Dolus: Cyber Defense using Pretense against DDoS Attacks in Cloud Platforms*

Roshan Lal Neupane
University of Missouri- Columbia
rlnzq8@mail.missouri.edu

Travis Neely
University of Missouri- Columbia
neelyt@missouri.edu

Nishant Chettri
University of Missouri- Columbia
nc5ff@mail.missouri.edu

Mark Vassell
University of Missouri- Columbia
mdvy96@mail.missouri.edu

Yuanxun Zhang
University of Missouri- Columbia
yzd3b@mail.missouri.edu

Prasad Calyam
University of Missouri- Columbia
calyamp@missouri.edu

Ramakrishnan Durairajan
University of Oregon
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 totally programmable as Software Defined everything Infrastructures (SDxI). At the same time, sophisticated targeted attacks such as Distributed Denial-of-Service (DDoS) are growing on an unprecedented scale threatening the availability of online businesses. In this paper, we present a novel defense system called *Dolus* to mitigate the impact of DDoS attacks launched against high-value services hosted in SDxI-based cloud platforms. Our Dolus system is able to initiate a 'pretense' in a scalable and collaborative manner to deter the attacker based on threat intelligence obtained from attack feature analysis in a two-stage ensemble learning scheme. Using foundations from *pretense theory in child play*, Dolus takes advantage of elastic capacity provisioning via 'quarantine virtual machines' and SDxI policy co-ordination across multiple network domains to deceive the attacker by creating a false sense of success. From the time gained through pretense initiation, Dolus enables cloud service providers to decide on a variety of policies to mitigate the attack impact, without disrupting the cloud services experience for legitimate users. We evaluate the efficacy of Dolus using a GENI Cloud testbed and demonstrate its real-time capabilities to: (a) detect DDoS attacks and redirect attack traffic to quarantine resources to engage the attacker under pretense, and (b) coordinate SDxI policies to possibly block DDoS attacks closer to the attack source(s).

*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.

ICDCN '18, January 4–7, 2018, Varanasi, India

ACM ISBN 978-1-4503-6372-3/18/01…$15.00
https://doi.org/10.1145/3154273.3154346

CCS CONCEPTS

• **Networks** → **Denial-of-service attacks**; **Cloud computing**;

KEYWORDS

Software-defined infrastructure, DDoS defense, Cloud services protection, Pretense theory

ACM Reference Format:
Roshan Lal Neupane, Travis Neely, Nishant Chettri, Mark Vassell, Yuanxun Zhang, Prasad Calyam, and Ramakrishnan Durairajan. 2018. Dolus: Cyber Defense using Pretense against DDoS Attacks in Cloud Platforms. In *ICDCN '18: 19th International Conference on Distributed Computing and Networking, January 4–7, 2018, Varanasi, India.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3154273.3154346

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 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.

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" [4] [5] have been proposed to protect networks and users against DDoS attacks by migrating networks and users from targeted virtual machines (VMs) to other healthy/safe VMs in a cloud platform. However, such strategies may cause the application response behavior to change to an extent that alerts the attacker that a high-value target has been hit. Given such a discovery that a service provider is moving a target in order to shelter from the attack impact, the attacker may then deflect more resources to seek ransom demands in order to stop the DDoS on the target.

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

In this 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 DDoS attacks launched against high-value services hosted in SDxI-based cloud platforms. The DDoS attack detection is performed in the Dolus system using the threat intelligence obtained from attack feature analysis in a two-stage ensemble learning scheme that we developed. The first stage focuses on anomaly detection to identify salient events of interest (e.g., connection exhaustion), and the second stage is invoked to distinguish the DDoS attack event type amongst the 5 common attack vectors: DNS (Domain Name System), UDP (User Datagram Protocol) fragmentation, NTP (Network Time Protocol), SYN (short for synchronize), SSDP (simple service discovery protocol). Our Dolus system[1] is novel owing to a scalable and collaborative defense strategy that uses 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 co-ordination across multiple network domains. Such a strategy is aimed at preventing the disruption of cloud-hosted services by deceiving the attacker through creation of a false sense of success, and by keeping the attacker from recognizing that a high-value target has been impacted and is being moved.

[1] Dolus system is openly available to the community here: [6]. A demonstration of Dolus, along with the attack generation scheme that we consider in this paper, is available here [7].

We evaluate the efficacy of our Dolus system using a GENI Cloud [10] testbed that 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 remainder of this paper is organized as follows: In Section 2, we discuss related works. We describe our overall system and defense methodology in Section 3. Section 4 presents our evaluation experiments in a GENI Cloud testbed. Section 5 concludes the paper.

2 RELATED WORK

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) [11] [12] [13]. 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 [14] 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 [15], which can then be used to "scrub" the flows through scrubbing SaaS services as a low-cost solution [16].

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 [17], where a network reacts to targeted attacks using accountability and content-aware supervision concepts. Similarly, using volume counting, authors in [18] 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 [19] present a programming framework. In another similar effort, authors in [20] 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 [21] 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 [22] 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) [5]. As an added defense

strategy, authors in works such as [23] present a survey of SDN-based mechanisms to detect attacks closer to the attackers/attack sources.

There have been efforts that seek to implement defense mechanisms using some form of 'trickery' to engage an attacker. For example, authors in [24] 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 [25]. 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 DDoS attacks is inspired by the authors' experiments where they show children (analogous to our attackers) various pictures of the animals along with a mismatch of the sounds made by the associated animals. Observations are made on how a pretense is effective based on how long it takes for a child to understand/protest that the information portrayed is actually false. In our case, the longer an attacker is tricked by our pretense, the more time a cloud service provider has to perform MTD mechanisms, strategize on patching identified vulnerabilities, as well as implement a SDxI-based infrastructure policy coordination for mitigation of the impact of a DDoS attack.

3 DOLUS DEFENSE METHODOLOGY

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

3.1 Dolus System Overview

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

(a) The basic assumption(s) or premise(s) that is used by a pretender on *what* is being pretended.
(b) Inferential elaboration which details of what goes into or what actually happens in the process of pretense.
(c) Appropriate behavior production which answers the question of whether the pretender was successful on the audience being tricked.
(d) Balancing and steering the effects of pretense.

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. Thus, we can see that an effective pretense in our case can be designed as shown in Figure 1 by creating pertinent stimulus from the target side i.e., redirecting attack traffic to a quarantine VM that mimics original target behavior, when our two-stage ensemble learning algorithm can blacklist the attacker flows from benign user flows. This in turn could help in keeping an attacker distracted for a brief period of time when the pretense is in effect.

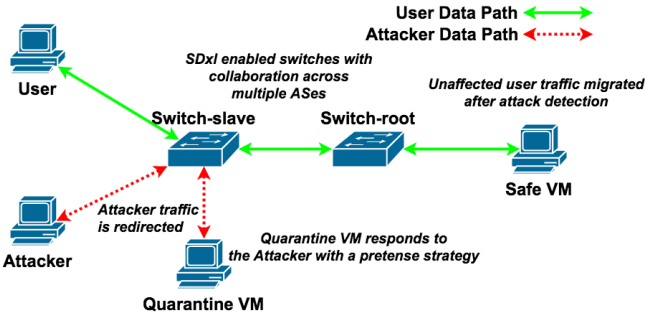


Figure 1: Illustration of the proposed Dolus system scheme wherein the attacker is *tricked* by redirection of the attack traffic to a quarantine VM for pretense initiation, while the providers work collaboratively to block the attack traffic closer to the source side.

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 [26] 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.

3.2 Attack Model

DDoS attacks aim to overwhelm network-accessible devices such as networks, firewalls and end-systems in enterprises by sending packets at excessively high rates. With cloud-hosted applications with large monetary value becoming highly common, DDoS attacks can cause 'Loss of Availability' for users/customers and can be used for extortion from vulnerable online businesses. Common DDoS attack event types are amongst the 5 common attack vectors: DNS (Domain Name System), UDP (User Datagram Protocol) fragmentation, NTP (Network Time Protocol), SYN (short for synchronize), SSDP (simple service discovery protocol). For the purposes of our work, we assume the DDoS attacker uses SYN [27] and ICMP/Ping [28] flooding. Such attacks typically inundate a networks' resources with Echo Request packets. We also assume that the attackers' traffic is sent constantly and may or may not solicit a response in return. Such attacks can bring the network to a standstill due to the high

volume of both incoming and outgoing traffic. To effectively capture the semantics of this attack model and to exhaust the target application services, we generate and emit synthetic ping and HTTP traffic using `hping3` [29] and `SlowHTTPTest` [30] tools, respectively. Furthermore, to capture the dynamics of an attacker, we randomly change the number of attack packets emitted by these tools.

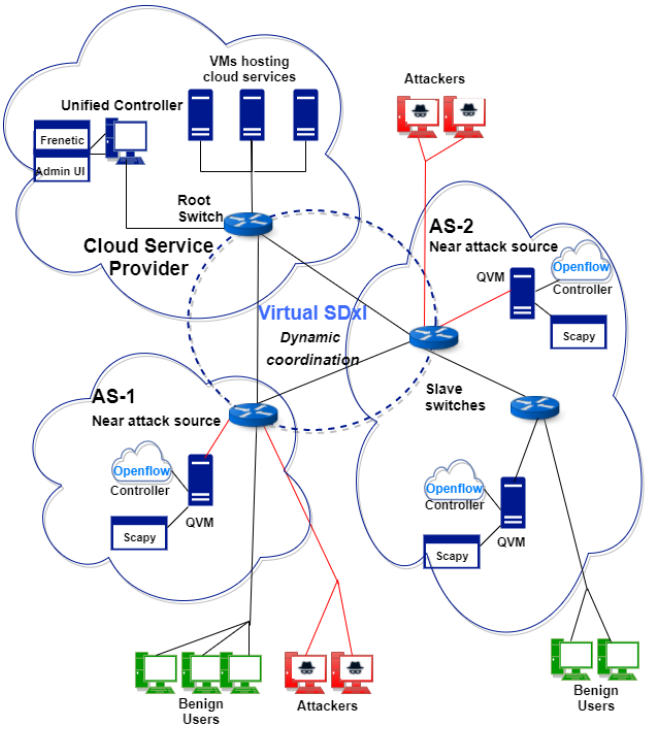


Figure 2: Cross-domain physical setup in a Dolus system deployment to share threat intelligence for a unified controller to coordinate policy management with a federation of ASes to block attack traffic closer to the source side.

3.3 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.

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 [26]-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).

Algorithm 1: Dolus system phases for spoofing pretense

Input: $attacker\_ID$ = attacker ID,
src_ip = source IP,
dst_ip = destination IP,
$no\_of\_packets$ = number of packets,
$spoof\_dst\_ip$ = spoofed IP,
$black\_ip$ blacklisted IP list
Result: Attack traffic will be redirected to the quarantine VM and DDoS blocking policy will be generated

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

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 [31, 32] 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 [33]. In our case, to achieve higher accuracy with a

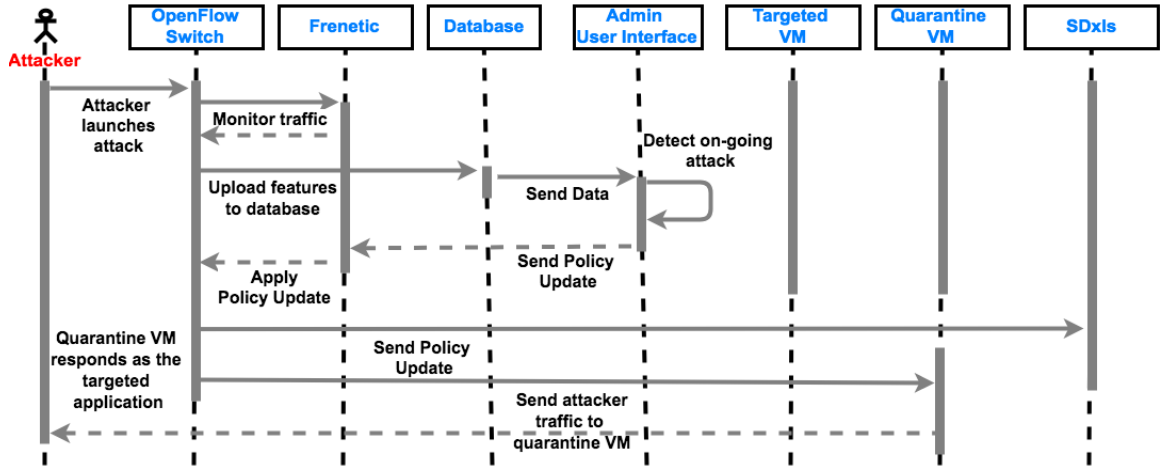


Figure 3: Sequence diagram of the Dolus system interactions for attack detection, quarantine setup, pretense initiation/maintenance and DDoS attack impact mitigation.

minimum size of the training dataset D, we use the Bayesian voting scheme [34] 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. 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 [35] 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).

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

Pretense Maintenance. Finally, we redirect the flow of the attack traffic by pushing a new policy from the unified controller running in the cloud to the switch(es). This will redirect the attacker's traffic that is intended for the targeted VM towards the QVM. The QVM then responds to the attacker's traffic as though it is the targeted VM/server under attack with spoofed IP address and hostname of the target, which creates the pretense effect, from an attacker's perspective, that the targeted DDoS attack is successful. Depending on the nature of the attack, we want the attacker to believe that services are no longer up/available on the targeted VM. We therefore allow the QVM to continue to respond to the attacker for a limited amount of time t. We tune t based on the type of attack traffic and how the targeted VM would respond if it was under attack. For example, if the targeted VM went down after 10 seconds of attack, the QVM would do the same by not responding at the same time with a variable random delay factor of [-1,1] seconds added. This allows the attacker to see that the services are available until, suddenly, they no longer are.

Policy Decision Making. In this sense, our defense maintains the pretense: gives the attacker the confirmation of a successful attack,

when in reality the service has not been affected at all as seen in the Figure 5 considering the scenario that the user is running a video gaming portal application. This also gives us sufficient time to collect information about the attackers and their attack patterns. We use the collected information to create a blacklist of attacker information. To help network administrators effectively manage the network in the face of attacks, our system also consists of a Administrator User Interface module and a unified controller module that can be customized in a Dolus system instance deployment depicted in Figure 2. The User Interface 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. That is, once if it has been determined that the attack traffic is no longer impacting the network, the policies can be updated to redirect the attacker traffic back to where it was prior to the start of pretense. There are several reasons to do this: (i) changes in the dynamics of the attack (e.g., bandwidth usage dropping back down to normal, absence of SYN packets in a SYN flooding attack, fixing of malware in an affected machine and hence it is no longer an attacker, etc.) calls for network policy changes so that the network resources can be effectively used, (ii) changes in traffic e.g., IP address change in incoming service requests sent from a benign user must be serviced to meet the service level agreement (SLA), and (iii) to save the operational cost of QVMs by reusing them for a different purpose e.g., periodic backups.

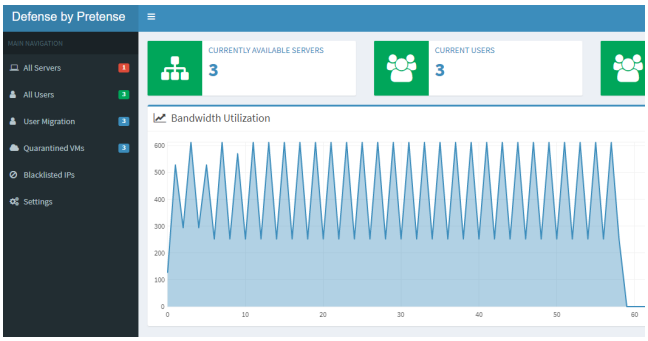


Figure 4: Administrator User Interface of an Dolus system instance.

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.

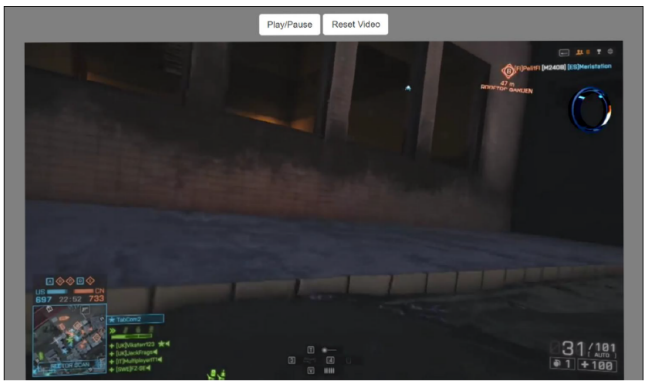


Figure 5: Video gaming portal application running in a SDxI-based cloud platform with cross-domain network collaboration.

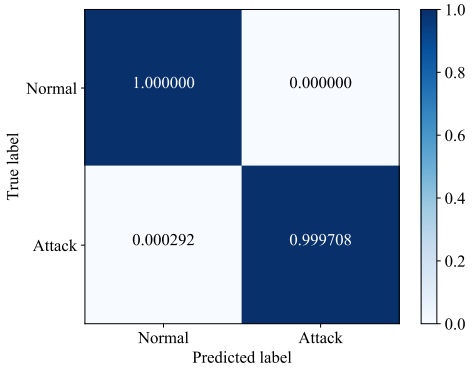
4 PERFORMANCE EVALUATION

We evaluate the efficacy of our Dolus system using a realistic, GENI Cloud [10] testbed as shown in Figure 9. The testbed contains three SDN switches, two slave switches and a single root switch. The slave switches are each attached to users and attackers, a quarantine VM, and a connection to the root switch. Likewise, the root switch is connected to elastic VMs, each of which could serve as a candidate for the target application (i.e., a video gaming portal) hosting that could be compromised by the attackers. All switches are connected to a unified SDN controller located in the cloud service provider domain, which directs the policy updates. 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.

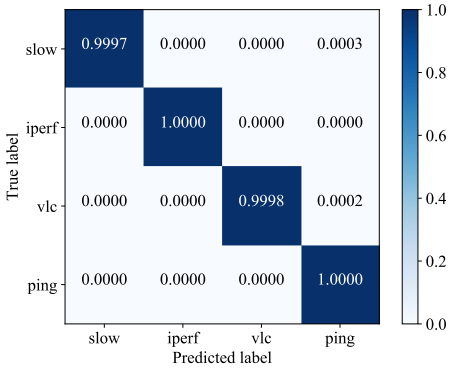
4.1 Attack Detection and Classification

Using the Dolus system, we monitor different types of data that are permitted to enter the GENI Cloud testbed depicted in Figure 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 6 shows the two confusion matrices

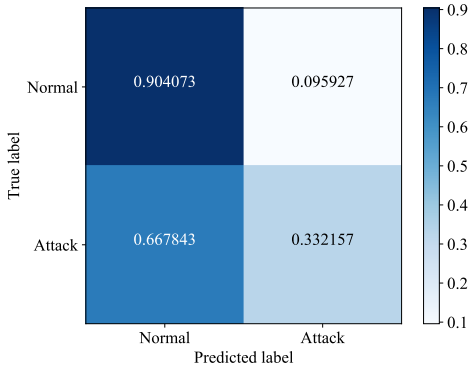


(a) Attack detection.

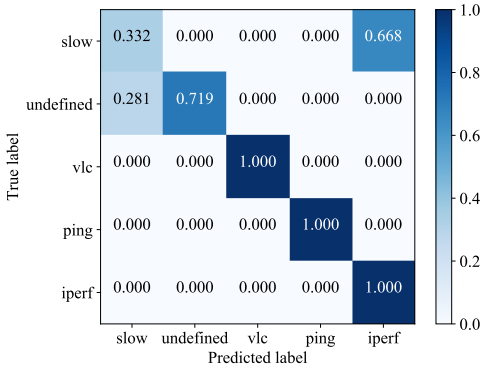


(b) Attack classification.

Figure 6: Confusion matrices for attack detection and classification for multiple traffic flows sent to a single server.



(a) Attack detection.



(b) Attack classification.

Figure 7: Confusion matrices for attack detection and classification for multiple traffic flows sent to multiple hosts.

Table 1: Overall Attack Detection and Classification Time and Accuracy

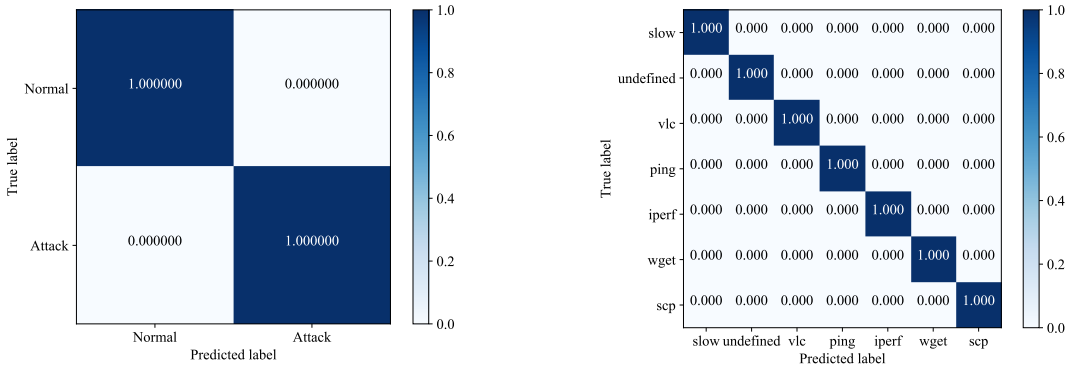
Tests	Time (in Seconds)	Accuracy (in %)
Single server stage 1	<1	99.99
Single server stage 2	<1	99.98
Multiple hosts stage 1	7	89.12
Multiple hosts stage 2	13	98.49

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 1, where stage 1 is the detection stage and stage 2 is the classification stage.

In the second instance, we consider multiple traffic types to multiple hosts. This instance is composed of 2.5 million rows per test, totaling 5 million rows of data. The types of traffic that we use to create this dataset include SlowHTTPTest, iperf, VLC, scp, wget, 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 7 shows the two confusion matrices in normalized form for attack detection and classification. Detection and classification of attack took ~7 and ~13 seconds, respectively. Despite the slowdown in attack detection/classification in comparison with the first instance, the accuracy of our approach is still high as shown in Table 1.

While the two-stage ensemble learning scheme is effective in detecting test data, a new attack that has not been used in training could initially go undetected and impact services. However, with pertinent labeling of attack traffic flows during training, the accuracy of the ensemble learning scheme can be improved significantly. We depict the outlier detection and classification for a trained cased in Figure 8, where we make use of 60% of the data as training data and 40% 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.



(a) Attack detection.

(b) Attack classification.

Figure 8: Confusion matrices for outlier detection and classification for multiple traffic flows comprising of familiar attack flows.

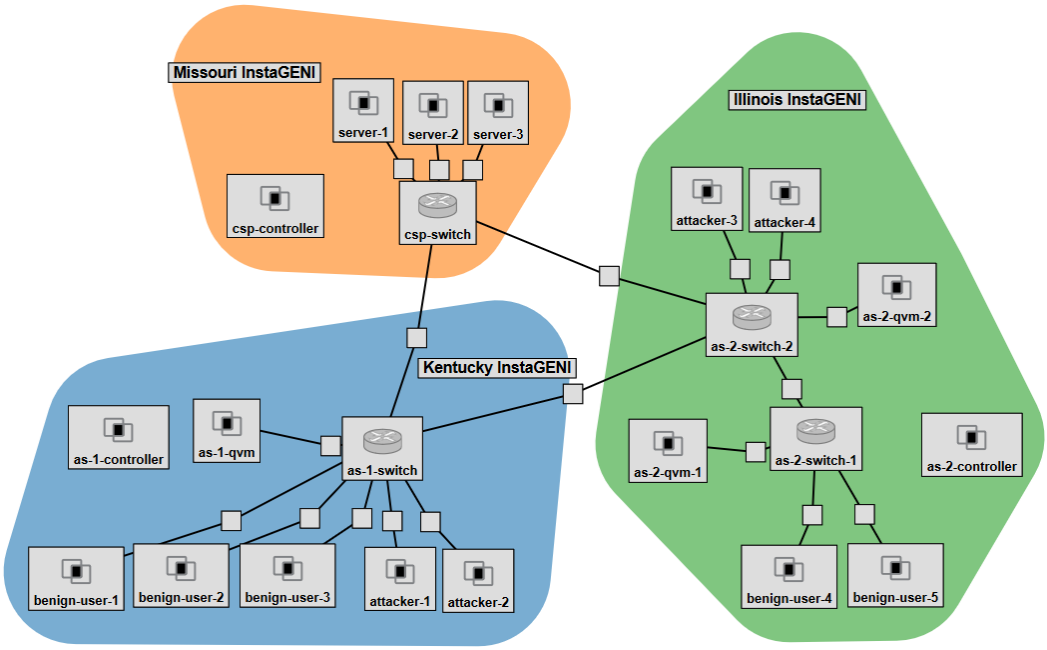


Figure 9: GENI Cloud testbed setup used to evaluate the Dolus system performance.

4.2 Time to Restore a Cloud-hosted Application Service

Figure 10 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. Similarly, the MTD-based defense strategy takes ~6 *seconds* to mitigate the attack traffic impact. However, our Dolus system supported service on the other hand, does not suffer from any loss of availability in comparison with the other two strategies. This is due to the sharing of attack intelligence between the slave switches and redirection of attack traffic to quarantine VMs closer to the attackers, making the cloud network completely oblivious to the attackers.

4.3 Amount of Traffic Processed at the Root Switch

Figures 11 and 12 depict the amount of traffic processed (in Bytes) at one of the slave switches and the root switch. From Figure 12, it is evident that the SDxI-based cloud network is oblivious to the attack traffic impact, complementing the result in Figure 10. Since the slave switch represented in Figure 12 redirects attack traffic to the quarantine VMs, we observe a $5X$ increase in the amount of traffic processed in comparison with the root switch.

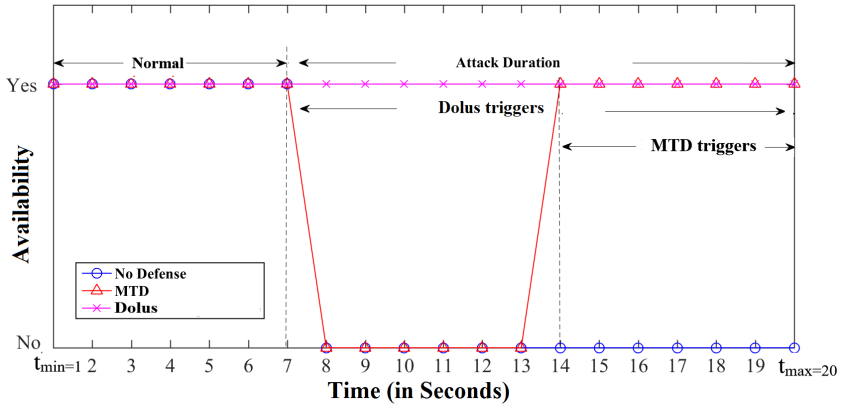


Figure 10: Comparison of the *cloud service restoration time* metric with cases of: no Defense, with MTD and with Dolus.

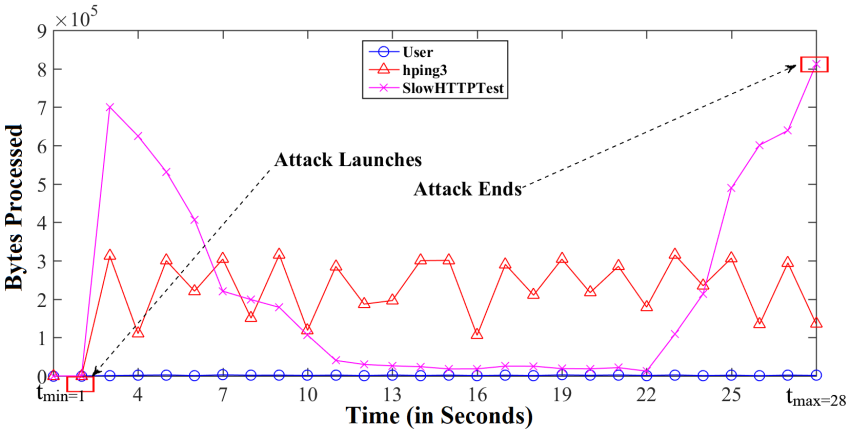


Figure 11: Traffic processed (in Bytes) in one of the slave switches.

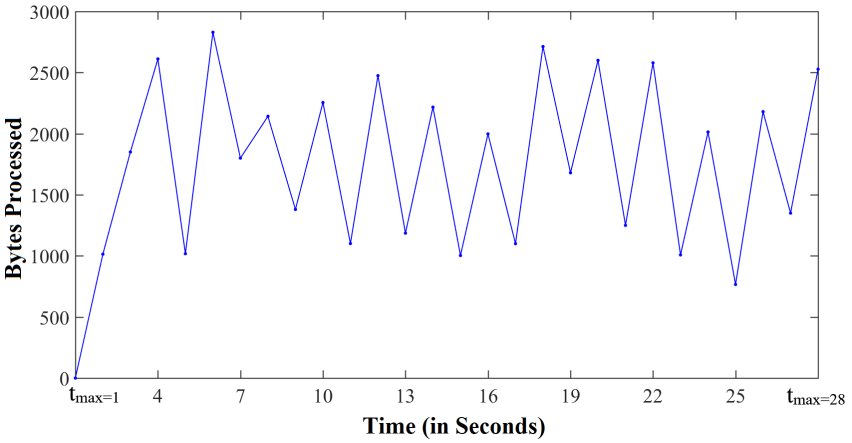


Figure 12: Traffic Processed at the root switch only shows user traffic proving that the attack traffic is redirected to quarantine VM.

Overall, we find that our Dolus can effectively detect DDoS attack and redirect traffic in real-time i.e., on the order of seconds depending on the knowledge of the DDoS attack pattern, and block it closer to the attack source in 1-2 seconds if automated policy updates

are possible in the cross-domain setting. However, if humans need to be brought into the loop, the time to block the attack can be adjusted so that there is enough time for cross-domain manual coordination during which an effective pretense of the quarantine VM is deceiving the attacker with a false sense of success.

5 CONCLUSION AND FUTURE WORK

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

Our above approach takes advantage of elastic capacity provisioning in cloud platforms to implement moving target defense techniques that does not affect the cloud-hosted application users, and contains the attack traffic in a quarantine VM(s). With the time gained through effective pretense initiation, cloud service providers could coordinate across a unified SDxI infrastructure involving multiple ASes to decide on policies that help in blocking the attack flows closer to the source side. Performance evaluation results of our Dolus system in a GENI cloud testbed show that our approach can be effective in filtering, detection and implementation of SDxI-based infrastructure policy coordination for mitigation of the impact of DDoS attacks.

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

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

[4] P. Kampanakis, H. G. Perros, and T. Beyene. Sdn-based solutions for moving target defense network protection. In *WoWMoM*, pages 1–6. IEEE Computer Society, 2014.

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

[6] Dolus: DefensebyPretense. https://bitbucket.org/rneupane93/dolus-defensebypretense, 2017.

[7] Cyber Defense Using Pretense in SDX-Driven Cloud Platforms - System Demo. https://www.youtube.com/watch?v=LppBPu477WU, 2017.

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

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

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

[11] S. Seufert and D. O'Brien. Machine learning for automatic defence against distributed denial of service attacks. In *2007 IEEE International Conference on Communications*, pages 1217–1222, June 2007.

[12] J. Choi, C. Choi, B. Ko, and P. Kim. A method of ddos attack detection using http packet pattern and rule engine in cloud computing environment. *Soft Computing*, 18(9):1697–1703, Sep 2014.

[13] T. Thapngam, S. Yu, W. Zhou, and S. K. Makki. Distributed denial of service (ddos) detection by traffic pattern analysis. *Peer-to-Peer Networking and Applications*, 7(4):346–358, Dec 2014.

[14] Y. Chen, S. Jain, V. K. Adhikari, Z. L. Zhang, and K. Xu. *A first look at inter-data center traffic characteristics via Yahoo! datasets*, pages 1620–1628. 2011.

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

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

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

[18] C. YuHunag, T. MinChi, C. YaoTing, C. YuChieh, and C. YanRen. A novel design for future on-demand service and security. In *2010 IEEE 12th International Conference on Communication Technology*, pages 385–388, Nov 2010.

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

[20] Y. Yu, C. Qian, and X. Li. Distributed and collaborative traffic monitoring in software defined networks. In *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, HotSDN '14, pages 85–90, New York, NY, USA, 2014. ACM.

[21] H. Tian and J. Bi. An incrementally deployable flow-based scheme for ip traceback. *IEEE Communications Letters*, 16(7):1140–1143, 2012.

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

[23] Q. Yan, F. R. Yu, Q. Gong, and J. Li. Software-defined networking (sdn) and distributed denial of service (ddos) attacks in cloud computing environments: A survey, some research issues, and challenges. *IEEE Communications Surveys Tutorials*, 18(1):602–622, Firstquarter 2016.

[24] A. Clark, K. Sun, and R. Poovendran. Effectiveness of IP address randomization in decoy-based moving target defense. In *Proceedings of the 52nd IEEE Conference on Decision and Control, CDC 2013, December 10-13, 2013, Firenze, Italy*, pages 678–685, 2013.

[25] T. Sochor and M. Zuzcak. Study of internet threats and attack methods using honeypots and honeynets. In *International Conference on Computer Networks*, pages 118–127. Springer, 2014.

[26] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker. Frenetic: A network programming language. In *ACM Sigplan Notices*, volume 46, pages 279–291. ACM, 2011.

[27] W. M. Eddy. Tcp syn flooding attacks and common mitigations. 2007.

[28] F. Gont. Icmp attacks against tcp. 2010.

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

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

[31] Y. Zhang, S. Debroy, and P. Calyam. Network-wide anomaly event detection and diagnosis with perfsonar. *IEEE Transactions on Network and Service Management*, 13(3):666–680, Sept 2016.

[32] Y. Zhang, P. Calyam, S. Debroy, and M. Sridharan. Pca-based network-wide correlated anomaly event detection and diagnosis. In *2015 11th International Conference on the Design of Reliable Communication Networks (DRCN)*, pages 149–156, March 2015.

[33] R S Boyer and J S Moore. Mjrty—a fast majority vote algorithm. In *Automated Reasoning*, pages 105–117. Springer, 1991.

[34] Thomas G Dietterich et al. Ensemble methods in machine learning. *Multiple classifier systems*, 1857:1–15, 2000.

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