Mitigation of Multi-vector Network Attacks via Orchestration of Distributed Rule Placement

Marinos Dimolianis, Adam Pavlidis, Dimitris Kalogeras, Vasilis Maglaris
Network Management & Optimal Design Laboratory (NETMODE)
School of Electrical & Computer Engineering,
National Technical University of Athens (NTUA), Greece
{mdimolianis, apavlidis, dkalo, maglaris} [at] netmode.ntua.gr

***Abstract*— In this paper we propose a framework for mitigating detected multi-vector anomalies in typical enterprise networks via the distribution of Access Control Rules. Our distributed, non-proprietary approach takes advantage of the capabilities offered by all devices along an attack path enhancing their mitigation potential. These devices are organized into distinct defense stages and network operators express their defense preferences for specific attack types. Our mechanism automatically assigns generic mitigation rules to each stage. Subsequently, device-specific access control rules are generated and seamlessly distributed to the corresponding defense stages of the network substrate via commonly used protocols. The proposed mitigation schema models the rule assignment to defense stages as a Generalized Assignment Problem. Items, i.e. generic mitigation rules, are assigned to bins, i.e. defense stages, based on capacity constraints and reward values guided by operator policies. Our approach considers reducing the GAP input size to enable reasonable execution of the resulting integer programming formulation. This is accomplished by aggregating malicious IP sources into prefixes and organizing rules into groups. The proposed mechanism is validated in a proof of concept prototype, used to mitigate realistic multi-vector attack scenarios.**

Keywords*— *Anomaly Mitigation, Multi-Vector Attack, DDoS, Generalized Assignment Problem, Integer Programming

I. Introduction

Multi-vector Distributed Denial of Service (DDoS) attacks constitute one of the most widely spread problems faced by network operators. This was recently illustrated by the Github (2018) and the Dyn (2016) attacks. Both highlighted the growth of DDoS attacks in terms of scale, diversity (multiple attack vectors) and sophistication. According to the *DDoS Mon* [1] there are 20,000 attacks detected daily, typically affecting both network/transport and application layers. Traditional detection and mitigation mechanisms often struggle against diverse and dynamic techniques devised by malicious attackers.

Mitigation techniques for network anomalies may be categorized as on-premises and cloud-based. Mechanisms targeting to alleviate attacks on-premises (e.g. within the enterprise network hosting the victim), either use a dedicated hardware appliance like [2], [3] or deploy on-demand protection [4]. Alternatively, many enterprises prefer to outsource protection mechanisms to cloud service providers [5], [6] redirecting their traffic to scrubbing centers whereby it is filtered and then redirected back. Cloud-scrubbing services grow in popularity; however, they raise privacy concerns, introduce additional latency and usually require considerable costs. On-premise specialized solutions are commonly based on the deployment of costly dedicated appliances that may require continuous support and upgrade from network administrators.

In this paper, we propose a distributed, non-proprietary on-premises anomaly mitigation schema. Our approach offers flexibility and cost-effectiveness by distributing access control rules over an enterprise network topology, deployed in diverse network nodes and operating at various protocol layers. Our framework leverages on the Software-Defined Network (SDN) paradigm that disassociates control-plane functionality from data-plane forwarding; the anomaly mitigation policies are separately implemented as northbound processes and communicate via diverse control plane mechanisms to the network substrate. Our motive is to enable network operators to appropriately mitigate network anomalies by enforcing custom security policies tailored to specific enterprise networks, while adhering to performance objectives and device-specific constraints.

II. Background & Related Work

Several approaches have been reported in the literature towards unified access control orchestration and on-premises mitigation of DDoS attacks. In [4] the *Bohatei* framework is presented by means of a DDoS *Mitigation-as-a-Service* platform. Incoming traffic is inspected and according to the type and the scale of the attack, *Bohatei* determines, allocates and deploys appropriate mitigation resources for a single attack vector. In our approach we investigate mitigation mechanisms for simultaneous multiple attack vectors.

In [7] the *VNGuard* high-level firewall policy language is defined for virtual networks configured within a cloud environment. Firewall instances are created and appropriate rules are placed within them based on an integer program formulation. Their objective is to minimize the number of virtual firewalls to be provisioned while respecting constraints on the number of rules, as specified by the cloud provider. Our work stems from a different perspective, notably the rule placement in existing attack mitigation resources (e.g. firewall instances) tailored to specific attack types based on capacity constraints and reward values guided by operator policies.

In [2] a two level approach is suggested for anomaly detection in legacy enterprise networks that triggers subsequent mitigation via a centralized *OpenFlow* middlebox. In [8] the *VGuard* two-layer DDoS mitigation mechanism is introduced. Traffic is classified according to the likelihood of malicious nature. Malignant flows are

This work was partially supported by the European Commission Horizon 2020 Framework Programme for Research and Innovation, Grant Agreement No. 731122 (GN4-2, GÉANT Project).

blocked, benign flows are routed to their destination as high priority traffic while suspicious flows are routed via low priority links. Our approach introduces an agile mitigation schema for multi-vector attacks that blocks attack traffic at various stages of an enterprise network.

In [9] Cloudflare introduced *Gatebot* a distributed mitigation system with centralized intelligence. This framework leverages *iptables* firewall on optimized commodity edge servers for advanced traffic filtering. Instead, we opted for an integrated SDN Controller to interface with distributed mitigation capable resources (either physical or virtual) via diverse/heterogeneous southbound protocols (not only *OpenFlow*).

III. Design Principles & Architectural Components

A. Design Principles

A high-level overview of the proposed framework is depicted in Fig. 1: Network environment capabilities and constraints together with security events generated by an Intrusion Detection System (*IDS*) are periodically collected by a *Pre-processor* (*PP*) component. This component correlates network security policies modeled as Event-Condition-Action (ECA) with security incidents, to formulate an optimization problem for distributing anomaly mitigation actions. In turn, this problem is assigned to the *Mitigation Resolver* (*MR*), a component tasked with computing a solution, i.e. an assignment of generic mitigation rules to network devices. These are converted to device-specific Access Control Rules (*ACRs*) and conveyed to the network infrastructure by the *Rule Handler* (*RH*) component through a range of supported southbound protocols.

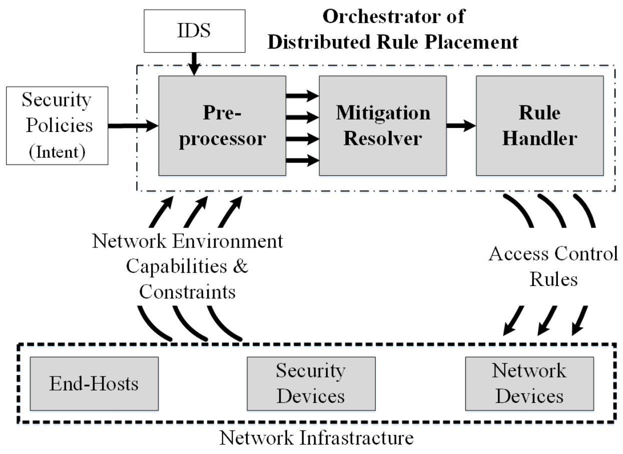

Fig. 1: Lifecycle of Orchestrator of Distributed Rule Placement

Our proposed framework embodies the following principles:

- *High-level abstraction of Access Control Rules (ACRs)*: For typical heterogeneous multi-vendor environments, we create an abstraction layer for high-level mitigation actions (primitives). Our framework maps these generic actions to device-specific *ACRs* through standardized operations and protocols (e.g. *Blackhole Routing*, *BGP Flowspec*) or vendor-specific implementations. Thus, network operators may leverage on the capabilities offered by the existing infrastructure to mitigate network anomalies in a uniform manner.
- *Orchestration of mitigation resources driven by optimization considerations*: Our proposed approach assigns generic mitigation rules to relevant network devices in various hierarchical layers. This was abstracted as a Generalized Assignment Problem (GAP) [10], whereby the process of assigning mitigation actions to network devices yields specific rewards. Reward values are based on: (i) firewall capabilities of a specific network environment, (ii) policies for different network anomalies/attacks and (iii) the actual network attack characteristics. Based on the computed solution for the described GAP, attack traffic is blocked across the attack path at the most appropriate stage of a hierarchical internetworking model [11]. Note that in terms of the GAP complexity, there is a trade-off between input size (e.g. size of attackers and defenders) and solution time.
- *Adaptive mitigation of multi-vector network attacks*: Enterprise network infrastructures are commonly structured in architectural hierarchies essentially defining distinct defense stages (i.e. core routers/switches, distribution switches, access switches and hosts). Modern sophisticated attacks consist of multiple attack vectors (volumetric, protocol-based, application-based) typically targeting specific network/host resources. Our approach implements an automated network security workflow (Fig. 1) whereby security incidents trigger mitigation actions assigned to various stages across the attack path.
- *Modular SDN / Intent-based approach*: Adhering to the concept of Intent Based Networking [12], we implement a security pipeline for mitigating network anomalies with minimal intervention based on (i) higher-level policies, (ii) network context awareness and (iii) externally generated *IDS* events. Our architecture complies to the core principles of the SDN paradigm whereby management policies are applied to the network substrate via a unified SDN controller.

B. Architectural Components

- *Pre-processor (PP)*: The purpose of this component is to properly formulate the GAP for a multi-vector attack tailored to a specific network environment. The Pre-processor component considers: (i) security events exported from an *IDS*, (ii) network security policies and (iii) environment-specific mitigation capabilities and constraints. Based on these, an appropriate input for GAP is structured and assigned to the *Mitigation Resolver* below.
- *Mitigation Resolver (MR)*: This component (i) receives data appropriately tailored to the GAP from the *PP*, (ii) attempts to reduce the input size of the algorithm to account for scalability issues, (iii) computes solutions for the GAP through a modular framework and (iv) exports the solution to the *Rule Handler*. The computed solution describes the generic mitigation rules to be distributed across the network elements.

- *Rule Handler (RH)*: This component is responsible for: (i) mapping the abstract mitigation rules to substrate-specific *ACRs* and (ii) distributing them to the network elements. The *RH* essentially materializes the SDN principle supporting a variety of mitigation techniques built on top of widely adopted southbound protocols.

C. Mitigation Techniques

In this section we outline mitigation techniques commonly used in SDN-enabled environments. Modern SDN Controllers such as *Ryu* [13], *OpenDaylight* [14], *ONOS* [15] enable device management via a wide variety of supported southbound protocols (e.g. *OpenFlow*, *BGP*, *NETCONF*). An abstract mitigation rule is defined as a typical firewall rule with the following 6-tuple: (*source_ip, destination_ip, source_port, destination_port, protocol, action*). Subsequently, this is mapped to device-specific *ACR* through various protocols/APIs via techniques such as:

C.1 Destination-based RTBH [16]: This mechanism is primarily used to prevent potential collateral damage during a DDoS attack (e.g. bandwidth and CPU utilization, service degradation). It is a destination-based filtering mechanism, in which the traffic destined to the victim is redirected to an edge router null interface. The dynamic redirection is triggered via *iBGP* using a device that retains *BGP* peering with the edge router. Particularly, we used *Ryu*'s *BGP Speaker* to propagate a static route to the victim's IP address via an appropriately configured null interface. As a result, both malicious and benign traffic destined to the victim are dropped.

C.2 Source-based RTBH [17]: Unlike the destination-based *RTBH* which renders the victim unreachable, the source-based *RTBH* drops packets from specific source IPs via the unicast Reverse Path Forwarding (*uRPF*) feature. Source-based *RTBH* also relies on *BGP* updates which contain routes to malicious IPs; attack packets from these sources are dropped on the *uRPF*-enabled router interface. Although it offers more granularity than destination-based *RTBH*, outgoing packets to legitimate destinations may be blocked if attackers employ *en route* and *fixed route* spoofing [18].

C.3 BGP Flowspec [19]: This mechanism extends the Network Layer Reachability Information (*NLRI*) format to disseminate traffic flow specification rules. These are transported over *BGP* and dynamically installed on appropriately configured devices.

C.4 OpenFlow [20]: *OpenFlow*-enabled devices contain flow tables and determine packet forwarding based on flow rule entries matched against packet headers. There is a plethora of matching capabilities and actions/instructions, used to instantiate typical firewall operations such as packet rejection/redirection.

C.5 Access Control Lists [21]: Access Control Lists (ACLs) are commonly used to implement firewall policies. In order to propagate ACLs in diverse multi-vendor environments administrators require a platform independent tool such as *Capirca* [22] designed by Google and employed in various automation frameworks e.g. *Salt* [23]. In *Capirca*, ACLs for various platforms are described in a generic data modeling language and converted to platform-specific format; the resulting ACLs may be disseminated via *SSH* to network devices.

We summarize the above techniques in the following table, with emphasis on their adoption and matching granularity they offer:

Mitigation techniques	***Distribution Protocol***	***Granularity***	***Adoption***
Destination–based RTBH	*BGP*	Low	High
Source-based RTBH	*BGP*	Medium	Medium
OpenFlow Firewall	*OpenFlow*	High	Low
Flowspec	*BGP*	High	Low
ACL	*SSH/NETCONF*	High	High

Table 1: Mitigation Techniques

Entries to Table 1 were partially inferred from *DDoS using BGP Flowspec* [24] by Juniper Networks.

IV. Orchestrator of Distributed Rule Placement (*ODRP*): Detailed Architecture

We elaborate upon implementation details pertaining to the architectural components and related sub-modules of the *ODRP*. An indicative setup for our proposed architecture is depicted in Fig. 2 consisting of the components described in *Subsections A*, *B* and *C* below.

A. Pre-processor (PP)

This component consists of four modules: the *Security Events Collector*, the *Reward Evaluator*, the *GAP Composer* and the *Capacity Collector*. These correlate security events, management policies and network environment details and formulate the input of the GAP algorithm.

A.1 Security Events Collector

This module extends the event handling capabilities offered by *Ryu*. It receives alerts generated by an external *IDS* e.g. *Suricata* [25] and processes them to extract relevant information pertaining to a network anomaly. Specifically, the *Security Events Collector* identifies: the exact type of the network anomaly and the network attributes required to create the 6-field tuple (e.g. source IP of the attacker). During a given time window (e.g. every 30 seconds), it generates structured data pertaining to observed alerts aggregated by the attack type and containing all assumed malicious source IPs. Finally, for scalability reasons, such information is conveyed periodically (every time window) to the *Reward Evaluator* module.

A.2 Reward Evaluator

This module receives the information from the *Security Events Collector* and in combination with operator-defined policies maps anomaly types to a specific defense stage among Core, Distribution, Access and Host. We provide below a simplified model based on Event-Condition-Action (ECA) policies that enables network operators to express mitigation policy statements.

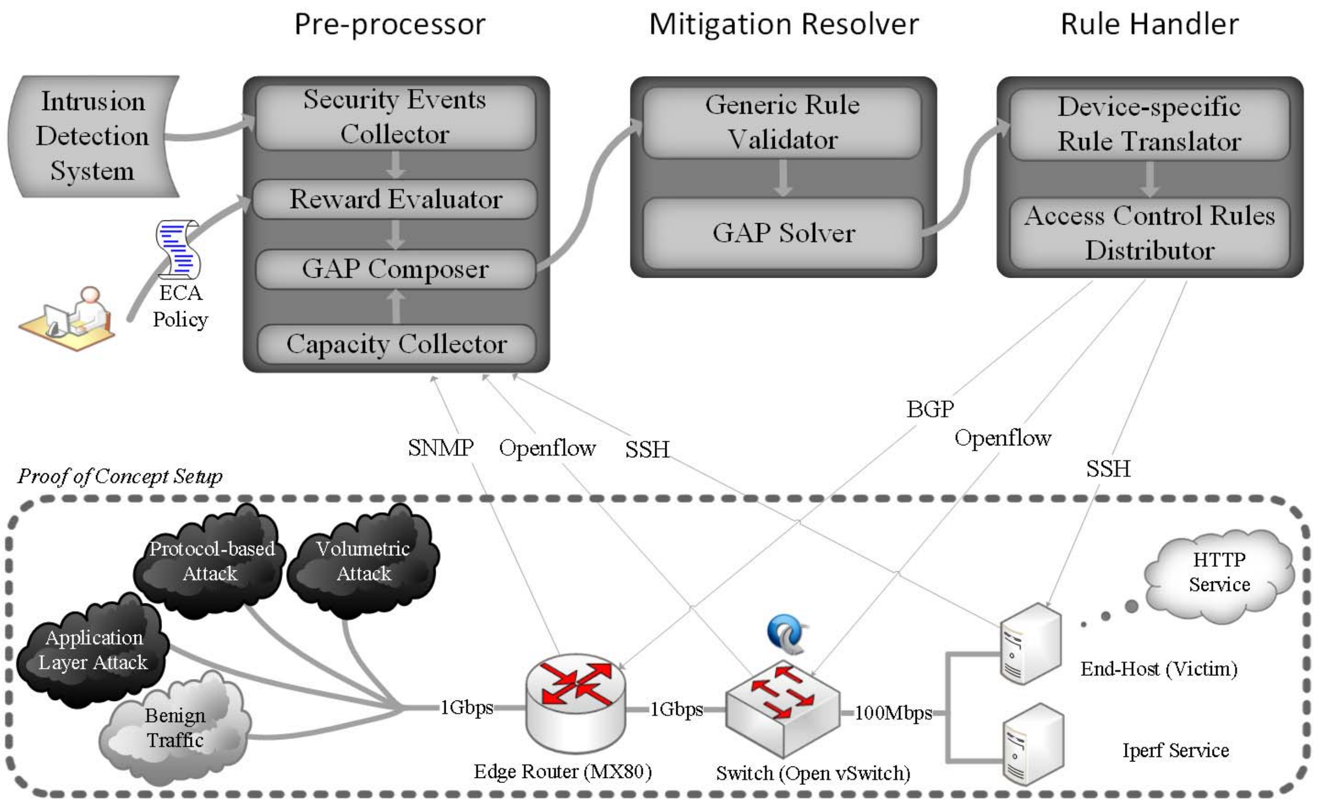

Fig. 2: Orchestrator of Distributed Rule Placement (ODRP): Detailed Architecture

ECA Syntax of high-level security policy
Event = {Network Anomaly Alert}
Condition = {Attack type}
    Attack type = {Volumetric | Protocol | Application}
Action = {Block Core | Block Distribution | Block Access | Block Host}

The security policy above associates anomaly alerts triggered by the *IDS* with a generic mitigation rule conditioned to the attack type. This action selects the stage at which the specific type of attack should be mitigated as defined by the network operator. The mitigation action specified by the security policy is used to generate a reward array for each type of anomaly.

In order to appropriately mitigate multi-vector attacks the *Reward Evaluator* formulates a reward array based on the desired defense stage for each attack vector. For volumetric attacks, it is reasonable to select the most upstream core stage, whereas application layer attacks may be blocked at the downstream access or host stages. Interim cases such as protocol attacks may be handled in transit stages. The *Reward Evaluator* assigns the highest reward to the selected defense stage, while lower values are assigned to the remaining defense stages that may be activated in case the selected stage cannot accommodate all required generic mitigation rules. In case the selected defense stage is the transit distribution stage, we assign the highest value to this stage, a lower interim value to the upstream core stage and the lowest to the downstream access stage. This way, links of the access network, generally of smaller bandwidth than the upstream links, are better protected. Note that, exact reward values are not important, just their relative order.

A.3 Capacity Collector

The *Capacity Collector* module obtains the number of currently installed *ACRs* from routers, switches and end-hosts via *SNMP*, *OpenFlow* and *SSH* respectively, defining a residual capacity vector. The maximum capacities may be estimated based on device specifications and operational experience related to performance degradation per specific network device.

A.4 GAP Composer

This module receives the reward array along with 6-tuple generic mitigation rules, generated per attack type, for all malicious IP sources observed during the previous time window. In addition, the residual capacity vector is received from the *Capacity Collector.* It subsequently exports to the *Generic Rule Validator* (i) generic mitigation rules per attack type, (ii) the reward array and (iii) residual capacities of network devices.

B. *Mitigation Resolver (MR)*

MR receives the generic mitigation rules, computes a solution to the problem formulated as a Generalized Assignment Problem (GAP) and exports to the *Rule Handler*. It consists of two modules: (i) the *Generic Rule Validator* and (ii) the *GAP Solver.*

B.1 Generic Rule Validator

This module compares candidate generic mitigations rules with the ones previously computed and maintained within the module. These are filtered to isolate new malicious source IPs identified during the previous time window.

B.2 GAP Solver

The Generalized Assignment Problem assumes *n* items to be assigned to *m* bins. Assignment of item *j* to bin *i* yields a reward r_{ij} and carries a weight of w_{ij}. A feasible solution is an assignment in which for each bin *i* the total weight of assigned items is at most c_i (the capacity of bin *i*). The goal is to assign each item to exactly one bin in order to maximize the sum of rewards. In our case, *items* are the generic mitigation rules per malicious source IP and attack type, *bins* are the defense stages, *weights* are assumed equal to 1, *capacities* are the available resources of each defense stage and *rewards* are provided by the reward array. The problem can be formulated as an integer program:

$$maximize \sum_{i=1}^{m}\sum_{j=1}^{n} r_{ij}x_{ij} \ (1)$$

$$subject\ to \sum_{j=1}^{n} w_{ij}x_{ij} \leq c_i, i = 1, \dots, m \ (2)$$

$$\sum_{i=1}^{m} x_{ij} = 1, j = 1, \dots, n \ (3)$$

$$x_{ij} \in \{0,1\}, i = 1, \dots, m, j = 1, \dots, n \ (4)$$

As GAP is an NP-hard problem, in order to reduce its solution time, we decrease the size of input to the integer programming algorithm. Namely, we first apply various prefix aggregation techniques [26] on malicious IP sources per attack type. The resulting set of generic mitigation rules are further organized into groups. The group size *g* is defined as the minimum between all the residual capacities across the attack path. In case of an attack requiring a small number of generic mitigation rules (equivalent to a small number of malicious source IPs), the rules are treated as a single group. In the reduced problem, the integer program formulation is transformed with items x_{ij} corresponding to assignments of groups *j* to stages *i* and *n*, the maximum value of *j*, divided by the group size *g*. Weights w_{ij} are equal to the number of the rules they contain (i.e. group size *g*) while upper bounds in (2), (3) are not affected.

The reduced problem is solved via a *branch price and cut* [27] method implemented on *Dippy* [28], a framework for advanced integer programming problems. The computed solution consists of the assigned groups of access control rules to the defense stages of the attack path and is conveyed to the *RH* component.

C. Rule Handler (RH)

The role of the *RH* is to: (i) translate the *MR* solution to device-specific firewall capabilities, (ii) withdraw *ACRs* that are considered obsolete e.g. after two consecutive time windows of no attack reported from a specific source (iii) avoid violation of the existing traffic flows and (iv) maintain and disseminate access control rules on each device. *RH* is configured appropriately to utilize the available southbound protocols (mitigation techniques, see *Section III*) on the specific network environment. It consists of the *Device-specific Rule Translator* and the *ACR Distributor* modules.

C.1 Device-specific Rule Translator

This module maps groups of generic mitigation rules to each network device along the attack path. Specific rules, actual device and southbound protocol are conveyed to the *ACR Distributor*.

C.2 ACR Distributor

This module uploads device specific Access Control Rules (*ACRs*) through a corresponding southbound protocol specified by each mitigation technique. Since our framework should not contradict existing network management policies (e.g. already deployed *OpenFlow* rules, routing preferences, end host firewalls), a tagging mechanism was implemented to separate *ODRP*-generated *ACRs* from existing forwarding rules. Indicatively:

- If the southbound protocol is *BGP*, our *ACRs* should match a predefined route policy which sets a high local preference, based on the IP address of the *BGP*-peer (*i.e.* the *BGP speaker* of *Ryu*).
- For *OpenFlow*-enabled devices we use the optional fields for flow identification: cookie, cookie-mask and flow priority. Our *ACRs* (i.e. *OpenFlow* rules) are tagged with a specific cookie identifier and a specified value for high priority.
- For end-host protection, we consider Linux-based machines which utilize the *iptables* firewall. We created a special-purpose *chain* labeled *ODRP* containing all the *ACRs* inserted by our framework. Their distribution on the end hosts is performed on top of the *SSH* protocol.

V. EXPERIMENTAL EVALUATION

A. Proof of Concept Experimental Setup

In order to validate our proposed framework in Fig. 2, we implemented a proof-of-concept testbed deployed in our laboratory based on *Ryu* Controller. Offered functionalities were implemented as distinct customized applications modules. We incorporated in our testbed specialized network devices, notably a Juniper MX80 router and an Open vSwitch (*OVS*) [29] with *Openflow* capabilities. End-hosts were implemented as Linux Containers (LXC), Virtual Machines and physical hosts. The *Ryu* Controller was customized to support *RTBH* over *BGP* on the MX80 router. Additional southbound interfaces were enabled on *Ryu*, notably *OpenFlow* for the *OVS* and *SSH* for Linux-based end-hosts.

Fig. 2 depicts an enterprise network section with a router placed in the upstream position, a switch as a transit device and Linux-based machines in the downstream end. Connectivity between these devices were selected to relatively represent typical hierarchical enterprise networks. Namely we assumed 1Gbps for upstream links and 100Mbps for the access network downstream.

B. Traffic Profiles for Anomaly Mitigation Experiments

Our testbed was used to experimentally evaluate our framework in multi-vector attack scenarios. The experimentation process considered the following types of traffic: (i) background traffic (benign) generated by the *iperf* traffic generator [30] emulating a dedicated TCP stream with bandwidth demand, (ii) benign HTTP requests via the *wrk* benchmarking tool [31] and (iii) artificially generated attack traffic via commonly used attack tools [32], [33]. Our artificially generated attack was tuned by considering attack characteristics inferred from a publicly available dataset [34]. This dataset contains real attack traffic generated from *Booters* offering *DDoS-as-a-Service*; it was captured during a controlled attack experiment at the University of Twente in collaboration with SURFnet, the Dutch National Research and Education Network.

To better highlight the potential benefits of mitigating the attack in distinct defense stages, we conducted an in-depth analysis of the aforementioned dataset (Booter 9). Specifically, considering the amount of total bytes sent and total packets sent, we clustered the attack sources into 3 distinct groups via the *k-means classifier* [35], as shown in Fig. 3.

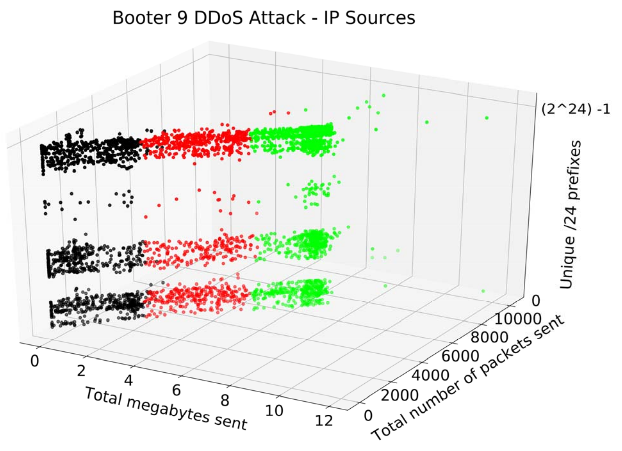

Fig. 3: Malicious sources distribution on the IPv4 address space (unique /24 prefixes) clustered based on the total megabytes/packets sent during the DDoS attack launched by Booter 9 [34].

Note that clustering is commonly suggested to classify network traffic [36] into benign and malicious traffic groups based on various network metrics such as quantity of bytes sent/received. Specifically, in [2] clustering based on malicious source IP prefixes was demonstrated as a means to significantly reduce the number of flow entries.

From Fig. 3 we considered the following mapping of attack types to groups:

- Volumetric attacks correspond to the *green* group exhibiting higher values of bytes per second (bps) and packets per second (pps).
- Protocol-based attacks correspond to the *red* group exhibiting interim values for bps and pps.
- Application layer attacks correspond to the *black* group, characterized by the lower values for bps and pps.

In our experiments, malicious attack traffic was generated via the *bonesi* attack simulator [32] following the mapping above from 3,779 malicious IP sources. In particular, for application layer attacks we assumed malicious HTTP requests that reserve application level resources; this was emulated via the *Slowloris* attack tool [33] from additional 1,500 different unique IP addresses. The resulting attack mix was combined to emulate a multi-vector attack scenario, in which the total amount of unique IP sources was 5,279 and the attack rate was 350 Mbps. Note that we downscaled the attack rate to this value due to the link size limitations and traffic generation constraints of our testbed.

Benign traffic was generated as 200 HTTP requests/sec to an Apache Web Server and a continuous 50 Mbps TCP stream via the *iperf* tool.

C. Experimental Evaluation of Anomaly Mitigation Mechanisms

Our experimental evaluation was conducted during 300 seconds per mitigation mechanism. During the first 30 seconds only benign traffic was present in the network; subsequently we launched the multi-vector attack targeting a victim host. We considered that malicious sources are detected within the next 30 seconds interval. Subsequently, mitigation countermeasures are instantiated (at the 60th second) and remain deployed for the duration of the experiment.

In order to evaluate our framework, we compared three different mitigation mechanisms: (i) *Single Firewall*, (ii) *Arbitrary Distribution* of *ACRs* and (iii) *ODRP* - Orchestrator of Distributed Rule Placement.

We considered that the *Single Firewall* mechanism is implemented with the *OVS* acting as a security middlebox [2]. We assumed that a moderate cost switch, emulated by *OVS*, can adequately support up to 4,000 flow entries.

For the distributed mitigation mechanisms (*Arbitrary Distribution* and *ODRP*), we considered that all network devices across the attack path, Fig. 2, may be used for the mitigation process. The device capacities are 2,000 routes for the router, 2,000 flow table entries for the *OpenFlow*-enabled switch and 1,500 rules for the *iptables* at the end-host firewall. Thus, all 5,279 malicious IP sources of *Subsection B* can be blocked with an appropriate rule distribution.

The *Arbitrary Distribution* of *ACRs*, was modeled by rules assigned in a round robin fashion while respecting the capacity constraints of our devices. Our proposed schema, *ODRP*, translates operator policies into rule assignments to stages based on the attack type. The *ODRP* preference is to block volumetric attacks on the router, protocol-based attacks on the switch and application layer attacks on the end host.

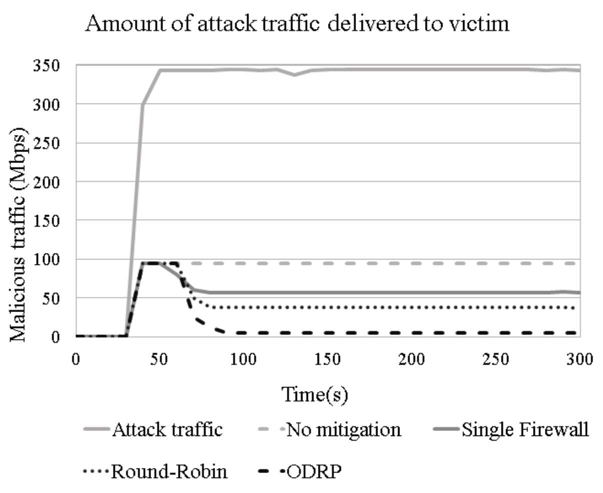

Fig. 4: The total attack traffic reaching the victim.

In Fig. 4 we illustrate the amount of attack traffic that is delivered to the victim while employing different mitigation approaches. The generated attack traffic is characterized by a bandwidth of 350 Mbps. However, only 94 Mbps actually reach the victim since the access link capacity in our testbed is 100 Mbps. While using the *Single Firewall* mitigation mechanism 62 Mbps of the attack traffic reaches the victim, since the installed 4,000 mitigation rules do not block all 5,279 malicious IP sources. The *Round-Robin* placement of rules performs slightly better, blocking all but 40 Mbps of attack traffic. Although this mechanism distributes mitigation rules to block all malicious sources, some were arbitrarily placed on the victim end-host. Thus, part of the attack reached the victim's access network. By contrast, *ODRP* distributes *ACRs* on the most appropriate defense stage, with volumetric attacks blocked early in the attack path. Hence, *ODRP* better protects the victim from malicious traffic.

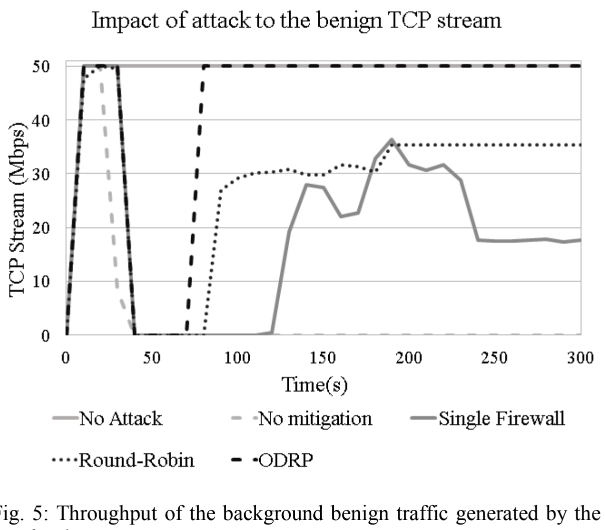

Fig. 5: Throughput of the background benign traffic generated by the *iperf* tool.

In Fig. 5 we depict the impact of the attack on a benign TCP stream by plotting its throughput in various scenarios. The baseline stream value with no attack present is 50 Mbps. Unmitigated attacks result in TCP stream throughput dropping to almost 0 Mbps, due to congestion caused by the attack. The *Single Firewall* mechanism is able to block 4,000 of the malicious sources, thus the TCP stream is stabilized at 18 Mbps (36% from its baseline value). Placement of *ACRs* in an *Arbitrary Round-Robin* fashion improves performance with rates varying from 30 to 33 Mbps, (60% and 66% respectively). Finally, *ODRP* outperforms both previous mechanisms, since volumetric and protocol-based interference is blocked early in the attack path, thus preserving the bandwidth of the access link utilized by the TCP stream.

In Fig. 6 we present an evaluation of the different mitigation mechanisms based on the percentage of successful HTTP transactions to a Web service running on the victim. Such a transaction was considered successful if completed within 1 second. Naturally with no attack injected, the percentage of successful HTTP transactions is 100%. Generated attacks had a two-fold impact: (i) HTTP Requests are dropped due to congestion and (ii) active sessions are consumed by maliciously crafted packets of *Slowloris* application-layer attack.

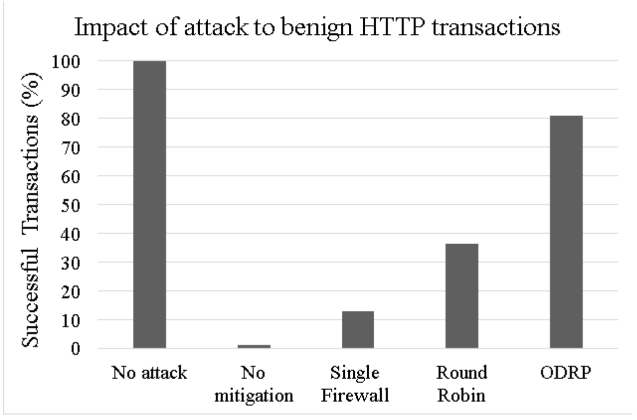

Fig. 6: The impact of the attack to benign HTTP transactions using as indicator the percentage of successful HTTP transactions.

Thus, unmitigated attacks exhibit a very low percentage of successful transactions, close to 1%. The amount of attackers blocked by the *Single Firewall* is not sufficient; unblocked attackers still manage to consume valuable bandwidth and server resources. The *Round-Robin* placement performs considerably better at 36% success. Finally, *ODRP* significantly outperforms both mechanisms reaching to 80% success. Note that in general, the effect of *Slowloris*-based attacks on the HTTP service persists even after blocking all malicious sources. This occurs because sessions reserved under false pretenses, stay open for a period of time even after a *Slowloris* attack is mitigated. Thus legitimate transactions attempted in the meantime might still fail. This affects every mechanism used in our experiment; including our *ODRP* approach that could not exceed the 80% successful transaction rate.

D. Complexity of Generalized Assignment Problem

The efficiency of our *ODRP* approach depends on solving GAP in a fraction of a detection time window. As already mentioned, GAP is an NP-hard problem, thus it may present scaling issues according to its input size. To address them, we have implemented a customized solver whereby the generic mitigation rules (*items*) are split into groups and assigned to 4 defense stages (*bins*). We are presenting below the execution time of our solver considering indicative values for the number of generic mitigation rules and possible groupings.

Group Size	Generic Mitigation Rules		
	1000	5000	10000
1	0.5	2.74	5.72
10	0.09	0.25	0.56
50	0.008	0.05	0.1
100	0.004	0.03	0.05

Table 2: GAP Execution Time in seconds

The results above, demonstrate that the grouping technique we employed, reduces the execution time of GAP solution in reasonable values within a 30 second time window. Even the worst case scenario (10,000 generic mitigation rules in 10,000 groups) performs in 5.72 seconds. In a case that a large botnet is used to launch an attack e.g. 100,000 malicious sources require 100,000 generic mitigation rules to block the attack. With a group size of 100 items, the assignment problem would be solved in approximately 6 seconds. Thus, our proposed *ODRP* approach is expected to solve GAP in a fraction of a detection time window for attacks emanating from a large number of unique malicious sources.

VI. Conclusions

In this paper we proposed a framework for mitigating detected multi-vector anomalies in typical enterprise networks via the distribution of Access Control Rules. Our approach emphasized on utilizing the capabilities offered by all devices present in an attack path, organized into distinct defense stages. Generic mitigation rules were assigned to each stage according to the defense preferences expressed by operators for specific attack types. Ultimately, device-specific access control rules were distributed to appropriate defense stages of the network substrate via commonly used protocols.

Our architecture abstracted rule assignment to defense stages by modeling the attack mitigation as a Generalized Assignment Problem (GAP): *Items* (generic mitigation rules) were assigned to *bins* (defense stages) based on *capacity* constraints and *reward* values, inferred from operator policies. To enable reasonable execution of the GAP algorithm we considered applying prefix-aggregation of the malicious IP sources. This reduced the number of mitigation rules (*items*). Further reduction of GAP input size may be achieved by organizing the rules into groups that are collectively assigned to defense stages (*bins*).

Our mitigation mechanism was validated by implementing a proof of concept setup in our laboratory that was also used in experimental evaluation of realistic heterogeneous attack scenarios. Our experiments demonstrated that intelligent distribution of access control rules yields noticeable improvements compared to rigid mitigation mechanisms.

As future work, we plan to investigate the applicability of our architecture in NFV-enabled cloud infrastructures whereby devices may be elastically deployed to mitigate network anomalies. This approach suits dramatic attack scenarios for which mitigation rule requirements exceed device capacity constraints. To that end, extensions might also include the integration of collaborative architectures as in [37]. Thus, our approach will consider extensions featuring smart contracts and reputation schemas, towards forging federations of autonomous systems to collaboratively enhance attack mitigation mechanisms.

References

[1] "DDoS Mon", available at: https://ddosmon.net, last visited July 2018

[2] K. Giotis, G. Androulidakis and V. Maglaris, "A Scalable Anomaly Detection and Mitigation Architecture for Legacy Networks via an OpenFlow Middlebox", in Security and Communication Networks, pp. 1958-1970, 2015

[3] "Thunder TPS", available at: https://www.a10networks.com/products/thunder-series/ddos-detection-protection-mitigation , last visited July 2018

[4] S. K. Fayaz, Y. Tobioka, V. Sekar and M. Bailey, "Bohatei: Flexible and Elastic DDoS Defense", in USENIX Security Symposium, pp. 817-832, August 2015

[5] "DDoS Protection services", available at: https://www.incapsula.com/ddos-protection-services.html, last visited July 2018

[6] "DDoS Protection from Akamai", available at: https://www.akamai.com/uk/en/resources/ddos-protection.jsp, last visited July 2018

[7] J. Deng *et al.*, "VNGuard: An NFV/SDN Combination Framework for Provisioning and Managing Virtual Firewalls", in IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN), pp. 107–114, 2015

[8] Carol J. Fung and B. McCormick, "Vguard: A Distributed Denial of Service Attack Mitigation Method using Network Function Virtualization", in Network and Service Management (CNSM), 2015 on 11th International Conference, pp. 64–70, 2015

[9] "Gatebot", available at: https://blog.cloudflare.com/meet-gatebot-a-bot-that-allows-us-to-sleep, last visited July 2018

[10] G. Ross and R. Soland "A Branch and Bound Algorithm for the Generalized Assignment Problem", Mathematical programm, 8(1), pp. 91-103, 1975

[11] "Hierarchical Internetworking Model" , available at: https://en.wikipedia.org/wiki/Hierarchical_internetworking_model, last visited July 2018

[12] "Fortinet Security Fabric Lays the Foundation for Intent-Based Network Security", available at: https://www.fortinet.com/corporate/about-us/newsroom/press-releases/2017/fortinet-security-fabric-lays-the-foundation-for-intent-based-network-security.html, last visited July 2018

[13] "Ryu the Network Operating System (NOS)", available at: https://ryu.readthedocs.io/en/latest, last visited July 2018

[14] J. Medved, R. Varga, A. Tkacik, and K. Gray, "OpenDaylight: Towards a Model-driven SDN Controller Architecture", in Proc. IEEE Int. Symp.World Wireless, Mobile Multimedia Networks, pp. 1–6, June 2014

[15] P. Berde *et al.*, "ONOS: Towards an Open, Distributed SDN OS", in Proc. 3rd Workshop Hot Topics Software Defined Networking, 2014

[16] D. Turk, "Configuring BGP to Block Denial-of-Service Attacks", RFC 3882, available at: http://www.ietf.org/rfc/rfc3882.txt, September 2004

[17] W. Kumari and D. McPherson, "Remote Triggered Black Hole Filtering with Unicast Reverse Path Forwarding (uRPF)", RFC 5635, available at: http://www.ietf.org/rfc/rfc5635.txt, August 2009

[18] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS Defense Mechanisms", in ACM SIGCOMM Computer Communication Review, vol. 34(2), pp. 39-53, 2004

[19] P. Marques, N. Sheth, R. Raszuk, B. Greene, J. Mauch and D. McPherson, "Dissemination of Flow Specification Rules", RFC 5575, available at: http://www.ietf.org/rfc/rfc5575.txt, August 2009

[20] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shrenker and J. Turner,. "OpenFlow:

enabling Innovation in Campus Networks", in ACM SIGCOMM Computer Commun. Review, 38(2), pp.69-74, March 2008

[21] "Access Control List", available at: https://en.wikipedia.org/wiki/Access_control_list, last visited July 2018

[22] "Capirca: Multi-platform ACL Generation System", available online: https://github.com/google/capirca, last visited July 2018

[23] "Saltstack Platform", available at: https://saltstack.com, last visited July 2018

[24] "DDoS using BGP Flowspec", available at: https://www.slideshare.net/apnic/ddos-mitigation-using-bgp-flowspec, last visited July 2018

[25] "Suricata Open Source IDS / IPS / NSM Engine", available at: https://suricata-ids.org, last visited July 2018

[26] F. Soldo, K. Argyraki, and A. Markopoulou, "Optimal Source-based Filtering of Malicious Traffic", IEEE/ACM Transactions on Networking, vol. 20, no. 2, pp. 381–395, April 2012

[27] C. Barnhart, E. Johnson, G. Nemhauser, M. Savelsbergh and P. Vance, "Branch-and-Price: Column Generation for Solving Huge integer programs", Operations Research, vol. 46, pp. 316–329, 1998

[28] M. O'Sullivan, Q. S. Lim, C. Walker, I. Dunning and S.Mitchell. "Dippy: A Simplified Interface for Advanced Mixed-integer Programming", Report 685, University of Auckland Faculty of Engineering, 2011

[29] B. Plaff, J. Pettit, T. Koponen, K. Amidon, M. Casado and S. Shenker, "Extending Networking into the Virtualization Layer", in 8th ACM Workshop on Hot Topics in Networks (HotNets-VIII), New York City, 2009

[30] "Iperf - The TCP/UDP Bandwidth Measurement Tool", available at: https://iperf.fr/, last visited July 2018

[31] "wrk, Modern HTTP Benchmarking Tool", available at: https://github.com/giltene/wrk2, last visited July 2018

[32] "The DDoS Botnet Simulator", available at: https://github.com/Markus-Go/bonesi, last visited July 2018

[33] "Low Bandwidth DoS Tool", available at: https://github.com/gkbrk/slowloris, last visited July 2018

[34] J.J. Santanna *et al*. "Booters—An Analysis of DDoS-as-a-Service Attacks", Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium, July 2015

[35] Anil K. Jain, "Data Clustering: 50 years beyond K-means", Pattern Recognition Letters, 31(8), pp. 651-666, 2010

[36] L. Keunsoo, K. Juhyun, H. K. Ki, H. Younggoo and K. Sehun, "DDoS attack detection method using cluster analysis", in Expert Systems with Application vol 34, pp. 1659-1665, April 2018

[37] K.Giotis, M. Apostolaki, and V. Maglaris, "A Reputation-based Collaborative Schema for the Mitigation of Distributed Attacks in SDN domains", in IEEE/IFIP Network Operations and Management Symposium, April 2016