

DOLUS: CYBER DEFENSE USING PRETENSE AGAINST DDOS ATTACKS IN CLOUD PLATFORMS

A Thesis presented to
the Faculty of the Graduate School
at the University of Missouri

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

by
ROSHAN LAL NEUPANE
Dr. Prasad Calyam, Thesis Supervisor
DECEMBER 2017

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

DOLUS: CYBER DEFENSE USING PRETENSE
AGAINST DDOS ATTACKS IN CLOUD PLATFORMS

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

Dr. Prasad Calyam

Dr. Rohit Chadha

Dr. Sanjeev Khanna

ACKNOWLEDGMENTS

My deepest gratitude, first of all, goes to my advisor, Dr. Prasad Calyam for supporting me through out my time during my Master's and of course the research work. His thorough support and guidance has been crucial in bettering me in and out of academia. His vision and ideas are truly motivational and his passion in the field of Cloud Computing played a pivotal role in dragging me into the field and I couldn't be more thankful. I would also like to thank my friends and team members, Nishant Chettri, Travis Neely and Mark Vassell, who showed great passion in this work. Their skills, dedication and assiduousness, I would say, play the most crucial part in completion of this work. Thank you all, without you, this work would have been far from complete. I would also like to thank my committee members, Dr. Rohit Chadha and Dr. Sanjeev Khanna for being a part of this thesis.

I am thankful to a great friend, Bidyut Mukherjee, who was and still is involved in helping and encouraging me in my works. Special thanks to Sai Shreya Nuguri for all her help. In addition, I would like to thank all the VIMAN lab members who helped me throughout my time at the lab. Thanks for always being there for me. I am grateful to all the professors who have shared their knowledge with me to keep up with all the advancements in technology and the field of computer science.

Finally, I would like to express my profound gratitude to my parents and to my little brother for being the best i could ever have and for providing continuous support and encouragement. I love you all.

Thank you all who have directly or indirectly been a part of this.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	ii
LIST OF TABLES	v
LIST OF FIGURES	vi
ABSTRACT	viii
CHAPTER	
1 Introduction	1
1.1 DDoS Attacks in Cloud Platforms	1
1.2 Need for Dolus (Defense using Pretense) System	3
1.3 Software Defined Network programmability and Frenetic	4
1.4 Thesis Outline	5
2 Background and Literature Survey	6
2.1 Flooding Attacks	6
2.2 Location-based SDN-enabled Defense	7
2.2.1 Network-based Mechanisms	7
2.2.2 Destination-based Mechanisms	8
2.2.3 Source-based Mechanisms	8
2.3 Defense through Trickery	8
2.4 Literature Review	9
2.4.1 Moving Target Defense	9
2.4.2 SDN for DDoS Survey	10
2.4.3 Software Defined Internet Exchange (SDX)	12
2.4.4 Pretense Theory	13

3	Dolus Defense Methodology	14
3.1	Dolus System Overview	14
3.2	Attack Model	16
3.3	Establishment of dummy-traffic based False Reality	17
3.4	Defense by Pretense Scheme	20
3.4.1	Attack Detection	21
3.4.2	Quarantine Setup	24
3.4.3	Pretense Initiation	25
3.4.4	Pretense Maintenance	25
3.4.5	Policy Decision Making	26
3.4.6	Threat Intelligence Sharing	27
4	Dolus System Testbed and Evaluation	28
4.1	Experiment Setup	29
4.1.1	Testbed Setup	29
4.1.2	Consumer UI	30
4.1.3	Administrative UI	31
4.2	Attack Detection and Classification	33
4.3	False reality establishment results	39
4.4	Time to restore Cloud-hosted Application Service	43
4.5	Amount of Traffic Processed at the Root Switch	44
5	Summary and concluding remarks	46
6	Future Work	48
	BIBLIOGRAPHY	49

LIST OF TABLES

Table	Page
2.1 DDoS attack types and examples	6
4.1 Overall Attack Detection and Classification Time and Accuracy	37

LIST OF FIGURES

Figure		Page
1.1	Akamai DDoS Attack Frequency By Industry Q2 2017-Q1 2017	2
2.1	MTD based VM migration technique against DDoS attacks	10
2.2	Classification of defense mechanisms against DDoS attacks using SDN . .	11
3.1	Illustration of the proposed Dolus system scheme wherein the attacker is <i>tricked</i> by redirection of the attack traffic to a quarantine VM for pretense initiation, while the providers work collaboratively to block the attack traf- fic closer to the source side.	16
3.2	Cross-domain physical setup in a Dolus system deployment to share threat intelligence for a unified controller to coordinate policy management with a federation of ASes to block attack traffic closer to the source side.	17
3.3	Dolus Scheme Flow Diagram	21
3.4	Sequence diagram of the Dolus system interactions for attack detection, quarantine setup, pretense initiation/maintenance and DDoS attack impact mitigation.	24
3.5	Scapy snippet setting up spoofed source IP address	25
4.1	GENI Cloud testbed setup used to evaluate the Dolus system performance. .	29
4.2	Video gaming portal application running in a SDxI-based cloud platform with cross-domain network collaboration.	30

4.3	Administrator User Interface of an Dolus system instance.	31
4.4	Administrator User Interface : Quarantined VMs	31
4.5	Administrator User Interface : Bandwidth utilization when access to root switch is disallowed	32
4.6	Administrator User Interface : Server History	32
4.7	Administrator User Interface : Attack History	33
4.8	Two stages transition diagram	34
4.9	Confusion matrices for attack detection and classification for multiple traf- fic flows sent to a single server.	35
4.10	Confusion matrices for attack detection and classification for multiple traf- fic flows sent to multiple hosts.	36
4.11	Confusion matrices for outlier detection and classification for multiple traf- fic flows comprising of familiar attack flows.	38
4.12	Performance comparison of reactive migration strategies with-and-without false reality: Average response time	39
4.13	Performance comparison of reactive migration strategies with-and-without false reality: Percentage of dropped packets	40
4.14	Comparison of successful identification of dummy traffic by the attacker . .	41
4.15	Benefits and costs of false reality in terms of average CPU utilization	42
4.16	Comparison of the <i>cloud service restoration time</i> metric with cases of: no Defense, with MTD and with Dolus.	43
4.17	Traffic processed (in Bytes) in one of the slave switches.	44
4.18	Traffic Processed at the root switch only shows user traffic proving that the attack traffic is redirected to quarantine VM.	45

ABSTRACT

Cloud-hosted services are being increasingly used in online businesses in e.g., retail, healthcare, manufacturing, entertainment due to benefits such as scalability and reliability. These benefits are fueled by innovations in orchestration of cloud platforms that make them totally programmable as Software Defined everything Infrastructures (SDxI). At the same time, sophisticated targeted attacks such as Distributed Denial-of-Service (DDoS) are growing on an unprecedented scale threatening the availability of online businesses. In this thesis, we present a novel defense system called Dolus to mitigate the impact of DDoS attacks launched against high-value services hosted in SDxI-based cloud platforms. Our Dolus system is able to initiate a pretense in a scalable and collaborative manner to deter the attacker based on threat intelligence obtained from attack feature analysis in a two-stage ensemble learning scheme.

Using foundations from pretense theory in child play, Dolus takes advantage of elastic capacity provisioning via quarantine virtual machines and SDxI policy co-ordination across multiple network domains. To maintain the pretense of false sense of success after attack identification, Dolus uses two strategies: (i) dummy traffic pressure in a quarantine to mimic target response time profiles that were present before legitimate users were migrated away, and (ii) Scapy-based packet manipulation to generate responses with spoofed IP addresses of the original target before the attack traffic started being quarantined. From the time gained through pretense initiation, Dolus enables cloud service providers to decide on a variety of policies to mitigate the attack impact, without disrupting the cloud services experience for legitimate users. We evaluate the efficacy of Dolus using a GENI Cloud testbed and demonstrate its real-time capabilities to: (a) detect DDoS attacks and redirect attack traffic to quarantine resources to engage the attacker under pretense, and (b) coordinate SDxI policies to possibly block DDoS attacks closer to the attack source(s).

Chapter 1

Introduction

1.1 DDoS Attacks in Cloud Platforms

Cloud computing has become an essential aspect of online services available to customers in major consumer fields such as e.g., retail, manufacturing, and entertainment. On-demand elasticity, and other benefits including diversity of resources, reliability and cost flexibility have led enterprises to pursue the development and operations of their applications in a “cloud-first” fashion [1]. In addition to above consumer fields, with the growing trend of hosting critical applications in e.g., finance, biotechnology, and healthcare on cloud platforms, there is a need to protect these applications from the security threats of cyber attacks. Sophisticated targeted attacks such as are growing on an unprecedented scale, threatening the availability of online businesses. Technological trends indicate that the aforementioned benefits typically rely on software-centric innovations in the orchestration of cloud resources. These innovations include cloud platforms based on Software Defined everything Infrastructures (SDxI) that allow programmability to achieve capabilities such as speed and agility [2] in elastic capacity provisioning. Additionally, they provide opportunities to create Software-Defined Internet Exchange Points (SDXs) between multi-

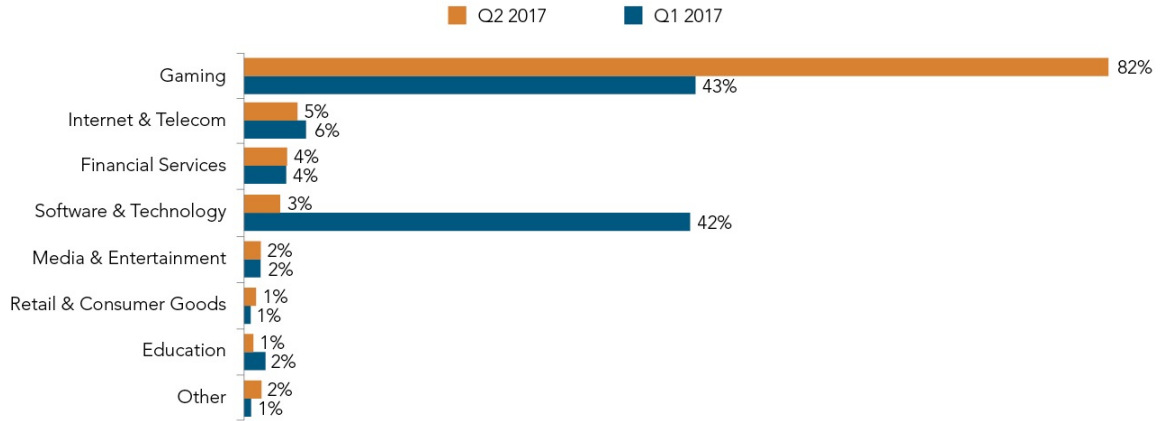


Figure 1.1: Akamai DDoS Attack Frequency By Industry Q2 2017-Q1 2017

ple Software-Defined Network (SDN) domains (or Autonomous Systems (ASes)) that can enable application-specific peering, knowledge sharing of cyber threats, and other cross-domain collaborations [3]. One of the most common type of cyber attacks targeting cloud platforms is the Distributed Denial of Service (DDoS) attack [4] that leads to Loss of Availability (LOA) through starvation of critical application resources serving legitimate users. Figure 1.1 depicts the Akamai Q2 2017 state of the internet report on DDoS attacks. The security report [5] reports that records on DDoS attacks have increased once again with more number of attacks recorded from last year.

The DDoS attack defense challenges within a cloud infrastructure are more severe than traditional cyber security risks in two ways. Firstly, a cloud infrastructure becomes a ‘vulnerability amplifier’ to traditional cyber security threats due to the highly elastic nature of the infrastructure resources designed to serve a large population of consumers. Secondly, new means of DDoS attacks exist that specifically target cloud infrastructures in vulnerable areas of application multi-tenancy within a virtual machine (VM), and within an internal network of a CSP. Moreover, traditional ‘detect-and-react’ defense approaches [6, 7] are largely ineffective in consistently maintaining the Service Level Agreements (SLA) when under DDoS attack due to their lack of: (a) agility in response to attack detection, (b) cost effectiveness for the CSP, and (c) sophistication to tackle intelligent attack strategies.

Consequently, as an alternative, the cloud security community and even federal organizations [8] are exploring ‘Cyber Agility and Defensive Maneuver’ (CAADM) mechanisms that can allow for real-time service restoration through agile cloud resource adaptations once an attack is detected. The same mechanisms can also limit proliferation of detected attacks within the cloud infrastructure through preventive VM resource maneuvers.

1.2 Need for Dolus (Defense using Pretense) System

Given the benefits of SDxI-based cloud platforms, the traditional Intrusion Prevention Systems (IPS) and Intrusion Detection Systems (IDS) solutions are undergoing major transformations. Recently, defense strategies such as SDN-based “moving target defense” [9] [10] have been proposed to protect networks and users against DDoS attacks by migrating networks and users from targeted virtual machines (VMs) to other healthy/safe VMs in a cloud platform. However, such strategies may cause the application response behavior to change to an extent that alerts the attacker that a high-value target has been hit. Given such a discovery that a service provider is moving a target in order to shelter from the attack impact, the attacker may then deflect more resources to seek ransom demands in order to stop the DDoS on the target.

Moreover, if the DDoS attack flows are blacklisted, traditional approaches allow defense only at the attack destination side i.e., any related traffic is dropped at the target-end. In such cases, the attacker still can escalate the DDoS attacks by crossing many other neighboring domain paths, who may not be inclined to drop the attack flow traffic assuming it may be legitimate traffic of a peer network. We suppose that SDxI-based cloud platforms can facilitate capabilities for coordination of policies and creation of incentives to block such targeted attack flows closer to the attack source side, which can then mitigate the impact on resource flooding for all the providers involved. However, this might require the target service provider to buy some time in order to bring ‘humans into the loop’ to actually

enforce attack traffic blocking measures closer to the attack source side.

In this thesis, we address the above challenges and present a novel defense system called *Dolus* (named after the spirit of trickery in Greek Mythology) to mitigate the impact of DDoS attacks launched against high-value services hosted in SDxI-based cloud platforms. The DDoS attack detection is performed in the Dolus system using the threat intelligence obtained from attack feature analysis in a two-stage ensemble learning scheme that we developed. The first stage focuses on anomaly detection to identify salient events of interest (e.g., connection exhaustion), and the second stage is invoked to distinguish the DDoS attack event type amongst the 5 common attack vectors: DNS (Domain Name System), UDP (User Datagram Protocol) fragmentation, NTP (Network Time Protocol), SYN (short for synchronize), SSDP (simple service discovery protocol). Our Dolus system is novel owing to a scalable and collaborative defense strategy that uses foundations from *pretense theory in child play* [11] [12] along with SDxI-based cloud platform capabilities for: (a) elastic capacity provisioning via ‘quarantine VMs’, and (b) SDxI policy co-ordination across multiple network domains. Such a strategy is aimed at preventing the disruption of cloud-hosted services by deceiving the attacker through creation of a false sense of success, and by keeping the attacker from recognizing that a high-value target has been impacted and is being moved.

1.3 Software Defined Network programmability and Fre-netic

Software Defined Networking (SDN) is an emerging technology which allows programmability in the network through decoupling of data plane and control plane. [13] provides a novel work by surveying capabilities of SDN programming with focus in defense mechanisms against DDoS attacks. SDN capabilities such as software-based traffic analysis, global viewing of network, centralized control, dynamic updating of forwarding rules make

it easy to detect and to react DDoS attacks in cloud platforms.

The programming language we use for our logical centralized controller is Frenetic [14]. Frenetic is a family of network programming languages which allows for programmable networks using Python and NetKat [15] (also developed by the creators of Frenetic to help in specifying OpenFlow policies). This allows one to write scripts in Python which can be run against the packets identified for investigation. These scripts will allow to determine where the current and future packets should go. Once a decision is made about where a packet goes, a network device can learn this decision and continue to forward those packets according to the rule without need of future investigation. Our goal with using Frenetic is to be able to quickly identify an attackers packets and teach each switch to send that traffic elsewhere. Frenetic can also run an HTTP Listener with which one can make RESTful web service calls. We use this functionality in order to GET data about the network as well as POST policy updates into Frenetic which then sends the policies downstream to the switches.

1.4 Thesis Outline

The remainder of this thesis is organized as follows: In Chapter 2, we describe the thesis background and literature review, that provide context to the solution approach. In Chapter 3, we elaborate on our solution and provide a detailed description of our approach with an overview and reference architecture. Chapter 4 evaluates the effectiveness of our system and show results of outlier detection and classification of our two-stage ensemble method. Chapter 5 discusses future work and provides information on extending the system for different targeted attacks. Finally, Chapter 6 concludes the thesis.

Chapter 2

Background and Literature Survey

In this chapter, we provide some background information on fundamental concepts behind the system. We then discuss the various literature work that have led to the idea and implementation of this research.

2.1 Flooding Attacks

There are various types of flooding attacks that have been utilized to trigger Loss of Availability (LOA) in web servers over the years. These attacks are mainly divided into types: a) Volume based or Volumetric DDoS attacks, b) Protocol attacks and c) Application Layer attacks. Table 2.1 list DDoS types and a few example of attacks that fall under those types.

Table 2.1: DDoS attack types and examples

DDoS types	Example
Volume based/ volumetric DDoS attacks	ICMP, UDP and spoofed-packet floods
Protocol Attacks	SYN floods, Ping of Death
Application Layer Attacks	GET/POST floods, web server vulnerabilities

Defense against flooding attacks such as DDoS typically involves attack traffic feature learning that provides intelligence on where the attack is coming from, and the specific attack type(s) [16] [17] [18]. Analysis of features such as source IP, destination IP, source port, destination port, size of packets, packet identifiers commonly help in subsequent filtering of flooding attacks. Authors in [19] show that the Internet traffic patterns are distinguishable, which can help filter and isolate attack traffic flows. Once attack flows are filtered, blacklists are created [20], which can then be used to “scrub” the flows through scrubbing SaaS services as a low-cost solution [21].

2.2 Location-based SDN-enabled Defense

Defense against flooding attacks have been utilized focusing on various locations. The research works using SDN for defense systems include at the Network, the source side and the destination side of the attack. They have been categorized and covered as below.

2.2.1 Network-based Mechanisms

DDoS defense strategies are typically handled in the network. A number of such in-network defense involves analysis of traffic and dynamic updation of rules to effectively reroute it. Such efforts include Choi *et al.* [22], which proposes a novel architecture that reacts to targeted attacks using accountability and content-aware supervision. Similarly, using volume counting, [23] provides a DDoS mechanism to monitor the traffic flow in OpenFlow switches. In the context of programmability of SDN switches to mitigate targeted attacks, Shin *et al.* [24] discusses a novel programming framework to that end. Yu *et al.* [25] proposes a memory-efficient system that uses Bloom filter and installs a monitoring tools into switch’s data plane. The work specifically focuses on the programmability and dynamic rule update characteristics of SDN to mitigate targeted attacks. [26] shows how a moni-

toring node that communicates with the controller can be used to build a global view of the network to monitor information from every user in the network. Idziorek *et al.* takes a traditional approach of intrusion prevention to prevent DDoS attacks [27].

2.2.2 Destination-based Mechanisms

Leveraging the dynamic rule update feature of SDN, Tian *et al.* [28] analyzes the probability that a flow is traced back across multiple Autonomous System (AS) hops by sampling the probability and the signature of the attack traffic. NetSight captures packet histories to investigate events of interest to trace the network state [29].

2.2.3 Source-based Mechanisms

Yan *et al.* [13] presents a survey of SDN-based mechanisms to detect attacks closer to the attackers/attack sources. Mehdi *et al.* [30] shows how to perform real-time traffic analysis on OpenFlow switches to detect mobile malware. Similarly, [31] uses SDN features such as dynamic update of rules and global view from centralized controller to effectively tackle the problem of source address validation.

2.3 Defense through Trickery

Clark *et al.*, in [32], shows the effectiveness of randomizing IP addresses in decoy-based MTD and tries to introduce the notion of tricking the attackers through IP randomization methods. However, our approach adds to [32] by preventing potential attack outcomes. Our notion of pretense is akin to Honeypots and Honeynets which are effective in gaining information about possible attacks based on minimal active interactions with attackers [33]. Primarily they are used in a setting to either gain more information about potential attacks or the behavior of attackers. Our work is complementary to Honeypots/Honeynets: we

employ pretense to deceive attackers by rerouting and responding to attack traffic using a QVM. Our pretense theory is more sophisticated where the foundation lies on the child psychology where we reach the assumption stimulus match [11].

Our work is looking at defense against flooding attacks which is mostly addressed by finding some attack feature learning that determines where the attack originates from and its type. Secondly, we use blacklisting and information gathering to block the attack traffic and to find out the flow and its origin. Thirdly, since we have cloud resources, we use elastic capacity provisioning, similar to MTD. We initiate pretense to keep the attacker guessing in a way that he doesn't figure out the trickery and chase back. We use some time before the attacker figures out the pretense, to coordinate across the unified SDxI infrastructure that could potentially help us block the attacker closer to the source. We instantiate this framework to collectively stop the attacker in collaborative way utilizing the scaling capability of a cloud, that altogether helps us get them closer to the source.

2.4 Literature Review

In this section, we provide reviews of work that motivated in building our research work on DDoS defense mechanism that is Dolus.

2.4.1 Moving Target Defense

Moving Target Defense (MTD) based resource obfuscation/adaptation mechanisms can be effective to protect critical cloud-hosted applications. For instance, MTD-based mechanisms can be used to perform both: (i) *proactive* resource adaptation, to detect a DDoS attack and act defensively before major damage is inflicted, and (ii) *reactive* resource adaptation, to act defensively after an attack has occurred. At the same time, MTD-based mechanisms are amenable to leverage the emerging SDN paradigm to achieve dynamic network

resource management. Work in [34] discuss a reactive and proactive defense solution against DDoS attack by utilizing cloud's resource adaptation and elastic capacity provisioning. The figure 2.1 show their system architecture that depicts VM migration when an attack occurs, leveraging SDN capabilities.

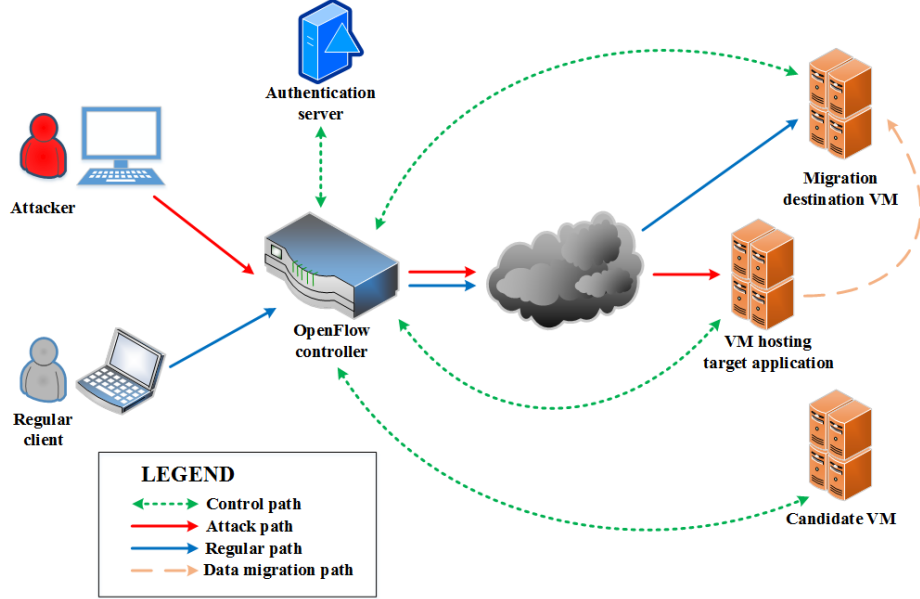


Figure 2.1: MTD based VM migration technique against DDoS attacks

Though this work presents both before attack and after attack methods to defend against DDoS attacks, they are done at the destination side i.e., at the cloud platform. Our work focus on idea of sharing knowledge across multiple domains for better chances of detecting attacks at the origin of attack. We use proactive mechanism in providing resource adaptation through elastic capacity provisioning in our False Reality establishment which is explained later in section 3.3.

2.4.2 SDN for DDoS Survey

SDN shows great promise in defending or responding to DDoS attacks. Open Networking Foundation (ONF) [35] is an organization dedicated to the development and commercialization of SDNs. SDNs are widely popular in defeating DDoS attacks because of their

good features in maintaining a secured network against different cyber attacks. Current interesting emerging technology can help reduce DDoS attacks significantly. Figure 2.2 shows classification of defense mechanisms using SDN. The survey in this novel work is informative and give ideas of different previous research works classifying the works based on source, destination or at the network.

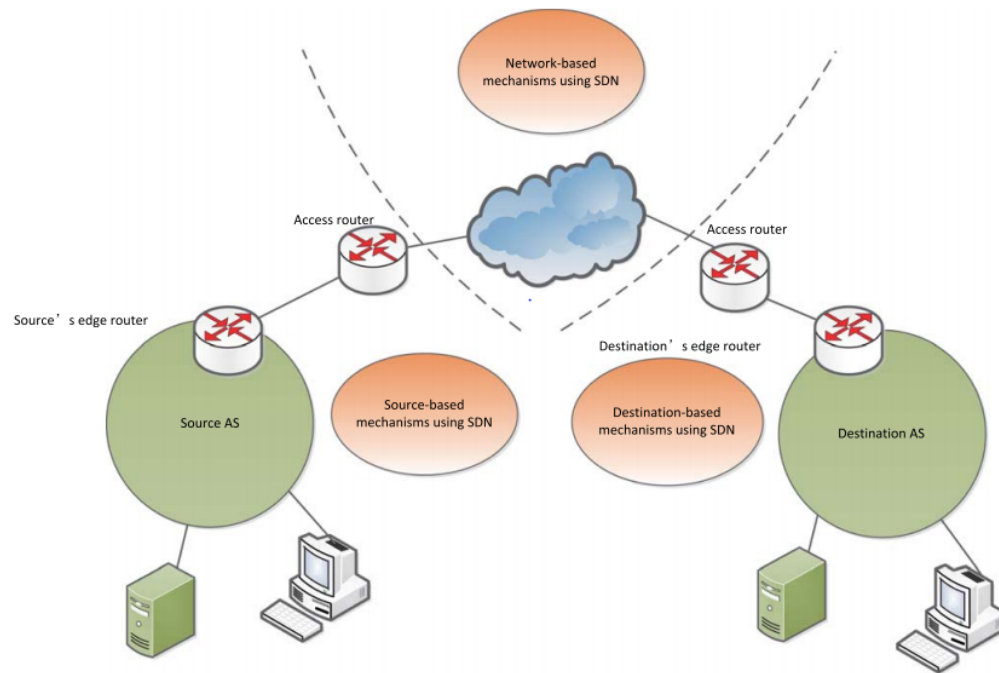


Figure 2.2: Classification of defense mechanisms against DDoS attacks using SDN

Compiled from [13], SDN provides

- Separation of control and data plane: helps in experimenting attacks and their detection which when successful, can be made operational. Also provides programmable networks
- Centralized controller: has global view of the network and permits dynamic quarantine of compromised hosts. SDX controller provide added ability to coordinate between one another across different Autonomous Systems (ASes) and IXPs
- Programmability through external applications: network can be programmed using

intelligent IDSes

- Software based traffic analysis: traffic can be analyzed to help switch improve its capabilities through the analysis
- Dynamic updating and forwarding rules

We utilize this survey of SDN capabilities in mitigation of DDoS attacks in cloud platforms and take it a step ahead to provide a global solution to DDoS defense. The following section gives a briefly discuss an emerging technology based on SDN. Authors in the research propose SDN in Internet Exchange Points (IXPs).

2.4.3 Software Defined Internet Exchange (SDX)

SDNs that are enabled at the Internet Exchange Points (IXP) [36] are called SDX. Key points that we gather from the paper to utilize in our work are: a) Knowledge sharing of cyber threats and b) cross-domain collaboration. DDoS attacks can be prevented from entering cloud platforms that are hosting the web services. For a successful mitigation, it is important that the attack is stopped near to its source rather than at the target destination. This can be obtained if there is proper knowledge sharing of cyber threats across multiple domains, or Autonomous Systems in our case, to detect the attack at source of attack origination.

In traditional approaches, if the DDoS attack flows are blacklisted, they are only applied at the attack destination side i.e., any related traffic is dropped at the target end. In such cases, the attacker still can escalate the DDoS attacks by crossing many other neighboring domain paths, who may not be inclined to drop the attack flow traffic assuming it may be legitimate traffic of a peer network. We suppose that SDxI-based cloud platforms, similar to our case, can facilitate capabilities for coordination of policies and creation of incentives to block such targeted attack flows closer to the attack source side, which can then mitigate the impact on resource flooding for all the providers involved. However, this might require

the target service provider to buy some time in order to bring ‘humans into the loop’ to actually enforce attack traffic blocking measures closer to the attack source side.

2.4.4 Pretense Theory

There have been efforts that seek to implement defense mechanisms using some form of ‘trickery’ to engage an attacker as discussed earlier in section 2.3. Dolus system’s pretense theory is mainly built upon the work in [11] and [12] belonging to the field of child pretend play psychology. Our novel *defense by pretense* mechanism for effective mitigation of DDoS attacks is inspired by the authors’ experiments where they show children (analogous to our attackers) various pictures of the animals along with a mismatch of the sounds made by the associated animals. Observations are made on how a pretense is effective based on how long it takes for a child to understand/protest that the information portrayed is actually false. In our case, the longer an attacker is tricked by our pretense, the more time a cloud service provider has to perform MTD mechanisms, strategize on patching identified vulnerabilities, as well as implement a SDxI-based infrastructure policy coordination for mitigation of the impact of a DDoS attack.

Chapter 3

Dolus Defense Methodology

In this chapter, we first present an overview of our proposed Dolus system. Following this, we describe the attack model that we assume to design our defense. Lastly, we detail our defense solution that uses a ‘defense by pretense’ scheme.

3.1 Dolus System Overview

The pretense in the Dolus system is designed to create stimulus from the target side that matches the initial expectation of an attacker that a high-value target has not yet been compromised through an automated bot activity. Pretense theory concepts from [11] motivate us to address the issue of how a cognitive agent can present a pretense world, which is different from the real world using the following four steps:

- (a) The basic assumption(s) or premise(s) that is used by a pretender on *what* is being pretended.
- (b) Inferential elaboration which details of what goes into or what actually happens in the process of pretense.

- (c) Appropriate behavior production which answers the question of whether the pretender was successful on the audience being tricked.
- (d) Balancing and steering the effects of pretense.

For use cases to guide our design, we borrow ideas from an example experiment from [12], where a child (i.e., the attacker in our case) is shown the image of a dog that makes the sound of a duck. In this situation, the child protests saying that it is not the sound that a dog makes. However, if the same child is shown an image that seemingly looks like a duck (in reality, it is not) and makes the sound of a duck, then there is no protest and the child falls for the pretense. However, given additional observation time, the child realizes he/she has been tricked and protests. Thus, we can see that an effective pretense in our case can be designed as shown in Figure 3.1 by creating pertinent stimulus from the target side i.e., redirecting attack traffic to a quarantine VM that mimics original target behavior, when our two-stage ensemble learning algorithm can blacklist the attacker flows from benign user flows. This in turn could help in keeping an attacker distracted for a brief period of time when the pretense is in effect.

From the time gained through such a pretense initiation, Dolus enables cloud service providers to decide on a variety of policies by dynamically generating network policies using Frenetic [14] to mitigate the attack impact, without disrupting the cloud services experience for legitimate users. In the worst case, destination-side blocking can be enforced. Alternately, if the cloud service provider uses the attack intelligence information and successful pretense time to coordinate the ‘humans in the loop’ of neighboring SDN-enabled domains, together they can direct a unified SDN controller that directs SDN-enabled switches to actually enforce attack traffic blocking measures closer to the attack source side.

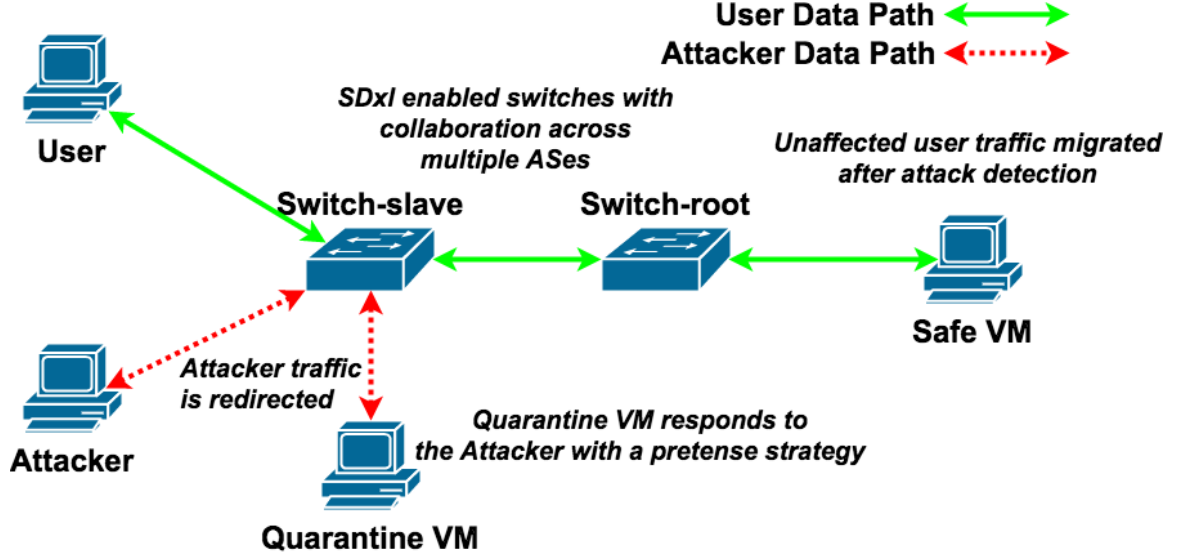


Figure 3.1: Illustration of the proposed Dolus system scheme wherein the attacker is *tricked* by redirection of the attack traffic to a quarantine VM for pretense initiation, while the providers work collaboratively to block the attack traffic closer to the source side.

3.2 Attack Model

DDoS attacks aim to overwhelm network-accessible devices such as networks, firewalls and end-systems in enterprises by sending packets at excessively high rates. With cloud-hosted applications with large monetary value becoming highly common, DDoS attacks can cause ‘Loss of Availability’ for users/customers and can be used for extortion from vulnerable online businesses. Common DDoS attack event types are amongst the 5 common attack vectors: DNS (Domain Name System), UDP (User Datagram Protocol) fragmentation, NTP (Network Time Protocol), SYN (short for synchronize), SSDP (simple service discovery protocol). For the purposes of our work, we assume the DDoS attacker uses SYN [37] and ICMP/Ping [38] flooding. Such attacks typically inundate a networks’ resources with Echo Request packets. We also assume that the attackers’ traffic is sent constantly and may or may not solicit a response in return. Such attacks can bring the network to a standstill due to the high volume of both incoming and outgoing traffic. To effectively capture the semantics of this attack model and to exhaust the target application services, we generate and emit synthetic ping and HTTP traffic using `hping3` [39]

and SlowHTTPTest [40] tools, respectively. Furthermore, to capture the dynamics of an attacker, we randomly change the number of attack packets emitted by these tools.

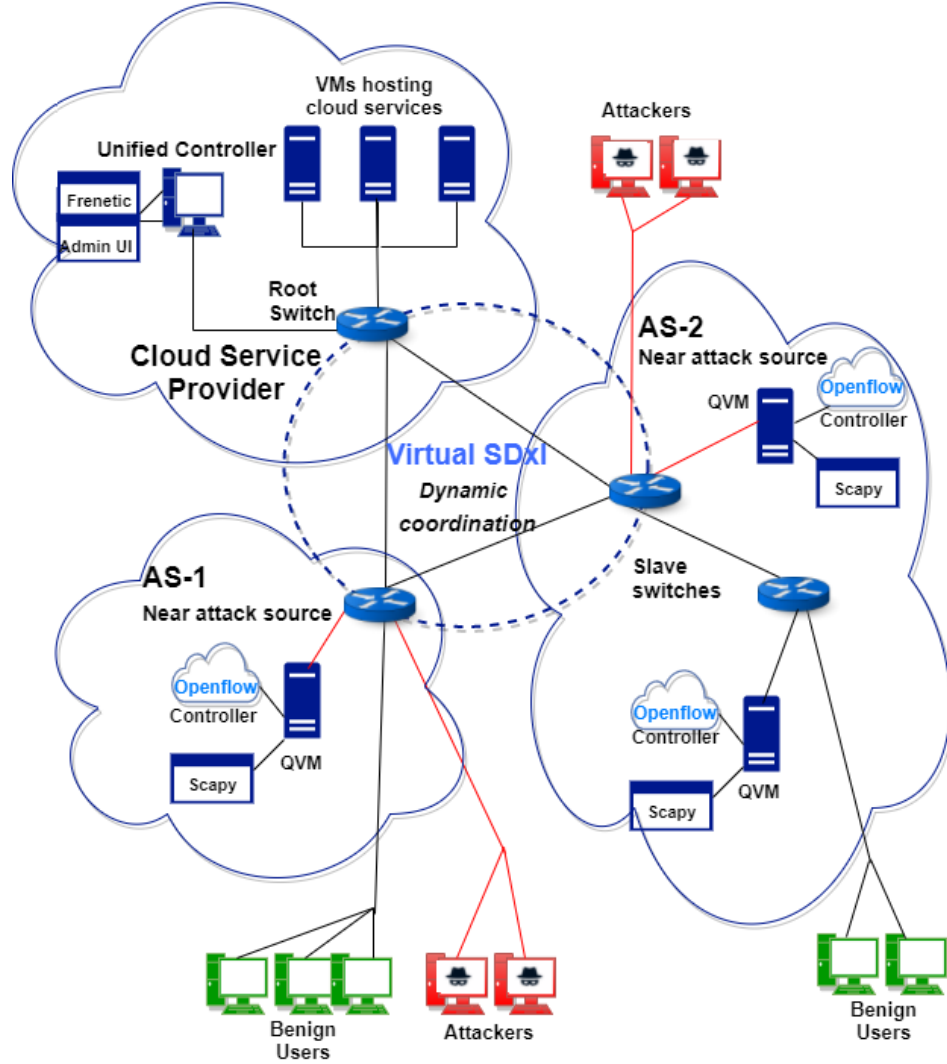


Figure 3.2: Cross-domain physical setup in a Dolus system deployment to share threat intelligence for a unified controller to coordinate policy management with a federation of ASes to block attack traffic closer to the source side.

3.3 Establishment of dummy-traffic based False Reality

As part of our reactive defense mechanism, we devise a dummy-traffic based *false reality* environment to trick attackers into thinking their attack is still in progress, while being

quarantined for monitoring and logging. As part of this pretense environment, a dedicated VM is used to create and send dummy user packets to the attacked server. This creates a false perception to the attacker that users are continuing to connect to the attacked server while in actuality the application has already been migrated to a new server, and all legitimate users properly redirected to it. As soon as the attack is detected, migration and *false reality* get triggered simultaneously so as to: (a) prevent attackers from recognizing attack failure, identification of a high-value target and retrying with greater resources, (b) sustain affected services for longer time to collect data to analyze the signature and pattern of attackers, to be prepared for future attacks with minimal increase in the overall CSP cost.

Statistically, an attacker can differentiate between dummy VM and a real VM hosting an application for multiple users by probing message response times and percentage of dropped packets. In [41], the authors have proved that if the mean response times for the dummy and real VMs are equal to $\frac{1}{rt_d}$, and $\frac{1}{rt_r}$, then the number of responses \mathcal{K} needed for the attacker to distinguish between dummy and real VMs within time T is given as:

$$\mathcal{K} = \arg \min_k \left\| \left(\frac{rt_r}{rt_d} \right)^k e^{-(rt_r - rt_d)T} > C \right. \quad (3.1)$$

where C is a parameter dependent on the rate of attacker probes and in turn on the attacker's *attack budget*.

Thus, in order for an attacker to quickly distinguish between decoy and real VMs, the attacker has to either: (a) increase the *attack budget*, or (b) rely on the difference between the response times from dummy and real VMs $\frac{1}{rt_d}$ and $\frac{1}{rt_r}$, respectively to be big enough. In other words, greater the difference, higher are the chances of dummy/real VM identification. However, for the first option, increasing the *attack budget* would mean a greater chance of getting detected for the attacker. Thus, in most cases attackers would rely on their ability to precisely differentiate between the response times. Therefore, from the perspective of a CSP, if the false reality can minimize the difference between the dummy and

real VM response times, the probability of the attacker detecting a dummy VM within finite amount of time will decrease. Thus, in our MTD with false reality implementation, we will use dummy VMs to generate just the adequate amount of dummy traffic to the VM under attack (after application migration and user redirection) that mimics the realistic traffic pattern of regular users. Such a dummy traffic will ensure that when the attacker returns to intermittent probing periods between prolonged flooding periods, the attacker does not experience any noticeable difference between the behaviors of the VM, with/without the regular users. For simplicity of analysis, we will assume the scenario of a single attacker for analysis, however our propositions will also be valid for multiple attacker scenarios.

Another important consideration while establishing a *false reality* environment is its cost-benefit analysis. Although the benefit of keeping an attacker entrapped and efforts to minimize the chances of future attacks is well motivated, we make a simplistic approach below to quantitatively analyze the cost and benefits of implementing our proposed *false reality* environment. The overall cost of *false reality* implementation (\mathcal{C}_{FR}) to the CSP through dummy VM installation and dummy traffic generation is essentially the cost of CPU utilization (C_{FR}^{C}) of the VM and the network cost (C_{FR}^{N}) for dummy traffic generation that can be expressed as:

$$\mathcal{C}_{\text{FR}} = C_{\text{FR}}^{\text{C}} + C_{\text{FR}}^{\text{N}} \quad (3.2)$$

On the other hand, qualitative benefits of *false reality* are the fact that the regular users experience no or little service interruption boosting the CSP revenue, and continued collection of attack statistics from the VM under attack for more efficient proactive migration strategy design. However, in order to quantify the benefits, we take the approach of measuring the “lost opportunity cost” of *false reality*, i.e., the amount of cloud resource (network and compute) saved by preventing attacker to migrate with the target application and relaunch an attack on the new VM, thereby jeopardizing new resources. Such resource saving is

in turn equal to the cost of a DDoS attack in terms of cloud resource wastage ($C_{\text{DDoS}}^{\text{C}}$ for compute resource and $dC_{\text{DDoS}}^{\text{N}}$ for network resource). Thus, if we ignore the benefits of attack statistics collection which is beyond the scope of this work, then the overall benefits of *false reality* in terms of “lost opportunity cost” is equal to or greater than the cost of DDoS attack, and can be expressed as:

$$\mathcal{B}_{\text{FR}} > (C_{\text{DDoS}}^{\text{C}} + C_{\text{DDoS}}^{\text{N}}) \quad (3.3)$$

Therefore, if we compare the costs and benefits of *false reality* from Equations (3.2) and (3.3) respectively, implementing *false reality* environment will only be cost effective or optimal, if the CSP defense mechanism obtained from the ‘Attack Profiler’ satisfies the following conditions:

$$C_{\text{DDoS}}^{\text{C}} > C_{\text{FR}}^{\text{C}} \quad \& \quad C_{\text{DDoS}}^{\text{N}} > C_{\text{FR}}^{\text{N}} \quad (3.4)$$

3.4 Defense by Pretense Scheme

Figure 3.2 depicts the cross-domain setup in a Dolus system deployment to implement a defense by pretense scheme. To complement Figure 3.2, interactions between different phases of a Dolus system configured for spoofing pretense are shown in Figure 3.4 and Algorithm 1, respectively.

Dolus flow diagram is depicted in figure 3.3 and lists different phases of Dolus system. This brief representation of how the flow of traffic happens in the network includes three stages.

- **Detection:** Our two-stage learning scheme analyzes network traffic to detect and differentiate benign user traffic from the attack traffic.
- **Control:** Frenetic based controller decides the rerouting of attack traffic to the Quar-

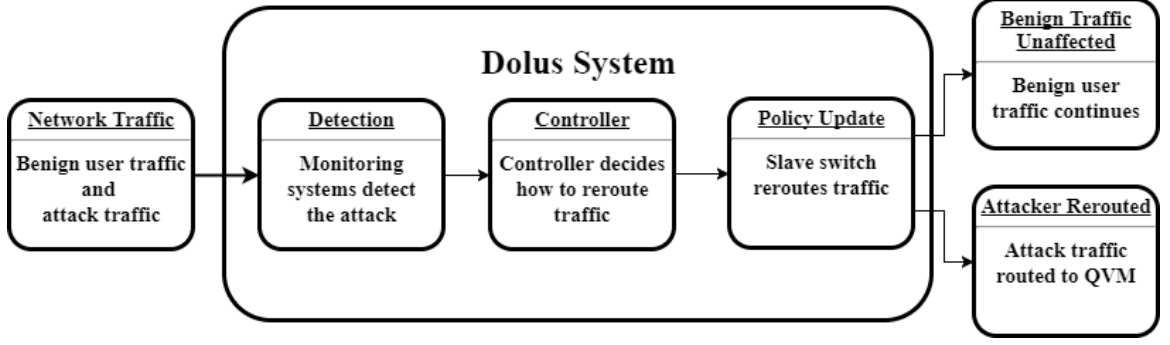


Figure 3.3: Dolus Scheme Flow Diagram

antine VMs.

- Policy Updating: Through JSON arrays, administrative UI shares knowledge of attack to possibly block the attack closer to the source of attack i.e., at slave switch closest to the attack origin.

3.4.1 Attack Detection

First, traffic within a cloud provider’s network (which is generated by the SDN switches) or across multiple transit provider ASes (which are composed of SDX plus SDN switch substrates) is monitored using a Frenetic runtime [14]-enabled monitoring subcomponent (line 24 of Algorithm 1). Next, in order to learn and classify the attacks (line 25 of Algorithm 1), we employ a two-stage ensemble learning scheme on the incoming traffic, both from the attackers and from the benign users. In order to differentiate attackers from benign users, the first stage handles outlier detection to identify salient events of interest (e.g., connection exhaustion), whereas the second stage handles outlier classification to distinguish different event types (e.g., DDoS attack).

Outlier Detection

We use basic/static methods such as multivariate Gaussian to detect outliers and build upon our prior work on detecting network-wide correlated anomaly events [42, 43] that

Algorithm 1: Dolus system phases for spoofing pretense

Input: *attacker_ID* = attacker ID,
src_ip = source IP,
dst_ip = destination IP,
no_of_packets = number of packets,
spoof_dst_ip = spoofed IP,
black_ip blacklisted IP list
Result: Attack traffic will be redirected to the quarantine VM and DDoS blocking policy will be generated

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

are typical of the traffic from multiple attack sources. Specifically, the outlier detection is a composition of many efficient, multivariate outlier detectors or hypotheses functions: $\mathcal{H} = \{h_1, h_2, \dots, h_n\}$ and the result, \mathcal{F} , is an ensemble of the different hypotheses. Furthermore, we note that the traditional methods for ensemble learning use averaging or majority voting [44]. In our case, to achieve higher accuracy with a minimum size of the training dataset D , we use the Bayesian voting scheme [45] as the ensemble method to predict the result for new data x , which can be represented as Equation 3.5.

$$\mathcal{F} = \sum_{h \in \mathcal{H}} h(x)P(h|D) \quad (3.5)$$

Final ensemble result \mathcal{F} consists of all of the hypotheses in \mathcal{H} , and each hypothesis h weighted by its posterior probability $P(h|D)$. The posterior probability is proportional to the likelihood of the training data D times the prior probability of h (3.6).

$$P(h|D) \propto P(h)P(D|h) \quad (3.6)$$

Outlier Classification

The outliers detected are classified into either interesting events (e.g., attacks) or erroneous conditions (e.g., router failure). We use a simple classifier to this end: if the final ensemble results of consecutive events (detected in the first stage) fall in the same range, we classify them as an attack; otherwise, we ignore those events. We remark that the above two-stage ensemble learning scheme requires a sizable amount of data to classify the attacks effectively. To overcome this challenge, we initially let the attacker(s) to attack the cloud services. However, we also monitor the incoming traffic carefully and make sure that the attack does not disrupt the network resources. Once an attack is classified, which are shown separately in Figure 3.2, we reroute the attack traffic using Frenetic runtime to quarantine VM (QVM) along with sample server responses (see lines 1 through 22 of Algorithm 1).

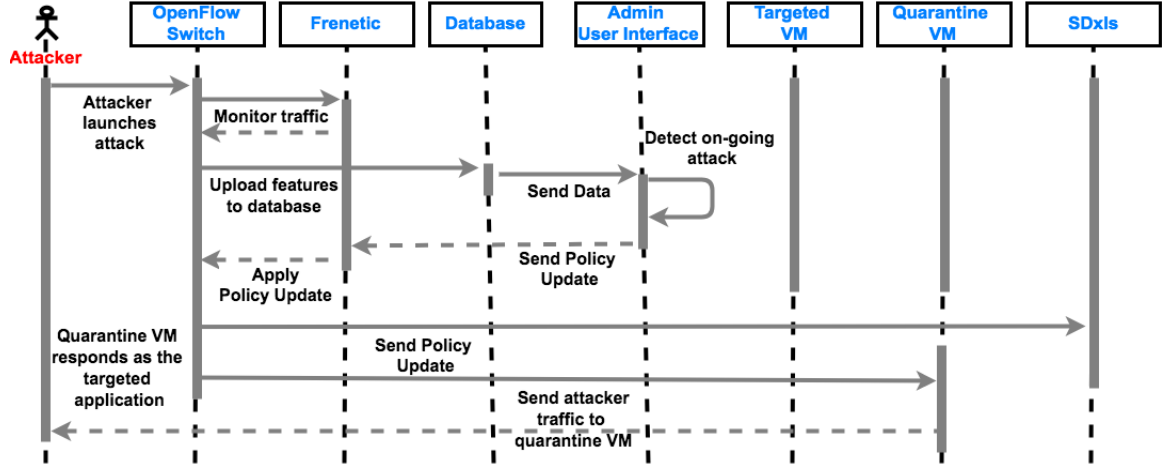
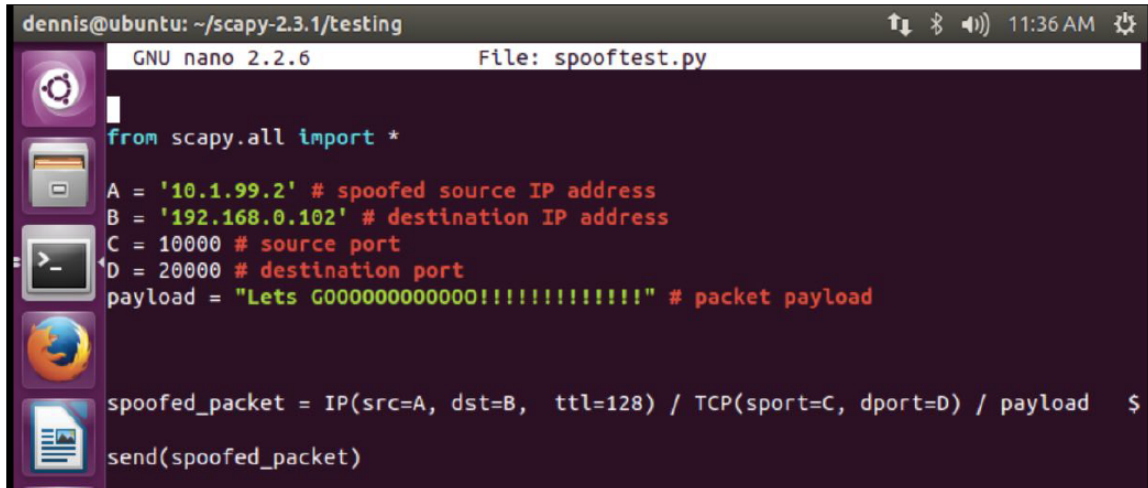


Figure 3.4: Sequence diagram of the Dolus system interactions for attack detection, quarantine setup, pretense initiation/maintenance and DDoS attack impact mitigation.

3.4.2 Quarantine Setup

Dolus calls the quarantine setup procedure (lines 1 to 9) where a new QVM is instantiated using a cloud platform’s elastic provisioning capability and the update policy routine is invoked (line 3). In the update policy routine (lines 10 to 14), we log the attack traffic to prevent future attack events as well as invoke the Frenetic runtime to generate new policies (line 12). Frenetic executes Python scripts to identify suspicious packets, learn from patterns and directs switches to redirect packets to QVMs. We then advertise this information (attack intelligence) to the neighboring switches (line 13), where, apart from the policy updates, the IP addresses of the attackers are blacklisted. Following this, based on the stored attack traffic logs, the QVM uses Scapy libraries [46] to generate responses with spoofed IP addresses and pretends as the targeted VM under attack from the perspective of the attacker(s) (lines 20 to 22). Scapy is a python based library which allowed us to write python scripts to spoof IPs. An instance of Scapy test script is shown in figure 3.5.

A screenshot of a terminal window on an Ubuntu system. The terminal title bar shows 'dennis@ubuntu: ~/scapy-2.3.1/testing'. The window contains the GNU nano 2.2.6 editor with a file named 'spooftest.py'. The script defines variables for a spoofed source IP, destination IP, source port, destination port, and a packet payload. It then constructs a spoofed packet using Scapy's IP and TCP classes and sends it.

```
dennis@ubuntu: ~/scapy-2.3.1/testing
GNU nano 2.2.6 File: spooftest.py

from scapy.all import *

A = '10.1.99.2' # spoofed source IP address
B = '192.168.0.102' # destination IP address
C = 10000 # source port
D = 20000 # destination port
payload = "Lets G000000000000!!!!!!!!!!!!!!" # packet payload

spoofed_packet = IP(src=A, dst=B, ttl=128) / TCP(sport=C, dport=D) / payload
send(spoofed_packet)
```

Figure 3.5: Scapy snippet setting up spoofed source IP address

3.4.3 Pretense Initiation

Subsequently, depending on the nature and volume of the incoming data, we decide either to move forward with the pretense or drop the traffic—which is the third step of production of appropriate behavior in pretense theory (lines 28 to 30). In order to gain more information about the attackers/attacks, we typically continue the process of pretense. While we continue the pretense, we routinely update attack intelligence such as the attacker's IP, targeted VM's IP where service(s) under attack is hosted, type of attacks, etc. Furthermore, we assume that an attacker has enough knowledge on how a successful attack should affect our system, which is another reason why we keep the attacker involved in the system as long as is usually expected. If we drop the attack traffic too early or keep it in the system for too long, they might potentially infer our pretense.

3.4.4 Pretense Maintenance

Finally, we redirect the flow of the attack traffic by pushing a new policy from the unified controller running in the cloud to the switch(es). This will redirect the attacker's traffic that is intended for the targeted VM towards the QVM. The QVM then responds to the attacker's traffic as though it is the targeted VM/server under attack with spoofed IP address and

hostname of the target, which creates the pretense effect, from an attacker’s perspective, that the targeted DDoS attack is successful. Depending on the nature of the attack, we want the attacker to believe that services are no longer up/available on the targeted VM. We therefore allow the QVM to continue to respond to the attacker for a limited amount of time t . We tune t based on the type of attack traffic and how the targeted VM would respond if it was under attack. For example, if the targeted VM went down after 10 seconds of attack, the QVM would do the same by not responding at the same time with a variable random delay factor of $[-1,1]$ seconds added. This allows the attacker to see that the services are available until, suddenly, they no longer are.

3.4.5 Policy Decision Making

In this sense, our defense maintains the pretense: gives the attacker the confirmation of a successful attack, when in reality the service has not been affected at all as seen in the Figure 4.2 considering the scenario that the user is running a video gaming portal application. This also gives us sufficient time to collect information about the attackers and their attack patterns. We use the collected information to create a blacklist of attacker information. To help network administrators effectively manage the network in the face of attacks, our system also consists of a Administrator User Interface module and a unified controller module that can be customized in a Dolus system instance deployment depicted in Figure 3.2. The User Interface shown in Figure 4.3 can be used for e.g., to enforce users to adhere to the policies generated by Frenetic runtime when they connect to the cloud. Policies generated by Frenetic internally are updated through the User Interface using JSON arrays. These policies (e.g., open/block flows) could be installed in the switches using the unified controller module, which is also linked with a back-end database that logs traffic characteristics and user profiles.

The after effects of our pretense only lasts for as long as they are needed. During the pretense, the attackers’ traffic continues to be redirected a QVM near the attacker. However,

this process need not continue indefinitely. That is, once if it has been determined that the attack traffic is no longer impacting the network, the policies can be updated to redirect the attacker traffic back to where it was prior to the start of pretense. There are several reasons to do this: (i) changes in the dynamics of the attack (e.g., bandwidth usage dropping back down to normal, absence of SYN packets in a SYN flooding attack, fixing of malware in an affected machine and hence it is no longer an attacker, etc.) calls for network policy changes so that the network resources can be effectively used, (ii) changes in traffic e.g., IP address change in incoming service requests sent from a benign user must be serviced to meet the service level agreement (SLA), and (iii) to save the operational cost of QVMs by reusing them for a different purpose e.g., periodic backups.

3.4.6 Threat Intelligence Sharing

Algorithm 1 runs in the monitor component and coordinates/shares intelligence with the switches deployed in the network and across different providers. This in turn enables a collaborative environment among providers such that the targeted attacks can be detected closer to the source *without* affecting the cloud infrastructure. A natural question is why would a provider share the attack intelligence, especially in a business that is driven by competition? We posit that the coordination among different ASes/providers is mutually beneficial for all the entities involved. Of course, a particular AS/provider can decide not to share the attack intelligence to others. However, if an AS experiences an attack and if it shares the intelligence with other ASes, a global and unified hardening of infrastructure against such targeted attacks can be achieved. In addition, any downtime is money lost in a business; sharing the attack intelligence in turn provides a cheaper alternative to lost downtime and business.

Chapter 4

Dolus System Testbed and Evaluation

In this chapter, we explain our testbed, user interfaces and then finally show the performance of the system, its schemes and components. This includes the two-stage ensemble learning algorithm, reactive false reality scheme and then the evaluation of Dolus implementation.

We,

- Test the accuracy of outlier detection and outlier classification using different datasets that include train and test data
- Present performance comparison of False Reality and Moving Target Defense and then the efficiency evaluation of False Reality in terms of cost
- Present results from two sets of experiments that were run for a maximum of 28 *seconds* to show how our Dolus implementation can be used in real-time to restore cloud services under DDoS attack situations

4.1 Experiment Setup

This section provides detailed information of our testbed, user interface and the administrator UI.

4.1.1 Testbed Setup

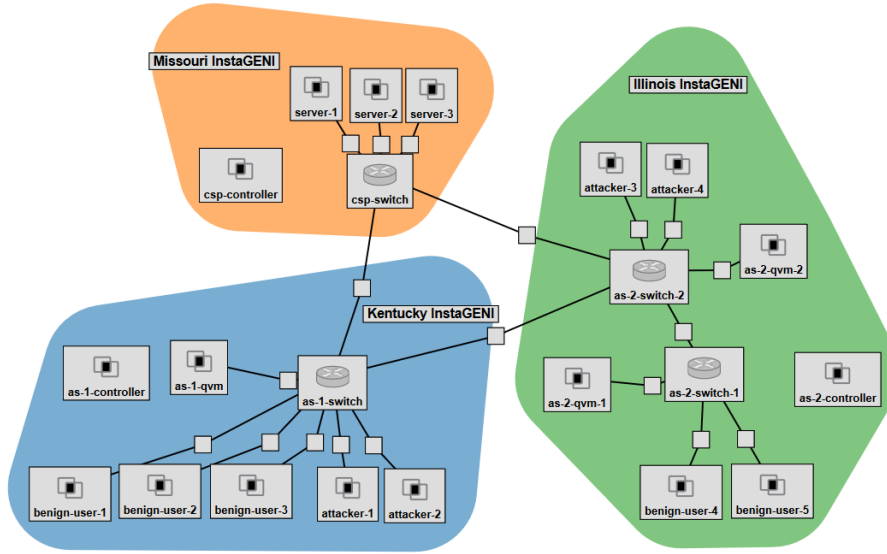


Figure 4.1: GENI Cloud testbed setup used to evaluate the Dolus system performance.

We evaluate the efficacy of our Dolus system using a realistic, GENI Cloud [47] testbed as shown in Figure 4.1. The testbed contains three SDN switches, two slave switches and a single root switch. The slave switches are each attached to users and attackers, a quarantine VM, and a connection to the root switch. Likewise, the root switch is connected to elastic VMs, each of which could serve as a candidate for the target application (i.e., a video gaming portal) hosting that could be compromised by the attackers. All switches are connected to a unified SDN controller located in the cloud service provider domain, which directs the policy updates. To emphasize on use of different ASes, we deploy the testbed spanned over three GENI aggregates, Missouri, Illinois and Kentucky InstaGENI.

4.1.2 Consumer UI

Our User Interface is a video streaming application that we use to mimic a gaming server for a good reason that gaming industries are the most DDoS attacks impacted industries. Most recent report as discussed in chapter 1 show that in the second quarter of 2017 saw numbers of gaming sites being repeatedly hit by DDoS attacks. Approximately 82% of DDoS attacks in that particular quarter were launched at gaming sites.

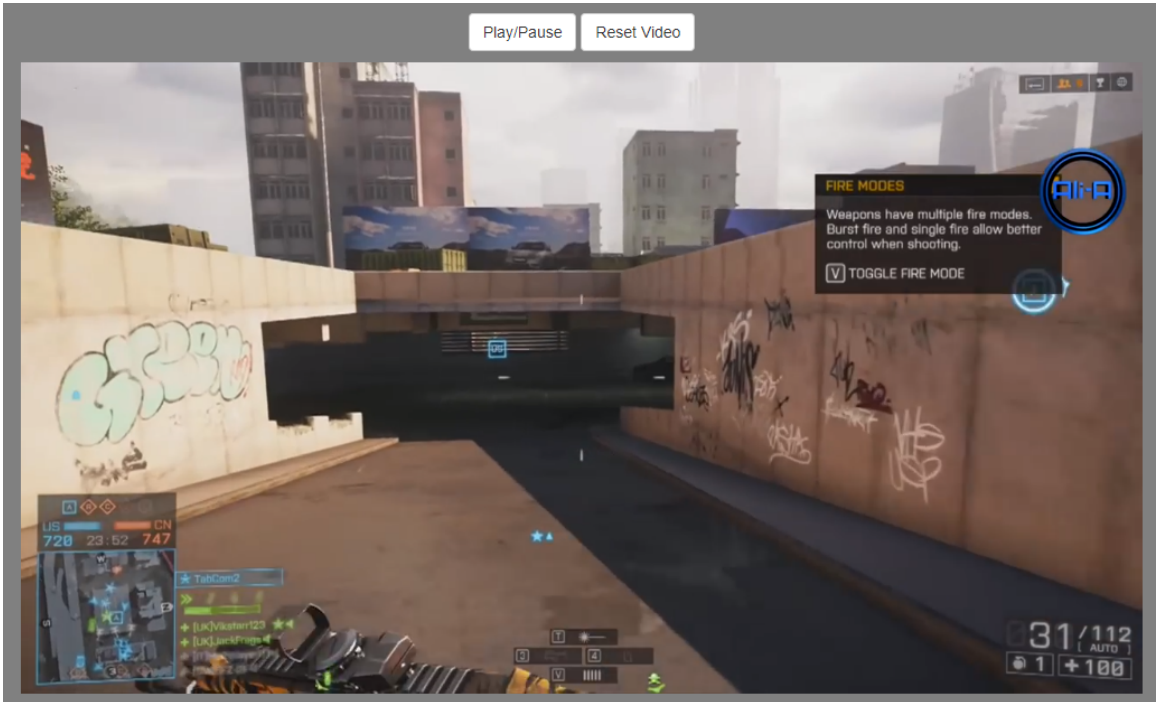


Figure 4.2: Video gaming portal application running in a SDxI-based cloud platform with cross-domain network collaboration.

The UI is built using LAMP stack that consists of a play/pause button and a reset button to provide users the ability to have control over the video. The web service also logs the timestamp of video at the time of pause or the time when the webpage is refreshed or closed. This allows users to start from where they left off.

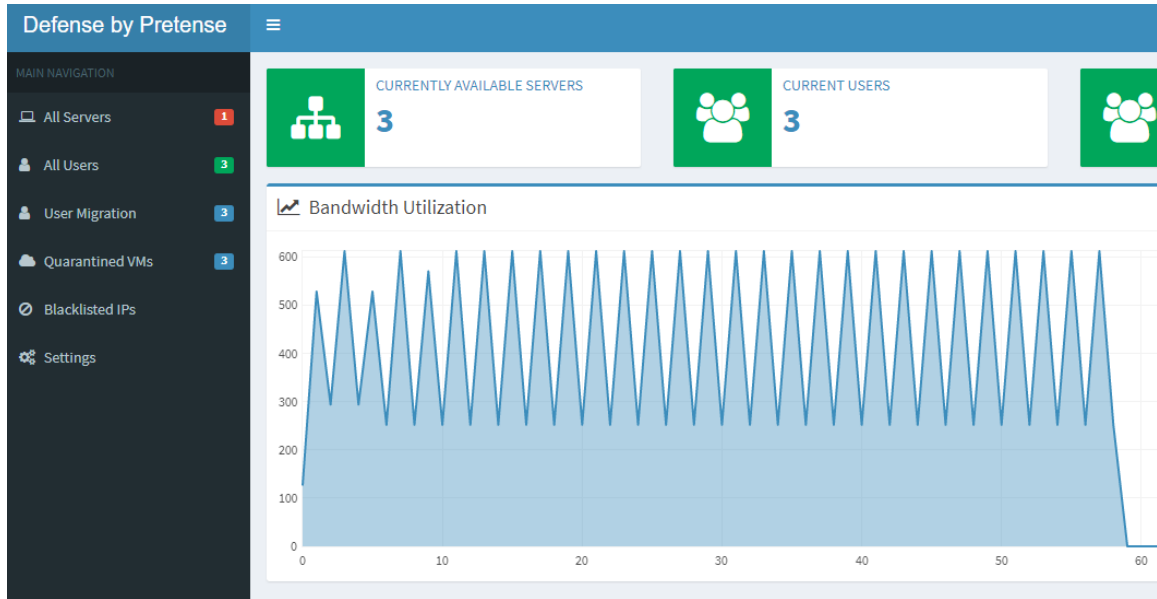


Figure 4.3: Administrator User Interface of an Dolus system instance.

4.1.3 Administrative UI

The backend of Dolus system, as depicted in figure 4.3 is highly customizable and provides administrators to have view of network through tracks of different features of network. The UI is built to keep log of everything that happens in the network and resides at the controller VM as shown in the testbed 4.1. We use Frenetic to program the system as mentioned earlier, and the UI can be used to enforce users to adhere to the policies generated by Frenetic runtime when they connect to the cloud. Policies generated by Frenetic are internally updated through the UI using JSON arrays.

The screenshot shows the 'Current Quarantined VMs' section of the dashboard. It includes a search bar and a table with 7 columns: S No, QVM Name, QVM IP Address, Start Time, Number of Attackers, and Currently Active. The table contains 3 entries. Below the table, it says 'Showing 1 to 3 of 3 entries' and has 'Previous', '1', and 'Next' navigation buttons.

S No	QVM Name	QVM IP Address	Start Time	Number of Attackers	Currently Active
3	cde.com	192.168.1.3	2017-04-26 00:00:00	1	●
2	xyz.com	192.168.1.2	2017-04-25 00:00:00	1	●
1	abc.com	192.168.1.1	2017-04-24 00:00:00	2	●

Figure 4.4: Administrator User Interface : Quarantined VMs

The UI is developed to automatically make policy updates in the system, to keep track of attack traffic and then blacklist the attacker's IP to stop future possible attacks. We show various instances of this UI, such as the list of quarantined VMs, the UI settings, and so on.

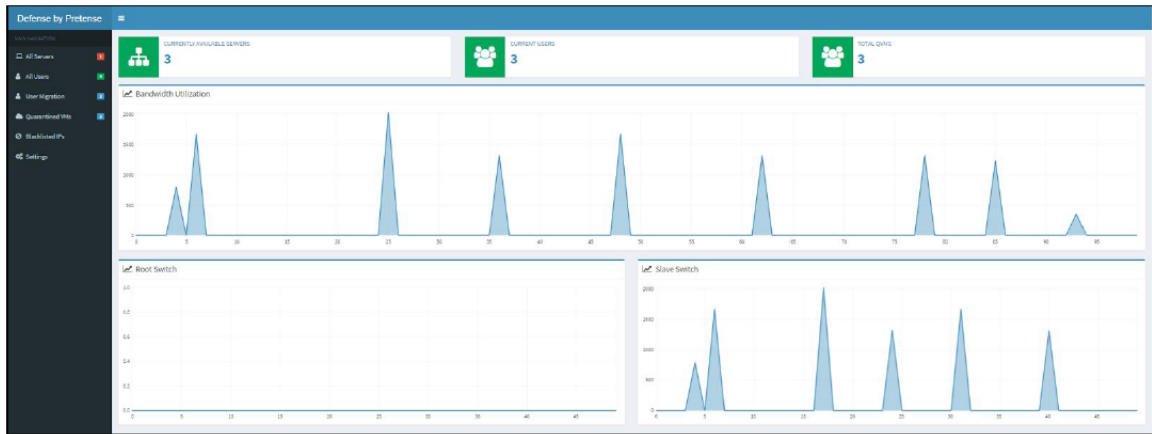


Figure 4.5: Administrator User Interface : Bandwidth utilization when access to root switch is disallowed

Figure 4.4 shows the list of quarantined VMs at the time when it was recorded. We use dummy data just for the purpose of explanation.

Similarly, figure 4.5 shows the network bandwidth utilization when Frenetic doesn't allow traffic to enter to the core switch. This figure shows the overall network along with the utilization at both slave and root switch. We can see that there is no traffic movement at the root switch since we are able to stop the traffic to get in the root switch.

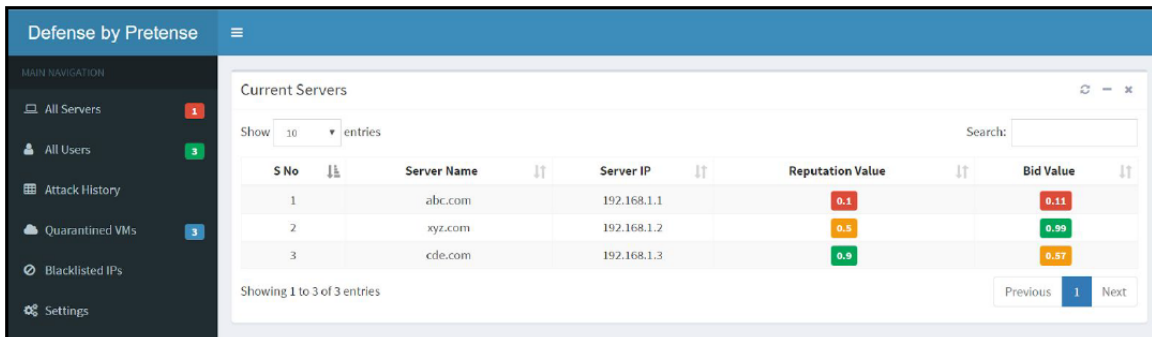


Figure 4.6: Administrator User Interface : Server History

In addition to the instances of UI explained earlier, we can also have record of servers'

history. 4.6 shows server histories logged based on their reputation. The reputation of a particular server is determined by the number of times it has been attacked or was targeted for an attack.

Admin UI is also capable of listing history of attacks that have occurred in the system. Origin of attack is logged in the system and through UI and using JSON arrays, policy updates in the system can automatically be made. Attack history is shown in figure 4.7 The automation also includes blacklisting of attacker’s IP.

Attack No	Source IP	Destination IP	Time To Live (TTL)	Start Time
1	192.168.1.1	192.168.1.1	128 seconds	2017-04-01 00:00:00
2	192.168.1.2	192.168.1.11	128 seconds	2017-04-01 00:00:00
3	192.168.1.3	192.168.1.21	128 seconds	2017-04-01 00:00:00
4	192.168.1.4	192.168.1.31	128 seconds	2017-04-01 00:00:00
5	192.168.1.5	192.168.1.41	128 seconds	2017-04-01 00:00:00
6	192.168.1.6	192.168.1.51	128 seconds	2017-04-01 00:00:00
7	192.168.1.7	192.168.1.61	128 seconds	2017-04-01 00:00:00
8	192.168.1.8	192.168.1.71	128 seconds	2017-04-01 00:00:00
9	192.168.1.9	192.168.1.81	128 seconds	2017-04-01 00:00:00
10	192.168.1.10	192.168.1.91	128 seconds	2017-04-01 00:00:00

Figure 4.7: Administrator User Interface : Attack History

4.2 Attack Detection and Classification

Using the Dolus system, we monitor different types of data that are permitted to enter the GENI Cloud testbed depicted in Figure 4.1. We send both normal and attack traffic (i.e., our datasets) to the targeted server to test the efficacy of our two-stage ensemble learning scheme. As shown in Figure 4.8, the algorithm module will be initialized in *Stage-1*. When an event is detected as an outlier (the probability is larger than ϵ , which is tuned in training phase) or if there is a manual trigger, the algorithm will transition to *Stage-2*, and may query more data or features to characterize this outlier event. Our evaluation results span

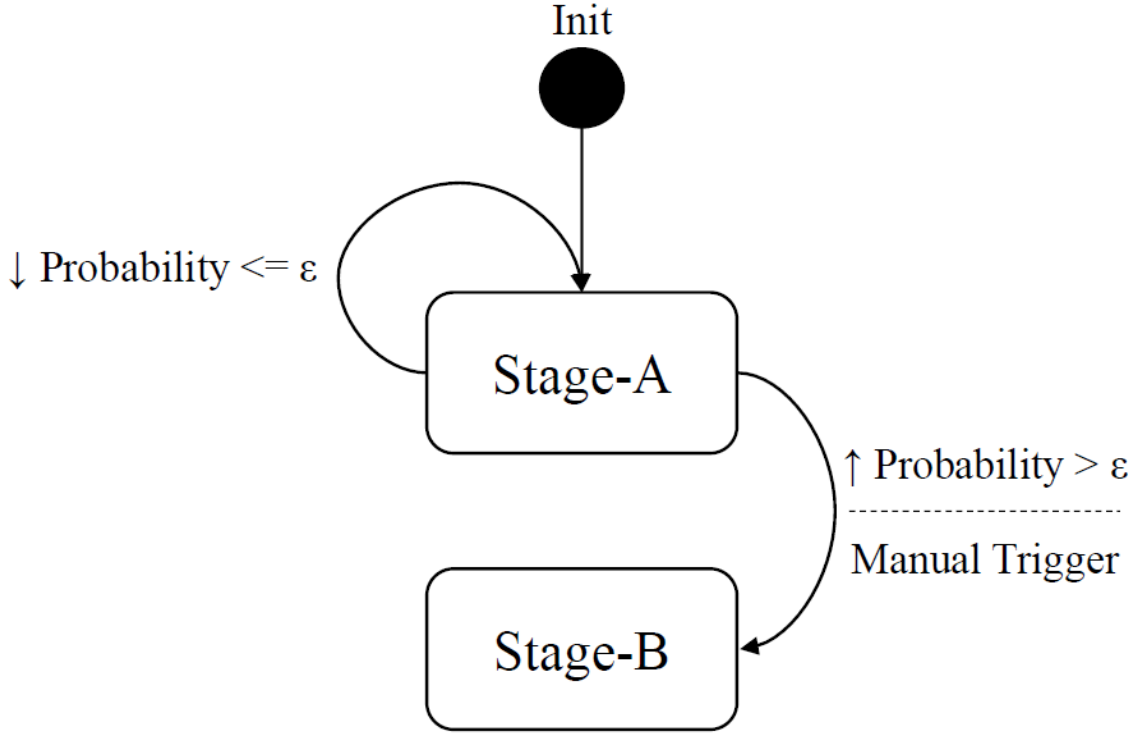
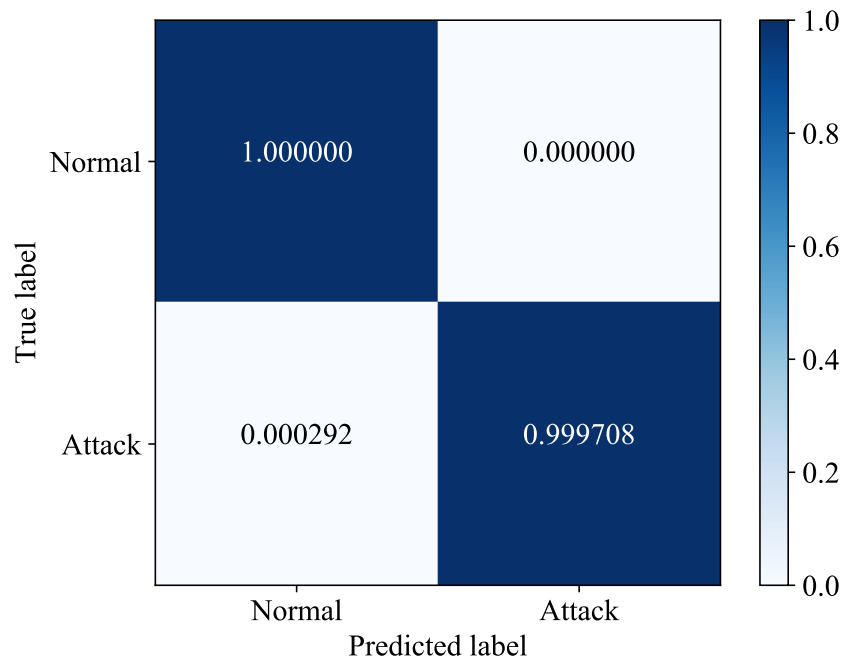


Figure 4.8: Two stages transition diagram

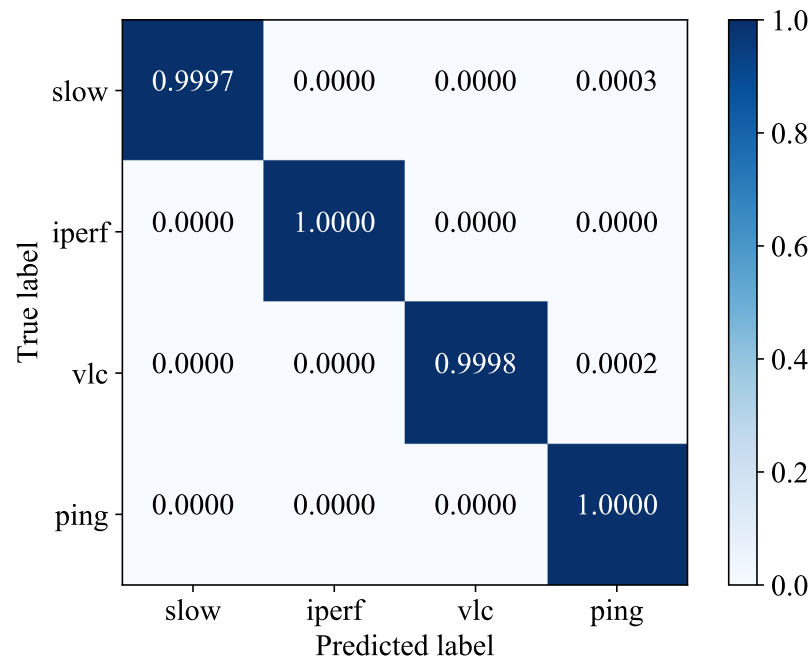
over two instances of learning of datasets as explained in the following.

The first instance shows multiple traffic types from a single attacker VM to a single target node. For this instance, we divide $\sim 180,000$ lines of data into two sets, one for training and the other to test the accuracy of our scheme. Furthermore, the types of traffic used to create these instances are composed of SlowHTTPTest, iperf, VLC and ICMP ping. Figure 4.9 shows the two confusion matrices for attack detection and classification in a normalized fashion. We note that both the detection and the classification of attack took less than a second. In addition to the rapid detection and classification, our approach is highly accurate as shown in Table 4.1, where stage 1 is the detection stage and stage 2 is the classification stage.

In the second instance, we consider multiple traffic types to multiple hosts. This instance is composed of 2.5 million rows per test, totaling 5 million rows of data. The types of traffic that we use to create this dataset include SlowHTTPTest, iperf, VLC, scp, wget,

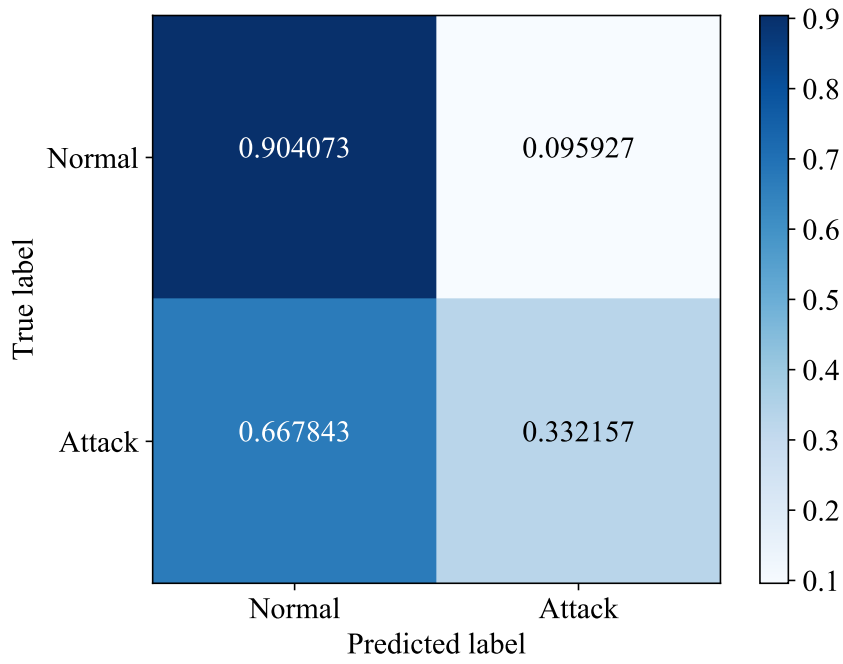


(a) Attack detection.

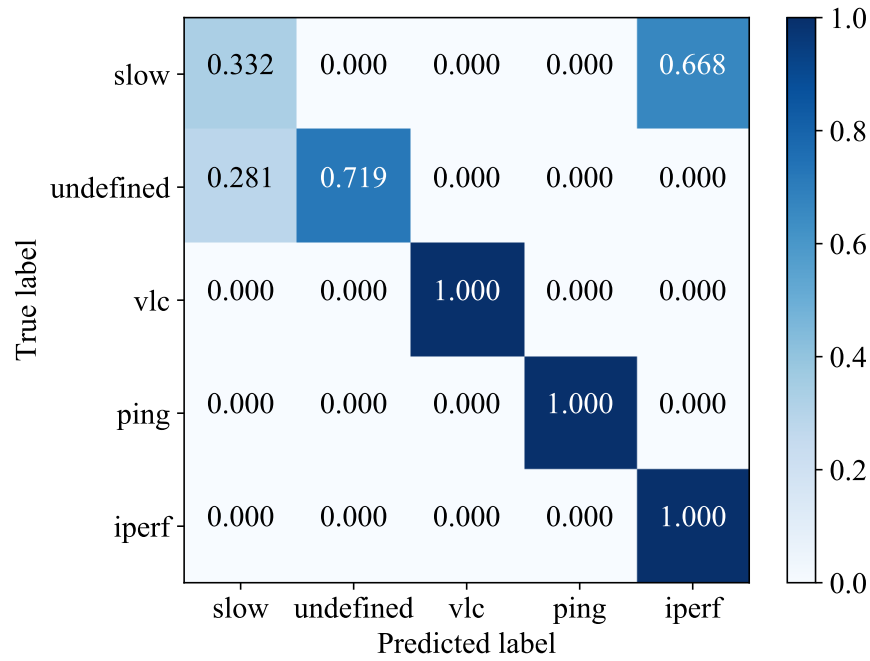


(b) Attack classification.

Figure 4.9: Confusion matrices for attack detection and classification for multiple traffic flows sent to a single server.



(a) Attack detection.



(b) Attack classification.

Figure 4.10: Confusion matrices for attack detection and classification for multiple traffic flows sent to multiple hosts.

and ICMP ping. This dataset also contains some unlabeled/undefined data for the scheme to assess and classify the training data to evaluate the effectiveness of our two-stage ensem-

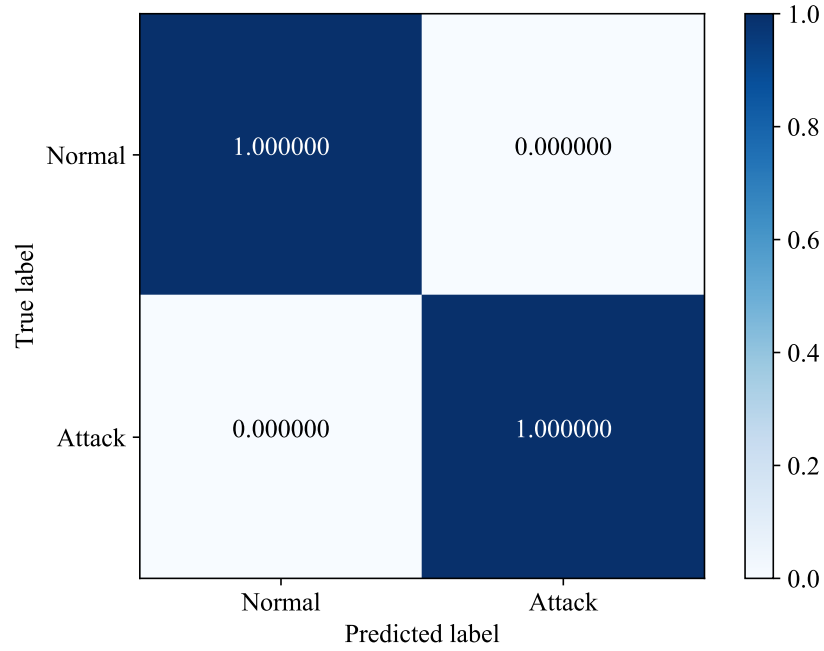
Table 4.1: Overall Attack Detection and Classification Time and Accuracy

Tests	Time (in Seconds)	Accuracy (in %)
Single server stage 1	1	99.99
Single server stage 2	1	99.98
Multiple hosts stage 1	7	89.12
Multiple hosts stage 2	13	98.49

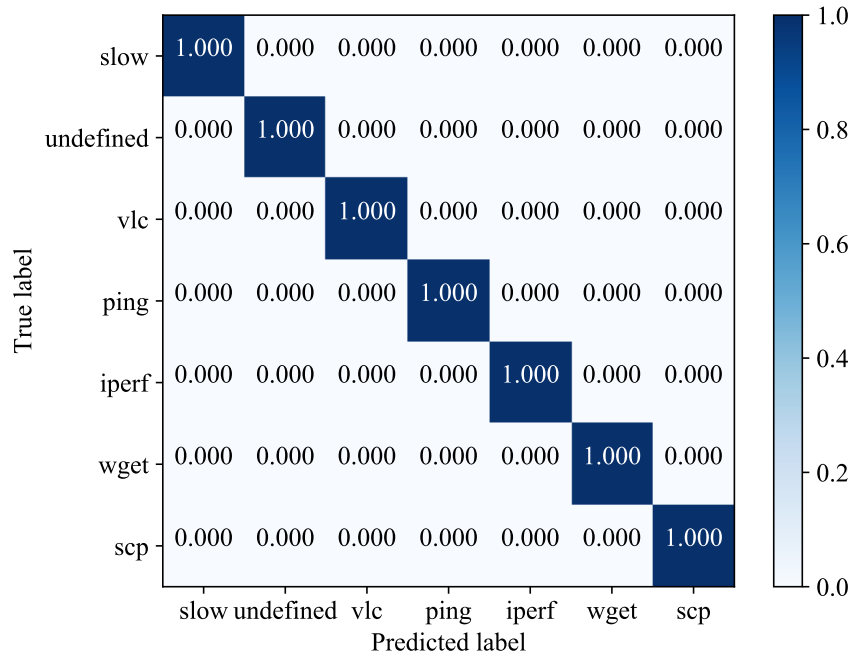
ble learning scheme. Figure 4.10 shows the two confusion matrices in normalized form for attack detection and classification. Detection and classification of attack took ~ 7 and ~ 13 seconds, respectively. Despite the slowdown in attack detection/classification in comparison with the first instance, the accuracy of our approach is still high as shown in Table 4.1.

While the two-stage ensemble learning scheme is effective in detecting test data, a new attack that has not been used in training could initially go undetected and impact services. However, with pertinent labeling of attack traffic flows during training, the accuracy of the ensemble learning scheme can be improved significantly. We depict the outlier detection and classification for a trained case in Figure 4.11, where we make use of 60% of the data as training data and 40% as test data for the same dataset used in the 2nd instance. For the purpose of our evaluation, the sorted dataset has randomized time stamps.

Though the dataset that we use is discrete with differences in traffic such as protocol, bytes transmitted, number of packets, source and destination addresses, our two-stage ensemble learning scheme is effective in detecting the attacks with good accuracy and efficiency. The ensemble learning scheme can further be modified based on other characteristics of network traffic, and such modifications are beyond the scope of the work in this paper.



(a) Attack detection.



(b) Attack classification.

Figure 4.11: Confusion matrices for outlier detection and classification for multiple traffic flows comprising of familiar attack flows.

4.3 False reality establishment results

False Reality evaluation supports in determining the amount of dummy traffic that needs to be synthetically created to instantiate the pretense. In our false reality establishment evaluation, we compare the results from false reality to a system that offers MTD based defense mechanism [34]. The comparisons are made in terms of the chances for the attacker to distinguish between VMs with real and dummy traffic. In Figures 4.12, and 4.13, we compare the average response time and percentage of dropped packets during the idle and probing phases for the different attack intensity settings. We observe that - for the migration without false reality, the difference between response times and percentage of dropped packets before and after migration/redirection is sufficient enough, from [34], for the at-

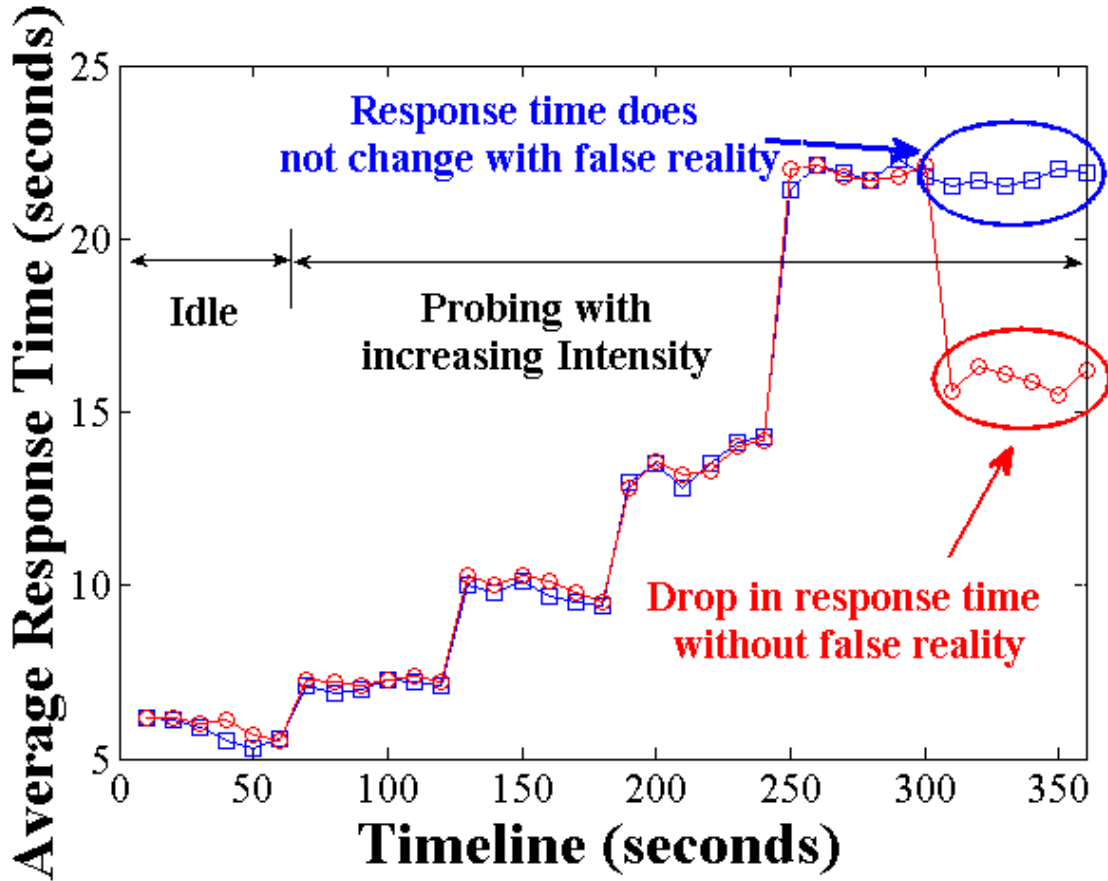


Figure 4.12: Performance comparison of reactive migration strategies with-and-without false reality: Average response time

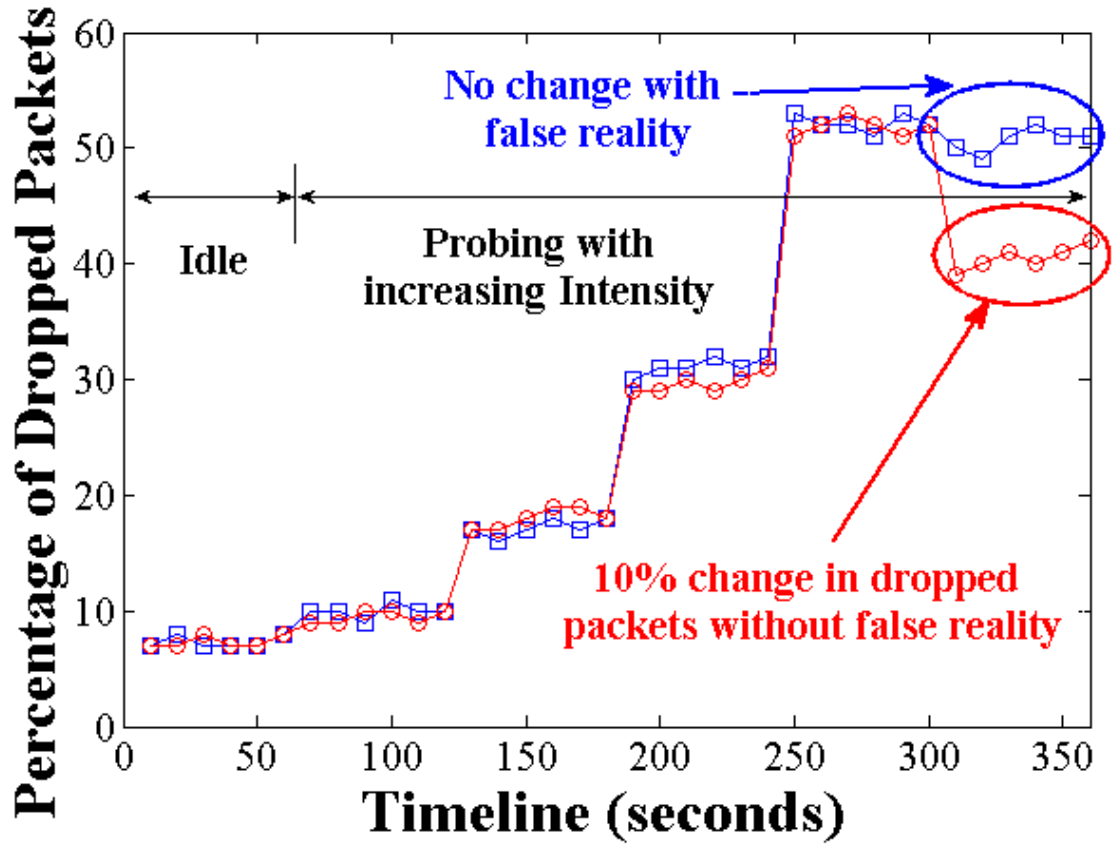


Figure 4.13: Performance comparison of reactive migration strategies with-and-without false reality: Percentage of dropped packets

tacker to detect migration/redirection scenarios. Such a detection suggests to the attacker that a high-value target has been found, and thus increases the chances of a future (more intensive) attack. Whereas, for our proposed scheme with false reality, the change in response time and dropped packets percentage after migration/redirection is negligible. The effectiveness of our false reality environment shown in Figures 4.12 and 4.13 are direct outcomes of the intelligent dummy traffic generation insights.

Next, we demonstrate the overall effectiveness of our proposed false reality pretense in terms of the success rate of an attacker in identifying the presence of dummy traffic in Figure 4.14. The success rate is calculated by the total number of successful dummy identifications for the target application consumption by small-to-large number of users. The results present an average of 20 reactive migration attempts involving different destination

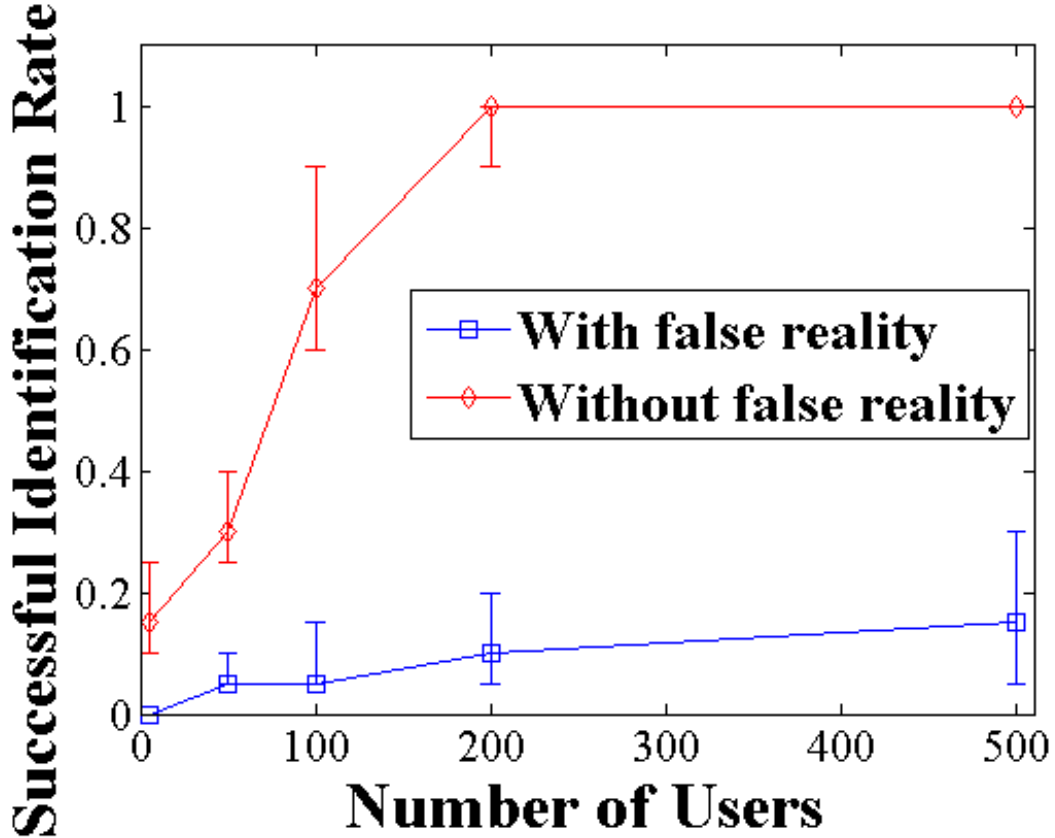


Figure 4.14: Comparison of successful identification of dummy traffic by the attacker

VMs with varied properties. Figure 4.14 shows that the false reality ensures considerably lower detection success in terms of identifying the dummy traffic patterns. We observe that with small number of users, the benefits of false reality is not pronounced enough as with small amount of traffic generated from small number of users using the target application. Consequently, the attacker cannot really experience any perceivable difference in response times when the users are redirected. Thus, it is possible for our schemes with-and-without false reality to be virtually indistinguishable. Moreover, schemes without false reality perform well when number of users using the target application is very small. However, with large number of users, the attacker can easily identify the migration/redirection with considerable change in response times and dropped packets in the absence of false reality (from Figures 4.12 and 4.13).

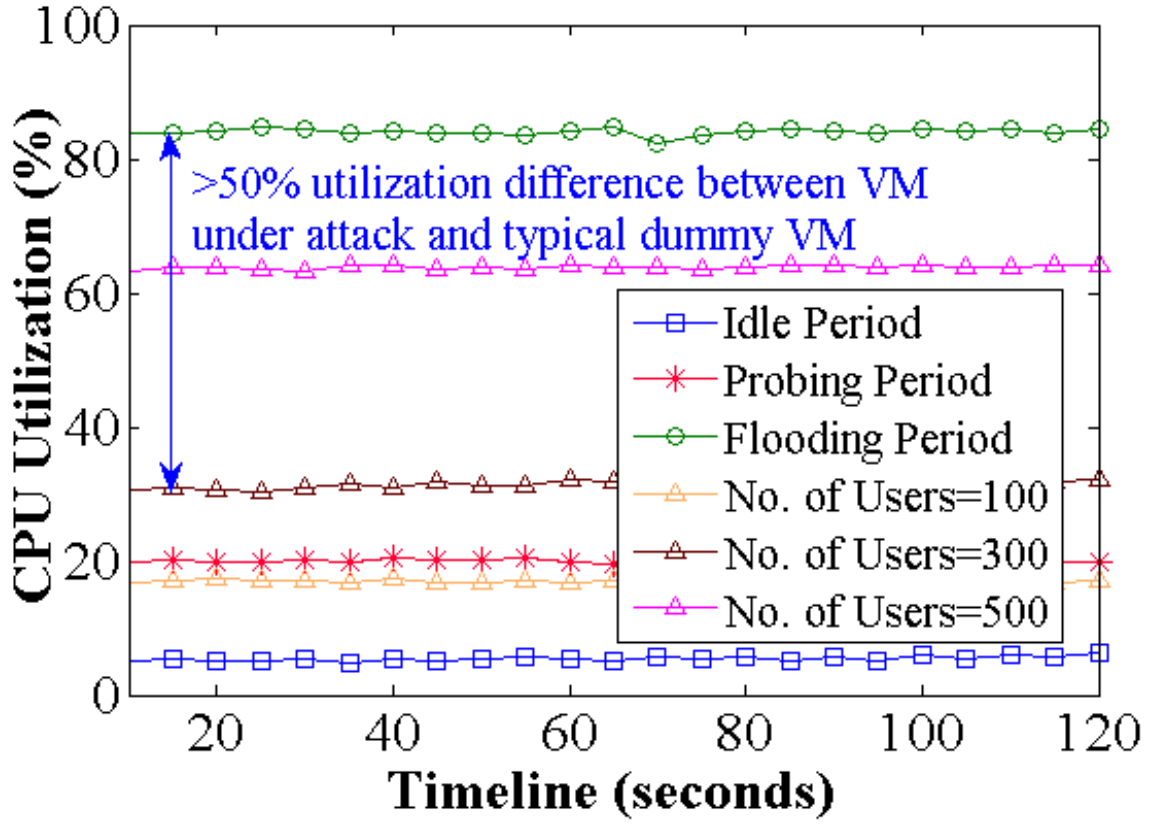


Figure 4.15: Benefits and costs of false reality in terms of average CPU utilization

Finally in Figure 4.15, we characterize the benefits of the false reality pretense in terms of the ‘lost opportunity cost’ of a DDoS attack and compare it with the cost of implementation of the false reality pretense. We perform the comparison in terms of average CPU utilization in order to ascertain whether the creation of false reality is viable for the CSPs. We calculated the CPU utilization of the VM hosting the target application during: idle, probing (200 packets/sec), and flooding periods. We then compared them with the CPU utilization of the standby VM generating dummy traffic for 100, 300, and 500 users, respectively. From Figure 4.15, it is evident that the utilization of a VM under DDoS attack is more than 50% higher than an average dummy VM generating traffic for around 300 users, thus satisfying Equation (3.4) for cost effectiveness of the false reality pretense. Interestingly while generating high loads of dummy traffic, i.e., mimicking more than 1000

users' traffic, the dummy VM's CPU utilization might be as high as a VM under attack, thus making false reality less beneficial in those circumstances (at least in terms of CPU utilization cost). These results overall not only signify the importance of the false reality pretense in creating an illusion of success for the attacker, but also indicate exactly how much dummy traffic needs to be generated in a cost effective manner to cope with a given DDoS attack state.

4.4 Time to restore Cloud-hosted Application Service

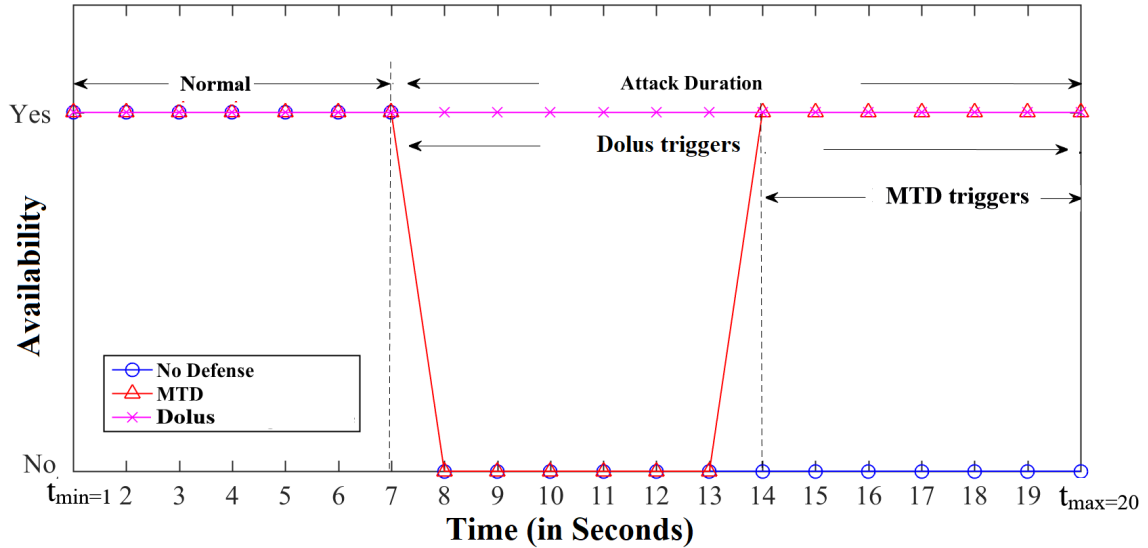


Figure 4.16: Comparison of the *cloud service restoration time* metric with cases of: no Defense, with MTD and with Dolus.

Figure 4.16 compares the time taken by our Dolus system to stop a DDoS attack versus MTD-based and no defense strategies. After a warm-up period of 6 *seconds*, we start the SlowHTTPTest and hping3 at the 7th *second* from the attackers. In a SDxI-based cloud network with no defense strategy, the services are immediately affected by the attack traffic. Similarly, the MTD-based defense strategy takes ~ 6 *seconds* to mitigate the attack traffic impact. However, our Dolus system supported service on the other hand, does not suffer

from any loss of availability in comparison with the other two strategies. This is due to the sharing of attack intelligence between the slave switches and redirection of attack traffic to quarantine VMs closer to the attackers, making the cloud network completely oblivious to the attackers.

4.5 Amount of Traffic Processed at the Root Switch

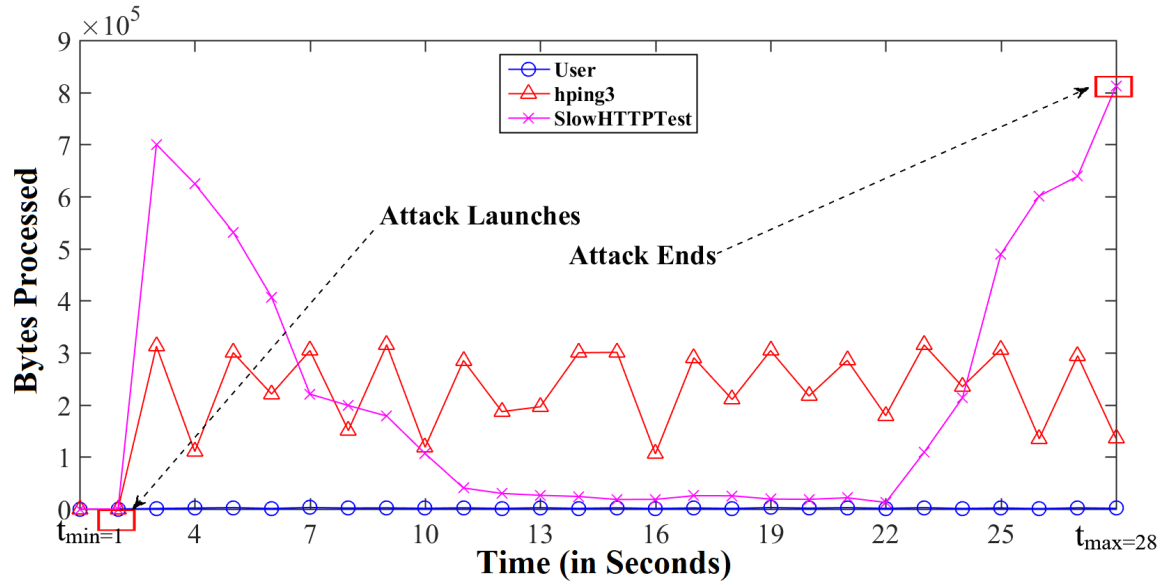


Figure 4.17: Traffic processed (in Bytes) in one of the slave switches.

Figures 4.17 and 4.18 depict the amount of traffic processed (in Bytes) at one of the slave switches and the root switch. From Figure 4.18, it is evident that the SDxI-based cloud network is oblivious to the attack traffic impact, complementing the result in Figure 4.16. Since the slave switch represented in Figure 4.18 redirects attack traffic to the quarantine VMs, we observe a 5X increase in the amount of traffic processed in comparison with the root switch.

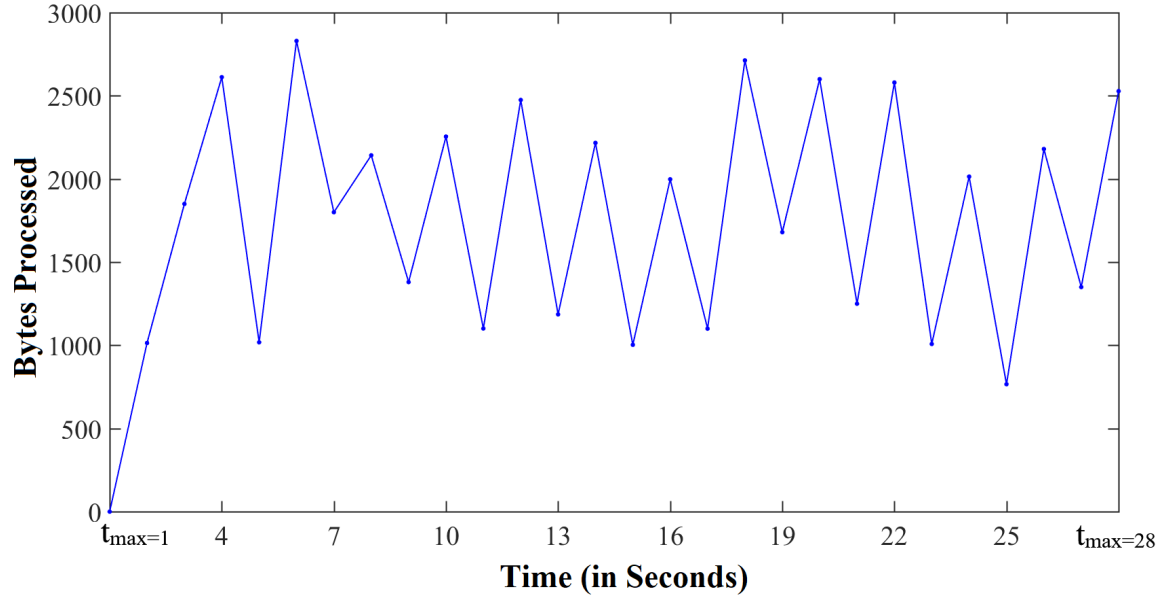


Figure 4.18: Traffic Processed at the root switch only shows user traffic proving that the attack traffic is redirected to quarantine VM.

Overall, we find that our Dolus can effectively detect DDoS attack and redirect traffic in real-time i.e., on the order of seconds depending on the knowledge of the DDoS attack pattern, and block it closer to the attack source in 1-2 seconds if automated policy updates are possible in the cross-domain setting. However, if humans need to be brought into the loop, the time to block the attack can be adjusted so that there is enough time for cross-domain manual coordination during which an effective pretense of the quarantine VM is deceiving the attacker with a false sense of success.

Chapter 5

Summary and concluding remarks

Recent innovations in the orchestration of cloud resources are fueled by the emergence of the Software-Defined everything (SDx) Infrastructure (SDxI) paradigm. At the same time, the sophistication of Distributed Denial-of-Service (DDoS) attacks are growing on an unprecedented scale, and online businesses in retail, healthcare and other fields are under constant threat. In this paper, we presented a novel defense system called *Dolus* to mitigate the impact of DDoS attacks against high-value services hosted in SDxI-based cloud platforms. We proposed a *defense by pretense* mechanism that can be used during defense against flooding attacks, which involves a two-stage ensemble learning algorithm to analyze features in order to determine where an attack originates from, and the attack type. Using blacklisting information, our pretense initiation builds upon pretense theory concepts in child play psychology to trick an attacker through creation of a false sense of success.

Our above approach takes advantage of elastic capacity provisioning in cloud platforms to implement moving target defense techniques that does not affect the cloud-hosted application users, and contains the attack traffic in a quarantine VM(s). With the time gained through effective pretense initiation, cloud service providers could coordinate across a unified SDxI infrastructure involving multiple ASes to decide on policies that help in blocking the attack flows closer to the source side. Performance evaluation results of our Dolus sys-

tem in a GENI cloud testbed show that our approach can be effective in filtering, detection and implementation of SDxI-based infrastructure policy coordination for mitigation of the impact of DDoS attacks.

Chapter 6

Future Work

Dolus system can be extended where its components address more complex targeted attacks such as Advanced Persistent Threats (APTs) as part of cyberhunting workflows. This will require advanced data sampling/analysis, as well as relevant machine learning techniques to help SDxI-based cloud service providers to visualize collateral effects in invoking one or more defense mechanisms.

Though there have been many approaches in using methods of pretense, the research work in the system we built, in our opinion, have just emerged and can be bettered by fully leveraging and implementing knowledge gathering and threat intelligence sharing across multiple domains. This will need cooperation from heterogeneous Autonomous systems, to comply with a base system as the coordination amongst different ASes/providers is mutually beneficial for all entities involved.

Bibliography

- [1] Verizon. State of the Market: Enterprise Cloud. http://www.verizonenterprise.com/resources/reports/rp_state-of-the-market-enterprise-cloud-2016_en_xg.pdf, 2017.
- [2] SDxCentral. SDxCentral. Software Defined Everything Part 3: SDx infrastructure. <https://www.sdxcentral.com/cloud/definitions/software-defined-everything-part-3-sdx-infrastructure>, 2017.
- [3] A. Gupta, N. Feamster, and L. Vanbever. Authorizing network control at software defined internet exchange points. In *Proceedings of the Symposium on SDN Research, SOSR '16*, pages 16:1–16:6, New York, NY, USA, 2016. ACM.
- [4] P.C. Hershey and C.B. Silio Jr. Procedure for detection of and response to distributed denial of service cyber attacks on complex enterprise systems. In *Systems Conference (SysCon), 2012 IEEE International*, pages 1–6, March 2012.
- [5] Q2 2017 State of the Internet Security Report. <https://www.akamai.com/de/de/multimedia/documents/state-of-the-internet/q2-2017-state-of-the-internet-security-report.pdf>, 2017.
- [6] P Stavroulakis and M Stamp. *Handbook of information and communication security*. Springer-Verlag, 2010.

- [7] A. Kartit, A. Saidi, F. Bezzazi, M. El Marakki, and A. Radi. A new approach to intrusion detection system. 36(2):284 – 289, February 2012.
- [8] Extreme ddos defense (xd3). <http://www.darpa.mil/program/extreme-ddos-defense/>.
- [9] P. Kampanakis, H. G. Perros, and T. Beyene. Sdn-based solutions for moving target defense network protection. In *WoWMoM*, pages 1–6. IEEE Computer Society, 2014.
- [10] S. Debroy, P. Calyam, M. Nguyen, A. Stage, and V. Georgiev. Frequency-minimal moving target defense using software-defined networking. *2016 International Conference on Computing, Networking and Communications (ICNC)*, 00:1–6, 2016.
- [11] S. Nichols and S. Stich. A cognitive theory of pretense. *Cognition*, 74(2):115–147, 2000.
- [12] J. W. Van de Vondervoort and O. Friedman. Young children protest and correct pretense that contradicts their general knowledge. *Cognitive Development*, 43:182–189, 2017.
- [13] Q. Yan, F. R. Yu, Q. Gong, and J. Li. Software-defined networking (sdn) and distributed denial of service (ddos) attacks in cloud computing environments: A survey, some research issues, and challenges. *IEEE Communications Surveys Tutorials*, 18(1):602–622, Firstquarter 2016.
- [14] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker. Frenetic: A network programming language. In *ACM Sigplan Notices*, volume 46, pages 279–291. ACM, 2011.
- [15] Carolyn Jane Anderson, Nate Foster, Arjun Guha, Jean-Baptiste Jeannin, Dexter Kozen, Cole Schlesinger, and David Walker. Netkat: Semantic foundations for networks. In *ACM SIGPLAN Notices*, volume 49, pages 113–126. ACM, 2014.

- [16] S. Seufert and D. O'Brien. Machine learning for automatic defence against distributed denial of service attacks. In *2007 IEEE International Conference on Communications*, pages 1217–1222, June 2007.
- [17] J. Choi, C. Choi, B. Ko, and P. Kim. A method of ddos attack detection using http packet pattern and rule engine in cloud computing environment. *Soft Computing*, 18(9):1697–1703, Sep 2014.
- [18] T. Thapngam, S. Yu, W. Zhou, and S. K. Makki. Distributed denial of service (ddos) detection by traffic pattern analysis. *Peer-to-Peer Networking and Applications*, 7(4):346–358, Dec 2014.
- [19] Y. Chen, S. Jain, V. K. Adhikari, Z. L. Zhang, and K. Xu. *A first look at inter-data center traffic characteristics via Yahoo! datasets*, pages 1620–1628. 2011.
- [20] L. Yang, T. Zhang, J. Song, J. S. Wang, and P. Chen. Defense of ddos attack for cloud computing. In *2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE)*, volume 2, pages 626–629, May 2012.
- [21] Internet2's ddos mitigation strategy. <https://www.internet2.edu/blogs/detail/12234>, 2016.
- [22] Yanghee Choi. Implementation of content-oriented networking architecture (cona): a focus on ddos countermeasure.
- [23] C. YuHunag, T. MinChi, C. YaoTing, C. YuChieh, and C. YanRen. A novel design for future on-demand service and security. In *2010 IEEE 12th International Conference on Communication Technology*, pages 385–388, Nov 2010.
- [24] S. Shin, P. A. Porras, V. Yegneswaran, M. W. Fong, G. Gu, and M. Tyson. Fresco: Modular composable security services for software-defined networks. In *NDSS*. The Internet Society, 2013.

- [25] Y. Yu, C. Qian, and X. Li. Distributed and collaborative traffic monitoring in software defined networks. In *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, HotSDN '14, pages 85–90, New York, NY, USA, 2014. ACM.
- [26] A. TaheriMonfared and C. Rong. *Multi-tenant Network Monitoring Based on Software Defined Networking*, pages 327–341. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
- [27] J. Idziorek, M. Tannian, and D. Jacobson. Detecting fraudulent use of cloud resources. In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, CCSW '11, pages 61–72, New York, NY, USA, 2011. ACM.
- [28] H. Tian and J. Bi. An incrementally deployable flow-based scheme for ip traceback. *IEEE Communications Letters*, 16(7):1140–1143, 2012.
- [29] N. Handigol, B. Heller, V. Jeyakumar, D. Mazières, and N. McKeown. I know what your packet did last hop: Using packet histories to troubleshoot networks. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, pages 71–85, Seattle, WA, 2014. USENIX Association.
- [30] S. A. Mehdi, J. Khalid, and S. A. Khayam. *Revisiting Traffic Anomaly Detection Using Software Defined Networking*, pages 161–180. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
- [31] G. Yao, J. Bi, and P. Xiao. Source address validation solution with openflow/nox architecture. In *2011 19th IEEE International Conference on Network Protocols*, pages 7–12, Oct 2011.
- [32] A. Clark, K. Sun, and R. Poovendran. Effectiveness of IP address randomization in decoy-based moving target defense. In *Proceedings of the 52nd IEEE Conference on Decision and Control, CDC 2013, December 10-13, 2013, Firenze, Italy*, pages 678–685, 2013.

- [33] T. Sochor and M. Zuzcak. Study of internet threats and attack methods using honeypots and honeynets. In *International Conference on Computer Networks*, pages 118–127. Springer, 2014.
- [34] S. Debroy, P. Calyam, M. Nguyen, A. Stage, and V. Georgiev. Frequency-Minimal Moving Target Defense using Software-Defined Networking. In *Conference on Computing, Networking and Communications (ICNC), IEEE International*, February 2016.
- [35] Open Networking Foundation. <https://www.opennetworking.org/>.
- [36] A. Gupta, L. Vanbever, M. Shahbaz, S. P. Donovan, B. Schlinker, N. Feamster, J. Rexford, S. Shenker, R. Clark, and E. Katz-Bassett. Sdx: A software defined internet exchange. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM ’14, pages 551–562, New York, NY, USA, 2014. ACM.
- [37] W. M. Eddy. Tcp syn flooding attacks and common mitigations. 2007.
- [38] F. Gont. Icmp attacks against tcp. 2010.
- [39] Kali Tools. hping3. ICMP or SYN flooding tool. <https://tools.kali.org/information-gathering/hping3>, 2014.
- [40] S. Shekhan. SlowHTTPTest. Application Layer DoS attack. <https://github.com/shekhan/slowhttptest/wiki>, 2011.
- [41] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. 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.
- [42] Y. Zhang, S. Debroy, and P. Calyam. Network-wide anomaly event detection and

- diagnosis with perfsonar. *IEEE Transactions on Network and Service Management*, 13(3):666–680, Sept 2016.
- [43] Y. Zhang, P. Calyam, S. Debroy, and M. Sridharan. Pca-based network-wide correlated anomaly event detection and diagnosis. In *2015 11th International Conference on the Design of Reliable Communication Networks (DRCN)*, pages 149–156, March 2015.
- [44] R S Boyer and J S Moore. Mjrtty—a fast majority vote algorithm. In *Automated Reasoning*, pages 105–117. Springer, 1991.
- [45] Thomas G Dietterich et al. Ensemble methods in machine learning. *Multiple classifier systems*, 1857:1–15, 2000.
- [46] Scapy: Packet manipulation tool. <http://www.secdev.org/projects/scapy/>, 2007.
- [47] M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, and I. Seskar. Geni: A federated testbed for innovative network experiments. *Comput. Netw.*, 61:5–23, March 2014.