Intelligent Defense using Pretense against Targeted Attacks in Cloud Platforms

Roshan Lal Neupane[1a], Travis Neely[1b], Prasad Calyam[b], Nishant Chettri[a], Mark Vassell[a], Ramakrishnan Durairajan[c]

[a]{rlnzq8, nc5ff, mdvy96}@mail.missouri.edu
[b]{neelyt, calyamp}@missouri.edu
[c]ram@cs.uoregon.edu

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 programmable as Software Defined everything Infrastructures (SDxI). At the same time, sophisticated targeted attacks such as Distributed Denial-of-Service (DDoS) and Advanced Persistent Threats (APTs) are growing on an unprecedented scale threatening the availability of online businesses. In this paper, we present a novel defense system called *Dolus* to mitigate the impact of targeted 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. 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 deceive the attacker by creating a false sense of success. We evaluate the efficacy of Dolus using a GENI Cloud testbed and demonstrate its real-time capabilities to: (a) detect DDoS and APT attacks and redirect attack traffic to quarantine resources to engage the attacker under pretense, (b) coordinate SDxI policies to possibly block attacks closer to the attack source(s).

Keywords: Software-defined Infrastructure, DDoS Attacks, Advanced Persistent Threats, Pretense Theory, Network Analytics for Targeted Attack Defense

[1] These authors contributed equally to this work

1. Introduction

Cloud computing has become an essential aspect of online services available to customers in major consumer fields such as e.g., retail, healthcare, 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].

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 multiple 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].

While the adoption of SDxI-based clouds is starting to mature, sophisticated targeted attacks such as Distributed Denial-of-Service (DDoS) attacks and Advanced Persistent Threats (APTs) are simultaneously growing on an unprecedented scale. DDoS attacks can have significant effects on cloud-hosted services (i.e., attack "targets") and are continual threats on the availability of online businesses to customers. If successful, they also cause significant loss of revenue/reputation for a large number of enterprises for extended periods of time. From the customers' perspective, application consumption interruptions due to cyber attacks can lower their overall Quality of Experience (QoE) and can lead to loss of trust, or in worst cases, the termination of cloud-hosted application provider services.

Different from DDoS attacks, APTs are a form of attacks that are characterized by computer viruses/trojans/worms, which hide on network devices (personal computers, servers, mobile devices). The nature of APT attack behavior is to exfiltrate data from within the network, to devices outside the network. While DDoS attacks are large scale and forthright with a goal of obvious disruption, APTs are quite the opposite, subtle and secretive while also ranging from small to large scale attacks. The aim of the long-term attack is to go unnoticed for as long as possible so that maximum exfiltration can occur. Many APTs will attempt to exploit both Zero-Day attacks (faults in software which have not been discovered by the application developers or hardware vendors and can be exploited) as well as human error (e.g., the curiosity of finding a flash drive in a parking lot, taking it, and attempting to use it). A combination of these methods are also used for initially breaking into an application system as well as spreading through an enterprise infrastructure [4].

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" [5] [6] 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. Threat intelligence collection and corresponding analytics can be developed to block malicious 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.

The above defense strategies in SDxI-based cloud platforms could also be applied to defend against APTs, however they pose a different set of challenges. Since the APTs attempt to be stealthy and commonly use Zero-Day attacks, it is difficult to detect them with existing IDS solutions. Many of these attacks go unnoticed for years, such as Red October, which was active for over five years [7]. With such long lasting and subtle attacks, new threat intelligence collection methods and corresponding analytics technologies are needed to detect APT related attacks quickly and defend against them before any further long term damage or exfiltration can be accomplished.

In this paper, 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 targeted attacks such as DDoS attacks and APTs launched against high-value services hosted in SDxI-based cloud platforms. Our Dolus approach is novel owing to a scalable and collaborative defense strategy which use foundations from *pretense theory in child play* [8] [9] along with SDxI-based cloud platform capabilities for: (a) elastic capacity provisioning via 'quarantine VMs', and (b) SDxI policy coordination across multiple network domains. Such a strategy is aimed at preventing the disruption of cloud-hosted services (i.e., Loss of Availability) and/or the exfiltration of data (i.e., Loss of Confidentiality) by deceiving the attacker through creation of a false sense of success, and by allowing the attacker to believe that a high-value target has been impacted or that high value data has been accessed or obtained.

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).

Dolus uses an automated defense strategy that we developed to mitigate APT attacks, which we refer to as Automated Defense against Advanced Persistent Threats (ADAPTs). Our goal in ADAPTs design is to detect which devices may be infected by an APT, by pursing tracking for data exfiltration outside of an enterprise network. Once a device is suspected of being infected by an APT, the device's traffic can be rerouted so that it does not leave the enterprise network, but can instead be analyzed to determine what is being exfiltrated or what has been compromised. In order to detect possible APTs and identify systems, which have been compromised by an APT, we use a concept called *Suspiciousness Scores* [10]. A Suspiciousness Score is assigned to each device on or off the network. Each device will be assigned a score which is calculated based upon its total number of unique destinations contacted, total number of connections, and total number of bytes transmitted. Using these scores we create a baseline for the entire network. Consequently, devices which are 'suspicious' will stand out with higher scores. Suspiciousness Scores are calculated for internal devices, external devices and domains. Consequently, an external device or domain, which we find to be suspicious can later be blocked from devices on the internal network.

In addition to the baseline scoring for individual devices and the overall network, we also introduce a novel concept of *Targeted Suspiciousness Scores*. Such a scoring overcomes limitations in effectiveness of tracking suspiciousness while accurately detecting a variety of attack traffic with unique characteristics. E.g., we can differentiate APT attacks involving malicious data exfiltration from their variants such as Advanced Persistent Miner (APM) attacks (also referred to as cryptojacking attacks), which involve malicious resource exfiltration by infecting user devices with miners such as CoinHive and geth for cryptocurrency mining purposes [11].

We evaluate the efficacy of our Dolus using two GENI Cloud [12] testbeds, one for DDoS detection and the other for APT attacks detection. The DDoS detection 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. Our experiment results demonstrate the real-time capabilities of our Dolus system 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) without affecting the (benign) cloud users/customers. The APT detection testbed with ADAPTs is similar to the DDoS detection testbed but features dynamic traffic manipulation, monitoring, and analysis to calculate Suspiciousness Scores for each device on the network.

We also present experiment results for targeted suspiciousness based detection of APM attacks before and after whitelisting in order to evaluate the detection accuracy effectiveness. Lastly, we demonstrate 'defense by pretense' traffic characteristics in comparison to normal traffic characteristics to show our pretense novelty. Specifically, we show a resource allocation 'fading pretense' defense mechanism, where resource limitations are emulated in order to diminish the value of the compromised resources in the middle of an APM attack. Our findings show how the defense mechanism eventually influences an attacker's decision to abandon the resource being exfiltrated when its money-making potential drops notably.

Another testbed development contribution that is used in our evaluation experiments is an Administrative User Interface (Admin UI) which we developed for a network administrator to defend against targeted attacks. The Admin UI acts as a central analytics hub for defense against both types of targeted attacks considered in this paper. The Admin UI informs the administrator of total bytes being transmitted through each switch connected to the controller. The administrator can also update policies dynamically and on-the-fly, thus allowing for customization of the network data flows allowing for human control of any data flowing through the network. The Admin UI also displays suspiciousness scores for each device on the network, as well as overall network suspiciousness. The administrator can then use this information to determine if a device has cause for closer investigation to determine if an APT exists on the device or if the device has been compromised by other means.

The remainder of this paper is organized as follows: In Section 2, we discuss related work. Section 3 provides an overview of the Dolus System design. In Section 4, we provide detailed description of Dolus defense methodology against DDoS attacks. Section 5 details our Dolus strategy for defense against APT attacks. Section 6 evaluates the performance of Dolus system in GENI Cloud testbeds. Section 7 concludes this paper.

2. Related Work

2.1. Attack Defense using Trickery

There have been efforts that seek to implement defense mechanisms using some form of ‘trickery’ to engage an attacker. For example, authors in [13] introduce the notion of tricking the attackers through IP randomization methods in decoy-based MTD efforts. In contrast, the notion of pretense in our Dolus approach is akin to Honeypots and Honeynets which are effective in gaining information about possible attacks based on minimal active interactions with attackers [14]. 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 quarantine VMs. Dolus system’s pretense theory is mainly built upon the work in [8] and [9] belonging to the field of child pretend play psychology. Our novel *defense by pretense* mechanism for effective mitigation of targeted 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 targeted attack.

2.2. Defense against DDoS Attacks

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) [15] [16] [17]. 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 [18] 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 [19], which can then be used to "scrub" the flows through scrubbing SaaS services as a low-cost solution [20].

A number of other network-based defense strategies have been proposed in efforts that involve analysis of traffic and dynamic updation of rules to effectively reroute malicious traffic. Such efforts include [21], where a network reacts to targeted attacks using accountability and content-aware supervision concepts. Similarly, using volume counting, authors in [22] provide a DDoS defense mechanism that involves monitoring SDN traffic flows in OpenFlow-enabled switches. In the context of programmability of SDN switches to mitigate targeted attacks, authors in [23] present a programming framework. In another similar effort, authors in [24] propose a memory-efficient system that uses Bloom filter and monitoring tools to dynamically update SDN rules to mitigate DDoS attacks. Also leveraging the dynamic rule update feature of SDN, authors in [25] analyze the probability that a flow is traced back across multiple ASes' hops by sampling the probability and the analyzing signatures of attack traffic flows.

Alternately, cloud service providers allow mitigation of DDoS attacks by utilizing the elastic capacity provisioning capabilities in the cloud platforms that allow "moving target defense" (MTD) techniques to be implemented. MTD basically involves replication and live migration [26] of compromised application services (with pre-attack state information) in new VM(s) to redirect legitimate users, and keep attackers in a quarantine VM(s) [6]. As an added defense strategy, authors in works such as [27] present a survey of SDN-based mechanisms to detect attacks closer to the attackers/attack sources.

2.3. Defense against APT Attacks

Techniques to detect APTs have been of interest to the community [28, 29, 30, 30, 10, 31, 32, 33, 34, 35, 36, 37]. This includes finite angular state velocity machines and vector mathematics to model benign versus attack traffic, allowing a network operator to easily view the differences [34], assesing the outbound network traffic [10, 31], using honeypots [38] and using distributed computing [37]. Another APT detection technique is based on a ranking system where all internal hosts are ranked based on number of bytes sent outside the network, number of data transfers initiated to an entity outside the network, and number of distinct destinations contacted outside the network per host [10]. Yet, another APT detection technique is to monitor attack traffic using a detector in an enterprise network [35].

Potential countermeasures against APTs are discussed in [4, 39, 40, 41, 42, 43, 44, 45]. Defense strategies include: (a) running routine software updates to avoid backdoors, bugs and vulnerabilities; (b) strengthening network access control and monitoring services; (c) enabling strict Internet access policies; and (d) dropping encrypted traffic from unknown hosts. Similarly, authors in [4] discuss a number of counter measures against APTs including training users about social engineering attacks, blacklisting hosts, dropping packets, etc. Futhermore, SDN-based defense [43] involves: (i) defining and maintaining a network baseline to identify any deviation from the baseline through analytical tools, and (ii) updation of flow policies for (re)directing and blocking traffic in any of the network segments. A framework that realizes such an SDN-based defense is discussed in [44]. In a similar vein, authors in [45] provide a sandbox environment using which a security professional can emulate the propagation of APTs across an enterprise network environment.

3. Dolus Defense Methodology

In this section, we first present an overview of pretense theory. Following this, we describe how the pretense theory is used in our Dolus system design.

3.1. Pretense Theory

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 [8] motivate us to address the issue of how a cognitive agent can

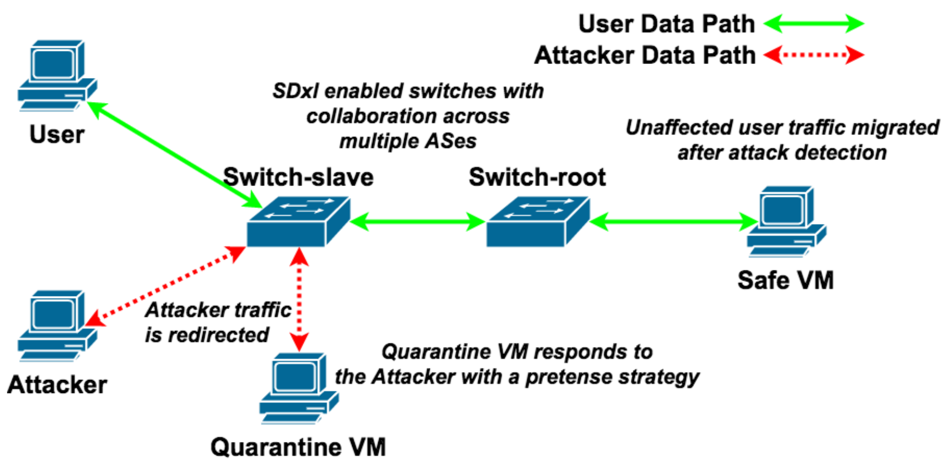


Figure 1: Illustration of the proposed Dolus system scheme wherein the attacker is *tricked* by redirection of the attack traffic to a quarantine VM for pretense initiation, while the providers work collaboratively to block the attack traffic closer to the source side.

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 [9], 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.

3.2. Pretense in Dolus System Design

In our Dolus system, effective pretense design methodology is illustrated in Figure 1. We create 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 (explained in Section 4.2) can identify and then blacklist the attacker flows while allowing benign user flows to continue unimpeded. This in turn could help in keeping an attacker distracted for a brief period of time while 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 [46] 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.

Table 1: Objectives for the pretense zero-sum game considered in the Dolus System design.

Objective	**Attacker**	**Defender**
Goal	Evade Defender's detection	Protect attacker's target(s)
Trick	Defender to provide access	Attacker to reveal presence
Time	Mislead defender to spend time on false positives	Mislead attacker to spend time on true negatives
Outcome	Make defender believe that an attack is simple	Make the attacker believe that the target attack is successful
Attribution	Hide attackers' identity	Induce attackers to believe that their identities are unknown

The goal of our Dolus approach is to model the notion of pretense as a zero-sum game. Specifically, a zero-sum game is one in which the sum of the individual payoffs for each outcome is zero. That is, (1) loss to an attacker is gain for a defender and vice versa, and (2) total sum of gain and loss is (roughly) zero. There are two strategies to play a zero-sum game, one from an attacker's perspective and the other from the defender's perspective. Specifically, we plan to employ the following two strategies: (a) `minimax` strategy: minimizing defenders own maximum loss (from defenders perspective), and (b) `maximin` strategy: maximize attacker's own minimum gain (from attacker's perspective). We explore these two strategies in our Dolus system on a number of objectives, which are summarized in Table 1.

Our guiding strategy for targeted attack defense using pretense is to use a form of *pretense machine learning* which we propose to be understood as - *If you don't know the enemy and don't know yourself, then you will succumb in every battle. If the attacker does not know you but you know the attacker, for some victories gained you will suffer some defeat. If the enemy knows you and you know yourself, you need not fear the result of a hundred battles*.[2]

For our defense by pretense strategy, we consider multiple vectors: (1) awareness of the attack surface, i.e., cloud/physical topology aware; (2) behavior and/or psychological aspects of the attacker, i.e., data science and pretense theory to understand an attacker's desire and to identify an effective countermeasure; (3) development of theory and algorithms to deceive the attacker into a false sense of success *without* affecting the network resources; and (4) sharing multi-domain threat intelligence across SDxI entities.

4. DDoS Attack Defense with Dolus

In this section, we first describe the DDoS attack model that we assume to design our Dolus defense. Subsequently, we detail our DDoS defense solution that uses a 'defense by pretense' scheme in the Dolus system.

4.1. 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 from multiple attack points. With cloud-hosted applications with large monetary value becoming highly common, DDoS attacks can cause LoA for users and 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 [47] and ICMP/Ping [48] flooding. Such attacks typically inundate a

[2] A quote modified from "The Art of War" by Sun Tzu.

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` [49] and `SlowHTTPTest` [50] tools, respectively. Furthermore, to capture the dynamics of an attacker, we randomly change the number of attack packets emitted by these tools.

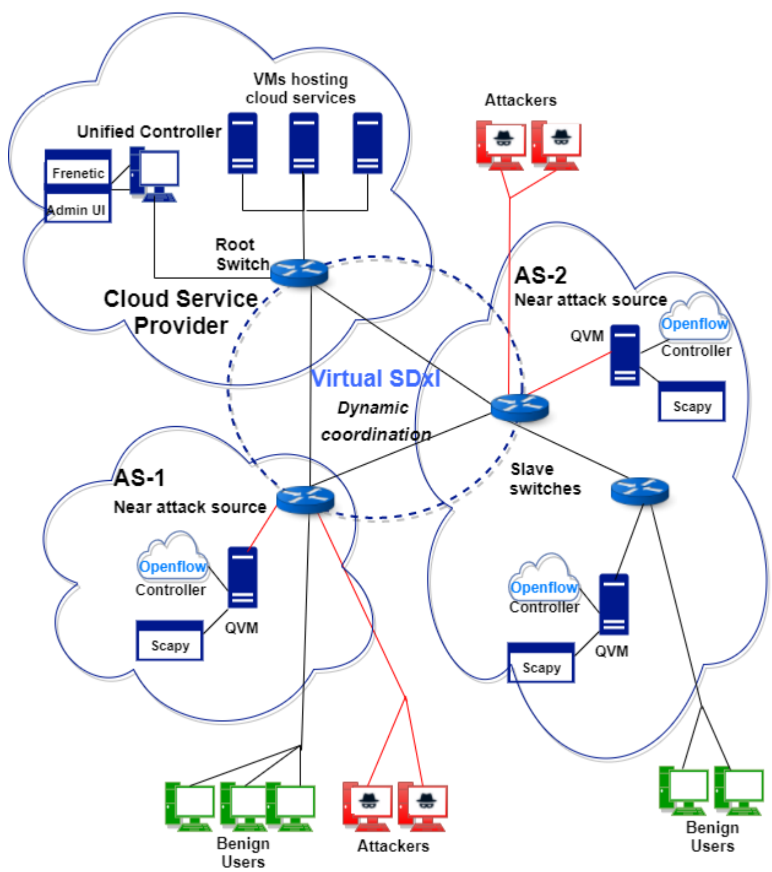


Figure 2: Cross-domain physical setup in a Dolus system deployment to share threat intelligence for a unified controller to coordinate policy management with a federation of ASes to block attack traffic closer to the source side.

4.2. *Defense by Pretense Scheme*

Figure 2 depicts the cross-domain setup in a Dolus system deployment to implement a defense by pretense scheme. To complement Figure 2, interactions between different phases of a Dolus system configured for spoofing pretense are shown in Figure 3 and Algorithm 1, respectively.

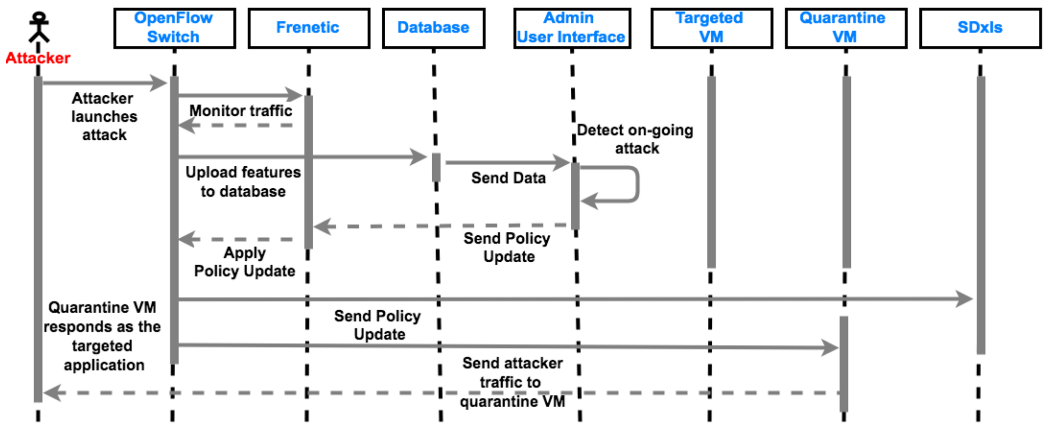


Figure 3: Sequence diagram of the Dolus system interactions for attack detection, quarantine setup, pretense initiation/maintenance and DDoS attack impact mitigation.

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 [46]-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 [51, 52] that 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: $\mathcal{H} = \{h_1, h_2, ..., h_n\}$ 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 [53].

In our case, to achieve higher accuracy with a minimum size of the training dataset D, we use the Bayesian voting scheme [54] as the ensemble method to predict the result for new data x, which can be represented as Equation 1.

$$\mathcal{F} = \sum_{h \in \mathcal{H}} h(x) P(h|D) \tag{1}$$

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 (2).

$$P(h|D) \propto P(h)P(D|h) \tag{2}$$

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 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).

Quarantine Setup for Pretense. 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 [55] 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).

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 threat 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 maintain it for too long, attacker might potentially infer our pretense.

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.

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 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 (Admin UI) module and a unified controller module that can be customized in a Dolus system instance deployment depicted in Figure 2. The Admin UI shown in Figure 4 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 i.e., 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.

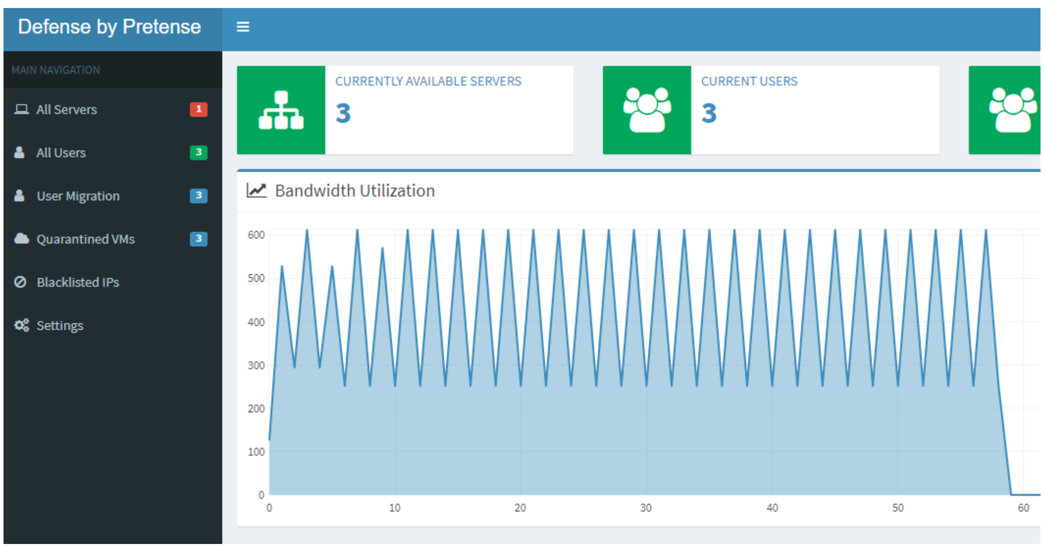


Figure 4: Administrator User Interface of an Dolus system instance.

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.

Algorithm 1: Dolus system defense algorithm against DDoS attacks

Input: $attacker\_ID$ = attacker ID,
src_ip = source IP,
dst_ip = destination IP,
$no\_of\_packets$ = number of packets,
$spoof\_dst\_ip$ = spoofed IP,
$black\_ip$ blacklisted IP list

Result: Attack traffic will be redirected to the quarantine VM and DDoS blocking policy will be generated

```
function initQuarantine()
    createVM();
    updatePolicy(src_ip);
    do
        redirectTraffic();
        pretense_data = generateUsingScapy();
        vmResponse(spoof_dest_ip, src_ip, dst_ip, pretense_data);
    while timeout == false;
end
function updatePolicy(src_ip)
    logAttackTraffic();
    new_policy = generateNewPolicy();
    collaborate(new_policy);
end
function collaborate(new_policy)
    advertisePoliciesToNeighbors(new_policy);
    black_ip = updateList(src_ip);
    redirectTraffic();
end
function redirectTraffic()
    sendTrafficToQuarantineVM();
end
function main()
    /* Receive incoming data from external machine */
    data = monitorPackets(attacker_ID, src_ip, no_of_packets, start_time,
    end_time);
    attack = twoStageEnsembleLearning(data);
    /* Update policy in case of attack detected */
    if attack == true then
        initQuarantine(src_ip);
    end
    decideToStopOrContinue();
end
```

5. APT Attack Defense with Dolus

5.1. Attack Model

APTs are long-term attacks and affect a target in four stages: *preparation, access, resident, and harvest* [7, 56, 57, 58, 59, 60, 61, 39]. In the preparation stage, attackers apply a reconnaissance tactic through social engineering (e.g., via social networks) to bootstrap the attack [4]. Once the attack is bootstrapped, attackers identify a vulnerability, and/or a vulnerable target and send malwares either through email (e.g., spear phishing) or through third-party software/service (e.g., watering-hole attack) in the access stage. Subsequently, the malwares establish external communication paths with attackers' Command and Control (C&C) server(s), and spread across other targets in the resident stage; which is a slow and a stealthy phenomenon. Finally, in the harvest stage, attackers extract any vital information in an on-going fashion for extended periods of time.

5.2. Defense by Pretense Scheme

Our novel Dolus system with ADAPTs is designed to automatically defend against APT attacks. Its design is similar to the original Dolus system algorithm (i.e., Algorithm 1) for DDoS attack defense described in Section 5, however the threat intelligence collection and defense are adapted towards mitigation of APT attacks. More specifically, ADAPTs consists of: (1) a Suspiciousness Score-based detection mechanism, which is robust against the threshold evasion problem; (2) internal quarantine VMs (iQVMs), which are a minimal version of honeypots to mimic hosts internal to an organization, along with performance/topology views to aid network administrators; (3) a coordination mechanism driven by enterprise defense policies to share threat intelligence about APTs among hosts; and (4) network policy update mechanism to mitigate attack spreading based on coordinated intelligence using iQVMs. We outline each one these mechanisms/components in the remainder of this sub-section.

Attack Detection. Inspired by the work of authors in [10] to identify hosts exhibiting suspiciousness in a network, we propose a Suspiciousness Score (SS) in a similar vein. We calculate SS based on captured network traces (.pcap) using three main features: *destinations (dst)*, *flows*, and *bytes*.

Table 2: Features captured from a network trace for APT attack defense analytics.

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

Table 2 shows the list of values/features captured in network traces for APT attack defense analytics. For each packet trace, a trace_id *t* is assigned. For each *t*, we perform the following: the features are normalized and their combined Root Mean Square Error (RMSE) values are calculated. Using the RMSE values, we calculate the Suspiciousness Scores of each device as follows. The *Min* and *Max* values (below) are assumptions made per device type regarding what one may expect the minimum and maximum values to be on the type of device, network and traffic expectations. These values are determined by the system/network administrators and could vary vastly depending on each ASeS or domain's threat monitoring objectives.

Destination suspiciousness for trace *t*:

$$dst_i = w_{dst} * \frac{numDst_i - numDistMin_i}{numDstMax_i - numDstMin_i}; w_{dst} \in [0.0, 1.0] \tag{3}$$

Flow suspiciousness for trace *t*:

$$flows_i = w_{flows} * \frac{numFlows_i - numFlowsMin_i}{numFlowsMax_i - numFlowsMin_i}; w_{flows} \in [0.0, 1.0] \tag{4}$$

Bytes suspiciousness for trace *t*:

$$bytes_i = w_{bytes} * \frac{numBytes_i - numBytesMin_i}{numBytesMax_i - numBytesMin_i}; w_{bytes} \in [0.0, 1.0] \tag{5}$$

Device suspiciousness for trace *t* is based on equations 3, 4 and 5 as shown below.

$$ss_i = \sqrt{\frac{dst_i^2 + flows_i^2 + bytes_i^2}{3}} \tag{6}$$

Note that for each device on the network *i* we calculate a Suspiciousness Score and the overall network suspiciousness for trace *t* is calculated based on ss for each individual device (equation 6) that is connected. That is, the sum of all ss for each devices on the network *n* is the *overall network suspiciousness SS* for that particular *t*.

$$SS_t = \sqrt{\frac{(ss_1^2 + ss_2^2 + ss_3^2 + ... + ss_n^2)}{n}} \tag{7}$$

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

$$\Delta ss_{i_t} = \frac{ss_{i_t} - \sqrt{\frac{ss_{i_1}^2 + ss_{i_2}^2 + ... + ss_{i_{t-1}}^2}{t-1}}}{\sqrt{\frac{ss_{i_1}^2 + ss_{i_2}^2 + ... + ss_{i_{t-1}}^2}{t-1}}} \tag{8}$$

In equations 3, 4 and 5, we assume the weight parameters i.e., w_{dst}, w_{flows} and w_{bytes} to be equal to 1 in a general case of SS calculations. As shown later through experiment findings in Section 6, assigning suitable weights for a variety of suspicious traffic can minimize the RMSE in the attack detection accuracy, as opposed to the general case. Consequently, we extend SS calculations as detailed in Algorithm 2 by introducing a novel concept of Targeted Suspiciousness Scores for specific traffic types that a system/network administrator would like to classify as suspicious. We remark that system/network administrators could whitelist certain traffic types, however none of the devices on the network will be entirely whitelisted. Thus, by using a targeted suspiciousness scoring for certain traffic types (e.g., for suspected APM-like traffic) can still be effective to detect malicious activities, even when whitelisting is performed for legitimate user traffic.

Quarantine Setup for Pretense. VMs which are internal to an organization and which implement minimal versions of honeypot-like hosts are internal quarantine VMs (iQVMs), whose Suspiciousness Scores are monitored continuously. These are also the hosts that play the game of pretense i.e., they create a false notion of *high-value targets within an organization with sensitive data* to the external world. An attacker is lured to attack iQVMs first; they maintain pretense by sending data similar to what a host with sensitive data would send. Apart from monitoring the

Algorithm 2: Targeted Suspiciousness Score Calculations

Input: $devices \in [1 \ldots n]$ = array of all devices on the network,
$maxTrace$ = Maximum number of packet traces to be evaluated,
t = current trace being evaluated

Result: Targeted Suspiciousness Scores calculated for each network device after traffic analysis

function $calcDst()$
function $calcFlows()$
function $calcBytes()$
function $calculateTargetedSuspiciousness(device_i)$
    $calcDst(device_i, w_{dst})$;
    $calcFlows(device_i, w_{flows})$;
    $calcBytes(device_i, w_{bytes})$;
end function $main()$
    do
        for each $device_i$;
        $calculateTargetedSuspiciousness(device_i)$;
    while $t <= maxTrace$;
end

data sent out of iQVMs, they also add weights to the calculated Suspiciousness Scores, overcoming the threshold evasion problem.

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

1. SS view: `Flot.js`-based bar and line graphs as depicted in Figure 5 excerpted from the Admin UI. SS per device or for the overall network can be viewed in temporal fashion as shown in Figure 6 extracted from the Admin UI. Moreover, when a suspiciousness score of a blacklisted device is shown to be above a certain threshold, an administrator can block all traffic from that device to the network or take an appropriate action.

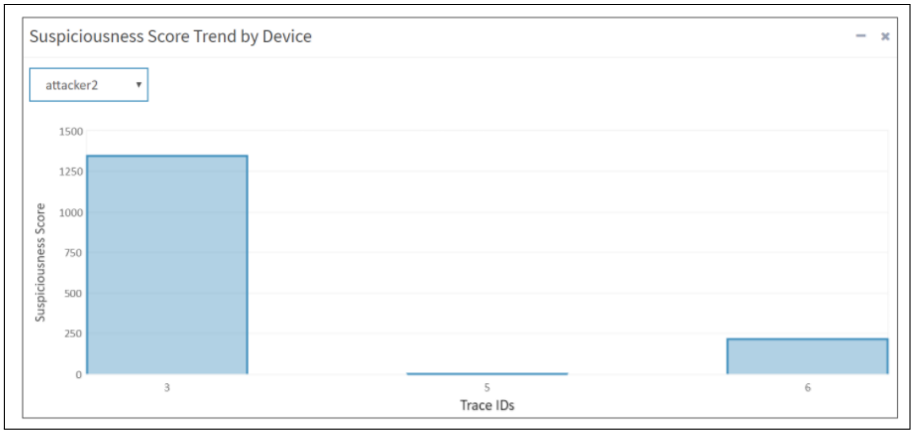


Figure 5: Suspiciousness score per device over time.

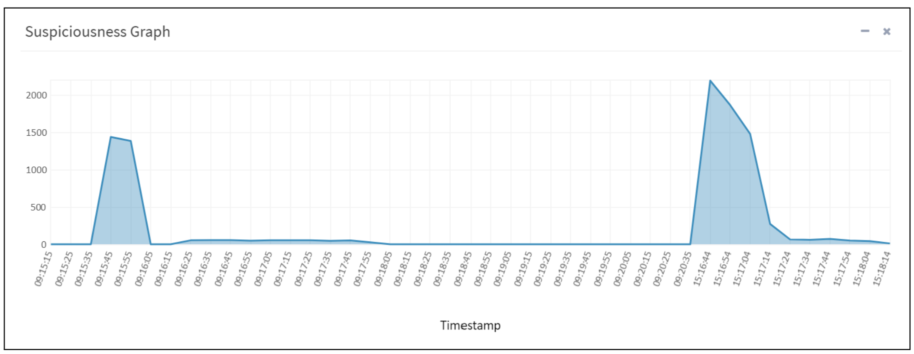


Figure 6: Overall network suspiciousness score over time.

2. Upload policy view: This view on the user interface enables administrators to push NetKAT-based policies [62] to a centralized database, which stores device configurations, thresholds, policies maintained by the organization. Interfaces are provided to select a specific device and a corresponding NetKAT policy to affect that device as shown in Figure 7.

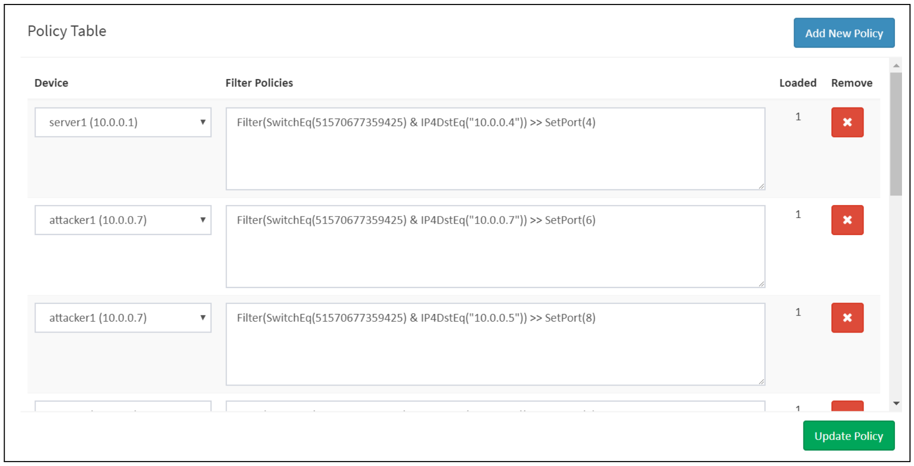


Figure 7: Policy table view.

3. Network view: a `vis.js`-based view to monitor the network as a graph of connected devices as depicted in Figure 8.

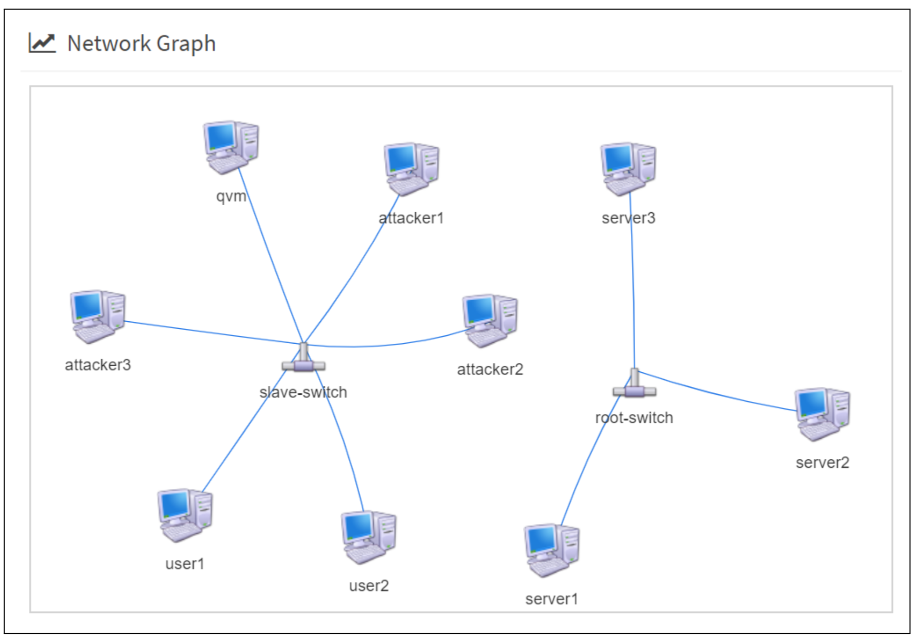


Figure 8: Network graph of all the connected devices.

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

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

Both these policies and the revocation/enabling functionalities are instantiated based on the policy updater mechanism, whose main objective is to simplify the learning curve for users/administrators to get proficient at writing policies (e.g., using network programming languages such as Frenetic [46])—a daunting and tedious task. With this in mind, the updater component can auto-generate policies based on simplified inputs that are provided via the user interface. For example, to minimize the process, the policy updater takes a generic command such as "user1 to server1" and all possible configuration policies would be generated by the updater. The updater works with the centralized database and is pre-programmed with the network architecture.

One of the advanced defense policies set by a system/network administrator could be a 'defense by pretense' policy. Such a policy can adopt our novel pretense concept that is adapted to a particular attack threat. Herein, we provide an example of one such policy viz., 'fading pretense' that can be used to defend against APM attacks. Invoking the defense policy on a compromised device essentially results in a situation where the defense mechanisms simulates resource limitations. Such a resource limitation diminishes the value of the compromised resources in the middle of an APM attack. This is because, by limiting the resource allocation through a hypervisor on a device that is part of a mining pool, the ability in terms of the computation speed of a miner software on that device to mine a single block (within a blockchain) can be slowed down. When the computation speed goes below a threshold, the overhead of communicating with this compromised miner device in a mining pool becomes excessively large, which in turn impacts the revenue for the attacker [63]. Eventually, such a scenario will influence an attacker to abandon the resources exfiltration on that device, and move onto other more-potent compromised devices with higher resource allocations. A realistic demonstration of the implementation of the 'fading pretense' policy is provided later in Section 6.

Threat Intelligence Sharing. Our iQVM monitors also coordinate and share the APT threat intelligence such as SS thresholds, policy updates, etc. with other hosts in the network. Apart from providing a collaborative environment amongst pertinent hosts to effectively counter APTs, the mechanism also provides a way to drill down on specific segments of the network with suspicious hosts. Furthermore, we believe that the coordination mechanism will pave the way to achieve a global and unified hardening of the enterprise network against APTs. In addition, any sensitive data sent out is money lost in a business; sharing the threat intelligence in turn provides a cheaper alternative to lost data and host/business downtimes.

6. Performance Evaluation

In this section, we describe the evaluation of our Dolus methodology in GENI Cloud testbeds for DDoS and APT attacks. For showing effectiveness of Dolus for each targeted attack type, we start by describing our testbed, followed by the experiments and results discussion. The source code and instructions to replicate below experiments are openly available at [64] [65].

6.1. Dolus Experiments for DDoS Attack Defense

6.1.1. Testbed Setup

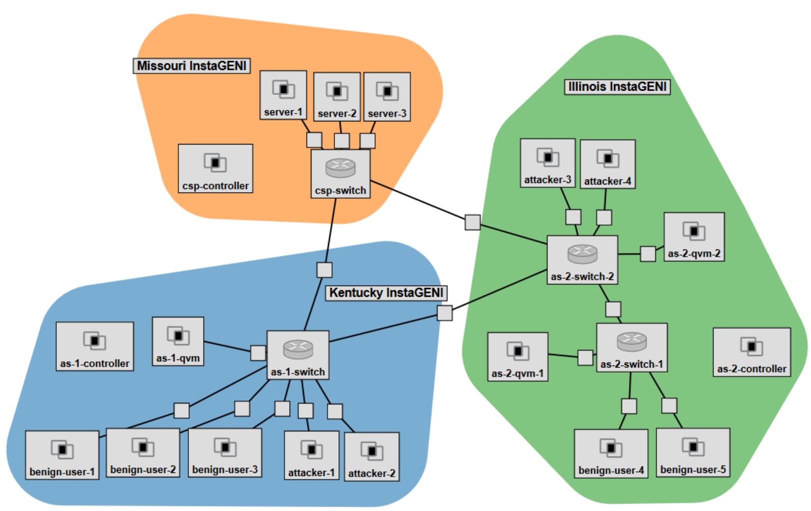
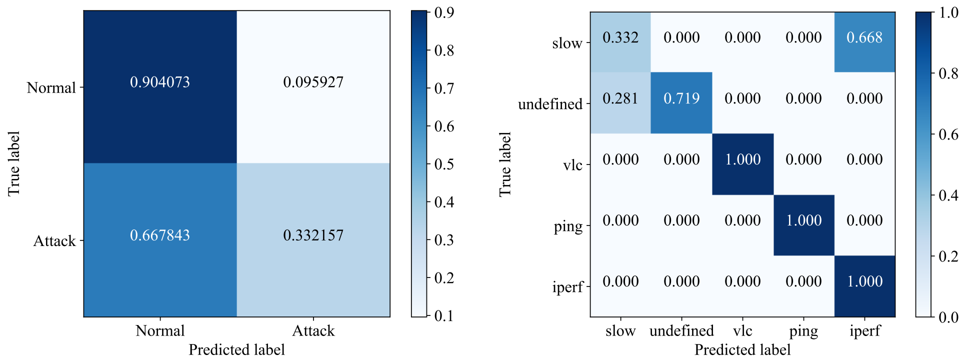


Figure 9: GENI Cloud testbed used to evaluate Dolus for DDoS attack defense.

We evaluate the efficacy of our Dolus system using a realistic, GENI Cloud [12] testbed as shown in Figure 9. The testbed contains three SDN switches, two slave



(a) Attack detection. (b) Attack classification.

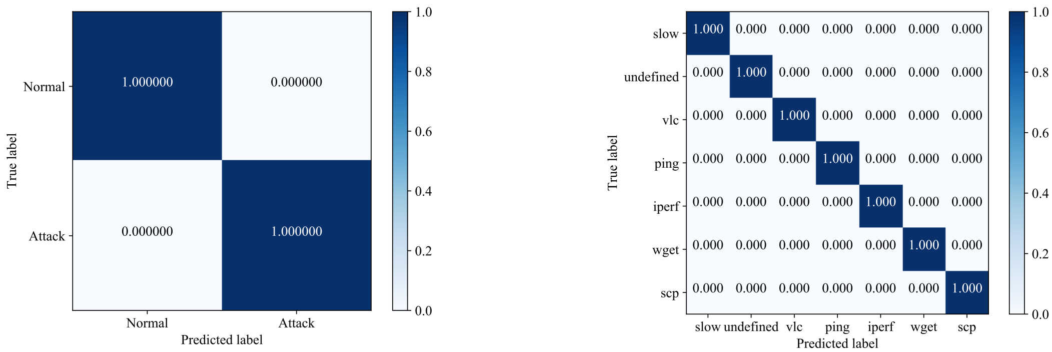
Figure 10: Confusion matrices for attack detection and classification for multiple traffic flows sent to multiple hosts.

switches and a single root switch. Such a system could also be extended to host many more switches and devices. 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. In the following, we show the attack detection and classification accuracy using our two-stage ensemble learning scheme and then 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.

6.1.2. Attack Detection and Classification Results

Using the Dolus system, we monitor different types of data that are permitted to enter the GENI Cloud testbed depicted in Figure 9. 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. Our evaluation results span 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 ∼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 15 shows the two confu-



(a) Attack detection.

(b) Attack classification.

Figure 11: Confusion matrices for outlier detection and classification for multiple traffic flows comprising of familiar attack flows.

Table 3: 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

sion 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 3, 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, 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 ensemble learning scheme. Figure 10 shows the two confusion matrices in normalized form for attack detection and classification. Detection and classification of attack took $\sim$7 and $\sim$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

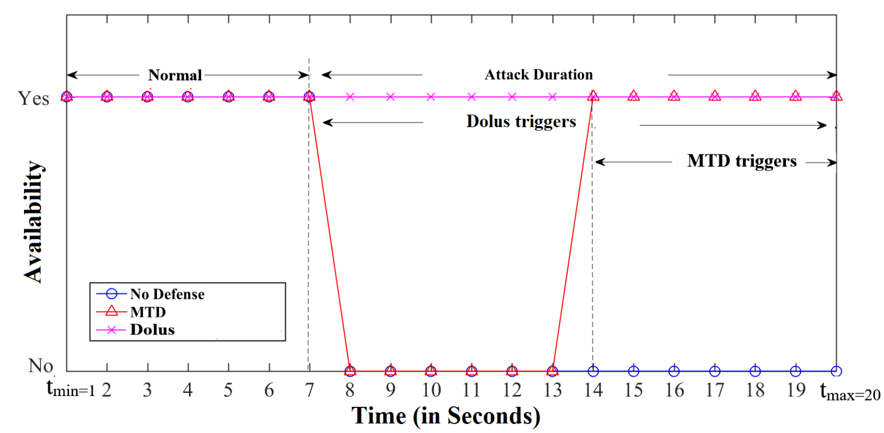


Figure 12: Comparison of the *cloud service restoration time* metric with cases of: no Defense, with MTD and with Dolus.

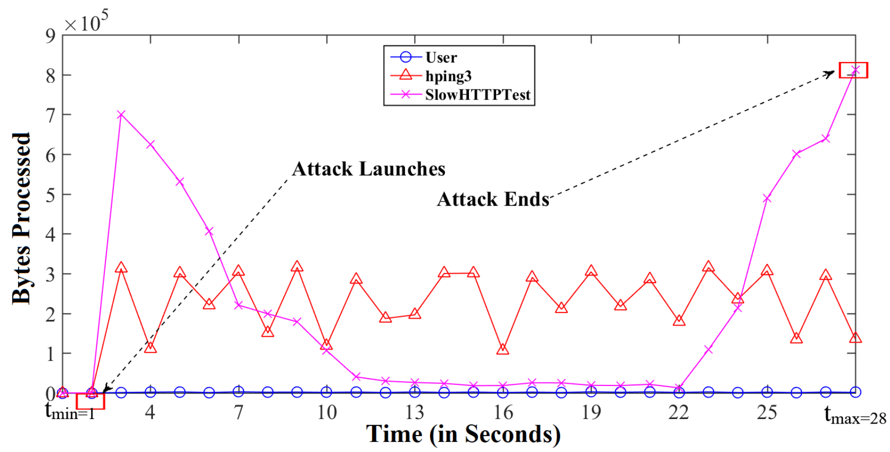


Figure 13: Traffic processed (in Bytes) in one of the slave switches.

Table 3.

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 cased in Figure 11, where we make use of 60% of the data as training data and 40%

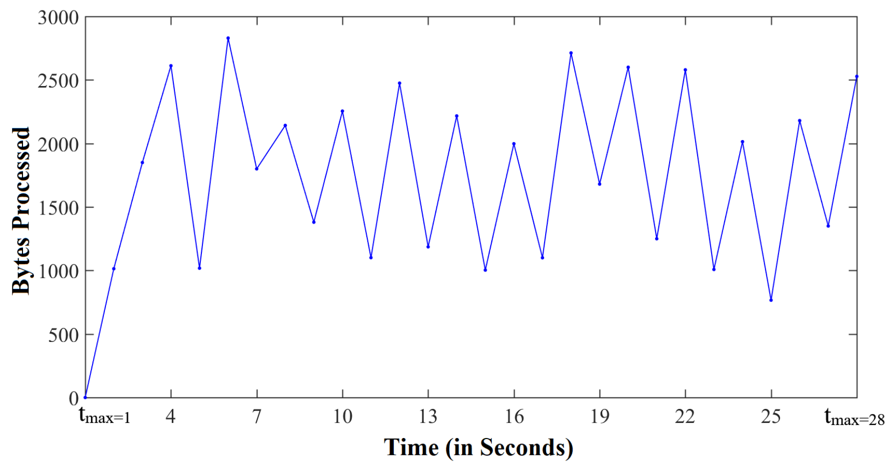
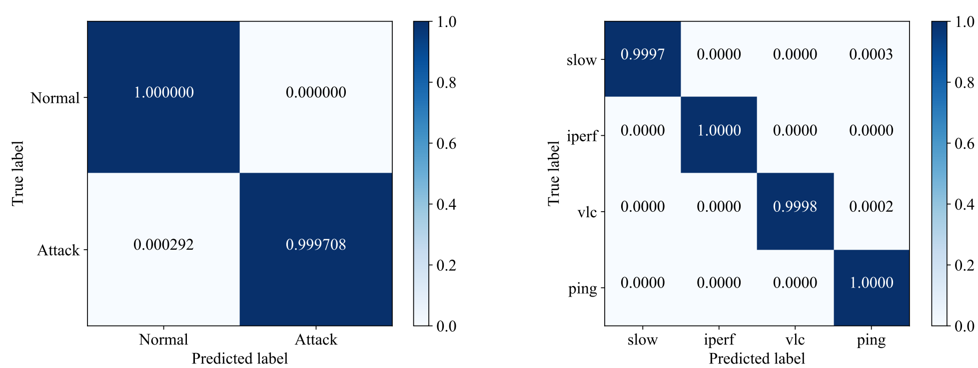


Figure 14: Traffic Processed at the root switch only shows user traffic proving that the attack traffic is redirected to quarantine VM.



(a) Attack detection.

(b) Attack classification.

Figure 15: Confusion matrices for attack detection and classification for multiple traffic flows sent to a single server.

as test data for the same dataset used in the 2^{nd} 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.

6.1.3. Time to Restore a Cloud-hosted Application Service

Figure 12 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 $7^{th}\ second$ from the attackers. In a SDxI-based cloud network with no defense strategy, the services are immediately affected by the attack traffic. Consequently, an absence of service availability after the $7^{th}\ second$ as shown in the graph results in a situation where cloud service restoration does not occur. MTD-based defense strategy is able to restore the service after taking $\sim 6\ seconds$ 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.

6.1.4. Amount of Traffic Processed at the Root Switch

Figures 13 and 14 depict the amount of traffic processed (in Bytes) at one of the slave switches and the root switch. From Figure 14, it is evident that the SDxI-based cloud network is oblivious to the attack traffic impact, complementing the result in Figure 12. Since the slave switch represented in Figure 14 redirects attack traffic to the quarantine VMs, we observe a $5X$ increase in the amount of traffic processed in comparison with the root switch.

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.

6.2. Dolus Experiments for APT Attack Defense

6.2.1. Testbed Setup

For the purposes of APT attack detection and defense, a modified GENI Cloud testbed was setup as shown in Figure 16. The purpose of the GENI Cloud Testbed is to simulate a collaborative cross-domain SDxI architecture with the core servers and services located at the switch-root in the Clemson InstaGENI (blue) domain. Correspondingly, the user traffic originates from three other separate domains with two distinct paths to where the services are located. Since an APT is not a distributed attack, there was no need to consider multiple attack vectors from many directions. However, due to the nature of an APT attack being secretive and stealthy, we assume that an APT can be hiding anywhere in an SDxI. Our testbed is comprised of multiple open vSwitches (a slaves and a single root), numerous nodes (which are hosts), and a controller. The slave switches connect all the user nodes, and the root switch connects all the servers hosting the application system and related services to the slave switches. The controller in the setup is a standalone node, running the monitor and policy updaters, calculating SS thresholds for nodes and the overall network, managing all the traffic and defense by pretense mechanisms of the Dolus system.

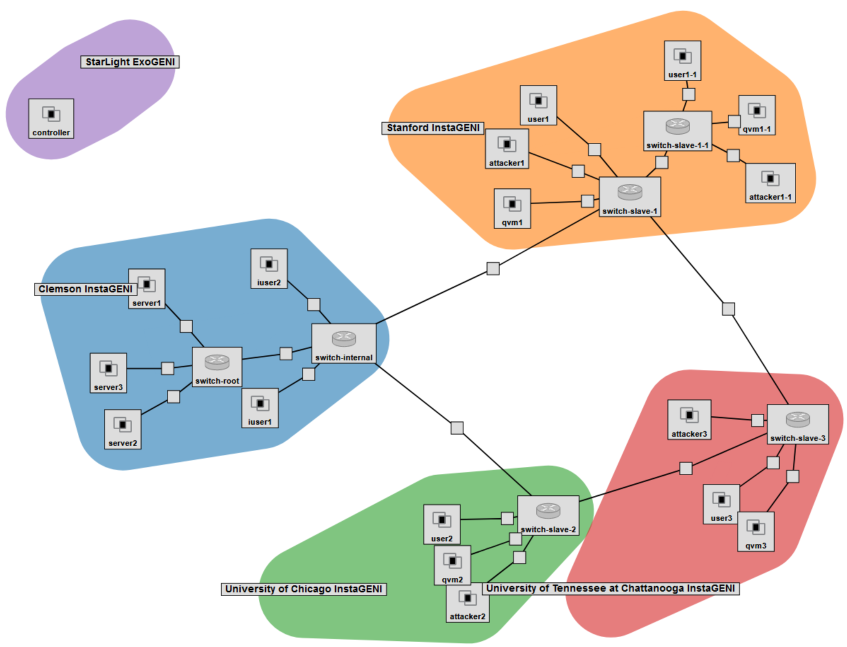


Figure 16: GENI Cloud testbed used to evaluate Dolus for APT attack defense.

6.2.2. Suspiciousness Score Calculation Results

In the first experiment, we randomly selected three hosts, and compromised them by running `slowhttp` attacks from attacker 1 and attacker 3, and a secure

Table 4: Suspiciousness Scores before Whitelisting

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

copy (`scp`) from attacker 2. This configuration allows us to compare the suspiciousness between a DDoS attack, and a file exfiltration attack. Before running the experiment, we specify minimum and maximum values for flows, connections, and bytes: the user and attacker nodes are each set to a minimum of 1 and a maximum of 10 connections, a minimum of 100 and a maximum of 1,000 flows, and a minimum of 10 and a maximum of 100,000 bytes. The servers had a minimum of 10 and a maximum of 1000 connections, a minimum of 1,000 and maximum of 10,000 flows, and minimum of 100,000 and a maximum of 100000000 bytes.

From the controller, we obtain the SS for these three attackers before (see Table 4) and after (see Table 5) whitelisting. Note that all devices have SS calculated for them, as we don't initially whitelist any devices or traffic on our testbed network. Attacker 2 exhibited the highest SS out of three, due to data exfiltration [10]. The traffic that is being exfiltrated generates a much higher score than the regular traffic in the network.

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

Table 5: Suspiciousness Scores after Whitelisting

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

Furthermore, we consider a case where - if attacker 1 compromised user 1 within our network and then used user 1 to exfiltrate data from server 1 to user 1 then from user 1 to attacker 1. In such a case, we are able to detect the suspiciousness between user 1 and attacker 1 since that is where the actual data exfiltration is taking place. As you can see in Table 5, we ignore the traffic between users and servers, even though there was data moving between them (as seen in Table 4). Moreover, by considering the whitelisting prior to the Suspiciousness Score calculations, we decrease the overall time spent on speed of the calculations since we will need to calculate scores for *only* a portion of the network.

6.2.3. Targeted Suspiciousness Score Effectiveness

As an extension of our overall Suspiciousness Scores calculation, we also performed experiments using our novel Targeted Suspiciousness Scores detailed in Section 5.2. Similar to the previous experiment, we tested a variety of different types of network traffic with the addition of a two new types i.e., BitTorrent and cryptocurrency mining traffic. For generating the cryptocurrency mining traffic, we use a miner software called 'geth' which could be used for CPU resource exfiltration. We also use variants of geth, where we limit the geth mining using various tools for both network rate limiting as well as CPU limiting. Both these variants could mimic the tools used by an attacker attempting to stealthily exfiltrate resources from an enterprise system without being detected. In addition to the cryptocurrency mining attack traffic, we also generated several types of benign, other attack, and suspicious traffic. We expected the BitTorrent traffic in particular to have similar characteristics and behave similar to the cryptocurrency mining traffic, since both use peer-to-peer protocols on distributed systems.

Figure 17 shows the Overall Suspiciousness Score results from the different tests we ran over a ten minute period. We surprisingly found that the Overall Suspiciousness Scores indicate that the cryptocurrency mining traffic is far less

Table 6: Test traffic types for Targeted Suspiciousness Score effectiveness evaluation experiments.

Test Traffic Types	
Devices	**Number of traces**
geth	Cryptocurrency mining
geth	Cryptocurrency mining with trickle (network rate limited to 10 kbps download and upload)
geth	Cryptocurrency mining with cpulimit (limited to 10% of total cpu)
ping	Ping traffic on one of the IP addresses contacted during crypto mining
wget	Traffic of a 121 MB video file download
scp	Traffic of the same 121 MB video file upload
DDoS	slowhttptest traffic against the same IP address used for scp
BitTorrent	Up to 200 connections relating to downloading/seeding Ubuntu 17.10 iso

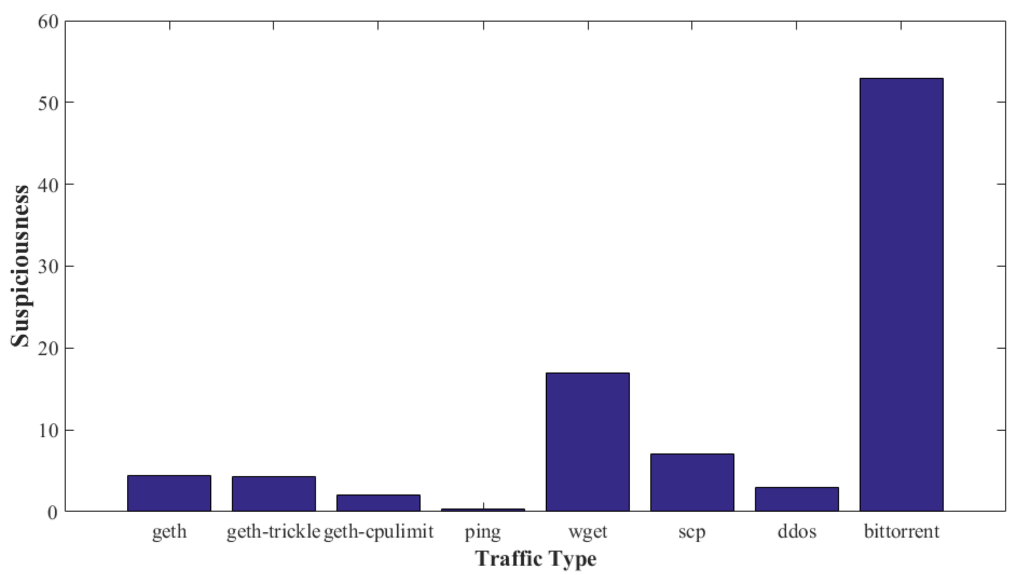


Figure 17: Overall Suspiciousness Score results that motivate the need for using targeted suspiciousness to improve APT attack detection accuracy.

suspicious than the BitTorrent traffic, and even the wget and scp traffic. This is mainly due to the fact that the suspiciousness calculations in the general case were created to detect data exfiltration attacks, and do not account for the resource exfiltration characteristics of a cryptocurrency mining attack. This motivated us to reconsider the use of Overall Suspiciousness Scores, and introduce weights for a variety of suspicious traffic in order to minimize the RMSE in the attack detection accuracy.

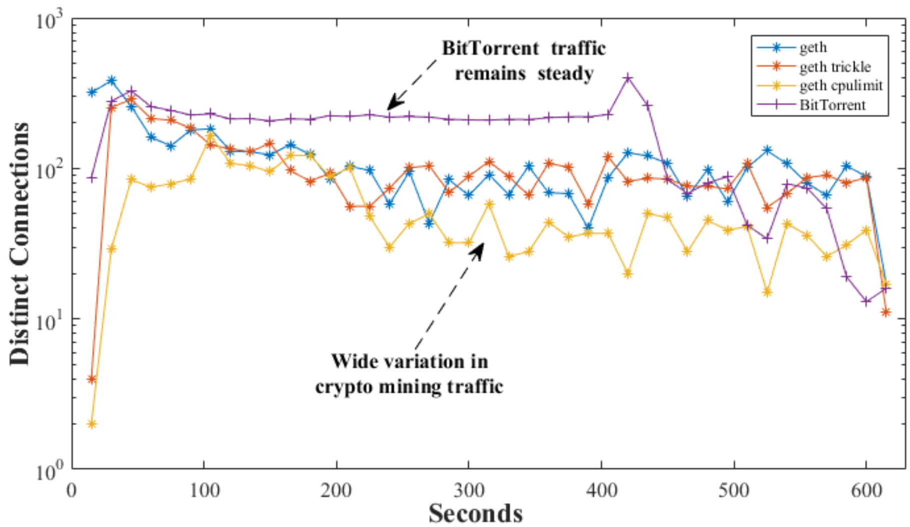


Figure 18: Distinct IP Addresses contacted per every 15 seconds.

Digging deeper as shown in Figure 18, we observed that - even though the Overall Suspiciousness Score for the benign bittorrent traffic was higher than cryptocurrency mining traffic, there is a distinct difference in the traffic variation over time. The cryptocurrency mining traffic was far more varied in total distinct IP addresses contacted at any given period of time. We also saw highly similar results with the total flows over a given time period, and also for the total bytes transmitted over the same time period. This led us to conclude that there is a distinct difference in the variation of traffic over time when comparing BitTorrent and cryptocurrency mining traffic. Specifically, BitTorrrent traffic will feature connections with a large number of IP addresses but will continue to maintain connections with the same number of IP addresses every few seconds with little or no change. In contrast, the cryptocurrency mining traffic will connect with many IP addresses, and the total number of distinct connections at any given period of time will flucutate over a wide range. With the knowledge of such attack traffic features, suitable weights can be assigned in Targeted Suspiciousness Scores that will not trigger BitTorrent traffic on a network as suspiciousness, but will accurately detect unauthorized cryptocurrency mining traffic that are part of APM attacks.

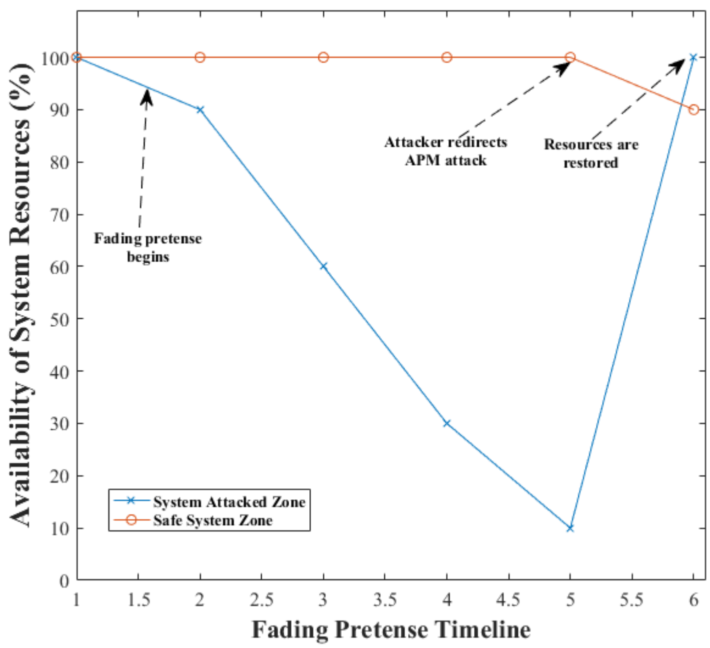


Figure 19: Demonstration of a Fading Pretense policy implementation to deter an APM attack after its detection on a compromised device.

6.2.4. Fading Pretense Policy Demonstration

The Fading Pretense policy can be implemented as an effective defense mechanism against APM attacks as described earlier in Section 5.2. Herein, we describe a demonstration of an experiment that illustrates how the fading pretense policy could deter an APM attacker in practice. Figure 19 shows the experiment conducted using a safe system zone, and an attacked system zone. The x-axis is an arbitrary unit of time progression that can be configured (on the order of several minutes, hours or even days) by the system/network administrator depending on the duration of the pretense policy being in effect. Behavioral psychology considerations also could be factored into determining the rate of progressive decline of the resource allocation on a compromised device. In any case, we can observe that three distinct phases of pretense should occur:

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

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

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

6.2.5. Time Overhead for Suspiciousness Score Calculation

Table 7 shows the time taken by ADAPTs to calculate the ss for devices, each running on a single core. It also shows the number of traces, and their corresponding processing times. As high as 1.8 million packets for 8 devices can be processed under 100 seconds, which demonstrates the efficacy of ADAPTs. However, there is a linear increase in time as the number of traces grow. If such a linear increase does not meet the threat monitoring objectives of a domain, a parallel implementation of ADAPTs can be extended and used on nodes with multi-core functionality to reduce the computation times in the Suspiciousness Score calculations.

Table 7: Processing time taken by ADAPT with single threaded processing

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

7. Conclusion

Recent innovations in the orchestration of cloud resources are fueled by emergence of the Software-Defined everything Infrastructure (SDxI) paradigm. At the same time, the sophistication of targeted attacks such as Distributed Denial-of-Service (DDoS) attacks and Advanced Persistent Threat (APT) attacks are growing on an unprecedented scale. Consequently, online businesses in retail, healthcare and other fields are under constant threat of targeted attacks. In this paper, we presented a novel defense system called *Dolus* to mitigate the impact of DDoS and APT 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 targeted attacks, which involves threat detection algorithms based on a number of attack vector features. 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 for DDoS and APT attacks defense 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 in the case of DDoS attacks, 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 system in a GENI cloud testbed for DDoS attacks show that our approach can be effective in filtering, detection and implementation of SDxI-based infrastructure policy coordination for mitigation of the impact of the DDoS attacks. In addition, we also showed how the Dolus system can be an effective defense using pretense against APTs and APMs. Using the general Suspiciousness Scores and a novel Targeted Suspiciousness Score concept, we proposed novel threat intelligence collection and accurate attack detection of subtle and secretive targeted attacks at a device level and also at a network-wide level. Further, we found that our Admin UI capability can greatly help network operators and cloud service providers to overcome their difficulty in determining which devices on an enterprise network or a cloud service deployment may be compromised. Lastly, we demonstrated how a pertinent defense strategy such as a fading pretense policy can be effective in the mitigation of APM attacks that target resource exfiltration within an SDxI-based infrastructure.

Future work can be pursued to investigate more sophisticated pretense schemes that use threat intelligence collection on effectiveness of a working pretense, and initiate more involved adaptations. In addition, data analytics extensions can be pursued for more sophisticated targeted attacks with significantly larger number of features that need to be involved in effective detection and defense schemes.

Acknowledgement

This material is based upon work supported by the National Science Foundation under Award Number: CNS-1205658. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[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, L. Vanbever, Authorizing network control at software defined internet exchange points, in: Proceedings of the Symposium on SDN Research, SOSR '16, ACM, New York, NY, USA, 2016, pp. 16:1–16:6.

[4] P. Chen, L. Desmet, C. Huygens, A study on advanced persistent threats, in: IFIP International Conference on Communications and Multimedia Security, Springer, pp. 63–72.

[5] P. Kampanakis, H. G. Perros, T. Beyene, Sdn-based solutions for moving target defense network protection., in: WoWMoM, IEEE Computer Society, 2014, pp. 1–6.

[6] S. Debroy, P. Calyam, M. Nguyen, A. Stage, V. Georgiev, Frequency-minimal moving target defense using software-defined networking, 2016 International Conference on Computing, Networking and Communications (ICNC) 00 (2016) 1–6.

[7] N. Virvilis, D. Gritzalis, T. Apostolopoulos, Trusted computing vs. advanced persistent threats: Can a defender win this game?, in: Ubiquitous intelligence and computing, 2013 IEEE 10th international conference on and 10th international conference on autonomic and trusted computing (uic/atc), IEEE, pp. 396–403.

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

[9] J. W. V. de Vondervoort, O. Friedman, Young children protest and correct pretense that contradicts their general knowledge, Cognitive Development 43 (2017) 182–189.

[10] M. Marchetti, F. Pierazzi, M. Colajanni, A. Guido, Analysis of high volumes of network traffic for advanced persistent threat detection, Computer Networks 109 (2016) 127–141.

[11] J. Leyden, CryptoLurker hacker crew skulk about like cyberspies, earn $$$, https://www.theregister.co.uk/2018/03/06/apt_cryptomining, 2018.

[12] M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, I. Seskar, Geni: A federated testbed for innovative network experiments, Comput. Netw. 61 (2014) 5–23.

[13] A. Clark, K. Sun, 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, pp. 678–685.

[14] T. Sochor, M. Zuzcak, Study of internet threats and attack methods using honeypots and honeynets, in: International Conference on Computer Networks, Springer, pp. 118–127.

[15] S. Seufert, D. O'Brien, Machine learning for automatic defence against distributed denial of service attacks, in: 2007 IEEE International Conference on Communications, pp. 1217–1222.

[16] J. Choi, C. Choi, B. Ko, P. Kim, A method of ddos attack detection using http packet pattern and rule engine in cloud computing environment, Soft Computing 18 (2014) 1697–1703.

[17] T. Thapngam, S. Yu, W. Zhou, S. K. Makki, Distributed denial of service (ddos) detection by traffic pattern analysis, Peer-to-Peer Networking and Applications 7 (2014) 346–358.

[18] Y. Chen, S. Jain, V. K. Adhikari, Z. L. Zhang, K. Xu, A first look at inter-data center traffic characteristics via Yahoo! datasets, pp. 1620–1628.

[19] L. Yang, T. Zhang, J. Song, J. S. Wang, P. Chen, Defense of ddos attack for cloud computing, in: 2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE), volume 2, pp. 626–629.

[20] Internet2's ddos mitigation strategy, https://www.internet2.edu/blogs/detail/12234, 2016.

[21] Y. Choi, Implementation of content-oriented networking architecture (cona): a focus on ddos countermeasure.

[22] C. YuHunag, T. MinChi, C. YaoTing, C. YuChieh, C. YanRen, A novel design for future on-demand service and security, in: 2010 IEEE 12th International Conference on Communication Technology, pp. 385–388.

[23] S. Shin, P. A. Porras, V. Yegneswaran, M. W. Fong, G. Gu, M. Tyson, Fresco: Modular composable security services for software-defined networks., in: NDSS, The Internet Society, 2013.

[24] Y. Yu, C. Qian, 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, ACM, New York, NY, USA, 2014, pp. 85–90.

[25] H. Tian, J. Bi, An incrementally deployable flow-based scheme for ip traceback., IEEE Communications Letters 16 (2012) 1140–1143.

[26] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, A. Warfield, Live migration of virtual machines, in: Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2, NSDI'05, USENIX Association, Berkeley, CA, USA, 2005, pp. 273–286.

[27] Q. Yan, F. R. Yu, Q. Gong, 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 (2016) 602–622.

[28] J. Kim, T. Lee, H.-g. Kim, H. Park, Detection of advanced persistent threat by analyzing the big data log, Cloud Security Alliance 13 (2013) 101–107.

[29] P. K. Sharma, S. Y. Moon, D. Moon, J. H. Park, Dfa-ad: a distributed framework architecture for the detection of advanced persistent threats, Cluster Computing 20 (2017) 597–609.

[30] P. Bhatt, E. T. Yano, P. Gustavsson, Towards a framework to detect multistage advanced persistent threats attacks, in: Service Oriented System Engineering (SOSE), 2014 IEEE 8th International Symposium on, IEEE, pp. 390–395.

[31] B. Binde, R. McRee, T. J. OConnor, Assessing outbound traffic to uncover advanced persistent threat, SANS Institute. Whitepaper (2011).

[32] J. de Vries, H. Hoogstraaten, J. van den Berg, S. Daskapan, Systems for detecting advanced persistent threats: A development roadmap using intelligent data analysis, in: Cyber Security (CyberSecurity), 2012 International Conference on, IEEE, pp. 54–61.

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

[34] G. Vert, A. L. Claesson-Vert, J. Roberts, E. Bott, A technology for detection of advanced persistent threat in networks and systems using a finite angular state velocity machine and vector mathematics, in: Computer and Network Security Essentials, Springer, 2018, pp. 41–64.

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

[36] M. Lee, D. Lewis, Clustering disparate attacks: mapping the activities of the advanced persistent threat, Last accessed June 26 (2013).

[37] P. Giura, W. Wang, Using large scale distributed computing to unveil advanced persistent threats, Science J 1 (2012) 93–105.

[38] Z. Saud, M. H. Islam, Towards proactive detection of advanced persistent threat (apt) attacks using honeypots, in: Proceedings of the 8th International Conference on Security of Information and Networks, ACM, pp. 154–157.

[39] N. Virvilis, D. Gritzalis, The big four-what we did wrong in advanced persistent threat detection?, in: Availability, Reliability and Security (ARES), 2013 Eighth International Conference on, IEEE, pp. 248–254.

[40] Z. Zulkefli, M. M. Singh, N. H. A. H. Malim, Advanced persistent threat mitigation using multi level security–access control framework, in: International Conference on Computational Science and Its Applications, Springer, pp. 90–105.

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

[42] P. Hu, H. Li, H. Fu, D. Cansever, P. Mohapatra, Dynamic defense strategy against advanced persistent threat with insiders, in: Computer Communications (INFOCOM), 2015 IEEE Conference on, IEEE, pp. 747–755.

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

[44] M. Ammar, M. Rizk, A. Abdel-Hamid, A. K. Aboul-Seoud, A framework for security enhancement in sdn-based datacenters, in: New Technologies, Mobility and Security (NTMS), 2016 8th IFIP International Conference on, IEEE, pp. 1–4.

[45] M. Saher, J. Pathak, Malware and exploit campaign detection system and method, 2014. US Patent App. 14/482,696.

[46] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, D. Walker, Frenetic: A network programming language, in: ACM Sigplan Notices, volume 46, ACM, pp. 279–291.

[47] W. M. Eddy, Tcp syn flooding attacks and common mitigations (2007).

[48] F. Gont, Icmp attacks against tcp (2010).

[49] K. Tools, hping3. ICMP or SYN flooding tool, https://tools.kali.org/information-gathering /hping3, 2014.

[50] S. Shekyan, SlowHTTPTest. Application Layer DoS attack , https://github.com/shekyan/slowhttptest/wiki, 2011.

[51] Y. Zhang, S. Debroy, P. Calyam, Network-wide anomaly event detection and diagnosis with perfsonar, IEEE Transactions on Network and Service Management 13 (2016) 666–680.

[52] Y. Zhang, P. Calyam, S. Debroy, 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), pp. 149–156.

[53] R. S. Boyer, J. S. Moore, Mjrty—a fast majority vote algorithm, in: Automated Reasoning, Springer, 1991, pp. 105–117.

[54] T. G. Dietterich, et al., Ensemble methods in machine learning, Multiple classifier systems 1857 (2000) 1–15.

[55] Scapy: Packet manipulation tool, http://www.secdev.org/projects/scapy/, 2007.

[56] M. K. Daly, Advanced persistent threat, Usenix, Nov 4 (2009) 2013–2016.

[57] W. A. M. T. Y. Enterprise, G. Hoglund, Advanced persistent threat (????).

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

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

[60] A. J. T.-F. Yen, Sherlock holmes and the case of the advanced persistent threat (2012).

[61] M. Ask, P. Bondarenko, J. E. Rekdal, A. Nordbø, P. Bloemerus, D. Piatkivskyi, Advanced persistent threat (apt) beyond the hype, Project Report in IMT4582 Network Security at Gjøvik University College, Springer (2013).

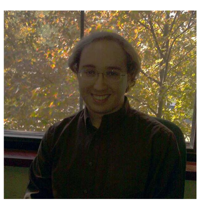
[62] C. J. Anderson, N. Foster, A. Guha, J.-B. Jeannin, D. Kozen, C. Schlesinger, D. Walker, Netkat: Semantic foundations for networks, in: Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14.

[63] A. Gervais, G. Karame, K. Wust, V. Glykantzis, H. Ritzdorf, S. Capkun, On the security and performance of proof of work blockchains, in: ACM SIGSAC Conference on Computer and Communications Security.

[64] T. Neely, M. Vassel, N. Chettri, R. Neupane, P. Calyam, R. Durairajan, Dolus for ddos attacks defense - open repository https://bitbucket.org/travisivart/dolus-defensebypretense, 2018.

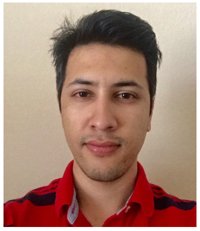
[65] T. Neely, M. Vassel, N. Chettri, R. Neupane, P. Calyam, R. Durairajan, Dolus for apt attacks defense - open repository https://github.com/travisivart/adapts, 2018.

Roshan Lal Neupane received his MS degree in Computer Science from University of Missouri - Columbia and his BE degree in Computer Science and Engineering from Visvesvaraya Technological University, Karnataka, India. His research interests include Cloud Computing, Computer Networking, Internet of Things, and Cyber Security. He is a student member of IEEE.

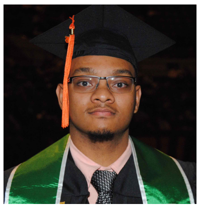


Travis Neely is currently pursuing his MS degree in Computer Science from University of Missouri - Columbia. He received his BS degree in Computer Science from Maryville University of Saint Louis. His research interests include Software-defined Networking, Data Science, and Enterprise Security.

Prasad Calyam received his MS and PhD degrees from the Department of Electrical and Computer Engineering at The Ohio State University in 2002 and 2007, respectively. He is currently an Assistant Professor in the Department of Computer Science at University of Missouri-Columbia. Previously, he was a Research Director at the Ohio Supercomputer Center. His research interests include: Distributed and Cloud computing, Computer Networking, and Cyber Security. He is a Senior Member of IEEE.



Nishant Chettri received his MS in Computer Science degree from the University of Missouri - Columbia, and his BS in Computing degree from London Metropolitan University, Kathmandu, Nepal. His research interests include Mobile and Web Development, and Database Security.



Mark Vassell is currently pursuing his MS degree in Computer Science from University of Missouri - Columbia. He received his BS degree in Computer Science from University of Missouri - Columbia. His research interests include IoT technologies for public safety, Data Science, and Cyber Security.

Ramakrishnan Durairajan received his MS and PhD degrees from the Department of Computer Science at the University of Wisconsin - Madison in 2014 and 2017, respectively. He is currently an Assistant Professor in the Department of Computer Science at University of Oregon. His research interests include: Internet Measurements, Computer Networking, and Cyber Security.