense* case to 989 milliseconds *Periodic RRM* case and to 438 milliseconds in *Strategic RRM* case.

In addition to the attacker with aggressive behavior, we also consider a stealthy one and represent the results in Fig. 7.4b. In this graph, we can see unbalanced results, which are basically due to the attacker's dynamic behavior. The stealthy attacker model can stay idle some of the time and can make longer reconnaissance attacks. Hence, it leads to such uneven consequences. Unlike the aggressive attack model, *Strategic RRM* has higher packet losses (i.e., 10%) compared to *Periodic RRM* case's packet loss (i.e., 7%), even after the belief gets time to be matured as shown in Table 7.5. The main reason behind this behavior is the characteristic of the stealthy attacker model where the attacker reduces and increases the attack intensity randomly, making the defender's belief change more often. Even though *Strategic RRM* has higher packet loss, the average transmission delay is 278 milliseconds for the *Strategic RRM* case while it is 380 milliseconds in *Periodic RRM*. Moreover, it should be noted that *Periodic RRM* brings additional overhead to the network even if there is not an attack targeting the network. Hence, we claim that it is reasonable to use *Strategic RRM* considering the overall advantages.

Furthermore, we run the same experiments with a decent attacker. As we defined earlier, the number of bots he has is half of the bots that the aggressive attacker has. The results are shown in Fig. 7.5, which are correlated with the ones in Fig. 7.4. One significant observation is that the numbers of packet losses are decreased by more than half. In other words, increasing the number of bots, as in the aggressive case, raises the damage significantly, more than a linear increase. It is easily noticed that the number of packet losses goes down to zero in the case of the decent attacker,

as shown in Fig. 7.5a and Fig. 7.5b. This is because there are fewer bots to be protected against. The other result to pay attention is that there is a more high number of packet losses consecutively in *Strategic RRM* compared to a strong attacker since less number of bots create less suspicion compared to more bots as it is shown in Equation (7.1). This interesting outcome can be seen in the comparison of Fig. 7.4b and Fig. 7.5b. In addition to that, *Strategic RRM* has almost the same overall contribution (e.g., 2% for aggressive, and 1% for stealthy) to the network defense compared to *Periodic RRM* (e.g., 2% for aggressive, and 0.7% for stealthy) as reported in Table 7.5. Yet, there is still a notable transmission delay decrease from 251 milliseconds to 200 milliseconds with Aggressive Attacker, and from 168 milliseconds to 141 milliseconds with Stealthy Attacker whenever *Strategic RRM* is used.

7.5 Conclusion

In this chapter, we present a signaling game-based dynamic MTD to defend against Crossfire attacks. We first model the attacker and the defender as a signaling game. Considering their payoffs, we compute the equilibria of the game, which represents the best strategies for each player considering the opponent is rational. According to the game results, we develop an algorithm, namely *Strategic RRM*. We implement and compare it with *Periodic RRM*. Our experimental results show that *Strategic RRM* can lessen the impact of Crossfire attacks similar to *Periodic RRM*, while it brings significantly less overhead.

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

In this dissertation, we proposed several defense frameworks against SLFAs. In this section, we summarize our works in each chapter. In our first work, we considered mitigating the SLFA attack before the network links incur significant damage. In other words, we changed the traffic paths before any communication link is congested. We showed that our framework is able to provide protection depending on the frequency of route updates. Furthermore, we observed that the benefit of TCP traffic is higher than that of the UDP.

In our second work, we proposed a framework for the ISP networks. In this framework, we employed virtual network switches that deceive the attacker's vision of the networking paths. We also created virtual collection points (VCPs) that behaves as a middle-box and analyzes all the interesting traffic. These VCPs were also used to do forensics investigations for any incident that happens in the network. In the experimental evaluations, we presented that the attacker has a much lower success rate planning the data phase of SLFA.

In the third work, we considered the effectiveness of the network debugging tools while employing MTD. In this manner, we utilized cloud-based services that provide necessary information to the requester while the malicious requester is not able to create target links for SLFA. We showed that the cloud-based overlay network contributes to the system's security with negligible additional delays.

Finally, in the last work, we strived to solve the problem of deciding and picking the best strategy to employ MTD. We modeled the attacker and the defender into a game and tracked the interaction between them. We showed that our game-based solution has far less number of packet loss compared to the periodical MTD application.

We present several key directions for future research directions:

- It would be interesting to improve the MTD mechanism by investigating possible integration of machine-learning for enhanced early detection of potential DDoS attacks. Furthermore, Intrusion Detection Systems such as SNORT [R+99] can be integrated into the system for further security analysis.
- It would be beneficial to investigate the Virtual Network Placement problem, which involves and explores the best usage of the virtual nodes in the network. In our framework, we considered virtual nodes installed in a static location without any resource constraints. Thus, the placement of virtual network nodes would be improved and chosen more effectively and wisely.
- Another future research direction would be to extend our strategic MTD-based framework for other DDoS-based emerging attacks. In other words, we would like to consider the interaction between the attacker and the defender for other similar attacks. For instance, a recent attack called the Maestro [MSS19] exploits Border Gateway Protocol (BGP) with poisoning messages which would be investigated to mitigate with considering the interaction between BGP routers.

REFERENCES

[AKK+13] Marios Anagnostopoulos, Georgios Kambourakis, Panagiotis Kopanos, Georgios Louloudakis, and Stefanos Gritzalis. Dns amplification attack revisited. *Computers & Security*, 39:475–485, 2013.

[AMMR17] Mohammad Taghi Adili, Amin Mohammadi, Mohammad Hossein Manshaei, and Mohammad Ashiqur Rahman. A cost-effective security management for clouds: A game-theoretic deception mechanism. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 98–106. IEEE, 2017.

[AMR05] William H Allen, Gerald A Marin, and Luis A Rivera. Automated detection of malicious reconnaissance to enhance network security. In *SoutheastCon, 2005. Proceedings. IEEE*, pages 450–454. IEEE, 2005.

[ARZMT06] Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose, and Andreas Terzis. A multifaceted approach to understanding the botnet phenomenon. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 41–52. ACM, 2006.

[ASA18] Abdullah Aydeger, Nico Saputro, and Kemal Akkaya. Utilizing nfv for effective moving target defense against link flooding reconnaissance attacks. In *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*, pages 946–951. IEEE, 2018.

[ASA19] Abdullah Aydeger, Nico Saputro, and Kemal Akkaya. A moving target defense and network forensics framework for isp networks using sdn and nfv. *Future Generation Computer Systems*, 94:496–509, 2019.

[ASAR16] A. Aydeger, N. Saputro, K. Akkaya, and M. Rahman. Mitigating crossfire attacks using sdn-based moving target defense. In *2016 IEEE 41st Conference on Local Computer Networks (LCN)*, pages 627–630, Nov 2016.

[ATT16] ATT. ECOMP (enhanced control, orchestration, management & policy) architecture white paper. `http://about.att.com/content/dam/snrdocs/ecomp.pdf`, March 2016. [Online; accessed 2020-5-14].

[BAS11] Theophilus Benson, Aditya Akella, and Aman Shaikh. Demystifying configuration challenges and trade-offs in network-based ISP services. *SIGCOMM Comput. Commun. Rev.*, 41(4):302–313, August 2011.

[BBH+14] Adam Bates, Kevin Butler, Andreas Haeberlen, Micah Sherr, and Wenchao Zhou. Let SDN be your eyes: Secure forensics in data center networks. In *Proceedings of the NDSS Workshop on Security of Emerging Network Technologies (SENT'14)*, pages 1–7, February 2014.

[BCAB15] Md Faizul Bari, Shihabur Rahman Chowdhury, Reaz Ahmed, and Raouf Boutaba. On orchestrating virtual network functions. In *2015 11th International Conference on Network and Service Management (CNSM)*, pages 50–56. IEEE, 2015.

[BSR13] Mitko Bogdanoski, Tomislav Suminoski, and Aleksandar Risteski. Analysis of the syn flood dos attack. *International Journal of Computer Network and Information Security (IJCNIS)*, 5(8):1–11, 2013.

[BSST09] Clark W Barrett, Roberto Sebastiani, Sanjit A Seshia, and Cesare Tinelli. Satisfiability modulo theories. *Handbook of satisfiability*, 185:825–885, 2009.

[CCFB14] Thomas E Carroll, Michael Crouse, Errin W Fulp, and Kenneth S Berenhaut. Analysis of network address shuffling as a moving target defense. In *Communications (ICC), 2014 IEEE International Conference on*, pages 701–706. IEEE, 2014.

[CFH+05] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live migration of virtual machines. In *Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2*, NSDI'05, pages 273–286, Berkeley, CA, USA, 2005. USENIX Association.

[CGC+07] Rafael Asorey Cacheda, Daniel Castro García, Antonio Cuevas, Francisco Javier González Castano, Javier Herrero Sánchez, Georgios Koltsidas, Vincenzo Mancuso, José Ignacio Moreno Novella, Seounghoon Oh, and Antonio Pantò. Qos requirements for multimedia services. In *Resource Management in Satellite Networks*, pages 67–94. Springer, 2007.

[CKM+18] William Casey, Ansgar Kellner, Parisa Memarmoshrefi, Jose Andre Morales, and Bud Mishra. Deception, identity, and security: the game theory of sybil attacks. *Communications of the ACM*, 62(1):85–93, 2018.

[CMN+14] William Casey, Jose A Morales, Thomson Nguyen, Jonathan Spring, Rhiannon Weaver, Evan Wright, Leigh Metcalf, and Bud Mishra. Cyber security via signaling games: Toward a science of cyber security. In *International Conference on Distributed Computing and Internet Technology*, pages 34–42. Springer, 2014.

[CRB12] Mosharaf Chowdhury, Muntasir Raihan Rahman, and Raouf Boutaba. Vineyard: Virtual network embedding algorithms with coordinated node and link mapping. *IEEE/ACM Transactions on Networking (TON)*, 20(1):206–219, 2012.

[CSA+20] Jin-Hee Cho, Dilli P Sharma, Hooman Alavizadeh, Seunghyun Yoon, Noam Ben-Asher, Terrence J Moore, Dong Seong Kim, Hyuk Lim, and Frederica F Nelson. Toward proactive, adaptive defense: A survey on moving target defense. *IEEE Communications Surveys & Tutorials*, 22(1):709–745, 2020.

[CSP15] Adrian Chavez, William M.S. Stout, and Sean Peisert. Techniques for the Dynamic Randomization of Network Attributes. In *Proceedings of the 49th Annual International Carnahan Conference on Security Technology*, pages 1–6, Taipei, Taiwan, Republic of China, Sept. 21–24, 2015.

[CY15] Andrew Robert Curtis and Praveen Yalagandula. Elephant flow detection in a computing device, September 1 2015. US Patent 9,124,515.

[DASJ13] Qi Duan, Ehab Al-Shaer, and Haadi Jafarian. Efficient random route mutation considering flow and network constraints. In *Communications and Network Security (CNS), 2013 IEEE Conference on*, pages 260–268. IEEE, 2013.

[DM04] Christos Douligeris and Aikaterini Mitrokotsa. Ddos attacks and defense mechanisms: classification and state-of-the-art. *Computer Networks*, 44(5):643–666, 2004.

[DRS+12] Jonathon Duerig, Robert Ricci, Leigh Stoller, Matt Strum, Gary Wong, Charles Carpenter, Zongming Fei, James Griffioen, Hussamuddin Nasir, Jeremy Reed, et al. Getting started with geni: A user tutorial. *ACM SIGCOMM Computer Communication Review*, 42(1):72–77, 2012.

[ESA12] Meisam Eslahi, Rosli Salleh, and Nor Badrul Anuar. Bots and botnets: An overview of characteristics, detection and challenges. In *2012 IEEE International Conference on Control System, Computing and Engineering*, pages 349–354. IEEE, 2012.

[FLH+00] Dino Farinacci, Tony Li, Stan Hanks, David Meyer, and Paul Traina. Generic routing encapsulation (gre). Technical report, RFC 2784, March, 2000.

[FRZ14] Nick Feamster, Jennifer Rexford, and Ellen Zegura. The road to sdn: an intellectual history of programmable networks. *ACM SIGCOMM Computer Communication Review*, 44(2):87–98, 2014.

[FTSB15] Seyed K Fayaz, Yoshiaki Tobioka, Vyas Sekar, and Michael Bailey. Bohatei: Flexible and elastic ddos defense. In *24th {USENIX} Security Symposium ({USENIX} Security 15)*, pages 817–832, 2015.

[G+92] Robert Gibbons et al. *A primer in game theory*. Prentice Hall, 1992.

[GKLD16] Dimitrios Gkounis, Vasileios Kotronis, Christos Liaskos, and Xenofontas Dimitropoulos. On the interplay of link-flooding attacks and traffic engineering. *ACM SIGCOMM Computer Communication Review*, 46(2):5–11, 2016.

[Gko14] Dimitrios Gkounis. Cross-domain dos link-flooding attack detection and mitigation using sdn principles. *MSc Th., ETH*, page 49, 2014.

[HAB+15] O. Haq, Z. Abaid, N. Bhatti, Z. Ahmed, and A. Syed. Sdn-inspired, real-time botnet detection and flow-blocking at isp and enterprise-level. In *2015 IEEE International Conference on Communications (ICC)*, pages 5278–5283, June 2015.

[HB16] Juliver Gil Herrera and Juan Felipe Botero. Resource allocation in nfv: A comprehensive survey. *IEEE Transactions on Network and Service Management*, 13(3):518–532, 2016.

[HGJL15] Bo Han, Vijay Gopalakrishnan, Lusheng Ji, and Seungjoon Lee. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, 53(2):90–97, 2015.

[HHB14] Fei Hu, Qi Hao, and Ke Bao. A survey on software-defined network and openflow: From concept to implementation. *IEEE Communications Surveys & Tutorials*, 16(4):2181–2206, 2014.

[HKP13] Bryan Harris, Eli Konikoff, and Phillip Petersen. Breaking the ddos attack chain. *Institute for Software Research*, pages 1–16, 2013.

[HTS15] T. Hirayama, K. Toyoda, and I. Sasase. Fast target link flooding attack detection scheme by analyzing traceroute packets flow. In *Information Forensics and Security (WIFS), 2015 IEEE International Workshop on*, pages 1–6, Nov 2015.

[HWJ05] J. Han, D. Watson, and F. Jahanian. Topology aware overlay networks. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 4, pages 2554–2565 vol. 4, March 2005.

[HWJ06] J. Han, D. Watson, and F. Jahanian. An experimental study of internet path diversity. *IEEE Transactions on Dependable and Secure Computing*, 3(4):273–288, Oct 2006.

[JASD12] Jafar Haadi Jafarian, Ehab Al-Shaer, and Qi Duan. Openflow random host mutation: Transparent moving target defense using software defined networking. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, pages 127–132, New York, NY, USA, 2012. ACM.

[JASD13a] Jafar Haadi Jafarian, Ehab Al-Shaer, and Qi Duan. Formal approach for route agility against persistent attackers. In *Proceedings of the Computer Security – ESORICS 2013: 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013.*, pages 237–254, 2013.

[JASD13b] Jafar Haadi Jafarian, Ehab Al-Shaer, and Qi Duan. Formal approach for route agility against persistent attackers. In *European Symposium on Research in Computer Security*, pages 237–254. Springer, 2013.

[JASD15] Jafar Haadi Jafarian, Ehab Al-Shaer, and Qi Duan. An effective address mutation approach for disrupting reconnaissance attacks. *Information Forensics and Security, IEEE Transactions on*, 10(12):2562–2577, 2015.

[JL14] EunHee Jeong and ByungKwan Lee. An ip traceback protocol using a compressed hash table, a sinkhole router and data mining based on network forensics against network attacks. *Future Generation Computer Systems*, 33:42 – 52, 2014.

[JSS13] Quan Jia, Kun Sun, and Angelos Stavrou. Motag: Moving target defense against internet denial of service attacks. In *Computer Communications and Networks (ICCCN), 2013 22nd International Conference on*, pages 1–9. IEEE, 2013.

[JYR+16] A. H. M. Jakaria, W. Yang, B. Rashidi, C. Fung, and M. A. Rahman. Vfence: A defense against distributed denial of service attacks using network function virtualization. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, volume 2, pages 431–436, 2016.

[KAA+14] Kamran Riaz Khan, Zaafar Ahmed, Shabbir Ahmed, Affan Syed, and Syed Ali Khayam. Rapid and scalable ISP service delivery through a programmable middlebox. *SIGCOMM Comput. Commun. Rev.*, 44(3):31–37, July 2014.

[KGW+16] Suleman Khan, Abdullah Gani, Ainuddin Wahid Abdul Wahab, Ahmed Abdelaziz, Kwangman Ko, Muhammad Khurram Khan, and Mohsen Guizani. Software-defined network forensics: Motivation, potential locations, requirements, and challenges. *IEEE Network*, 30(6):6–13, 2016.

[KHRH14] Marc Kührer, Thomas Hupperich, Christian Rossow, and Thorsten Holz. Exit from hell? reducing the impact of amplification ddos attacks. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pages 111–125, 2014.

[KK06] Aleksandar Kuzmanovic and Edward W Knightly. Low-rate tcp-targeted denial of service attacks and counter strategies. *IEEE/acm transactions on networking*, 14(4):683–696, 2006.

[KLG13] Min Suk Kang, Soo Bum Lee, and Virgil D. Gligor. The crossfire attack. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, SP '13, pages 127–141, Washington, DC, USA, 2013. IEEE Computer Society.

[Koz05] Charles M Kozierok. *The TCP/IP guide: a comprehensive, illustrated Internet protocols reference*. No Starch Press, 2005.

[KPB14] Panos Kampanakis, Harry Perros, and Tsegereda Beyene. Sdn-based solutions for moving target defense network protection. In *A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on*, pages 1–6. IEEE, 2014.

[KRV+15] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, Jan 2015.

[KT05] Jon Kleinberg and Eva Tardos. *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[LC15] Y. Li and M. Chen. Software-defined network function virtualization: A survey. *IEEE Access*, 3:2542–2553, 2015.

[LHM10] Bob Lantz, Brandon Heller, and Nick McKeown. A network in a laptop: Rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, pages 19:1–19:6, New York, NY, USA, 2010. ACM.

[Lin] Linux. Ping(8) - linux man page. `https://linux.die.net/man/8/ping`. [Online; accessed 2020-4-29].

[LKD16] Christos Liaskos, Vasileios Kotronis, and Xenofontas Dimitropoulos. A novel framework for modeling and mitigating distributed link flooding attacks. In *IEEE INFOCOM*, pages 1–9, San Francisco, CA, USA, Apr 2016.

[LKG13] Soo Bum Lee, Min Suk Kang, and Virgil D Gligor. Codef: Collaborative defense against large-scale link-flooding attacks. In *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*, pages 417–428, 2013.

[LM07] Zhi Li and Prasant Mohapatra. On investigating overlay service topologies. *Comput. Netw.*, 51(1):54–68, January 2007.

[LWC14] Yue-Bin Luo, Bao-Sheng Wang, and Gui-Lin Cai. Effectiveness of port hopping as a moving target defense. In *2014 7th International Conference on Security Technology*, pages 7–10. IEEE, 2014.

[MA07] B. McLaughlan and K. Akkaya. Coverage-based clustering of wireless sensor and actor networks. In *IEEE International Conference on Pervasive Services*, pages 45–54, July 2007.

[MAB+08] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[Man13] Pratyusa K Manadhata. Game theoretic approaches to attack surface shifting. In *Moving Target Defense II*, pages 1–13. Springer, 2013.

[Mas11] Mark Masse. *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. O'Reilly Media, Inc., 2011.

[MGT+15] Jon Matias, Jokin Garay, Nerea Toledo, Juanjo Unzilla, and Eduardo Jacob. Toward an sdn-enabled nfv architecture. *IEEE Communications Magazine*, 53(4):187–193, 2015.

[Mic] Microsoft. Pricing calculator — microsoft azure. `https://azure.microsoft.com/en-us/pricing/calculator`. [Online; accessed 2020-4-15].

[MMMZ16] Amin Mohammadi, Mohammad Hossein Manshaei, Monireh Mohebbi Moghaddam, and Quanyan Zhu. A game-theoretic analysis of deception over social networks using fake avatars. In *International Conference on Decision and Game Theory for Security*, pages 382–394. Springer, 2016.

[MMZ15] Monireh Mohebbi Moghaddam, Mohammad Hossein Manshaei, and Quanyan Zhu. To trust or not: a security signaling game between service provider and client. In *International Conference on Decision and Game Theory for Security*, pages 322–333. Springer, 2015.

[MR04] Jelena Mirkovic and Peter Reiher. A taxonomy of ddos attack and ddos defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2):39–53, 2004.

[MS15] Douglas C MacFarland and Craig A Shue. The SDN shuffle: Creating a moving-target defense using host-based software-defined networking. In *Proceedings of the Second ACM Workshop on Moving Target Defense*, pages 37–41. ACM, 2015.

[MSS19] Tyler McDaniel, Jared M Smith, and Max Schuchard. The maestro attack: Orchestrating malicious flows with bgp. *arXiv preprint arXiv:1905.07673*, 2019.

[MTL+18] Roland Meier, Petar Tsankov, Vincent Lenders, Laurent Vanbever, and Martin Vechev. Nethide: secure and practical network topology obfuscation. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 693–709, 2018.

[Net] Netscout. Network security infrastructure report — netscout. `https://www.netscout.com/report/`. [Online; accessed 2020-4-1].

[Net14] Arbour Networks. 10th annual worldwide infrastructure security report (wisr). `http://www.arbornetworks.com`, 2014. [Online; accessed 2016-9-20].

[NIT09] NITRD. National cyber leap year summit 2009, co-chairs report. *Tech. Rep., Federal Networking and Information Technology Research and Development (NITRD) Program*, pages 1–58, 2009.

[Noe88] Thomas H Noe. Capital structure and signaling game equilibria. *The Review of Financial Studies*, 1(4):331–355, 1988.

[OHBS14] H. Okhravi, T. Hobson, D. Bigelow, and W. Streilein. Finding focus in the blur of moving-target techniques. *IEEE Security Privacy*, 12(2):16–26, Mar 2014.

[Ope] Opendaylight. Home - opendaylight. `https://www.opendaylight.org/`. [Online; accessed 2020-05-14].

[PJN10a] Emmanuel S Pilli, Ramesh C Joshi, and Rajdeep Niyogi. Network forensic frameworks: Survey and research challenges. *digital investigation*, 7(1-2):14–27, 2010.

[PJN10b] Emmanuel S. Pilli, R.C. Joshi, and Rajdeep Niyogi. Network forensic frameworks: Survey and research challenges. *Digital Investigation*, 7(12):14 – 27, 2010.

[Pro] Floodlight Project. Floodlight controller. `https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/overview`. [Online; accessed 2020-5-14].

[R+99] Martin Roesch et al. Snort: Lightweight intrusion detection for networks. In *13th Large Installation System Administration Conference (Lisa)*, volume 99, pages 229–238, 1999.

[RGAS+16] Usman Rauf, Fida Gillani, Ehab Al-Shaer, Mahantesh Halappanavar, Samrat Chatterjee, and Christopher Oehmen. Formal approach for resilient reachability based on end-system route agility. In *Proceedings of the 2016 ACM Workshop on Moving Target Defense*, MTD '16, pages 117–127, New York, NY, USA, 2016. ACM.

[RKWM08] Ranga S Ramanujan, Maher N Kaddoura, Xiaoming Wu, and Kevin S Millikin. Protecting networks from access link flooding attacks, April 8 2008. US Patent 7,356,596.

[RMAS13] Mohammad Ashiqur Rahman, Mohammad Hossein Manshaei, and Ehab Al-Shaer. A game-theoretic approach for deceiving remote operating system fingerprinting. In *Conference on Communications and Network Security (CNS)*, pages 73–81. IEEE, 2013.

[Rod15] Chris Rodriquez. The expanding role of service providers in DDoS mitigation. In *Stratecast Perspectives and insight for Executives (SPIE), Vol 15, Number 10*, pages 1–10. Frost & Sullivan, 2015.

[SGT+11] A Srivastava, BB Gupta, A Tyagi, Anupama Sharma, and Anupama Mishra. A recent survey on ddos attacks and defense mechanisms. In *International Conference on Parallel Distributed Computing Technologies and Applications*, pages 570–580. Springer, 2011.

[SLB08] Yoav Shoham and Kevin Leyton-Brown. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2008.

[SLXC11] Shigen Shen, Yuanjie Li, Hongyun Xu, and Qiying Cao. Signaling game based strategy of intrusion detection in wireless sensor networks. *Computers & Mathematics with Applications*, 62(6):2404–2416, 2011.

[SMW02] Neil Spring, Ratul Mahajan, and David Wetherall. Measuring isp topologies with rocketfuel. *ACM SIGCOMM Computer Communication Review*, 32(4):133–145, 2002.

[SP09] Ahren Studer and Adrian Perrig. The coremelt attack. In *Proceedings of the 14th European Conference on Research in Computer Security*, ESORICS'09, pages 37–52, Berlin, Heidelberg, 2009. Springer-Verlag.

[Sym] Symantec. Internet security threat report. `https://docs.broadcom.com/doc/istr-24-2019-en`. [Online; accessed 2020-5-14].

[TCB17] Daniel R Thomas, Richard Clayton, and Alastair R Beresford. 1000 days of udp amplification ddos attacks. In *2017 APWG Symposium on Electronic Crime Research (eCrime)*, pages 79–84. IEEE, 2017.

[Tea] Mininet Team. Mininet: An instant virtual network on your laptop. `http://mininet.org/`. [Online; accessed 2020-5-14].

[Wes13] Baugh Wesley. Random network generator. `https://gist.github.com/bwbaugh/4602818`, 2013. [Online; accessed 2018-8-20].

[WLJW16] Lei Wang, Qing Li, Yong Jiang, and Jianping Wu. Towards mitigating link flooding attack via incremental sdn deployment. In *2016 IEEE Symposium on Computers and Communication (ISCC)*, pages 397–402. IEEE, 2016.

[WRH15] M. Wichtlhuber, R. Reinecke, and D. Hausheer. An SDN-based CDN/ISP collaboration architecture for managing high-volume flows. *IEEE Transactions on Network and Service Management*, 12(1):48–60, March 2015.

[WW16] Li Wang and Dinghao Wu. Moving target defense against network reconnaissance with software defined networking. In *International Conference on Information Security*, pages 203–217. Springer, 2016.

[WWL+18] Juan Wang, Ru Wen, Jiangqi Li, Fei Yan, Bo Zhao, and Fajiang Yu. Detecting and mitigating target link-flooding attacks using sdn. *IEEE Transactions on Dependable and Secure Computing*, 16(6):944–956, 2018.

[WXZ+17] Qian Wang, Feng Xiao, Man Zhou, Zhibo Wang, Qi Li, and Zhetao Li. Linkbait: Active link obfuscation to thwart link-flooding attacks. *arXiv preprint arXiv:1703.09521*, 2017.

[WZS02] Haining Wang, Danlu Zhang, and Kang G Shin. Detecting syn flooding attacks. In *Proceedings. Twenty-First Annual Joint Conference of the*

IEEE Computer and Communications Societies, volume 3, pages 1530–1539. IEEE, 2002.

[XLCZ14] Lei Xue, Xiapu Luo, Edmond W. W. Chan, and Xian Zhan. Towards detecting target link flooding attack. In *Proceedings of the 28th USENIX Conference on Large Installation System Administration*, LISA'14, pages 81–96, Berkeley, CA, USA, 2014. USENIX Association.

[XML+18] Lei Xue, Xiaobo Ma, Xiapu Luo, Edmond WW Chan, Tony TN Miu, and Guofei Gu. Linkscope: Toward detecting target link flooding attacks. *IEEE Transactions on Information Forensics and Security*, 13(10):2423–2438, 2018.

[YWL+18] Bo Yi, Xingwei Wang, Keqin Li, Min Huang, et al. A comprehensive survey of network function virtualization. *Computer Networks*, 133:212–262, 2018.

[YYRC08] Minlan Yu, Yung Yi, Jennifer Rexford, and Mung Chiang. Rethinking virtual network embedding: substrate support for path splitting and migration. *ACM SIGCOMM Computer Communication Review*, 38(2):17–29, 2008.

[YYWZ05] B. Yang, M. Yang, J. Wang, and S. Q. Zheng. Minimum cost paths subject to minimum vulnerability for reliable communications. In *8th International Symposium on Parallel Architectures,Algorithms and Networks (ISPAN'05)*, pages 1–6, Dec 2005.

[ZBA10] Jun Zhuang, Vicki M Bier, and Oguzhan Alagoz. Modeling secrecy and deception in a multiple-period attacker–defender signaling game. *European Journal of Operational Research*, 203(2):409–418, 2010.

[ZJT13] Saman Taghavi Zargar, James Joshi, and David Tipper. A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks. *IEEE communications surveys & tutorials*, 15(4):2046–2069, 2013.

[ZLG17] Zheng Zhao, Fenlin Liu, and Daofu Gong. An sdn-based fingerprint hopping method to prevent fingerprinting attacks. *Security and Communication Networks*, pages 1–12, 2017.

VITA

ABDULLAH AYDEGER

2013 B.S., Computer Engineering
Istanbul Technical University University (ITU)
Istanbul, Turkey

2016 M.Sc., Computer Engineering
Florida International University (FIU)
Miami, Florida

2020 Ph.D., Electrical and Computer Engineering
Florida International University (FIU)
Miami, Florida

SELECTED PUBLICATIONS AND PRESENTATIONS

- N. Saputro, S. Tonyali, A. Aydeger, K. Akkaya, M. Rahman, and S. Uluagac, "A Review of Moving Target Defense Mechanisms for Internet of Thing", Book Chapter in *Modeling and Design of Secure Internet of Things,* Wiley-IEEE Press, 2020 (To Appear).
- A. Aydeger, N. Saputro and K. Akkaya, "A Moving Target Defense and Network Forensics Framework for ISP Networks using SDN and NFV", in *Future Generation Computer Systems*, Vol. 94, pp. 496-509, 2019.
- A. Aydeger, N. Saputro, K. Akkaya and A. Uluagac, "SDN-enabled recovery for Smart Grid teleprotection applications in post-disaster scenarios", in *Journal of Network and Computer Applications*, Vol. 138, pp. 39-50, 2019.
- A. Aydeger, N. Saputro, K. Akkaya, "Utilizing NFV for Effective Moving Target Defense against Link Flooding Reconnaissance Attacks", in the *Proceedings of Military Communications (Milcom)*, pp. 946-951, Los Angeles, FL, Oct. 2018.
- A. Aydeger, N. Saputro, K. Akkaya, "Assessing the Overhead of Authentication during SDN-Enabled Restoration of Smart Grid Inter-substation Communications", in the *Proceedings of Consumer Communications & Networking Conference (CCNC)*, pp. 1-6, Las Vegas, NV, Jan. 2018.
- A. Aydeger, N. Saputro, K. Akkaya and M. Rahman, "Mitigating Crossfire Attacks using SDN-based Moving Target Defense", in the *Proceedings of IEEE Local Computer Networks (LCN'16)*, pp. 627-630, Dubai, UAE, Nov. 2016.