SDNScore: A Statistical Defense Mechanism Against DDoS Attacks in SDN Environment

Kübra Kalkan*, Gürkan Gür*,† and Fatih Alagöz*

*SATLAB, Dept. of Computer Engineering
†Telecommunications & Informatics Technologies Research Center (TETAM)
Bogazici University, Istanbul, Turkey
Email: {kubra.kalkan, gurgurka, fatih.alagoz}@boun.edu.tr

***Abstract*—Software Defined Networking is a promising solution for addressing challenges of future networks. Despite its advantages such as flexibility, simplification and low costs, it has several drawbacks that are largely induced by the centralized control paradigm. Security is one of the most significant challenges related to centralization. In that regard, Distributed Denial of Service (DDoS) attacks pose crucial security questions in software-defined networks. In SDN architecture, switches send all packets to the controller if they do not have any applicable rules in their flow tables. Basically, controller is the key place that can take initiative in decisions. However, this characteristic results in large communication overhead and delay until a DDoS attack is detected and appropriate action is activated against attack packets. Therefore, in this work we propose a hybrid mechanism, namely SDNScore, where switches are not simply data forwarders. Instead, they can collect statistics and can decide if DDoS attack is in action. Then they coordinate with the controller and decide on attack packets in cooperation. SDNScore is a statistical and packet-based defense mechanism against DDoS attacks in SDN environment. Since it has a statistical scoring method, it can detect not only known but also unknown attacks. In addition, it does not drop all packets in a flow which includes both attack and legal packets, but rather acts on attack packets using packet-based analysis.**

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

I. Introduction

Software Defined Networking (SDN) is a recent emerging technology which defines a new design and management approach for networking [1]. The main property of this paradigm is the separation of control and data planes. In traditional networks, routers apply high level routing algorithms and decide where data packets should be forwarded. In SDN, decision and forwarding functionalities are separated. Decision process is provided by SDN controller whereas data forwarding is handled by switches. Since decision algorithms do not run on network devices, simpler network devices can be utilized rather than complicated routers [2]. Moreover, in traditional networks, each router has its own security, link failure and forwarding mechanisms. If any of these mechanisms needs to be updated, each network device should be handled individually. However, one can ideally manage all these issues at a central point in SDN architecture.

Despite its advantages, SDN has several inherent challenges such as reliability, scalability, latency, and controller placement [3]. Security can be counted as one of the most vital problems. In that regard, Denial of Service (DoS) attacks provide a favorable way for attackers to damage security of these systems. The main agenda of DoS attackers is to make network-resident services unavailable for legal users. An attacker generates enormous number of attack packets and makes the system busy such that it cannot serve the requests of legal users. If several machines participate in this attack, it is called Distributed Denial of Service (DDoS) attack. It can be created with steadily diminishing amounts of effort in today's pervasive Internet whereas its detection can be difficult since malicious packets show up as legal but with very large quantities.

SDN environment is favorable for DDoS attacks since it is managed by the centralized controller by design [4]. When a packet comes to a switch from an IP unmatched in its flow table, it is forwarded to the controller as the default behavior. Then, the controller sends a flow rule to the switch for this IP. If attackers send a large number of packets from several IPs, these will be forwarded to the controller. Then this traffic will consume all available resources of the controller and make the system unavailable for legal users. Besides, the same attack can cripple the system by exploiting the table capacity of switches. When a huge number of spoofed packets are received by the switch, its memory will be totally occupied. Similarly, the link between switch-controller can become unavailable because of the congestion by malicious traffic. All these issues pose SDN vulnerable for DDoS attacks and thus DDoS defense as a critical research topic for SDN.

A critical characteristic of SDN architecture that nondeliberately serves DDoS attacks is the limited passive capabilities of switches. Since they send all packets with unmatched IP addresses to the controller, their medium becomes attractive for DDoS attacks. In addition, they do not have enough resources for very large volumes of traffic. In order to solve these problems, security-oriented intelligence can be integrated to switches. That will help to keep traffic in data plane as much as possible. Several works have suggested this approach such as [5], [6]. In our SDNScore model, we also utilize switches with relevant processing and intelligence capabilities for security.

In the literature, there are some works proposed for DDoS

defense in SDN. However, it is still an immature area since there is no dominant solution and all models have some drawbacks. In this work, we propose a packet-based statistical defense mechanism against DDoS attacks for SDN. We have been inspired by another packet based model, PacketScore [7], for traditional networks. Our proposal can cope with not only traditional attacks but also unknown new attacks. In order to make statistical analysis, some properties are utilized in packet scoring. The main difference of our proposal from PacketScore is the selection of appropriate attributes for each attack. This crucial property is provided by the support of the controller. Our results suggest that SDNScore is an effective method for DDoS defense in SDN.

This paper is structured as follows: In the next section, a literature review is presented about defense mechanisms against DDoS attacks in SDN. In Section III, our proposed mechanism SDNScore is described. Section IV discusses experimental results. Finally, Section V concludes the paper.

II. Related Work

SDN brings flexibility and simplification while having potential vulnerabilities for new threats. SDN architecture inherits two main security challenges: trusted controller-switch communication and single-point failure problem. Since switches are controlled by one point, the whole system can be damaged by capturing the controller. In addition, in order to provide trusted communication between controller and switch, authentication and authorization issues between controller and switches need to be handled. SDN security studies in the literature can be classified into two categories [2]: *SDN used as a security enabler* or *security provisioning for SDN*. Early studies are mostly focused on the first group and they investigate DDoS attack defense [8], [9] . However, the works in the second group suggest that there are vulnerabilities in SDN itself considering various attacks [5], [10]. Thus, defense mechanisms should be developed and optimized for SDN. Our proposal SDNScore falls into this latter group.

In order to defend against DDoS attacks in SDN, a relatively small number of works has been presented in the literature. Some of these works suggest to bring some intelligence to SDN switches [5], [6]. 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. Facilitating switches with some minor intelligence features does not compromise the main paradigm of SDN.

Avant-Guard [5] is a framework to improve the security and resiliency against DDoS and scanning attacks with greater involvement of switches. It introduces two modules on switches: connection migration and actuating triggers. Connection migration proxies TCP SYN requests and classifies them. If these are regarded as legitimate, they are authorized and migrated to the real target. Actuating triggers module perceives changes and triggers an event. Then, flow rule installation is handled automatically and response time is reduced. The most conspicuous side effect of this mechanism is the performance penalty. Since it utilizes connection migration, each flow needs to be classified. In addition, this module can only defend against one type of DDoS attack (TCP SYN Flood).

Another similar model in [6] proposes an entropy-based lightweight DDoS fooding attack detection model running in the OpenFlow edge switch. In this mechanism, entropy is calculated for destination IP address. If entropy decreases under a threshold, DDoS is detected. It determines the victim, but it is not possible to dissociate the legal packets from the attack ones. It achieves a distributed anomaly detection in SDN and reduces the flow collection overload on the controller.

The proposed model in our work called SDNScore does not have the limitations of the approaches cited above. It has detection and mitigation properties for DDoS attacks including unknown ones. In addition, it discards all detected attack packets and does just rate-limit. SDNScore is a statistical approach that is inspired by PacketScore proposed in [7]. PacketScore is a statistical filtering mechanism wherein each packet is analyzed according to its attribute values and then correspondingly scores are calculated according to them for packet filtering. However, this method has a trade-off between memory and accuracy. It does not have attribute selection property in profile generation. Instead, it generates profiles with predetermined fixed attributes. For instance, having more attributes for more accurate decisions means more complex profiles which results in larger memory footprint. The source of the problem in this model is the lack of appropriate attribute selection for each attack type. In SDNScore method, such drawbacks are omitted. The following section explains how these advantages are provided.

III. SDNScore Mechanism

SDNScore consists of four modules, *profiler*, *actuator*, *comparator* and *scorer*, to be loaded on the switch(es) and another module, *PairProfiler*, for the controller. These components cooperate for DDoS detection and mitigation in SDN environment.

A. SDNScore Modules for Switches

Profiler generates nominal profiles in an attack-free period whereas *actuator* inspects traffic and starts packet-based inspection if an attack is detected. *Comparator* finds most appropriate pair for packet attributes. According to that information, *Scorer* analyzes incoming packets and makes selective discarding. These modules are illustrated in Figure 1 and detailed in the following subsections. Similarly, the terminology utilized in this paper is listed in Table I.

1) Profiler: Each switch generates a nominal profile during an attack-free period. It counts the number of packets that have the same attribute value. The following properties are considered as attributes although this set can be extended: *source IP*, *destination IP*, *source port*, *destination port*, *protocol type*, *packet size*, *TTL value* and *TCP flag*. During nominal profiling period, the profiler sends headers of all packets to the controller. Accordingly, the controller generates pair

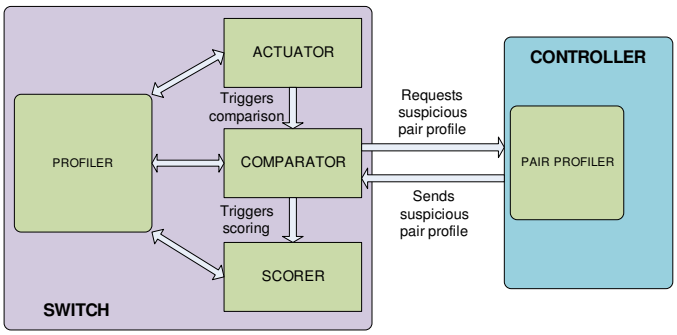

Fig. 1. SDNScore architecture: switch and controller modules.

TABLE I
SYSTEM PARAMETERS

Term	Explanation
$NominalProfile$	Profile in an attack-free period
$CurrentProfile$	Profile in an attack period
$SuspiciousPair$	Attribute pair with most probable signs for current attack
A, B	Attribute A and B
$A = a_p$	Attribute A with a value in packet p
$B = b_p$	Attribute B with b value in packet p
$PCP_{(A=a_p,B=b_p,\ldots)}$	The number of packets in a current profile that have the property of a_p for attribute A and b_p for attribute B
$PNP_{(A=a_p,B=b_p)}$	The number of packets in a nominal profile that have the property of a_p for attribute A and b_p for attribute B
$TPNP$	Total number of packets in a nominal profile
$TPCP$	Total number of packets in a current profile
S_p	Score value of packet p
Th	Threshold score value for packet discarding
ϕ	Acceptable traffic
ψ	Total current incoming traffic

nominal profiles for each combination. At the end of this period, it has each pair nominal profile whereas the switches have single nominal profiles. Pair profiles are not stored in the switches because that would occupy a large amount of memory. Quantitatively, a switch needs to store eight tables with above mentioned attributes whereas the controller needs to store $\binom{8}{2} = 28$ tables. In addition to nominal profile generation, all profiling instructions that will be held in packet inspection period are also provided by the *profiler*. When the *actuator* detects a congestion, it also informs *profiler* and starts to generate current profiles. A *current profile* is constructed during an attack period. Pair profiles are needed during current traffic analyses whereas single profiles are only used for choosing the most appropriate attribute selection. Pair profiles are utilized during all attack periods since they provide more detailed information about the traffic and enables more accurate decisions. These generated profiles are used by the other modules.

2) Actuator: In normal conditions, flow-based network monitoring is provided in switches. *Actuator* inspects bandwidth usage to determine surges. When traffic exceeds a bandwidth threshold, system monitoring is switched to packet-based inspection, and *comparator* and *profiler* modules are activated. Packet based inspection provides generation of new tables as single current profiles in *profiler*.

Since *actuator* monitors congestion consistently, when traffic returns to normal intensity and drops below the threshold, it informs other modules to switch to stand-by mode. Then *Actuator* stops packet inspection and the system continues on flow-based monitoring.

3) Comparator: After traffic surge is detected, *actuator* activates this module. Since current single profiles are generated by *profiler*, this module compares single nominal profiles with single current profiles. It determines the two specific attributes that have the most deviation from nominal profiles. These pairs are chosen as the most probable signs for ongoing attack. Thus, it is called as $SuspiciousPair$. Then *comparator* requests nominal profile of this pair from the controller. For instance, if it detects that protocol type and destination port have more deviation from nominal profile, the controller sends the nominal profile comprised of protocol type and destination port values and corresponding number of packets. Then it triggers *scorer* to start packet inspection and selective discarding.

4) Scorer: This module is activated after all tables are generated. Scorer's three main responsibilities are as follows:

a) Score calculation: Each packet's score is calculated considering $SuspiciousPair$'s corresponding value. If $SuspiciousPair$ is determined as A and B, then packet p with the attributes $A = a_p$ and $B = b_p$ will have the score S_p as follows:

$$S_p = \frac{PNP_{(A=a_p,B=b_p)}/TPNP}{PCP_{(A=a_p,B=b_p,\ldots)}/TPCP} \quad (1)$$

b) Threshold calculation: The score of a packet needs to be compared with a threshold Th. This threshold value is determined according to the cumulative distribution of scores by using load shedding algorithm [11]. That distribution is shown as $CDF(Th) = \Phi$ where Φ is the portion of traffic that should be dropped. The fraction of traffic permitted to pass is $1 - \Phi = \frac{\phi}{\psi}$ whereas ϕ is the amount of acceptable traffic and ψ is the total current incoming traffic.

c) Selective discarding: Each packet's score value is compared with the threshold. If it exceeds the threshold, this packet is assumed to be malicious and discarded. Otherwise, it is forwarded to the destination.

B. SDNScore Module for Controller: PairProfiler

The controller employs a module called *PairProfiler* to provide necessary capabilities for SDNScore. In normal conditions, while there is no surge condition, switches forward the packets that do not have an entry for their IP addresses to the controller. The controller decides how a flow packet should be handled by executing applicable decision logic. Then, it sends and inserts these appropriate rules for unknown IPs into the flow table of the source switch. In our proposal, it

has additional duties before and after the attack detection. *PairProfiler*'s primary work is to create pair nominal profiles. While single profiles are generated by the *profiler*, all the packets' headers are also sent to the controller. This can be a burden but it is required only once in an attack-free period as explained in the following subsection. Since eight attributes are utilized in profiling and each pair combination is generated as pair nominal profiles, $\binom{8}{2} = 28$ profiles are generated in the controller. After a congestion is detected by a switch and the *comparator* determines the $SuspiciousPair$, it requests the nominal profile of this pair from the controller. Then, the controller responds to this request with the expected information.

1) Analysis of Communication Overhead: In SDNScore, a switch needs to send TCP and IP header data of each packet in the profiling period. Therefore, it is important to calculate the communication burden on the system. Since our model utilizes a scoring technique similar to [7], we choose packet-based intervals as periods which guarantees that a sufficient number of packets are processed for profile creation. For our analysis, we aim to calculate the maximum possible overhead. Thus, we suppose that controller and switch communicate via IPv6 since IPv6 header size is larger compared to IPv4. In that case, a "payload-less" packet will need 24 bytes for Ethernet header, 40 bytes for IPv6 header and 40 bytes for TCP header which leads to 104 bytes. Since the switch also needs to send all TCP and IP header information of the incoming packets, we need to add this information length. If the incoming packet is an IPV6 packet over TCP, it will be $40\ bytes + 40\ bytes = 80\ bytes$. Then, the size of a packet that just contains header information is 184 bytes in total. Since there are TPNP packets in a profiling period and TPNP = 5000 in our system setup, this communication incurs $184\ bytes \times 5000 = 920\ KB$ as overhead. However, it is a negligible burden for the communication capacity of today's routers since this is only needed in an attack-free period.

IV. Simulations and Performance Evaluation

For performance analysis, we simulate our SDNScore mechanism and *An Entropy-Based Distributed DDoS Detection Mechanism in Software-Defined Networking (EBS)* presented in [6] and compare their experimental results. Network topology and dataset, attack types, performance metrics and simulation results are discussed in the following subsections.

A. Network Topology and Dataset

We simulated our SDNScore and EBS in C++ on a 3.3 GHz Intel Core i5 processor with 4 GB memory. For nominal profile generation, 5000 packets are active whereas during the DDoS attacks, ten-fold of this nominal traffic is generated per period. Figure 2 depicts the topology used for experiments. There are two switches managed by the controller. One switch has five legal users (L1- L5) and five attacker users (A1-A5), whereas the other switch has the victim and two more hosts (H1, H2). These five attackers are assumed to be legal users in the past. Then they are compromised by a malicious user

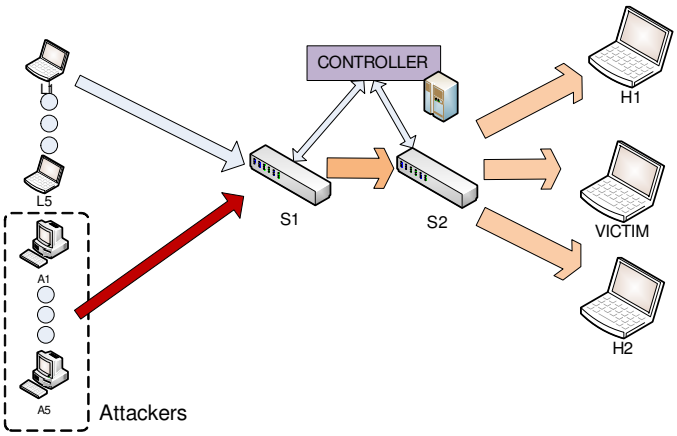

Fig. 2. Network topology used in the experiments.

and they constitute a botnet to facilitate DDoS attacks. A real dataset from MAWI Working Group Traffic Archive [12] is utilized in traffic generation. It is a popular traffic dataset used in the literature [13]–[15]. 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 data that we have used in our simulations was collected on Jan 12, 2014. This dataset is utilized to generate nominal profiles.

B. Attack Types

In the experiment topology, attack and legitimate nodes are connected to a switch. Legitimate nodes generate legitimate traffic that has similar properties and similar amount of traffic to nominal profile packets. Nominal profile packets are created according to the dataset as mentioned in Section IV-A. Attack nodes generate both legitimate and attack traffic. In our simulations, we perform attacks by creating new packets that are similar to non-attack period's traffic. We simulate the following attacks:

- *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.
- *SQL Slammer Worm Attack*: The protocol type of all attack packets is UDP and the destination port is set to 1434. Also, the packet size is between 371- 400 bytes. Other attributes are randomized.
- *DNS Amplification Attack*: The protocol type of attack packets is DNS and the destination port is set to 53. Also, the attack packet size is 60 bytes. Other attributes are randomized.
- *NTP Attack*: The protocol type of attack packets is NTP and the destination port is set to 123. Also, the attack packet size is 90 bytes. Other attributes are randomized.

In addition to these known attacks, other attack types are analyzed to compare the performance of our model and EBS for unknown attacks. These attacks are setup as follows:

- *Generic Attack*: In this attack, attacker generates attack packets with attribute values that are selected randomly in their respective ranges.

- *Generic Attack with Determined Attributes*: Attacker chooses several attributes and assigns most common values to them whereas it uses random values for other attributes. Choosing the most common values for determined attributes enables the attacker to form attack packets similar to legal packets. This approach makes the attacker more powerful and gives the ability to get through the filtering mechanism of the victim. If the attacker utilizes two attributes it is named as *Generic Attack with Determined Two Attributes (GAD-2A)*. For instance, it chooses protocol type "TCP" and destination port "80". Then, the attack packets with these attributes can be mixed up with the legal packets. Similarly, if the attacker chooses three and four attributes, that attack is denoted as *GAD-3A* and *GAD-4A*, respectively.

C. 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 a binary classification problem, we utilize these metrics.

In our case, TP is the number of legal packets that are identified correctly by the system and reach the destination safely. TN is the number of attack packets dropped in the network and stonewalled to prevent reaching the destination. 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. Following metrics calculated via these parameters are utilized for performance measurement:

- *Precision (PN)*: What fraction of the forwarded packets corresponds to legal packets? $\Rightarrow PN = \frac{TP}{TP+FP}$
- *Recall (RL)*: What fraction of legal packets were forwarded to destination?$\Rightarrow RL = \frac{TP}{TP+FN}$
- *Accuracy (AY)*: What fraction of the decisions were correct? $\Rightarrow AY = \frac{TP+TN}{TP+TN+FP+FN}$
- *F-measure (FM)*: This metric combines precision and recall. In other words, it deals with the system's success regarding legal packets. It is the harmonic mean of precision and recall: $\Rightarrow FM = \frac{2\times PN\times RL}{PN+RL}$

D. Experimental Results

In this section, we demonstrate experimental results according to the metrics explained in Section IV-C. Table II shows precision, recall, accuracy and F-measure values for SDNSec and EBS. EBS system detects the attack according to the entropy of the destination port attribute. Then the corresponding destination IP is determined as the victim which is to receive most of the packets. When it determines that victim, it drops all the attack packets including the legal ones destined to that destination. According to the results in Table II, SDNScore outperforms EBS for majority of the metrics. For each attack type, appropriate attributes are to be considered. For TCP SYN Flood attack, protocol type and TCP flag are the arbiters, whereas protocol type, destination port and packet size are identifiers for DNS and NTP attacks. Protocol type and destination port are the identifier pair for SQL Slammer Worm attack. Since SDNScore uses the same pair for DNS and NTP attack, results for both schemes are almost identical for these attacks.

TABLE II
PRECISION, RECALL, ACCURACY AND F-MEASURE

Attack Type	Model	PN	RL	FM	AY
TCP-SYN Flood Attack	EBS:	0.57	1	0.73	0.75
	SDNScore:	0.98	1	0.99	0.99
SQL Slammer Worm Attack	EBS:	0.58	1	0.72	0.79
	SDNScore:	1	1	1	1
DNS Attack	EBS:	0.57	1	0.74	0.94
	SDNScore:	0.99	1	0.99	0.99
NTP Attack	EBS:	0.56	1	0.75	0.93
	SDNScore:	0.99	1	0.99	0.99

Precision and recall metrics represent the system performance for legal traffic. Since F-Measure is the harmonic mean of them, it also deals with the success of the system on legal packets. Thus, SDNScore outperforms EBS as far as it does not care about the legal packets that goes to the victim. The results suggest that F-Measure values are about 0.70 for EBS whereas it is 0.99 for SDNScore. Accuracy metric considers not only legal but also attack packets. The results suggest that SDNScore's decisions are nearly perfect for the attacks whereas EBS cannot meet SDNScore's performance.

In order to compare the performance of our model and EBS for unknown attacks, generic attack and generic attacks with deterministic attributes are performed. Experimental results are shown in Table III. In generic attack, attacker chooses random values for each attribute. SDNScore mechanism gives moderate results of 0.84 accuracy as seen in Table III. It gives 0.85 success on legal packets whereas it allows some attack packets to pass. On the other hand, EBS gives 0.84 accuracy whereas it performs about 0.75 success on legal packets. In SDNScore, since all attributes are chosen randomly, some packets' corresponding values for $SuspiciousPair$ attribute are exactly same as legal packets and scores of these packets do not exceed the threshold since there is not a high amount of packets with these properties. Therefore, these packets can manage to get through SDNScore.

If an attacker becomes more intelligent and tries to generate attack packets more similar to the legal ones, it can choose two attributes and give most popular values for them. Other values are chosen randomly in that case. This corresponds to the GAD-2A case. For instance, it makes attack packets with protocol type = "TCP" and destination port = "80". Then SDNScore results in perfect decision on attack packets (FN=0) whereas it drops some legal packets and thus TN rate increases. Accordingly, its precision value decreases considerably. In this case, SDNScore cannot cope with the legal packets that have the same value with the attack packets. The legal TCP packets that are sent to port 80 are given higher scores

TABLE III
PRECISION, RECALL, F-MEASURE AND ACCURACY FOR UNKNOWN ATTACKS

Attack Type	Model	PN	RL	FM	AY
Generic Attack	EBS:	0.57	1	0.73	0.75
	SDNScore:	0.99	0.75	0.85	0.84
GAD-2A	EBS:	0.57	1	0.73	0.74
	SDNScore:	0.55	1	0.71	0.80
GAD-3A	EBS:	0.57	1	0.73	0.75
	SDNScore:	0.81	1	0.90	0.98
GAD-4A	EBS:	0.57	1	0.73	0.75
	SDNScore:	0.81	1	0.90	0.98

since there is high amount of such packets. As their score exceeds the threshold, they are marked as attack packets. The performance level of SDNScore is on par with EBS figures for this attack.

If the attacker increases the number of attack attributes to three (GAD-3A) and gives most popular values for these attributes, it tries to make his packets more similar to legal ones. For instance, it generates attack packets with protocol type = "TCP", destination port = "80" and packet size = "40". In this case, SDNScore gives perfect results for attack packets whereas it increases the accurate decision performance for legal packets. As the number of utilized attributes increases, the number of legal packets who have the determined properties decreases. Thus, the number of legal packets with scores higher than the threshold decreases. Accordingly F-measure value increases. SDNScore clearly outperforms EBS which is insensitive to the number of packet attributes as seen in Table III.

If the number of determined attributes is further increased to four, SDNScore gives exactly the same result with three attributes. This is because the number of legal packets who have the same values with attack packets for four attributes is almost equal to the number of legal packets who have the same values with attack packets for three attributes. That situation corresponds to the case of saturation where information gain does not contribute to the performance gain anymore.

All these results suggest that SDNScore performs elegantly for unknown attacks. Even in the conditions that the attacker generates packets that are very similar to legal packets, our mechanism gives favorable results and increases the accuracy whereas EBS results do not change and thus fall behind SDNScore.

V. CONCLUSION

In this work, SDNScore, which is a statistical defense mechanism, is proposed against DDoS attacks in SDN environment. Some intelligence embedded in suggested modules is deployed on SDN switches and controller to enable this mechanism. It is a defense mechanism including detection and mitigation against DDoS attacks in software-defined networks. It is effective against unknown attacks since it utilizes statistical analysis and makes comparison with nominal traffic. It determines most appropriate attributes for current traffic and provides considerable improvement in accuracy. We compare SDNScore with an existing entropy based model (EBS) [6] according to precision, recall, f-measure and accuracy metrics. The results suggest that SDNScore outperforms EBS in almost all types of attacks.

As future work, we plan to perform experiments in an SDN testbed. In addition, the role of the controller can be enhanced for prevention. For instance, it can coordinate different switches for preventing future attacks. Besides, it can combine statistics from several switches to provide more accurate scoring.

ACKNOWLEDGMENT

This work is supported by the Turkish State Planning Organization (DPT) under the TAM Project, number 2007K120610.

REFERENCES

[1] H. Selvi, S. Güner, G. Gür, and F. Alagöz, "The controller placement problem in software defined mobile networks (SDMN)," *Software Defined Mobile Networks (SDMN): Beyond LTE Network Architecture*, pp. 129–147, 2015.

[2] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 1, pp. 27–51, Firstquarter 2015.

[3] M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li, "Software defined networking: State of the art and research challenges," *Computer Networks*, vol. 72, pp. 74–98, 2014.

[4] K. Kalkan and F. Alagöz, "A distributed filtering mechanism against DDoS attacks: ScoreForCore," *Computer Networks*, vol. 108, pp. 199–209, 2016.

[5] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Avant-Guard: Scalable and vigilant switch flow management in software-defined networks," in *Proceedings of the 2013 ACM SIGSAC conference on computer & communications security*. ACM, 2013, pp. 413–424.

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

[7] Y. Kim, W. C. Lau, M. C. Chuah, and H. J. Chao, "PacketScore: Statistics-based overload control against distributed denial-of-service attacks," in *INFOCOM 2004*, vol. 4, 2004, pp. 2594–2604.

[8] S. A. Mehdi, J. Khalid, and S. A. Khayam, "Revisiting traffic anomaly detection using software defined networking," in *Recent Advances in Intrusion Detection*. Springer, 2011, pp. 161–180.

[9] G. Yao, J. Bi, and P. Xiao, "Source address validation solution with OpenFlow/NOX architecture," in *Network Protocols (ICNP), 2011 19th IEEE International Conference on*. IEEE, 2011, pp. 7–12.

[10] N. Gde Dharma, M. F. Muthohar, J. Prayuda, K. Priagung, and D. Choi, "Time-based DDoS detection and mitigation for SDN controller," in *Network Operations and Management Symposium (APNOMS), 2015 17th Asia-Pacific*. IEEE, 2015, pp. 550–553.

[11] S. Kasera, J. Pinheiro, C. Loader, M. Karaul, A. Hari, and T. LaPorta, "Fast and robust signaling overload control," in *Network Protocols, 2001. Ninth International Conference on*. IEEE, 2001, pp. 323–331.

[12] R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda, "MAWILab: Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking," in *ACM CoNEXT '10*, Philadelphia, 2010.

[13] H. Jiang, S. Chen, H. Hu, and M. Zhang, "Superpoint-based detection against distributed denial of service (DDoS) flooding attacks," in *Local and Metropolitan Area Networks (LANMAN), 2015 IEEE International Workshop on*, April 2015, pp. 1–6.

[14] Q. Chen, W. Lin, W. Dou, and S. Yu, "CBF: A packet filtering method for DDoS attack defense in cloud environment," in *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on*, Dec 2011, pp. 427–434.

[15] K. Juszczyszyn and G. Kołaczek, "Motif-based attack detection in network communication graphs," in *Communications and Multimedia Security*. Springer, 2011, pp. 206–213.