JESS: Joint Entropy-Based DDoS Defense Scheme in SDN

Kübra Kalkan, Levent Altay, Gürkan Gür, and Fatih Alagöz

***Abstract*— Software-defined networking (SDN) is a communication paradigm that brings cost efficiency and flexibility through software-defined functions resident on centralized controllers. Although SDN applications are introduced in a limited scope with related technologies still under development, operational SDN networks already face major security threats. Therefore, comprehensive and efficient solutions are crucial. Especially, large-scale security threats such as distributed-denial-of-service (DDoS) attacks are jeopardizing safety and availability of data and services in these systems. A DDoS attack is aimed at making resources unavailable to legitimate users via overloading systems with excessive superfluous traffic from distributed sources. In this paper, we describe and evaluate a joint entropy-based security scheme (JESS) to enhance the SDN security with the aim of a reinforced SDN architecture against DDoS attacks. In particular, our proposed model devises a statistical solution to detect and mitigate these hazards. To the best of our knowledge, JESS is the first model that utilizes joint entropy for DDoS detection and mitigation in the SDN environment. Since it relies on a statistical model, it mitigates not only known attacks but also unfamiliar types in an efficient manner.**

***Index Terms*— SDN, network security, DDoS, filtering, network defense mechanisms.**

I. Introduction

SOFTWARE-DEFINED Networking (SDN) is a key paradigm that is envisaged to meet burgeoning demands and requirements of Future Internet while addressing the challenges of current networking and communications infrastructure. It enables central management of networks by separating forwarding and control planes and provides network programmability. However, this innovation also introduces some security vulnerabilities. In addition to the single-point-of-failure problem which is introduced by central management, SDN provides new attack vectors such as the controller itself or the links between data plane elements and the controller [1]. In that regard, information security is yet to be adequately addressed since SDN is a novel concept and academic studies have especially aimed at providing a functional SDN infrastructure [2]–[5]. Accordingly, recent surveys indicate that security is one of the most critical problems in the softwarized networks [6]–[9].

In that vein, DDoS attacks provide an effective way for attackers to compromise the availability of these systems. SDN environment is favorable for such attacks since the centralized controller (or a controller cluster in a more advanced setting) inherently manages the network and constitutes a high value target as a potential single-point-of-failure [10]. When a packet comes from an unmatched flow to a switch, the switch forwards it to the controller as the default behavior. Then, the controller sends back a flow rule for this flow according to the logic defined in the network application(s). If attackers send a large number of packets from several IP addresses, implying unknown flows for switches, these packets are forwarded to the controller. Then excessive amount of attack packets will consume available communication-, computation- and storage-related resources of the controller and make the system unavailable for legitimate users. Likewise, that attack can cripple the system by capturing the forwarding table capacity of the switches. When the switch receives a vast number of spoofed packets, illegal traffic totally occupies the table space. Furthermore, the link between the switch and controller can become unavailable because of congestion due to this malicious traffic.

Several solutions against DDoS attacks are proposed in the literature for software-defined networks. They are classified as intrinsic and extrinsic solutions [11]. Intrinsic solutions [12]–[17] are related to structural attributes of SDN environment, whereas extrinsic solutions [18]–[22] mostly rely on the properties of traffic flows in the network. Since intrinsic solutions need structural changes they need more fundamental alterations in SDN environment, which is not preferable for feasibility and compatibility. The latter class, extrinsic solutions, can be applicable to existing SDN systems since their implementation requirements are related to network flows. Those can be further categorized into two groups — statistical vs. machine learning based. Statistical solutions are more favorable from the performance perspective since machine learning based approaches have heavier computational burden and challenging data requirements. However, existing statistical solutions exhibit several drawbacks such as having only rate-limiting or detection functions but lacking the mitigation capability. Thus, it is essential to have a statistical DDoS detection and mitigation mechanism for software-defined networks

Manuscript received May 2, 2018; revised August 18, 2018; accepted August 28, 2018. Date of publication September 17, 2018; date of current version November 28, 2018. This work was supported in part by the Scientific and Technical Research Council of Turkey (TUBITAK) under grant number 117E165 and in part by the Turkish State Planning Organization (DPT) under the TAM Project, number 2007K120610. *(Corresponding author: Kübra Kalkan.)*

K. Kalkan is with the Department of Computer Science, University of Oxford, Oxford OX1 2JD, U.K. (e-mail: kubra.kalkan@cs.ox.ac.uk).

L. Altay and F. Alagöz is with the Department of Computer Engineering, Bogazici University, Istanbul 34342, Turkey (e-mail: levent.altay@boun.edu.tr; fatih.alagoz@boun.edu.tr).

G. Gür is with the Institute of Applied Information Technology (InIT), Zurich University of Applied Sciences (ZHAW), 8401 Winterthur, Switzerland (e-mail: gueu@zhaw.ch).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSAC.2018.2869997

which has reasonable computational complexity and data requirements [11].

Entropy is a statistical measure that shows the randomness in a data set [23]. Since an attacker sends similar type of packets and these illegal packets make up most of the traffic during a DDoS attack, randomness in packet properties necessarily decreases. For this reason, entropy calculation is a useful tool for DDoS detection. Since this technique relies on statistical calculation, it does not put a heavy burden on the system unlike machine learning based approaches. Thus, this paper builds up on our previous preliminary work [24] and proposes a joint entropy-based scoring system (JESS) to detect and mitigate DDoS attacks. It utilizes joint entropy as a statistical tool to provide defense capability. Our key contributions in our work can be summarized as follows:

- a statistical mitigation method that selects the most appropriate attributes for tackling with the active attack.
- dynamic rule generation by the controller for each switch in a way which does not significantly tax the switches.
- utilization of joint entropy calculation for not only detection, but also mitigation against DDoS attacks. According to the best of our knowledge, this is the first integration of joint entropy metric into DDoS defense for mitigation in SDN.
- mitigation for not only known but also unfamiliar attacks that utilize packets with similar attributes.
- a tunable accuracy capability in defense mechanism such that systems with sufficient resources can utilize an ordered profile generation to provide more accurate nominal profiles and accordingly more effective results.

In the next section, we provide an overview of related work on defense mechanisms against DDoS attacks in SDN. In Section III, we describe our proposed defense scheme JESS focusing on its components and stages. Section IV introduces the system setup and presents experimental results followed by the performance evaluation. We also discuss different aspects of JESS including, *inter alia*, parameter selection, scalability and overhead. Finally, Section V concludes the paper.

II. Related Work

The frequency and magnitude of DDoS attacks have been constantly increasing with detrimental impact on information and communication systems [25]. Accordingly, there is a burgeoning body of research in that topic. In this section, we focus on SDN and network virtualization related works and discuss the relevant literature in two parts. Firstly, the studies related to network virtualization and virtual security are examined. Then DDoS defense mechanisms in SDN are reviewed to render the state-of-the-art in that field.

A. Network Virtualization and Virtual Security

Network virtualization (NV) separates the roles of supervision the physical infrastructure and creating virtual networks by combining resources from Internet service and infrastructure providers [26]. NV represents a software-based network by combining Network Function Virtualization (NFV) and SDN concepts [27] and initiates two novel foundations, network softwarization and network slicing notions. Network softwarization organizes, manages, and re-configures network components while network slicing guarantees the equitable sharing of the services by separating network resources logically and physically. Looking at the current studies, new approaches are emerging for the efficient operation of transport networks by combining SDN and NFV, and consequently, the mentioned softwarization and slicing technologies. However, the construction and design of these novel approaches will bring emerging problems together. Especially, slicing the network brings various levels of security required by individual slices, separated technological domains of the slices, and orchestration of various aspects [28], [29].

One of the proposed solutions against specific security threats is virtual security [30] ensuring the ideal resource management to enable network slicing efficiently. Besides, virtual security provides efficient restoration mechanism to overcome failure of network functions to ensure service resilience [31]. In order to support network virtualization and ensure the security of transport networks it is necessary to solve the fundamental issues which are already present. Especially against the increasing DDoS attacks in local networks and data centers, measures, similar to our proposition, will also help to ensure the healthy functioning of virtualization environments in softwarized networks.

B. DDoS Defense in SDN

Various works are presented in the literature for DDoS defense in SDN. AvantGuard [19] is a framework to improve the security against DDoS and provides resiliency and attack detection with greater involvement of switches. AvantGuard introduces two modules on switches: *connection migration (CM)* and *actuating triggers (AT)*. CM proxies and classifies TCP SYN requests. If it regards requests as proper, they are authorized and transferred to the destination. This mechanism provides protection against SYN Flood attacks. If CM realizes an enormous volume of requests, AT module triggers an event in the controller to insert a flow rule into the flow table automatically so that AvantGuard reduces response time. The most remarkable side effect of this mechanism is the performance penalty. Since it utilizes connection migration, each flow needs to be classified. Additionally, this module can handle only defense against a single type of DDoS attack (TCP SYN Flood).

The communication between SDN controller and switches can also be a vulnerability enabling attack impact. Scotch by Wang *et al.* [21] is a mitigation method that scales up SDN control plane using virtual switch (vSwitch) based overlay. The authors did an experiment to find where the real problem is in the case of DDoS attacks or flash crowds. Their results indicate that the bottleneck is the communication from a switch to the controller. When the network traffic overloads the physical switch, this model tunnels new flows to multiple vSwitches.

Piedrahita *et al.* [18] propose the FlowFence mechanism against DDoS attacks. In this work, network switches monitor traffic and detect the congestion condition by checking the bandwidth utilization. When congestion occurs, the switch notifies the controller. Moreover, the controller requests statistics from every switch that sends packet flows to that congested link. It sorts out the packet flows and determines misbehaving ones, using their excessive bandwidth utilization. Then, the controller sends commands to switches to rate-limit those flows. FlowFence results present a fast and straightforward solution that prevents starvation. Nevertheless, this mechanism is specifically a rate-limiting mechanism that does not halt an attack completely.

Some of the works in the literature such as [18]–[20] suggest to integrate some additional capabilities to SDN switches. The motivation for this approach is based on the idea of keeping flows in the data plane as much as possible. When more flows are forwarded to the controller, it is more prone to attacks by malicious users. If switches become more capable on flow decisions, it will be safer for the controller. In that regard, our previous work SDNScore [24] is a packet-based statistical model that relies on capable switches. It analyzes each packet and calculates scores according to their attribute values similar to the work in ScoreForCore model [32]. In SDNScore, switches can gather statistics and determine if DDoS attack is in action, rather than being simple data forwarders. Moreover, they coordinate with the controller and make a decision on the attack packets in cooperation. However, the "capable switch" concept is a controversial issue in the literature since it brings some capabilities to switches whereas SDN's key rationale is to provide decisions in a centralized manner with a very streamlined data plane. Therefore, in this work, we devise a novel approach that also addresses this drawback and propose a joint entropy based defense scheme that translates the main burden/requirements related to DDoS defense from switch to the controller.

Various statistics-based solutions for DDoS defense are proposed in the literature. One of them is presented in [33] where Sahay *et al.* propose an SDN-based autonomic mitigation framework (ArOMA). They utilize dynamic programmability and global view features of SDN. In their work, the detection function is provided by switches whereas mitigation is provided by the controller which provides quick response for recovering benign traffics performance. Thus, the controversial issue related to "capable switch" concept is also applicable for this paper. Furthermore, it can only mitigate the known attacks such as UDP, TCP/SYN, and ICMP flood attacks. Another statistics-based approach is presented in [34]. In their work, Huong et al. propose a DDoS defense method called SDN Onepacket DDoS Mitigation (SODM). This method drops all one-packet flows as soon as DDoS attack is suspected. They introduce a security analyzer which is utilized to analyze the traffic flow statistics and identify some malicious indicators. However, this additional entity brings new security attack vectors which are the link between the security analyzer and controller and the link between the security analyzer and switches. Additionally, SODM drops all one-packet flows if an attack is identified. They provide a case analysis and state that normal flows have very small number of one-packet flows. However, it can cause serious problems since the most of the control flows have only one packet but they contain critical messages. In that sense, the most attractive property of our work JESS is the ability of providing defense for not only known but also unfamiliar attacks while protecting most of the benign traffic, with the help of the dynamic attribute selection property.

There are several works that utilize entropy in traditional networks such as [23], [35], and [36]. However, there are only a few works in SDN domain. One of them is [37] that proposes an early detection scheme of DDoS attacks against SDN controllers. This mechanism runs on the controller and calculates entropy by considering destination IP address. If entropy decreases below a threshold, a DDoS attack is detected. Although, it enables to determine the victim, it is not possible to dissociate the legal packets from the attack ones. Another similar entropy-based lightweight DDoS flooding attack detection model runs on the OpenFlow (OF) edge switch in [20]. This model achieves a distributed anomaly detection in SDN and reduces the flow collection overload on the controller. Another work [38] proposes a detection and mitigation method that utilizes entropy calculation. Giotis *et al.* [38] also separate the data gathering function from the controller by utilizing sFlow [39] packet sampling. They provide reduction in data gathering with sampling and use entropy for anomaly detection. Their model runs on the controller, and their mitigation strategy is to cut off all the flows to the host under attack. However, this approach does not protect the legal packets.

In JESS, we utilize joint entropy for both detection and mitigation. We do not only focus on destination IP address entropy, but also the combinations of IP address and TCP layer attributes. Several hosts under a switch can be the victims at the same time; thus considering solely destination IP address does not always give accurate results. A reliable method should also take the other attributes into account. For instance, for TCP SYN Flood attack, protocol type and TCP flag are the arbiters. These attributes give possible signs of potential malicious activity. On the other hand, in SQL Slammer Worm, DNS and NTP attacks, the protocol type, destination port and packet size properties of the traffic are the identifiers. Thus, to be ready for an unfamiliar attack, a model should not stick to predetermined attributes statically. Instead it should be able to select appropriate attributes dynamically for the current attack. For this reason, joint entropy is utilized during the appropriate pair selection in our model. It allows JESS to function in an independent manner from statically-set statistical parameters of an attack (i.e. not just addressing familiar attack types) despite being a statistics-based method.

III. Joint Entropy Based DDoS Defense Scheme in SDN (JESS)

JESS has three main phases: nominal, preparatory, and active mitigation stages. It creates nominal information in a non-attack period named as nominal stage. When the traffic bandwidth exceeds the nominal threshold, the second phase called preparatory stage goes into action. In this phase, the DDoS attack is detected and relevant parameters are determined. Then the last phase denoted as active mitigation stage which defends the system during the attack goes online. All these stages are shown in Figure 1 and 2.

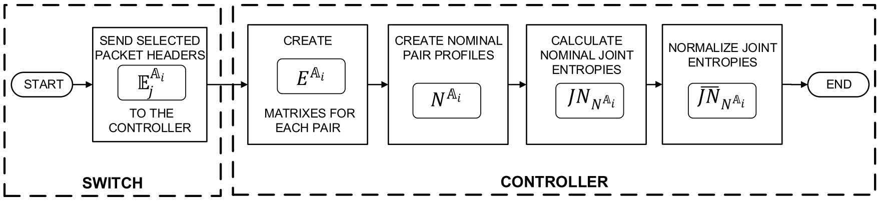

START
SEND SELECTED PACKET HEADERS
$\mathbb{E}_j^{\mathbb{A}_i}$
TO THE CONTROLLER
CREATE
$E^{\mathbb{A}_i}$
MATRIXES FOR EACH PAIR
CREATE NOMINAL PAIR PROFILES
$N^{\mathbb{A}_i}$
CALCULATE NOMINAL JOINT ENTROPIES
$JN_{N^{\mathbb{A}_i}}$
NORMALIZE JOINT ENTROPIES
$\overline{JN}_{N^{\mathbb{A}_i}}$
END
SWITCH
CONTROLLER

Fig. 1. Nominal stage in JESS.

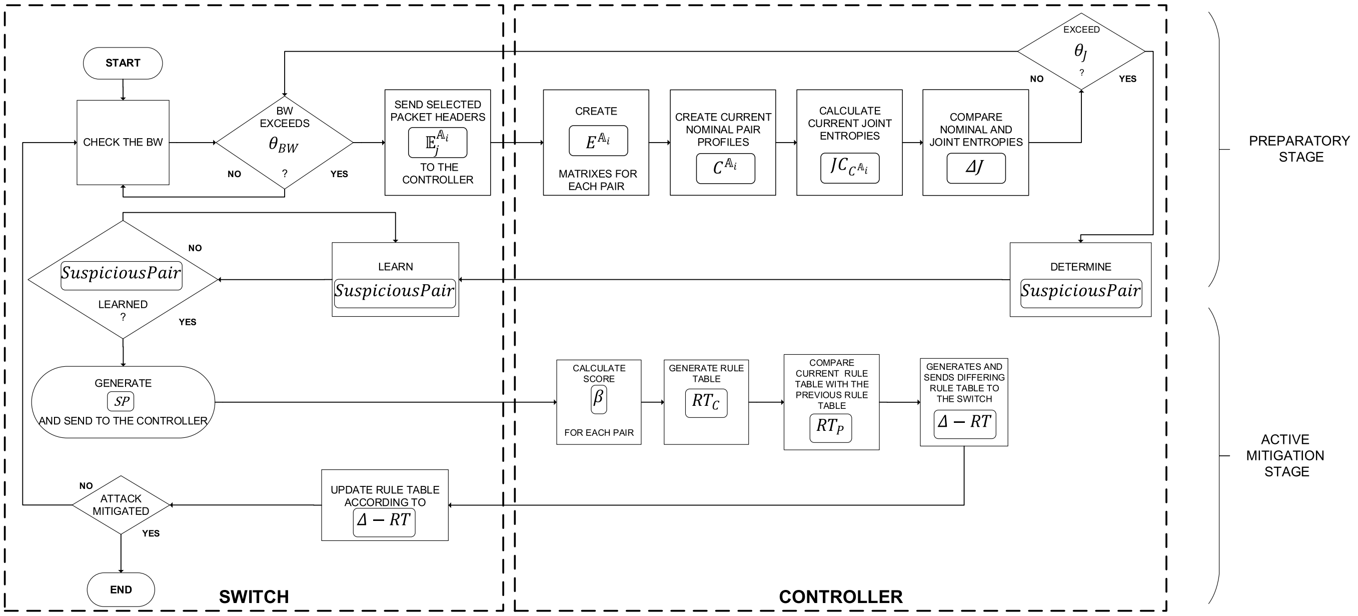


Fig. 2. Overall operation of JESS.

Before going into the details of JESS operation, we first provide a higher level description to render the overall scheme as follows:

- In an attack-free period called *nominal stage*, baseline information is generated by creating nominal pair profiles for each attribute pair. During this period, the switch sends headers of all packets to the controller. At the end of nominal stage, when the nominal profiles are ready, the controller calculates the joint entropies of each pair.
- After a congestion is detected, *preparatory stage* is commenced. The switch starts to send the headers of packets to the controller. The controller generates the current pair profiles for each attribute pair and calculates the joint entropies. If any of the difference exceeds θ_J, a DDoS attack is detected. Then, the pair which has the maximum difference is determined as the $SuspiciousPair$. This generated information is sent to the switch.
- When the $SuspiciousPair$ is determined, *active mitigation stage* is started. This phase consists of profile update periods and continues until the DDoS-driven congestion is mitigated. During a period in this stage, the switch creates current profile SC of identified $SuspiciousPair$ and at the end of each period it sends SC to the controller. The controller calculates the corresponding score for each entry in the profile and generates a score table ST. Subsequently, a current rule table RT_C is generated by comparing the score of each pair with a designated threshold θ_S. If a score for an entry is under this threshold, the corresponding rule is determined as “forward”, otherwise it is a “drop” rule. Then, the current rule table RT_C is compared with the previous rule table RT_P. To minimize update traffic for rule table management, only the differing rules $\Delta - RT$ are sent to the switch since they have changed after the previous period. Then the switch updates its rule table according to $\Delta - RT$.

The relevant terms and parameters in JESS are described in Table I with details explained in the following subsections.

A. Nominal Stage

This stage consists of the preparation steps before the attack, where JESS generates the necessary information that will be used during an attack period. Particularly, nominal information is gathered in an attack-free period to be utilized for comparisons in the attack period to detect traffic anomalies. The pair nominal profiles and joint entropy for each pair are prepared in this stage. We describe how this information is obtained in the following parts.

1) Nominal Profile Generation: A profile is a set of data representing the significant features of an entity. The controller examines specific fields of the TCP and IP packet headers

TABLE I
SYSTEM PARAMETERS

Term	Explanation
$\mathbb{P}$	Collection of the used TCP and IP packet headers
k	The number of elements in a subset of $\mathbb{P}$ ($k = 2$ in our experiments)
$\mathbb{A}_i$	ith element of $\binom{\mathbb{P}}{2}$
$E^{\mathbb{A}_i}$	A matrix of properties for $\mathbb{A}_i$ in a period
$E_j^{\mathbb{A}_i}$	jth parsed packet in $E^{\mathbb{A}_i}$
α	The number of packets in a period ($\alpha = 1000$ in our experiments)
PAC	The number of packets with corresponding properties of $E_j^{\mathbb{A}_i}$
$N^{\mathbb{A}_i}$	Pair nominal profile for the ith attribute pair
$N_p^{\mathbb{A}_i}$	Order of p pair nominal profile for the ith attribute pair ($p = 1$ in our experiments)
m	The number of unique $\mathbb{A}_i^j$'s
$C^{\mathbb{A}_i}$	Current profile for the ith attribute in an attack period
$JN_{N^{\mathbb{A}_i}}$	Joint entropy of a nominal profile for ith attribute pair
$JC_{N^{\mathbb{A}_i}}$	Joint entropy of a current profile for ith attribute pair
$SuspiciousPair$	Attribute pair with the most probable signs for the current attack
SN	Nominal profile of the $SuspiciousPair$
SC	Current profile of the $SuspiciousPair$
PN	Fraction of packets having $\mathbb{A}_i^n$ attribute pair in a nominal period
PC	Fraction of packets having $\mathbb{A}_i^n$ attribute pair in a current period
$\Delta J_{A,B}$	Difference between joint entropy of current and nominal profiles for the pair A and B
β	Score value of an entry
ST	Score table
RT_C	Current rule table
RT_P	Previous rule table
$\Delta - RT$	Differing rule table
θ_{BW}	Threshold value for the bandwidth
θ_J	Threshold value for the joint entropy
θ_S	Threshold score value for packet discarding
θ_C	Threshold score value of the current period
θ_P	Threshold score value of the previous period
ϕ	Acceptable traffic volume
ψ	Total current incoming traffic

to create nominal profiles. In that regard, some of the most important fields of TCP and IP packet headers are: source IP address (IP_{src}), destination IP address (IP_{dst}), source port (P_{src}), destination port (P_{dst}), protocol type ($PROT$), packet size (PKT_{size}), Time To Live value (TTL), and TCP flag (TCP_{flag}). Unordered collection of these fields can be represented as

$$\mathbb{P} = \{IP_{src}, IP_{dst}, P_{src}, P_{dst}, PROT, PKT_{size}, TTL, TCP_{flag}\}.$$

For $1 \leq k \leq |\mathbb{P}|$ and $k \in \mathbb{N}$, k-element subsets of $\mathbb{P}$, $\binom{\mathbb{P}}{k}$ is the set of all possible profiles based on these selected fields. For instance, for $k = 2$, $\binom{\mathbb{P}}{2} = \{\{IP_{src}, IP_{dst}\}, \{IP_{src}, P_{src}\}, ..., \{TTL, TCP_{flag}\}\}$. Besides, each element of $\binom{\mathbb{P}}{2}$ has different instances according to the corresponding packet field definition. Let $\mathbb{A}_i$ be i^{th} element of the $\binom{\mathbb{P}}{2}$ and $i = 1, 2, .., \left|\binom{\mathbb{P}}{2}\right|$. For example the possible number of elements in $\mathbb{A}_1$ and $\mathbb{A}_2$ can be shown as $|\mathbb{A}_1 = \{IP_{src}, IP_{dst}\}| = 2^{32} * 2^{32}$, $|\mathbb{A}_2 = \{IP_{src}, P_{src}\}| = 2^{32} * 2^{16}$, and so on.

A switch captures and parses the TCP/IP packets then forwards the field information to the controller. The time period for this operation is determined according to the designated total number of received packets. In PacketScore scheme [40], the authors made comparison in terms of time based vs. packet based period determination. Their results suggest that packet volume based intervals are more effective. Time-based window may not allow creation of meaningful profiles since it may not have sufficient amount of packets. Since our model utilizes a scoring technique similar to [40], we choose packet based period determination and process α packets which form a period. For α packets in a nominal period, ensemble of parsed packets can be represented as an $\alpha * k$ matrix, $E^{\mathbb{A}_i}$. $\mathbb{E}_j^{\mathbb{A}_i}$ is the j^{th} parsed packet and it has a_j and b_j values for the corresponding attributes. Alternatively, it can be represented as the following tuple, $\mathbb{E}_j^{\mathbb{A}_i} = (a_j, b_j)$. Each row of the matrix corresponds to $\mathbb{E}_j^{\mathbb{A}_i} \in \mathbb{A}_i$. Let $\mathbb{A}_i \in \binom{\mathbb{P}}{2}$, $i = 1, 2, 3, .., \left|\binom{\mathbb{P}}{2}\right|$, $j = 1, 2, 3, ..., \alpha$, and α be the number of captured packets in a nominal period. Then the ensemble matrix $E^{\mathbb{A}_i}$ is given as:

$$E^{\mathbb{A}_i} = \begin{bmatrix} a_1 \in \mathbb{E}_i^1 & b_1 \in \mathbb{E}_i^1 \\ a_2 \in \mathbb{E}_i^2 & b_2 \in \mathbb{E}_i^2 \\ .. & .. \\ .. & .. \\ a_\alpha \in \mathbb{E}_i^\alpha & b_\alpha \in \mathbb{E}_i^\alpha \end{bmatrix} \quad (1)$$

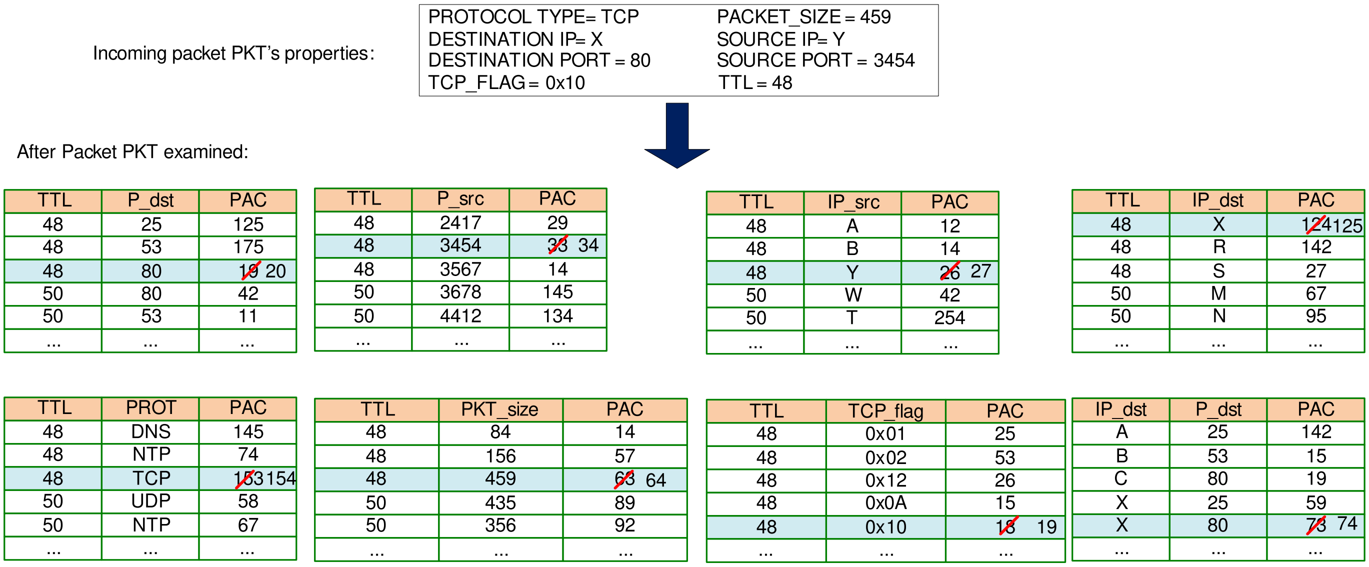
Based upon the $E^{\mathbb{A}_i}$ matrix, a Pair Nominal Profile $N^{\mathbb{A}_i}$ for $\mathbb{A}_i$ is created by adding a new element, $PAC^{A_i^j}$, for each different tuple $\mathbb{E}_i^j = (a_j, b_j)$ in $E^{\mathbb{A}_i}$.

Let $\mathbb{A}_i^j = (x_j, y_j)$ be a tuple that is composed of x_j and y_j properties for the corresponding attribute. It is a subset of $\mathbb{A}_i \in \binom{\mathbb{P}}{2}$, $i = 1, 2, 3, .., \left|\binom{\mathbb{P}}{2}\right|$, $j = 1, 2, 3, ..., m_i$ where m_i is a system specific positive integer that depends on the number of different properties of the packets in a corresponding period. The Pair Nominal Profile for $\mathbb{A}_i$, $N^{\mathbb{A}_i}$, in an α packet period is represented as a $m_i \times (k + 1)$ matrix shown as (2):

$$N^{\mathbb{A}_i} = \begin{bmatrix} x_1 \in \mathbb{A}_i^1 & y_1 \in \mathbb{A}_i^1 & PAC^{\mathbb{A}_i^1} \\ x_2 \in \mathbb{A}_i^2 & y_2 \in \mathbb{A}_i^2 & PAC^{\mathbb{A}_i^2} \\ .. & .. & .. \\ .. & .. & .. \\ x_{m_i} \in \mathbb{A}_i^{m_i} & y_{m_i} \in \mathbb{A}_i^{m_i} & PAC^{\mathbb{A}_i^{m_i}} \end{bmatrix} \quad (2)$$

Each switch generates nominal profiles for each pair attribute during an attack-free period by counting the number of packets that have the same attribute value. During this profiling period, the switch sends the headers of packets to the controller. Accordingly, the controller generates pair nominal profiles for each combination. An example nominal profile for a pair is given in Table II. Similarly, an example update action of the nominal profiles $N^{\mathbb{A}_i}$, when a packet PKT is examined, is shown in Figure 3.

2) Nominal Profile of Order p: For higher defense performance, nominal profile generation may incorporate the previous nominal profiles. Increasing the number of enrolled periods enhances the accuracy of how the nominal state of the network is realized. In JESS, the nominal profile at order p is calculated from p separate past periods.



Incoming packet PKT's properties:

PROTOCOL TYPE= TCP
DESTINATION IP= X
DESTINATION PORT = 80
TCP_FLAG= 0x10
PACKET_SIZE = 459
SOURCE IP= Y
SOURCE PORT = 3454
TTL = 48

After Packet PKT examined:

TTL	P_dst	PAC
48	25	125
48	53	175
48	80	~~19~~ 20
50	80	42
50	53	11
...	...	...

TTL	P_src	PAC
48	2417	29
48	3454	~~33~~ 34
48	3567	14
50	3678	145
50	4412	134
...	...	...

TTL	IP_src	PAC
48	A	12
48	B	14
48	Y	~~26~~ 27
50	W	42
50	T	254
...	...	...

TTL	IP_dst	PAC
48	X	~~124~~125
48	R	142
48	S	27
50	M	67
50	N	95
...	...	...

TTL	PROT	PAC
48	DNS	145
48	NTP	74
48	TCP	~~153~~154
50	UDP	58
50	NTP	67
...	...	...

TTL	PKT_size	PAC
48	84	14
48	156	57
48	459	~~63~~ 64
50	435	89
50	356	92
...	...	...

TTL	TCP_flag	PAC
48	0x01	25
48	0x02	53
48	0x12	26
48	0x0A	15
48	0x10	~~18~~ 19
...	...	...

IP_dst	P_dst	PAC
A	25	142
B	53	15
C	80	19
X	25	59
X	80	~~73~~ 74
...	...	...

Fig. 3. An example of how nominal profiles are updated.

TABLE II

AN EXAMPLE OF PAIR NOMINAL PROFILE $N^{\mathbb{A}_i}$ FOR TTL VALUE & DESTINATION PORT ATTRIBUTES

TTL Value	Destination Port Number	Number of Packets
48	25	125
48	53	175
48	80	19
50	80	42
...	...	...

A pair nominal profile of order p for $\mathbb{A}_i$ is denoted as $N_p^{\mathbb{A}_i}$. It is a $m \times (k+1)$ matrix in p periods and for each period there are α packets as noted in Table I. This matrix is shown as follows:

$$N_p^{\mathbb{A}_i} = \begin{bmatrix} x_1 \in \mathbb{A}_i^1 & y_1 \in \mathbb{A}_i^1 & pPAC^{A_i^1} \\ x_2 \in \mathbb{A}_i^2 & y_2 \in \mathbb{A}_i^2 & pPAC^{A_i^2} \\ .. & .. & .. \\ .. & .. & .. \\ x_m \in \mathbb{A}_i^m & y_m \in \mathbb{A}_i^m & pPAC^{A_i^m} \end{bmatrix} \tag{3}$$

Since each period may have a different set of $\mathbb{A}_i^j$, the controller identifies unique $\mathbb{A}_i^j$'s per period. In this measurement, the number of different unique pairs for each attribute pair A_i is denoted as m_i. For instance, assume that A_1 is the attribute pair for TTL value and $Destination\_Port$. If A_1 is the profile in Table II, then $m_1 = 4$ since it has four unique pairs for A_1. Similarly, in nominal profile based on p periods, controller identifies all unique $\mathbb{A}_i^j$'s in that time span and then calculates m, which is equal to the number of these $\mathbb{A}_i^j$'s. This process is illustrated in Figure 4 where each color represents different pairs and all different colors form the nominal profile from p separate periods. Figure 4 shows the collection of the pair profiles during p periods and the formation of the nominal profile of order p. In this process, any TCP/IP packet is parsed by the switch according to the monitored attributes. Parsed headers are assigned to the corresponding tuple $\mathbb{E}_i^j$ in each period. The same colored tuples show the tuples with the identical first and second characteristics, $a_1, b_1 \in \mathbb{E}_i^j$. Looking at the tuples stored for the 27^{th} attribute pair (A_{27}), as an example, let's say it shows the values for TCP_flag and TTL. There are only five different tuples appearing in the first period (Green, Dark Blue, Blue, Red, and Yellow colored tuples). For the same feature in the second period, four different tuples appear (Red, Blue, Purple, and Green colored tuples). When set A_j^{27} is created for order p, the intersection of all these different tuples is taken into consideration (Red, Green, Blue, Purple, Yellow, and Dark Blue colored tuples) and nominal profile $N_p^{\mathbb{A}_{27}}$ is created where $m_{27} = 6$.

3) Joint Entropy Calculation for Nominal Profiles: JESS utilizes joint entropy in DDoS detection and appropriate pair attribute selection. Since entropy shows randomness in a data set, it is a good indicator for DDoS detection [36]. When similar packets are sent in DDoS situation, entropy will decrease and DDoS can be detected. Joint entropy shows the amount of randomness we observe when we consider more than one attribute at a time. In JESS, the controller calculates joint entropy JN for each nominal pair profile and uses it in detection function. When the joint entropy values of the current profiles and nominal profiles are compared, DDoS is detected if the difference of any pair exceeds a threshold θ_j. Then, the pair with the largest difference will be utilized as the determinant pair for the current attack.

Specifically, joint entropy of a nominal profile $N^{\mathbb{A}_i}$ in a nominal period is calculated as follows:

$$JN_{N^{\mathbb{A}_i}} = -\sum_{n=1}^{m_i} (PAC^{A_i^n}/\alpha) \log(PAC^{A_i^n}/\alpha)) \tag{4}$$

We drop p since we are using $p = 1$ in our experiments. For the ease of representation in the previous equation, $PN_{A_i^n}$ term is defined as:

$$PN_{A_i^n} = PAC^{A_i^n}/\alpha \tag{5}$$

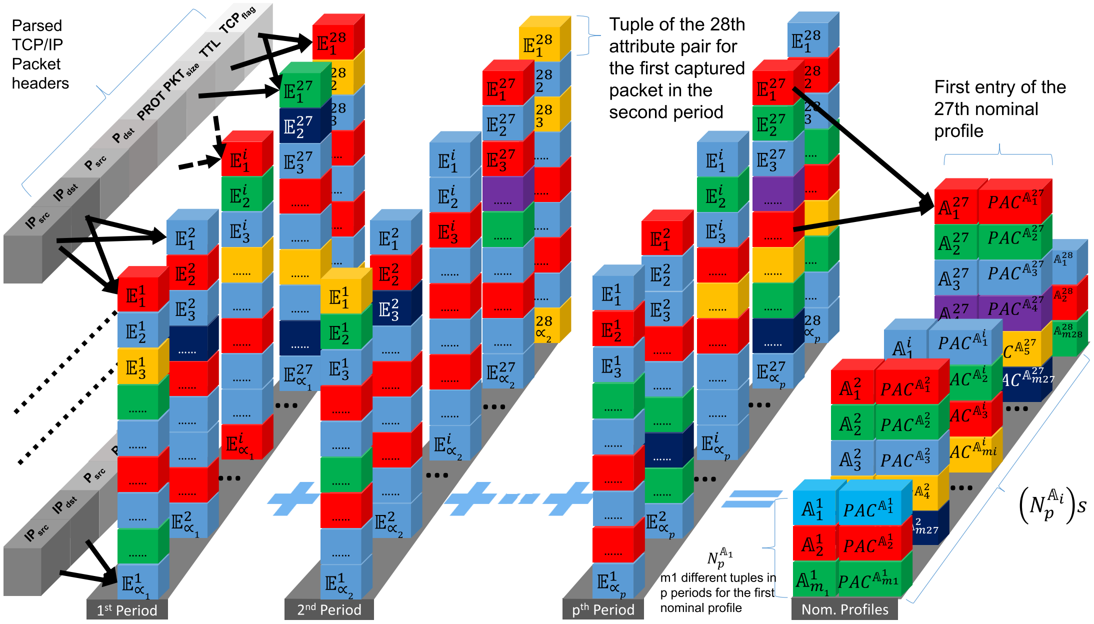

Parsed
TCP/IP
Packet
headers
IP src
IP dst
P src
P dst
PROT
PKT size
TTL
TCP flag
Tuple of the 28th attribute pair for the first captured packet in the second period
First entry of the 27th nominal profile
$(N_p^{\mathbb{A}_i})s$
$N_p^{\mathbb{A}_1}$
m1 different tuples in p periods for the first nominal profile
1st Period
2nd Period
pth Period
Nom. Profiles

Fig. 4. Nominal profiles ($N_p^{\mathbb{A}_i}$) from p separate periods (Nominal profile of order p).

Then, $JN_{N^{\mathbb{A}_i}}$ can be simplified as:

$$JN_{N^{\mathbb{A}_i}} = -\sum_{n=1}^{m_i} PN_{A_i^n} \log(PN_{A_i^n}) \tag{6}$$

In order to have the entropy value in $[0, 1]$, we normalize its value as follows:

$$\overline{JN}_{N^{\mathbb{A}_i}} = JN_{N^{\mathbb{A}_i}}/log[\alpha] \tag{7}$$

In this work, all entropy values are normalized. Thus when we use joint entropy term, it means that its value is normalized, i.e, $\overline{JN}_{N^{\mathbb{A}_i}}$, $\overline{JC}_{N^{\mathbb{A}_i}}$.

At the end of this stage, nominal profiles $N_p^{A_i}$ and joint entropy $JN_{N^{A_i}}$ values are ready to be used in case of an attack.

B. Preparatory Stage

When the bandwidth exceeds its nominal value, the preparatory stage is activated. This stage detects the attack and determines the most appropriate pair for profiling in active mitigation periods of the attack. As explained below, there are three phases in this stage which involve generation of current current profiles, joint entropy calculation for them and comparison operation.

1) Current Profile Generation: Each switch generates current profiles for each pair attribute, $\mathbb{A}_i$ during the attack period. It counts the number of packets that have the same attribute value. During this profiling period, the switch sends the headers of all the packets to the controller. Accordingly, the controller generates pair current profiles for each combination. The i^{th} current profile $C^{\mathbb{A}_i}$ is shown in the (8):

$$C^{\mathbb{A}_i} = \begin{bmatrix} x_1 \in \mathbb{A}_i^1 & y_1 \in \mathbb{A}_i^1 & PAC^{\mathbb{A}_i^1} \\ x_2 \in \mathbb{A}_i^2 & y_2 \in \mathbb{A}_i^2 & PAC^{\mathbb{A}_i^2} \\ .. & .. & .. \\ .. & .. & .. \\ x_{m_i} \in \mathbb{A}_i^{m_i} & y_{m_i} \in \mathbb{A}_i^{m_i} & PAC^{\mathbb{A}_i^{m_i}} \end{bmatrix} \tag{8}$$

2) Joint Entropy Calculation for Current Profiles: Joint entropy JC is calculated for each pair in order to provide the comparison with JN. Similar to the $PN_{A_i^n}$ term $PC_{A_i^n}$ is represented as follows:

$$PC_{A_i^n} = PAC^{A_i^n}/\alpha \tag{9}$$

Then, JC is calculated as follows:

$$JC_{N^{\mathbb{A}_i}} = -\sum_{n=1}^{m_i} PC_{A_i^n} \log(PC_{A_i^n})) \tag{10}$$

3) Comparison: Whenever the joint entropy values of current profiles are ready, they are compared with corresponding nominal joint entropy values and $\Delta J_{N^{\mathbb{A}_i}}$ for each pair is calculated:

$$\Delta J_{N^{\mathbb{A}_i}} = JC_{N^{\mathbb{A}_i}} - JN_{N^{\mathbb{A}_i}} \tag{11}$$

If any of the difference values exceeds the threshold θ_J, then DDoS attack is detected. This threshold is a system parameter that needs to be tuned to an optimal value according to security requirements and operational environment.

After the detection, JESS determines the maximum difference for $|\mathbb{P}| = 8$ and $k = 2$ according to the operation

$$\Delta J_{MAX} = \max\{\Delta J_{N^{\mathbb{A}_1}}, \Delta J_{N^{\mathbb{A}_2}}, \ldots, \Delta J_{N^{\mathbb{A}_{28}}}\},$$

and identifies the pair which has the maximum difference. Suppose that $\Delta J_{N^{\mathbb{A}_2}}$ has that maximum difference. Then it will be marked as the $SuspiciousPair$ which is regarded as the most appropriate pair for mitigation of the current attack traffic. Accordingly, the controller sends a signaling message to the switch to inform it about that $SuspiciousPair$.

C. Active Mitigation Stage

After the controller determines the $SuspiciousPair$ and informs the switch, the preparatory stage is completed and the active mitigation stage is initialized. Attack traffic is mitigated

TABLE III

AN EXAMPLE RULE TABLE (RT_C) FOR THE SUSPICIOUS PAIR {TTL VALUE, DESTINATION PORT} ATTRIBUTES

TTL Value	Destination Port Number	Decision
48	25	PASS
48	53	PASS
48	80	DROP ①
50	80	PASS ②
...	...	...

TABLE IV

RT_P FOR THE SUSPICIOUS PAIR {TTL VALUE, DESTINATION PORT} ATTRIBUTES

TTL Value	Destination Port Number	Decision
48	25	PASS
48	53	PASS
48	80	PASS ①
50	80	DROP ②
...	...	...

by dropping the attack packets whereas the legal packets are protected. This stage continues to operate until the traffic bandwidth decreases to acceptable values. The following five steps are realized in this stage.

1) SuspiciousPair Profile Generation: All packet headers are sent from the switch to the controller in the previous two stages. This mechanism may apparently create high amount of communication burden on a switch. This overhead is decreased in this phase. Since the switch knows the $SuspiciousPair$, it starts to generate current profile SC for that pair. It does not send the headers of all packets, instead it just sends the current profile SC to the controller at the end of each period.

2) Score Calculation: After the SC generation, the controller has the nominal profile SN and current profile SC of the $SuspiciousPair$. Each entry of SC is taken and its score β is calculated by considering its corresponding value in SN. After $SuspiciousPair$ is determined, the score β for each entry is calculated as follows:

$$\beta = \frac{PC_{A_i^n}}{PN_{A_i^n}} \qquad (12)$$

All β values are stored in a data structure, namely score table ST.

3) Threshold Determination: The score of a packet needs to be compared with a threshold value θ_S for deciding on the drop or pass action. The controller determines the current threshold value θ_C according to the cumulative distribution of scores in ST by using load shedding algorithm [41]. This distribution is shown symbolically as $CDF(\theta_C) = \Phi$ whereas Φ is the ratio of traffic that should be dropped. The portion of the traffic permitted to pass is $1 - \Phi = \frac{\phi}{\psi}$ whereas ϕ is the acceptable traffic and ψ is the total current incoming traffic. The threshold θ_S is determined with the current θ_C and the previous period's θ_P and thus is calculated as follows:

$$\theta_S = (\theta_C + \theta_P)/2 \qquad (13)$$

4) Rule Generation: After the controller calculates the scores and the threshold, the system is ready to generate the filtering rules. Since the score of each entry in SC is calculated, decision for DROP or PASS can be provided via comparison with that value. If the score value exceeds the threshold, the packet is supposed to be malicious and discarded. Otherwise, it is forwarded to the destination. The controller generates a current rule table RT_C for each entry in SC. An example rule table is shown in Table III.

5) Differing Rules ($\Delta - Rules$) Determination: The controller determines the differing rules by comparing the current rule table RT_C and previous rule table RT_P. Same rules that have the same decisions will not be sent to the switch for higher efficiency and overhead minimization. For instance, the following table RT_P in Table III is compared with RT_C in Table IV and only the last two rules (① and ②) of RT_C are determined as differing rules $\Delta - RT$, since they are different from the previous ones. Then, they are sent to the switch.

Considering all the components and algorithmic procedures described in this section, JESS operation can be summarized based on its three main stages: nominal, preparatory and active mitigation. Baseline information is generated by creating nominal pair profiles for each attribute pair in the *Nominal Stage*. Controller calculates the joint entropies for each pair. After a congestion signaling a DDoS attack is detected, *preparatory stage* is started where controller generates current pair profiles and calculates joint entropies for each pair. Then it compares these current entropies with the nominal ones to determine if there is actually an attack. Accordingly, the pair which has the maximum difference is marked as the $SuspiciousPair$. After that pair is determined, *Active Mitigation stage* is started. The switch creates current profile SC of $SuspiciousPair$ and at the end of each period it sends SC to the controller. The controller calculates score for each entry in the profile and generates a score table ST. A current rule table RT_C is generated. If the score of a pair is under the threshold, it is forwarded otherwise it is discarded, i.e. DDoS attack is mitigated. Moreover, only the differing rules $\Delta - RT$ are sent to the switch since they have changed after the previous period. Then the switch updates its rule table according to $\Delta - RT$.

IV. SIMULATIONS AND PERFORMANCE EVALUATION

We perform experiments via simulations for performance evaluation of JESS. The dataset, simulation environment, network topology, attack types, performance metrics and simulation results are presented in the following subsections.

A. Dataset

In our work, we need a real view of the Internet traffic. Therefore, we use real network traffic dataset from MAWI Working Group Traffic Archive [42]. MAWILab works on traffic measurement analysis in long-term on global Internet. It was started in 2002 and it is still collecting data from Internet. The part of the data that we have used in our simulations is collected on Jan 12, 2014. These data are utilized to generate nominal profile.

B. Simulation Environment

In order to evaluate our work, we utilize Mininet [43] as SDN network simulator. It enables us to create a realistic network topology. Moreover, it is convenient for experimenting

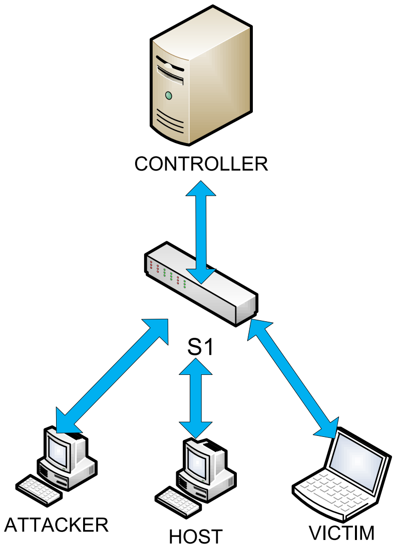


Fig. 5. Network topology used in the experiments.

with SDN solutions due to its design and capabilities. Our Mininet environment is integrated with Ryu [44] as the SDN controller. The test environment is run on a PC which has 8 GB RAM and Intel Core i7-3610QM 2.3 GHz CPU with Ubuntu 14.04 OS.

Figure 5 depicts the topology used for experiments. There is a switch which is managed by the controller and connected to three hosts. One of the hosts is attacked by a malicious user and it starts to generate DDoS attacks to the other host named as *Victim*. There is also one more host that is neither a victim nor an attacker which generates legitimate traffic in the network. Although this is not a large-scale topology, the attacker is able to generate malicious packet streams which seem like they are coming from different IP addresses (we do not utilize incoming physical switch port information in our scheme for a fair analysis). Thus, this environment emulates a much larger topology in practice. Furthermore, this experimental setup is more convenient to measure and evaluate the performance related phenomena, and thus to pinpoint the effect of our scheme more precisely.

C. Packet Analysis in Mininet

In order to reach the attribute values for each packet we need to know the corresponding values in each header. Source IP and destination IP addresses, TTL value, packet size and protocol type information are obtained from IP packet header. Similarly, destination port, source port and TCP flag attributes are obtained from TCP header. The corresponding names for attributes in headers are provided in Table V.

D. Attack Types

In the experiments, legitimate nodes generate benign traffic that has similar properties and thus characteristics to the nominal profile packets. Those nominal profile packets are created according to the employed dataset described in Section IV-A. The attacker generates both legitimate and attack traffic. In our simulations, we perform attacks by creating new packets that are also similar to the nominal traffic. We simulate the following attacks for performance evaluation:

- *TCP SYN Flood Attack*: The protocol type of an attack packet is TCP and TCP flag is set to SYN flag. Other attributes are randomized.
- *DNS Amplification Attack*: The protocol type of attack packets is DNS and destination port is set to 53. Also, attack packet size is 60 bytes. Other attributes are randomized.
- *NTP Attack*: The protocol type of attack packets is NTP and destination port is set to 123. Also, attack packet size is 90 bytes. Other attributes are randomized.
- *Generic Attack*: All attributes are selected randomly in their respective ranges. This can be stated as an unfamiliar type of attack since packet characteristics do not manifest a statistical coherence.
- *Mixed Attack*: TCP SYN Flood Attack, SQL Slammer Worm Attack, DNS Amplification Attack and NTP Attacks are superposed and applied at the same time to the victim node(s).

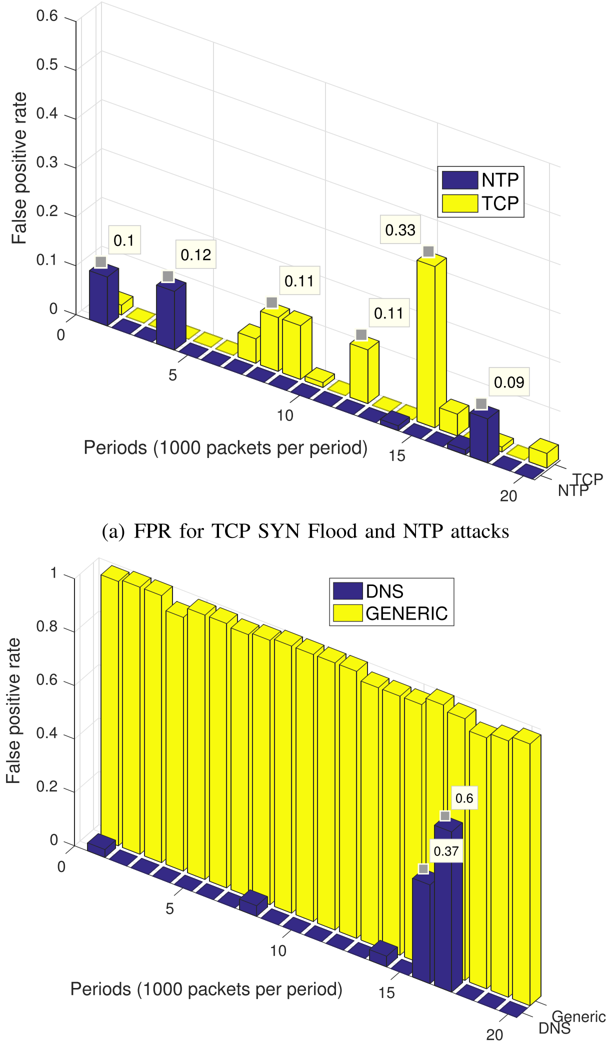
TABLE V
ATTRIBUTES IN PACKET HEADERS

Header Type	Attribute	Corresponding name in header
IPv6	packet size	payload_length
IPv6	TTL	hop_limit
IPv6	protocol type	next_header
IPv6	source IP address	src
IPv6	destination IP address	dst
IPv4	packet size	total_length
IPv4	TTL	ttl
IPv4	protocol type	proto
IPv4	source IP address	src
IPv4	destination IP address	dst
TCP	source port	src_port
TCP	destination port	dst_port
TCP	TCP flag	bits

E. Performance Metrics

In pattern recognition and information retrieval, true positive (TP), true negative (TN), false positive (FP) and false negative (FN) based metrics for binary classification are instrumental to measure system performance. Since DDoS attack packet identification is also fundamentally a binary classification problem, we utilize these metrics for our investigation.

In our case, TP is the number of attack packets dropped in the network and stonewalled to prevent reaching the destination whereas TN is the number of legal packets that are identified correctly by the system and reach the destination safely. In that vein, FP is the number of legal packets that are falsely discarded whereas FN is the the number of attack packets that are falsely forwarded to the destination. False positive rate FPR and accuracy ACC metrics are calculated via these parameters and utilized in performance analysis.



(a) FPR for TCP SYN Flood and NTP attacks

(b) FPR for DNS Amplification and Generic attacks

Fig. 6. FPR performance of JESS during DDoS attacks.

FPR deals with the success of the system on legal packets and calculates what percentage of the legal packets are dropped:

$$FPR = FP/(FP + TN) \tag{14}$$

ACC deals with the success of the system and calculates what percentage of the decisions are correct:

$$ACC = (TP + TN)/(TP + TN + FP + FN) \tag{15}$$

F. Experimental Results

In this section, we demonstrate our performance results according to the metrics explained above. For investigating JESS performance, We first elaborate on the key case where we consider different attack types. Then we extend our analysis with the case where JESS is augmented with a sliding window mechanism for packet analysis. Finally, we examine the performance characteristics under different attack intensities.

1) For Different DDoS Attack Types: In order to show the system performance on legal packets, FPR results are given in Figure 6. The lower values for FPR is better since it shows the ratio of the legal packets that are discarded erroneously. In Figure 6(a), the performance of JESS is measured against TCP SYN Flood and NTP attacks for fifteen periods. Similarly, in Figure 6(b) the performance is measured against DNS Amplification and Generic attacks. Since during nominal and preparatory stages switches do not have the updated rules yet, these results do not include these warm-up periods. Our mechanism starts to eliminate the attack packets in active mitigation stage. The results suggest that while there is TCP SYN Flood attack, nearly all packets can reach to the victim safely. There are a few small surges but they die out immediately, which shows that legal packets can be identified successfully by JESS with a small response time. It rises to 33% only in period for TCP, the reason being that there are more packets which have same characteristics with the attack packets. Therefore, the legal ones are also marked more as malicious and thus dropped. Similarly during NTP and DNS attacks, in most of the periods all legal packets can reach to the victim safely. During DNS attack, FPR increases rapidly to 60% for a single period. When we analyze the legal packets in this period, we again see that there are more packets "similar" to the attack packets. For instance, JESS selects PKT_{size} and P_{dst} pair as $SuspiciousPair$ while DNS attack packets have the properties as $P_{dst} = 53$ and $PKT_{size} = 60$. Then all the packets with these properties will be dropped. If there are more legal packets with these properties in this period, FPR increases. In order to improve this metric, sliding window periods are utilized, which is explained in the following subsection. The most distinctive result is obtained during Generic attack. This is a difficult attack type for JESS, since it needs at least two fixed properties of an attack to determine $SuspiciousPair$. However in Generic attack, all attributes are randomized and thus, FPR values are high during this attack.

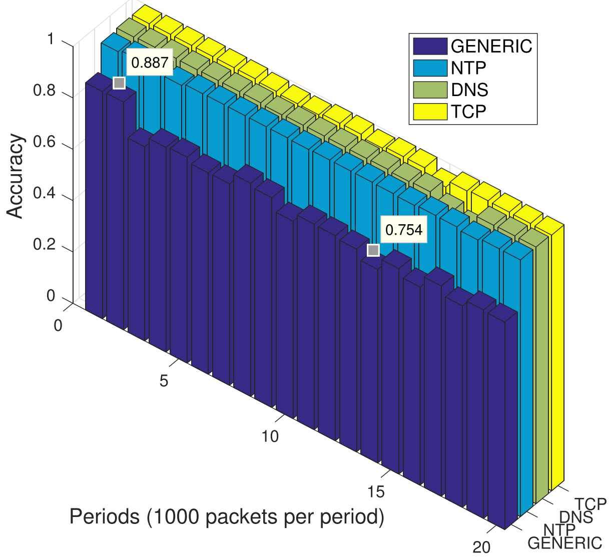


Fig. 7. Accuracy performance of JESS during TCP SYN Flood, NTP, DNS Amplification and Generic attacks.

Figure 7 shows the system performance on both legal and attack packets. Apparently, the accuracy of JESS is almost perfect during TCP SYN Flood, NTP and DNS Amplification attacks. JESS gives about 80% accuracy during Generic

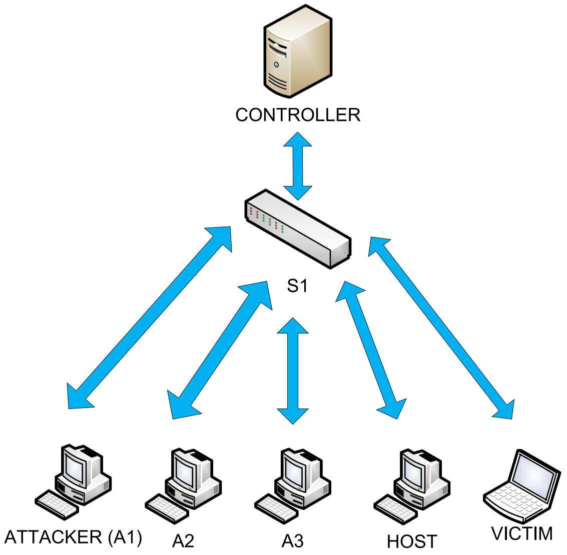


Fig. 8. Network topology used in the mixed attack experiment.

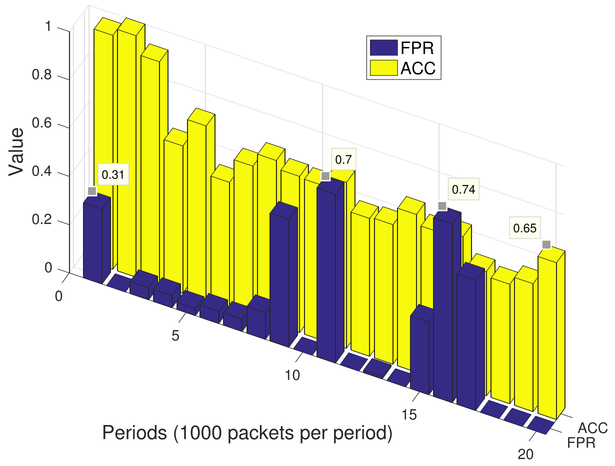


Fig. 9. Accuracy and FPR performance of JESS during Mixed attack.

attack, which shows that it performs 80% success even on unfamiliar attacks. Its performance on legal packets is not very high but it can differentiate attack packets easily and drop them.

Another attack type called *Mixed attack* is realized on a different network topology to render its performance more clearly. The topology in Figure 8 is used for that attack experiment. All attacker hosts apply different attacks on the victim, i.e. TCP SYN Flood, DNS Amplification and NTP attack packets are sent to the victim simultaneously. Moreover, the legal host tries to communicate with the victim in a benign manner. This composite attack is one of the most difficult situations for DDoS defense. The results are illustrated in Figure 9. During most of the periods FPR is under 10%. This means that more than 90% of legal packets can manage to reach the victim. There are several peaks but this is anticipated since the attack packet types are varied. In addition, it is favorable since the system recovers itself after at most three periods. Overall, average ACC is 70% which represents a decent performance against such a difficult attack type.

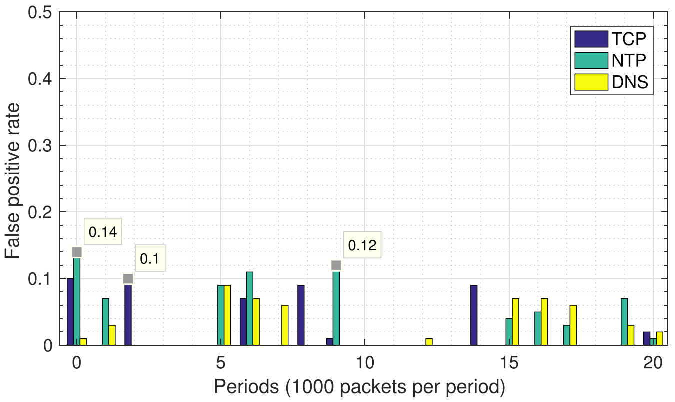
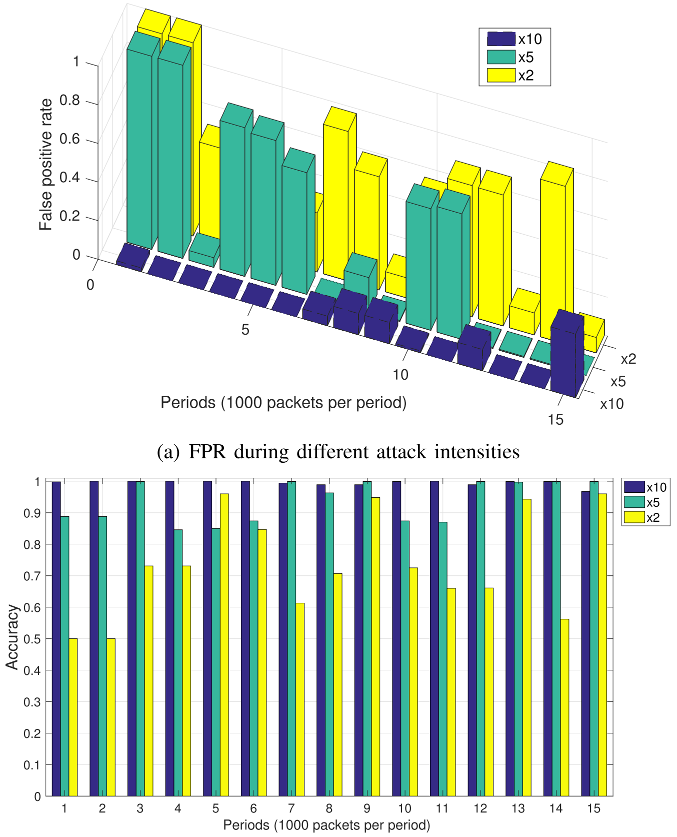


Fig. 10. FPR during TCP SYN Flood, NTP and DNS Amplification with sliding windows.

2) Integration of Sliding Window Periods: In JESS, packet based discrete periods are utilized since using time-based periods may not allow creating meaningful profiles due to lack of packets in a fixed-size time window. In Figure 6(a) and 6(b), there are some FPR peaks which are more than 20%. Especially for TCP attack, it may result in problems since the legal users cannot start a session due to adverse effect on the establishment phase. In addition, as far as there is a slow start mechanism in TCP, when the legal user's machine cannot get an acknowledgement message, the system pulls the brake at initial stage and throttle or even fail to establish the connection. In order to get rid off this problem, we integrate sliding window periods to JESS operation. This will decrease the peaks since they consider not only the current but also the previous period's packets. In sliding window mode, JESS needs to use some packets that arrived in the previous period. However, it can not update the profile since it does not know which packet properties belong to which entry. For this reason, JESS needs to store the headers of each packet until they are not used anymore. At the end of each period, SN is created by considering W number of packet headers where W is the size of the sliding window. Since in our simulations, each period has 1000 packets, our window size is $W = 1000$. If the incoming packet is an IPv6 and TCP packet then it will have 40 bytes for IP header and and 40 bytes for TCP header and header size becomes 80 bytes in total. Then the needed extra storage for a switch is $1000 \times 80 = 80$ KB, which is affordable for current switches. In addition, sliding range sld should be determined, in order to express how many of the new packets are involved in the new period. In our simulations $sld = 500$. The results of FPR with sliding window mechanism during different attacks are illustrated in Figure 10. The peaks of TCP and DNS are mostly about 0.1 whereas the peaks are also below 0.15 for NTP. These results suggest that JESS gives favorable results with sliding windows with the additional expense of affordable storage overhead in the switches.

3) Performance Under Different Attack Intensities: Performance results of JESS are analyzed against different attack intensities. Figure 11(a) shows the FPR of JESS under different rates of DDoS traffic. The results show that it gives about 10% FPR during 10 times of nominal traffic. However,



(a) FPR during different attack intensities

(b) Accuracy during different attack intensities

Fig. 11. Performance of JESS during different attack intensities.

it has worse results for 5 and 2 times of nominal traffic. It is easier to determine the legal packets when the ratio of the attack traffic to the nominal traffic increases. It is more difficult to differentiate legal packets when the number of legal packets are equal to the number of attack packets as in the case ×2 in Figure 11(a). However, JESS still gives promising ACC results for low attack densities. As illustrated in Figure 11(b), it gives perfect result for 10 times (×10). For ×5 case, accuracy rate is about 90% percent whereas its meanX falls to around 70%s for ×2. A decrease in accuracy while the attack traffic rate decreases is an expected result since it becomes difficult to differentiate the attack packets as their entropy decreases.

G. Algorithmic Analysis

In this section, JESS algorithm is analyzed in terms of time complexity, processing time, storage and communication requirements.

1) Complexity: JESS steps for controller and switch, and their corresponding time complexities are shown in Table VI. All profile generations have $O(N^2)$ complexity since they are generated from pair attributes which have two dimensions. Similarly, rule table generation and comparison operations have $O(N^2)$ complexities. The switch has a relatively small number of operations but as $SuspiciousPair$'s current profile SC is generated in the switch, the time complexity of our algorithm is $O(N^2)$ on both switch and controller. This complexity can be improved by utilizing different types of data structures like dictionaries or hash trees. However, this is beyond the scope of our work.

TABLE VI

TIME COMPLEXITY ANALYSIS FOR STEPS IN JESS

Location	Operation	Complexity
Controller	Generation of nominal profiles	$O(N^2)$
Controller	Joint entropy calculation of nominal profiles	$O(N)$
Controller	Generation of current profiles	$O(N^2)$
Controller	Joint entropy calculation of current profiles	$O(N)$
Controller	Compare Joint Entropies	$O(N)$
Switch	Generation of current suspicious pair profile	$O(N^2)$
Controller	Score table generation	$O(N)$
Controller	Rule table generation	$O(N^2)$
Controller	Rule table comparison	$O(N^2)$
Switch	Update rule table	$O(N)$

TABLE VII

THE NUMBER OF DIFFERENT VALUES (DV) FOR EACH ATTRIBUTE

	IP_{src}	P_{dst}	$PROT$	TCP_{flag}	TTL	PKT_{size}
DV	324	80	7	8	59	67

2) Processing Load: In our simulations, we utilize packet based periods. Thus, JESS does not consider the time during the analysis. In this section, we provide a simple emprical analysis to investigate if JESS's periods are meaningful in terms of timing, i.e, how long does a period correspond to and can a switch handle the packet processing or is there a need for buffering to meet line-rate requirements?

We base our analysis on the MAWI dataset. In that trace, the average packet size is about 500 bytes. Then, each period has $500 \times 8 \times 1000 = 4$ MB if a period is 1000 packets long. Assuming that average transmission rate of a switch is 600 Mbps, then a period's packets processing takes 0.06 seconds. If we analyze the dataset and look at the duration between the first and 1000^{th} packets arrival, this duration is 0.182 seconds. As 0.06 is much less than 0.18, the packets are not expected to be buffered extensively or dropped in the long-term.

3) Storage Analysis: Most of the storage is provided by the controller since high performance computers are typically utilized as controller platforms and the practical impact of storage requirements is minimal. Nevertheless, we calculate storage requirement for both the switch and controller.

For that purpose, the real dataset that we have utilized in our simulations is analyzed in terms of number of different values for each attribute named as DV. These values are displayed in Table VII. According to these values, the largest pair attribute profile requires $SR = 96 \times 324 \times 80$ bits, being equal to 311 KB. If IPv6 is used for all IP packets, then a pair profile needs $324 \times 80 \times (128 + 32 + 32) \cong 622$ KB.

As far as there are 8 attributes, there are $\binom{8}{2} = 28$ profiles for each current and nominal operation. Thus there is a need for 311 KB$\times 28 \times 2$ Bytes $= 17.4$ MB. If we consider IPv6 for destination and source IP addresses, the pairs which have IP addresses will have 622 KB size. The number of profiles which have 622 KB will be 13 whereas the remaining has 311 KB profiles for each current and nominal operations. Then there is a need for $((311KB \times 15) + (622KB \times 13)) \times 2 \cong 25.5$ MB. This is a small burden for a controller which has a much more advanced hardware setup. Also, the switch only needs storage allocation for one profile of $SuspiciousPair$ which is 311 KB or 622 KB. These are again insignificant values even for a basic switch.

4) Communication Analysis: The communication between the switch and the controller can be analyzed with respect to stages. In *Nominal Stage*, the switch needs to send TCP and IP headers of each packet. If the controller and the switch communicate via TCP on IPv6, an empty packet will have 24 bytes for Ethernet header, 40 bytes for IPv6 header and 40 bytes for TCP header. Thus, in total it needs 104 bytes without any information. Since the switch also needs to send all TCP and IP packet header information of the incoming packets, we need to add this information length. If the incoming packet is an IPv6 and TCP packet then it will be 40 bytes + 40 bytes = 80 bytes. Then in total, the size of a packet that contains header information is 184 bytes. In our simulations, 1000 packets were utilized in the nominal stage. Then it takes 184 bytes × 1000 = 184 KB. In the active mitigation stage, at the end of each period current profile of the $SuspiciousPair$ should be sent to the controller. As far as a pair profile takes about 311 KB and its header lengths will be at the order of bytes, it takes about 311 KB.

The analysis performed above is done for IPv6 and TCP packets since they have larger header sizes. If IPv4 is utilized, header size will be 20 bytes instead of 40 bytes. Similarly, for UDP packet header size will be 8 bytes instead of 40 bytes. Overall, the communication overhead at the order of KBs is negligible considering the communication capabilities of the switch(es) and the controller.

V. Discussions

An important issue for JESS or any statistical defense scheme is the need of different nominal profiles for different times of day. This topic is also analyzed in [40]. Their analysis suggests that the traffic profiles are most similar among the traffic at the same day and at the same time. In addition, a traffic profile is still very similar for a different time or day. The authors compared morning and evening profiles and they suggest that it may be enough to keep one profile. But still, it is possible to compare the stability of the profiles and if there is too much difference, it may be better to keep more multiple profiles for different times of a day. In order to decide how much difference occurs or how many profiles should be stored, sFlow [39] can be utilized to get some samples and compare the samples periodically. If there is a very large disparity between samples, the number of profiles could be increased. However, it is obvious that maintaining different profiles per site and per time of day would increase operational overhead. Thus, there is a trade-off between more accurate profiling and operational overhead. In our simulations, we suppose that profiles are very similar during the day and one profile is utilized for the nominal traffic.

Another issue is related to period lengths. As we utilize packet-based periods, we need to determine how many packets are supposed to be in a period. There should be enough number of packets in a profile to have accurate information about the network. But then, at the end of each period in the active mitigation stage of JESS, current profile of $SuspiciousPair$ SC is sent to the controller. For this reason, that data should be in an affordable size for a communication link which is under attack. In our experiments, we have utilized 1000 packets since that configuration provides enough information while the respective profile generates a traffic at the order of KBs when it is sent to the controller. In order to provide this data, the switch also needs to have some adjustments or additional functions. However, since these are attainable via software changes, it does cause too much operational burden on the network administrators.

JESS aims at detecting and mitigating DDoS attacks using joint entropy and can be effectively used on any virtual or non-virtualized network environment. Architecturally, JESS is not bound to have a controller-centric deployment. However, the key advantage of such a design is the simplicity. Moreover, the controller is expected to have a very level of resilience and protection measures since it is the single most valuable function (or target) for software-defined operation. As explained in the Section II, network virtualization and related technologies can also cause many additional security vulnerabilities, which may lead to other security issues. However, using the southbound interface JESS related functions can be packaged into VNFs and deployed in a network function virtualization paradigm. Therefore, studies utilizing the methods used in JESS as a separate network function or in a VNF approach may be considered as future work.

From high availability and resilience perspective, multi-controller configurations can also be considered in DDoS defense. Although multicontroller settings such as ONOS and OpenDayLight support physically distributed controllers, they are logically centralized. Moreover, DDoS against the controller is still feasible since switch queue, communication channel bandwidth of southbound interface and processing capacity of the controller is limited. Multiple controller can alleviate some of these limits, however, basically all they can do is to increase the required DDoS magnitude (packet per second or bandwidth) to cripple the control plane rather than detection and mitigation capabilities. Due to DoS attacks, SDN multicontroller model may additionally face the risk of the cascading failures of controllers [45]. Therefore, it does not provide the ultimate solution despite being being beneficial for resilience.

Another general issue with any DDoS defense scheme is the flash crowd effect. The flash crowd is a typical DDoS false alarm source which causes a large amount of legitimate traffic being sent to a specific endpoint similar to DDoS. One way we can often differentiate between a flash mob and a DDoS is by the number of requests per client. A DDoS can have a high or low number of clients with a very high number of requests each client. A flash mob will have a very high number of clients but a relatively low number of requests per client. This is because in the flash mob, the client requests are being sent from a human and not an automated script. Moreover, the external flash crowd factor is typically assumed to be managed using other means such as security experts or other flash crowd detection mechanisms which are necessary for fundamental network management functions such as resource management. Therefore, this aspect is out of scope for our work despite being an interesting research direction.

Finally, a general discussion on scalability and overhead characteristics of JESS is worthwhile for a more balanced analysis. In the first two stages of JESS, packet headers traversing through switches need to be sent to the controller, which deserves caution in terms of overhead and scalability when applied in practical environments. For that purpose, JESS may install many specific rules in the switch, which can copy the packet headers towards the controller, leading to memory overhead in the switch and traffic overhead on the control channel between the switch and controller. Scalability may also be challenging for the controller regarding incoming message processing for very large networks. However, please note that the packet header transmission occurs only during the profile generation steps of nominal and preparatory stages. During the nominal stage, a DDoS attack is not present which also implies a much lower traffic load. Additionally, this overhead is for only p periods. Similarly, it is very limited in the preparatory stage since it occurs only for a single period before determining the suspicious pair.

To address scalability and overhead related challenges, there are also potential remedies such as data plane programmability for optimized operation, adaptive bandwidth allocation in the control channel, inclusion of measurement related middle boxes for packet traffic instead of sole switch involvement, compression approaches for transmitted data and isolated computational resources for JESS packet header processing in the controller platform. Moreover, in our work JESS is expected to be deployed in edge switches of a network which are relatively few in number and with higher capabilities compared to core switches serving different segments of a network. Nevertheless, the challenges and intricacies for extensive scale-up of our approach for very large-scale networks is an interesting research topic for further research and left as future work.

VI. Conclusion

In this work, a statistical defense scheme JESS is proposed against DDoS attacks in SDN environment. This is the first model that utilizes joint entropy in a defense mechanism against DDoS attacks. There are three stages in JESS, namely *nominal*, *preparatory* and *active mitigation* stages. In the nominal stage, baseline profiles are generated in an attack-free period. In the preparatory stage, attack is detected rapidly. Then the active mitigation stage continues until that attack is prevented or stopped. Since it relies a statistical approach, JESS is effective against not only known but also unfamiliar attacks which use attack packets with similar attributes. The results suggest that it gives perfect results for several known attacks. Besides, it performs with 80% success rate for an unfamiliar attack type denoted as *generic attack*. It also provides 70% success for a mixed attack. For rendering the algorithmic characteristics, we carry out time, storage and communication burden analyses of JESS. The results show that it can work efficiently with low storage and processing requirements.

References

[1] I. Farris, T. Taleb, Y. Khettab, and J. S. Song, "A survey on emerging SDN and NFV security mechanisms for IoT systems," *IEEE Commun. Surveys Tuts.*, to be published.

[2] D. L. C. Dutra, M. Bagaa, T. Taleb, and K. Samdanis, "Ensuring end-to-end QoS based on multi-paths routing using SDN technology," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2017, pp. 1–6.

[3] A. Ksentini, M. Bagaa, and T. Taleb, "On using SDN in 5G: The controller placement problem," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–6.

[4] A. Ksentini, M. Bagaa, T. Taleb, and I. Balasingham, "On using bargaining game for optimal placement of SDN controllers," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–6.

[5] A. Aissioui, A. Ksentini, A. M. Gueroui, and T. Taleb, "Toward elastic distributed SDN/NFV controller for 5G mobile cloud management systems," *IEEE Access*, vol. 3, pp. 2055–2064, 2015.

[6] Z. Shu, J. Wan, D. Li, J. Lin, A. Vasilakos, and M. Imran, "Security in software-defined networking: Threats and countermeasures," *Mobile Netw. Appl.*, vol. 21, no. 5, pp. 764–776, 2016.

[7] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes, "Software-defined networking security: Pros and cons," *IEEE Commun. Mag.*, vol. 53, no. 6, pp. 73–79, Jun. 2015.

[8] S. Scott-Hayward, G. O'Callaghan, and S. Sezer, "SDN security: A survey," in *Proc. IEEE SDN Future Netw. Services (SDN4FNS)*, Nov. 2013, pp. 1–7.

[9] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A survey of security in software defined networks," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 623–654, 1st Quart., 2016.

[10] K. Kalkan, G. Gür, and F. Alagöz, "Filtering-based defense mechanisms against DDoS attacks: A survey," *IEEE Syst. J.*, vol. 11, no. 4, pp. 2761–2773, Dec. 2017.

[11] K. Kalkan, G. Gur, and F. Alagoz, "Defense mechanisms against DDoS attacks in SDN environment," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 175–179, Sep. 2017.

[12] H. T. N. Tri and K. Kim, "Assessing the impact of resource attack in software defined network," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2015, pp. 420–425.

[13] N. P. Katta, J. Rexford, and D. Walker, "Incremental consistent updates," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw.*, 2013, pp. 49–54.

[14] S.-W. Hsu *et al.*, "Design a hash-based control mechanism in vSwitch for software-defined networking environment," in *Proc. IEEE Int. Conf. Cluster Comput. (CLUSTER)*, Sep. 2015, pp. 498–499.

[15] S. Lim, S. Yang, Y. Kim, S. Yang, and H. Kim, "Controller scheduling for continued SDN operation under DDoS attacks," *Electron. Lett.*, vol. 51, no. 16, pp. 1259–1261, 2015.

[16] A. Zaalouk, R. Khondoker, R. Marx, and K. Bayarou, "OrchSec: An orchestrator-based architecture for enhancing network-security using network monitoring and SDN control functions," in *Proc. IEEE Netw. Oper. Manage. Symp. (NOMS)*, May 2014, pp. 1–9.

[17] D. Chourishi, A. Miri, M. Milić, and S. Ismaeel, "Role-based multiple controllers for load balancing and security in SDN," in *Proc. IEEE Canada Int. Humanitarian Technol. Conf. (IHTC)*, May 2015, pp. 1–4.

[18] A. F. M. Piedrahita, S. Rueda, D. M. F. Mattos, and O. C. M. B. Duarte, "FlowFence: A denial of service defense system for software defined networking," in *Proc. Global Inf. Infrastruct. Netw. Symp. (GIIS)*, Oct. 2015, pp. 1–6.

[19] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Avant-guard: Scalable and vigilant switch flow management in software-defined networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2013, pp. 413–424.

[20] R. Wang, Z. Jia, and L. Ju, "An entropy-based distributed DDoS detection mechanism in software-defined networking," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, vol. 1, Aug. 2015, pp. 310–317.

[21] A. Wang, Y. Guo, F. Hao, T. V. Lakshman, and S. Chen, "Scotch: Elastically scaling up SDN control-plane using vSwitch based overlay," in *Proc. 10th ACM Int. Conf. Emerg. Netw. Exp. Technol. (CoNEXT)*, 2014, pp. 403–414.

[22] R. T. Kokila, S. T. Selvi, and K. Govindarajan, "DDoS detection and analysis in SDN-based environment using support vector machine classifier," in *Proc. 6th Int. Conf. Adv. Comput. (ICoAC)*, Dec. 2014, pp. 205–210.

[23] K. Kumar, R. C. Joshi, and K. Singh, "A distributed approach using entropy to detect DDoS attacks in ISP domain," in *Proc. Int. Conf. Signal Process., Commun. Netw.*, Feb. 2007, pp. 331–337.

[24] K. Kalkan, G. Gür, and F. Alagöz, "SDNScore: A statistical defense mechanism against DDoS attacks in SDN environment," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2017, pp. 669–675.

[25] M. Özçelik, N. Chalabianloo, and G. Gür, "Software-defined edge defense against IoT-based DDoS," in *Proc. IEEE Int. Conf. Comput. Inf. Technol. (CIT)*, Aug. 2017, pp. 308–313.

[26] N. M. M. K. Chowdhury and R. Boutaba, "Network virtualization: State of the art and research challenges," *IEEE Commun. Mag.*, vol. 47, no. 7, pp. 20–26, Jul. 2009.

[27] D. Kreutz, F. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[28] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2429–2453, 3rd Quart., 2018.

[29] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1657–1681, 3rd Quart., 2017.

[30] Y. Khettab, M. Bagaa, D. L. C. Dutra, T. Taleb, and N. Toumi, "Virtual security as a service for 5G verticals," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2018, pp. 1–6.

[31] T. Taleb, A. Ksentini, and B. Sericola, "On service resilience in cloud-native 5G mobile systems," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 3, pp. 483–496, Mar. 2016.

[32] K. Kalkan and F. Alagöz, "A distributed filtering mechanism against DDoS attacks: ScoreForCore," *Comput. Netw.*, vol. 108, pp. 199–209, Oct. 2016.

[33] R. Sahay, G. Blanc, Z. Zhang, and H. Debar, "ArOMA: An SDN based autonomic DDoS mitigation framework," *Comput. Secur.*, vol. 70, pp. 482–499, Sep. 2017.

[34] T. T. Huong and N. H. Thanh, "Software defined networking-based one-packet DDoS mitigation architecture," in *Proc. 11th Int. Conf. Ubiquitous Inf. Manage. Commun.*, 2017, p. 110.

[35] A. Wagner and B. Plattner, "Entropy based worm and anomaly detection in fast IP networks," in *Proc. 14th IEEE Int. Workshops Enabling Technol., Infrastruct. Collaborative Enterprise (WETICE)*, Jun. 2005, pp. 172–177.

[36] S. Yu, W. Zhou, R. Doss, and W. Jia, "Traceback of DDoS attacks using entropy variations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 3, pp. 412–425, Mar. 2011.

[37] S. M. Mousavi and M. St-Hilaire, "Early detection of DDoS attacks against SDN controllers," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2015, pp. 77–81.

[38] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments," *Comput. Netw.*, vol. 62, no. 5, pp. 122–136, 2014.

[39] M. Wang, B. Li, and Z. Li, "sFlow: Towards resource-efficient and agile service federation in service overlay networks," in *Proc. 24th Int. Conf. Distrib. Comput. Syst.*, Mar. 2004, pp. 628–635.

[40] Y. Kim, W. C. Lau, M. C. Chuah, and H. J. Chao, "PacketScore: Statistics-based overload control against distributed denial-of-service attacks," in *Proc. 23rd IEEE Int. Conf. Comput. Commun. (INFOCOM)*, vol. 4, Mar. 2004, pp. 2594–2604.

[41] S. Kasera, J. Pinheiro, C. Loader, M. Karaul, A. Hari, and T. LaPorta, "Fast and robust signaling overload control," in *Proc. 9th IEEE Int. Conf. Netw. Protocols (ICNP)*, Nov. 2001, pp. 323–331.

[42] *MAWI Working Group Traffic Archive*. Accessed: May 2017. [Online]. Available: http://mawi.wide.ad.jp/mawi/

[43] *Mininet*. Accessed: Jul. 2017. [Online]. Available: http://mininet.org/

[44] *Ryu*. Accessed: Jul. 2017. [Online]. Available: https://osrg.github.io/ryu/

[45] T. Wang, H. Chen, G. Cheng, and Y. Lu, "SDNManager: A safeguard architecture for SDN DoS attacks based on bandwidth prediction," *Secur. Commun. Netw.*, vol. 2018, Jan. 2018, Art. no. 7545079, doi: 10.1155/2018/7545079.

Kübra Kalkan received the B.S. and M.S. degrees from the Computer Science and Engineering Department, Sabanci University, Istanbul, Turkey, in 2009 and 2011, respectively, and the Ph.D. degree from Bogazici University in 2016. She was a researcher with various institutions, such as EPFL, Microsoft Redmond, and Northeastern University. She is currently a Post-Doctoral Researcher with the University of Oxford. Her current research interests include network security, computer networks, wireless networks, and software-defined networking.

Levent Altay received the B.S. degree in system engineering and the M.S. degree in computer engineering from the ASTIN-Aeronautics and Space Technologies Institute in 1999 and 2011, respectively. He is currently pursuing the Ph.D. degree in computer engineering with Bogazici University. He is a member with the Satellite Networks Research Laboratory, Bogazici University. His research interests include cognitive radios, information security, and complex attacks.

Gürkan Gür received the B.S. degree in electrical engineering and the Ph.D. degree in computer engineering from Bogazici University, Istanbul, Turkey, in 2001 and 2013, respectively. He is currently a Senior Researcher with the Zurich University of Applied Sciences (ZHAW). He is also a member with the Satellite Networks Research Laboratory, Bogazici University. His research interests include information security, future Internet, next-generation wireless networks, and information-centric networking.

Fatih Alagöz received the B.Sc. degree in electrical engineering from Middle East Technical University, Turkey, in 1992, and the D.Sc. degree in electrical engineering from The George Washington University, Washington, DC, USA, in 2000. He is currently a Professor with the Computer Engineering Department, Bogazici University, Turkey. His current research areas include cognitive radios, wireless networks, network security, and UWB communications.