RAD: A Statistical Mechanism Based on Behavioral Analysis for DDoS Attack Countermeasure

Mosayeb Hajimaghsoodi and Rasool Jalili

***Abstract*— Nowadays, Distributed Denial of Service (DDoS) attacks are among the most prevailing and costly attacks across the networks which challenge a variety of services. While many defense mechanisms are presented to detect and mitigate DDoS attacks, attackers constantly explore alternative approaches for orchestrating novel DDoS attacks. Distribution of the mechanism and its deployment into different zones can improve the accuracy and coverage of DDoS attack varieties. In this paper, we propose a 3-phase DDoS attack countermeasure, named *RAD*, based on a statistical model for scoring users in order to detect DDoS attacks. In the first phase, users are classified into either suspicious or benign based on their traffic behavior, being indicated by the number of flows, packets, concurrent connections, and amount of user-generated traffic. In the second phase, we identify a potential attack state using the drop, jitter, and delay processing parameters. In the third phase, relevant policies are enforced on the suspicious class of users and its effects are assessed continuously in order to reduce false alarms. *RAD* is evaluated through the UNB CICDDoS2019 dataset and is compared with four well-known DDoS detection algorithms. *RAD* counters DDoS attacks with more than 80% precision, 99% recall, and 89% F1-Measure in CICDDoS2019.**

***Index Terms*— Statistical defense mechanism, user behavioral analysis, DDoS attack countermeasure, event correlation.**

I. Introduction

THE denial of service is one of the most dangerous threats to the availability of online services. Since victims usually prefer not to reveal all details about the attack, the exact estimation of the cost incurred by these attacks is a difficult task. On the other hand, Distributed Denial of Service (DDoS) attacks could be easily deployed using botnets which can be rented as a service. DDoS attacks have grown significantly over the past years. According to the latest Kaspersky quarterly DDoS attacks report, the number of DDoS attacks in Q1 2021 increased by 47% compared to Q4 2020, and in Q2 2020 it increased three times in comparison to Q2 2019 [1]. The number of DDoS attacks serving over 100 Gb/s of data increased 967% between 2019 and 2020 [2]. Furthermore, the average DDoS attack volume increased to 200% in Q1 2019 compared to the same time period in 2018.

Manuscript received 22 October 2021; revised 4 February 2022 and 20 March 2022; accepted 16 April 2022. Date of publication 3 May 2022; date of current version 3 August 2022. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Issa Traore. *(Corresponding author: Mosayeb Hajimaghsoodi.)*

The authors are with the Department of Computer Engineering, Sharif University of Technology, Tehran 11365-11155, Iran (e-mail: hajimaghsoodi@ce.sharif.edu; jalili@sharif.edu).

Digital Object Identifier 10.1109/TIFS.2022.3172598

In a DDoS attack scenario, many attackers (bot) being controlled by a master, simultaneously send requests to the victim [3]. The sophistication of bots is rising in terms of their behavioral proximity with human browsing characteristics which makes it difficult to detect. In particular, using legitimate IP addresses, bots are able to perform connection establishment procedures, like a benign user, to portray a false resource demand by sending multiple requests to overload the server [4]. Out of the total bots identified in recent years, 46% were advanced bots, 53% of bots were able to deal with complex logical web elements, and 39% were able to mimic humans [4].

There are two main deployment options for a defense system, standalone at the target server or distributed in the network. Due to the nature of DDoS attacks, researchers usually prefer to deploy their approaches at the victim side. The existence of different deployment locations can improve the accuracy and increase the coverage variety of DDoS attacks. On the other hand, most of the available methods only examine features at the packet level or at the level of traffic flow. Considering the user inter-flow relationships, in addition to reducing the computational overhead, can provide a better presentation of the user actual behavior. In other words, to understand the actual behavior of a user, the relationship among her flows is examined. After DDoS attacks detection and enforcing the relevant policy, correlating events from different locations and continuing assessment of users behavior can decrease false-alarms. In this paper, a defense mechanism, named *RAD*, is introduced. *RAD* consists of a distributed architecture with three phases: **R**anking of users, **A**nnouncement of the victim server state, and **D**ynamic policy enforcement. To the best of our knowledge, *RAD* is the first collaborative statistical model that utilizes the advantages in different zones of defense mechanism for DDoS detection and mitigation. The main contributions of this paper are as follows:

- Enhancing detection accuracy and false positives reduction using continuous monitoring and dynamic policy enforcement on suspicious traffic.
- Mitigating both high-rate DDoS attacks and low-rate types in an efficient statistical model.
- Proposing an inter-flow relation formally which affords a comprehensive set of parameters, by which all users could be represented. The relation can be considered as a tuning parameter, to consider the evolution of traffic over time.

The rest of this paper is organized as follows: Section II discusses the relevant background of the work and previously proposed approaches. Our proposed mechanism is introduced in Section III. The evaluation results are presented in Section IV. Finally, the ongoing challenges and future directions are discussed in Section V.

II. Related Work

Security applications are generally used to countermeasure cyber-attacks such as DDoS attacks on the network. User behaviors are monitored and abnormal traffic flows are identified. Well-known detection mechanisms of DDoS attacks are typically aimed at recognizing attacks based on a specific signature (misuse detection), or at detecting anomalous events [5]–[7].

A. Misuse Detection in DDoS Attacks

The aim of misuse detection is the identification of previously known attacks. However, it cannot uncover novel or mutated intrusions. Due to the variety and variable nature of the network traffic, the distinguishing of different applications based on their pattern is a difficult task for misuse detection mechanisms. Existing defense mechanisms at the application layer have been reviewed by Praseed and Thilagam [8], where different features, such as user puzzles, are used for the detection of these attacks. In [9], the authors suggested using the concept of application fingerprint to detect DDoS attacks. They gathered features from two levels: (i) Packet-level features such as IP address, source and destination port, header length of IP packets, length of packets, and TTL value of packets; (ii) Stream-level features such as total bytes of session packets, the number of packets from client to server, and vice versa. Such features have been used to define the so-called application fingerprint for each application. A clustering framework was introduced according to this fingerprint, which can be used to identify the level of DDoS attacks. They have applied a Bayesian clustering on the information to find the thresholds for the first and second steps of the fingerprint.

In [10], authors evaluated a dataset of botnets and concluded in some important findings such as access patterns of the geographical location of attack sources. In [11], [12], by assessing attack traffic, patterns, and degree of attacks, the authors attempted to provide a new approach in analyzing DDoS attacks.

In the counter-based DDoS detection mechanism, an entropy-based approach is used. David and Thomas [13] and Qin *et al.* [14] have created an array of different entropies, features, and parameters of the data stream and have detected attacks by modeling behavioral patterns of sessions using clustering algorithms [13], [14]. Kalkan *et al.* published a series of three papers on attack predictions to identify patterns of legal packets. First, a distributed filtering mechanism for scoring the packets is proposed [15]. Subsequently, the utilization of the statistical defense mechanism for predicting legal traffic is discussed [16]. Finally, the joint entropy-based DDoS defense scheme in SDN is described [17].

Such solutions have many limitations and challenges. As most of the available methods only examine features at either packets level or at the level of traffic flows, the main challenge, especially in large-scale networks, is the overhead of processing by either per packet or per flow mechanism. Another challenge is having a significant false-positive in the case of selecting inappropriate features for clustering the traffic. Since attackers are acting as benign users and some attack packets do not have any malicious parts, it is not easy to tackle the problem. In other words, the presence of control packets (like the packets with TCP flags) can cause mistakes in packet level calculations.

Nevertheless, there are few mechanisms that consider user inter-flow relationships in the literature. In addition to reducing the computational overhead, considering the user inter-flow relationships can provide a better presentation of the user actual behavior.

B. Anomaly Detection in DDoS Attacks

The aim of anomaly detection is to detect novel and mutated attacks by attempting to define the normal network situation and behavior. DDoS attacks sometimes mimic legitimate user behavior. In order to add complexity to botnet detection, the bots can pick benign messages from a behavior emulation dictionary, then they send a relatively low number of admissible requests to make their behavior non-suspicious [18]. The research community attempted to find properties to distinguish between normal and abnormal behavior. In [18], authors provided a theoretical analysis of the message innovation rate to characterize a botnet identification strategy with multiple emulation dictionaries. Feature extraction can be done either manually or using different methods of machine learning. Anomaly detection mechanisms consider property selection and profiling of normal and abnormal behavior [19]–[21]. Authors in [5], carried out an experiment-based fair comparison to quantify achievable performance and trade-off with different artificial neural networks solution. Their review has provided time complexity data and useful performance metrics that laid the basis for a trade-off analysis across different artificial neural networks. Authors in [22], suggested an anomaly detection method to filter the abnormal traffic flow by checking the first few bytes of each flow. They used two deep learning models for classifying spatial and temporal features of traffic flows respectively. A hybrid approach named AE-MLP that combines two deep learning-based models for effective DDoS attack detection and classification was proposed in [23]. The authors extracted the most important and relevant features for detecting malicious DDoS network payloads, using Auto-Encoder model. They fed the compressed features to the multilayer perceptron to effectively classify different DDoS attack types. Although the machine learning and counter-based mechanisms are used to detect anomalies in the traffic, the machine learning mechanisms look at the derivation of normal values in parameters to detect attacks; while the counter-based mechanisms are more eager to detect anomalies using the traffic itself relying on methods such as traffic classification or deep packet inspection (DPI) [10], [11].

Different features are used to calculate entropy and to form an array. In [24]–[26], to consider DDoS attacks in IoT, features related to requests and traffic packets like the number of

requests per unit, the number of packets per unit, the average size of packets, and also features such as inter-request times or average time between request and answers are gathered.

In [27], the authors proposed a method based on Shannon entropy and granular computing to select some potential features of DDoS attacks. In [28], a variation of Lyapunov exponent is proposed to detect anomalies in network traffic, based on entropy. If the Lyapunov index is zero, it means source and destination IPs have the same distribution which implies that traffic is benign. As this index value increases the attack is more probable.

In [14], the authors used five features including source and destination IPs, destination port, packet length, and session length in order to obtain entropy. They used a clustering algorithm like K-means to train a model, aiming at modeling normal traffic patterns and identifying thresholds for detecting attacks. They calculated entropy for each session and found the cluster having the minimum distance with the entropy array of the session. If this distance was less than the diameter of cluster traffic, the session would be normal; otherwise, it would show an attack.

A well-defined systematic review has been conducted by Singh *et al.* [29] that formulated certain research questions, such as detection attributes, to capture various aspects of the identified primary studies. Their results show that as fortifying every facet of an application is nearly impossible, DDoS attacks focus on individual susceptible areas instead of unleashing a huge traffic flood towards the victim. Also, there is a significant difference between the attributes selection and detection processes for lower layer and application layer DDoS attacks. For example, the detection of HTTP-GET flood attacks mainly relies on monitoring the browsing semantics of users whereas lower layer attack detection mainly considers traffic-related characteristics.

In [4], two variables of Up-time and Down-time are specified. Up-time is the time interval that a user sends a request to the server and Down-time is the time interval that the user does not send a new request and consumes the resources that received in the previous up-time. Users can request multiple resources from the server simultaneously, and so, down-time should be extended. Now, if we consider the proportion of down-time with respect to up-time as a request rate, we will see this parameter is completely different for benign traffic and attack traffic.

Anomaly-based mechanisms suffer from some problems and limitations. Finding thresholds for assessment is one of these challenges. Because the incoming traffic contains innocent users, a bad threshold helps attackers to reach their goals. Finding a good threshold is necessary for different types of users (like home, office, or research users) and also for different applications. For example, the threshold for a video streaming application could be very different from a news website or email application.

The above summary highlights the potential application of the methods in the context of application-layer DDoS attacks. These methods have certain limitations and the complexity of the identification algorithm increases quadratically with an increase in network size [18]. This problem can be overcome due to the relationship between the performance parameters at the network level and the processing overhead of the requests sent to the victim. Hence, certain processing parameters in a victim system can be used as features for anomaly detection. A cooperative defense mechanism gives more accurate decisions since it can be very effective against DDoS attacks with more knowledge of the network.

To detect and mitigate the effects of DDoS attacks, using a method that can analyze the behavior of the user after enforcing the policy could be very useful. In many previous solutions, attack detection and policy enforcement is done only after the attack has happened and no feedback has been gathered from the behavior of the blocked traffic.

III. The Proposed Defense Mechanism

Event correlation in different network entities is an important approach for DDoS detection. This approach can be feedbacked with monitoring of the result of policies enforced on suspicious traffic. In this paper, we use the correlation between events in different locations to mitigate the effect of DDoS attacks and decrease the false positive generated by a detection system.

In this paper, a defense mechanism is introduced to detect DDoS attacks, named RAD that uses a collaborative statistical model and utilizes the advantages in different deployment locations of defense mechanism for DDoS detection and mitigation. RAD has a distributed architecture with three phases: **R**anking of users, **A**nnouncement of the victim server state, and **D**ynamic policy enforcement. As shown in Fig. 1, in the first phase, a profile is created for each user based on her previous behavior considering the network performance parameters. A statistical model is used to rank all users based on their benignity. This task is done in the middlebox devices. Simultaneous to the first phase, the processing parameters are measured in order to detect abnormal situations at the server-side. In cases where the processing parameters exceed a nominal threshold, an announcement of the attack state is raised and sent to the next phase. Ultimately, on receiving the announcement an appropriate policy is enforced on the suspicious traffic and at the same time, the behavior of the supposed attackers is dynamically monitored. If the response of those users is similar to a normal node, we mark them as benign; otherwise, we continue blocking them.

In RAD, we consider a time interval (T_i) for each phase that is separately tuned. The parameters are calculated at the end of each time interval in each phase. The value of (T_i) differs depending on parameters such as type and number of users, type of incoming traffic, and also the number of available computing resources at each phase. The initial value for the time interval of the first, second, and third phases are set to ($T_1 = 30sec$), ($T_2 = 60sec$), and ($T_3 = 15sec$) respectively.

The motivation of having (T_3) less than the others is to decrease the negative effect of probable false detection of users. In other words, after inspecting blocked flows, if a flow is mistakenly considered as an attack, it will be recovered to its normal state after 15 seconds. In the following sub-sections, we elaborate on the three phases of our mechanism.

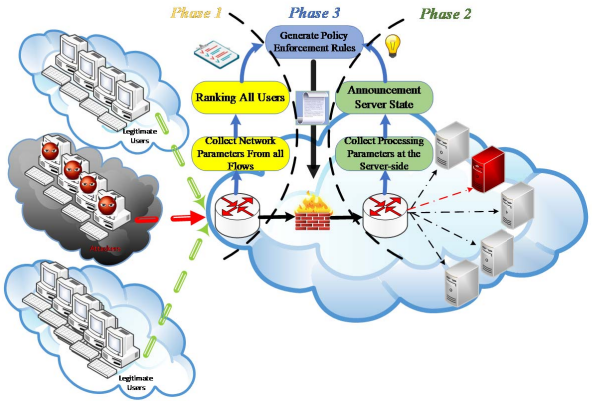


Fig. 1. The phases of RAD.

A. The First Phase: Detection and Classification of Suspicious Users

Where RAD generates the necessary statistical information in the first phase in its distributed architecture, middle-box nodes such as firewalls create a profile for different users using network performance parameters of their traffic. This phase includes the preparation steps for ranking of users that will be used during attack mitigation. In this phase, no policy is enforced on traffic. The parameters and relevant terms in the first phase are described in Table I with details explained in the following subsections:

1) Selection of User Behavior Parameters: To identify the behavior of a user, we should examine the users' inter-flow relationship. Authors in [30] define a flow data structure f $=<$ source-IP, source-port, destination-IP, destination-port, TCP/UDP protocol, ea_1, …, ea_d $>$. The first five attributes in the tuple are the well-known flow specification, the other attributes are extended to describe the flow more accurately [30]. As the calculations are based on packets from different user flows (system input), at the first, we should define a structure to store the necessary information of each packet. Then, the relation between packets forming a flow and the relation of different flows determining the user traffic should be defined formally. We define a packet structure to describe typical packet information.

$$PKT_s = \{(IP_{src}, PRT_{src}, IP_{dst}, PRT_{dst}, PRTCL_{L4}, PKT_{dir}, TS, SZ)\}. \quad (1)$$

PKT_s consists the set of extracted necessary information for each packet. PKt_{dir} can be either CtS for the packets transmitted from client to server, or StC for the packets transmitted from the opposite direction. TS indicates the packet received instantly at the middle-box and generated packet structure (PKT_s set). Keep in mind that the timestamp data type is used for TS and includes a trailing fractional seconds part. Eventually SZ stores the size of each packet in bytes.

Packets belong to a specific flow if the first six attributes in the PKT_s had the same values or if the packets direction had not the same value, the source IP and source port of each one had the same values as the destination IP and destination port of another one. The binary relation R over PKT_s means that the packet belongs to a flow, as defined formally:

$$\begin{aligned}
&\forall a, b \in PKT_s \\
aRb \iff & IP_{src}(a) = IP_{src}(b), \\
PRT_{src}(a) &= PRT_{src}(b), \\
IP_{dst}(a) &= IP_{dst}(b), PRT_{dst}(a) = PRT_{dst}(b), \\
PRTCL_{L4}(a) &= PRTCL_{L4}(b), PKT_{dir}(a) = PKT_{dir}(b) \\
&\quad | \\
IP_{src}(a) &= IP_{dst}(b), PRT_{src}(a) = PRT_{dst}(b), \\
IP_{dst}(a) &= IP_{src}(b), PRT_{dst}(a) = PRTsrc(b), \\
PRTCL_{L4}(a) &= PRTCL_{L4}(b), PKT_{dir}(a) \neq PKT_{dir}(b).
\end{aligned}$$

TABLE I
MECHANISM TERMS AND PARAMETERS

Term	*Explanation*
DPI	*Abbreviation of Deep Packet Inspection*
PRT	*The port number of the packet*
$PRTCL$	*The type of transport layer protocol of the packet*
TS	*The time of packet received at the middle-box*
SZ	*The size of the packet in byte*
CtS	*The packet direction from client (user) to server*
StC	*The packet direction from server to client (user)*
PKT_s	*The set contains extracted necessary information for each packet*
R	*The binary relation shows the packet belongs to a flow*
F	*A set indicates a partition of the PKT_s set under the R relation*
IP	*A set consists all possible value for IP address*
$Port$	*A set consists all possible value for Port in the transport layer*
$Protocol$	*A set consists only TCP or UDP protocol*
$Time$	*A set consists positive integers for packets received time*
$Count$	*A set consists positive integers for numerating the packets*
S	*A set indicates the codomain for MFS relation*
MFS	*A relation indicates inter-flow relationship*
x	*The equivalence class of "a" under R relation*
U	*The set of all users in the traffic*
UF	*A relation demonstrates number of active flows for the user u*
UoP	*The user associated with "a" packet*
$Precede$	*The previous packets belong to flow that are received before "a" packet*
MPS	*A data structure to aggregate the necessary metadata of user traffic*
$FpS(u)^{(t)}$	*The number of new flows created from user u in the $T_1(t^{th})$ time interval*
$PpS(u)^{(t)}$	*The number of transmitted packets for user u in the $T_1(t^{th})$ time interval*
$UGT(u)^{(t)}$	*The summation of the size of all received packets for user u in the $T_1(t^{th})$ time interval*
$CC(u)^{(t)}$	*The number of created and active flows for user u in the $T_1(t^{th})$ time interval*
$\overline{du}(t)^{(t)}$	*The average of flows duration flows for user u in the $T_1(t^{th})$ time interval. $CC(u)^{(t)}$ and $\overline{du}(t)^{(t)}$ is used interchangeably in this paper*
$ULP(useru)$	*The probability of a user being legal using parameters related to her flows*
N_U	*The total number of users ($\|U\|$) which is the sum of attackers (N_A) and benign (N_B) users*
N_A	*The number of attackers*
N_B	*The number of benign users*

In relation R; each packet "a" belonging to PKT_s, relates to itself ($\forall a \in PKT_s : aRa$) thus R has the reflexive property. On the other hand, it can be proved that the binary relation R has symmetric ($\forall a, b, c \in PKT_s : aRb, bRa$) and transitive ($\forall a, b, c \in PKT_s : aRb, bRc \longleftrightarrow aRc$) properties. Thus, the binary relation R is an equivalence relation. The set of all traffic flows under relation R is defined as follows:

$$F = \{[a]_R | a \in PKT_S\}.$$

The F set is a partition of the PKT_s set and consists of all classes under the R relation such that every two elements of the given PKT_s set are equivalent to each other, if and only if they belong to the same equivalence class (same flow). So, F presents all flows in the network traffic. In order to indicate inter-flow relationship we define the relation $MFS(.)$ to describe a meta-flow structure. Lets:

$$\begin{aligned}
IP &= \{(w, x, y, z) | 0 \leq w, x, y, z \leq 255\} \\
Port &= \{p | 0 \leq p \leq 65535\} \\
Protocol &= \{UDP, TCP\} \\
Time &= \{t | t \in Z^+\} \\
Count &= \{c | c \in Z^+\} \\
S &= \{(x_1, x_2, x_3, x_4, x_5, x_6, x_7) | x_1, x_3 \in IP, \\
&\quad x_2, x_4 \in Port, x_5 \in Protocol, \\
&\quad x_6 \in Time, x_7 \in Count\}. \\
&\quad MFS : F \longrightarrow S; \forall x \in F, \\
x &= [a]_R, PKT_{dir}(a) = CtS \\
MFS(x) &= (f_1(x), f_2(x), f_3(x), f_4(x), f_5(x), f_6(x), f_7(x)) \\
&\quad \begin{cases} f_1(x) = IP_{src}(a), f_2(x) = PRT_{src}(a), \\ f_3(x) = IP_{dst}(a), \\ f_4(x) = PRT_{dst}(a), f_5(x) = PRTCL_{L4}, \\ \forall b \in [x] : f_6(x) = min(TS(b)), f_7(x) = |x| \end{cases}
\end{aligned} \tag{2}$$

$f_6(x)$ and $f_7(x)$ indicate the start time of a flow and the sequence number of received packets in the flow. After receiving the last packet, the $f_7(x)$ value is updated in the flow structure. According to $MFS(.)$, the set of all users (U) can be defined:

$$U = \{u \in IP | \forall x \in F, u = f_1(x)\}.$$

Flows belong to the user u; if the source IP address is $IP(u)$ and the value of packet direction is CtS, or the destination IP address is $IP(u)$ and the value of packet direction is StC. In the following, flows belonging to user u are defined formally.

$$\begin{aligned}
UF : U \longrightarrow F; \forall u \in U \\
UF(u) = \{x \in F | f_1(x) = u\}.
\end{aligned}$$

A data structure is also proposed to aggregate the necessary metadata of user traffic (meta-packet structure). For this purpose, we introduce the $UoP(.)$ and $Precede(.)$ relations that demonstrate the user associated with the packet and previous packets belonging to the flow that are received before "a."

$$\begin{aligned}
UoP : PKT_s \longrightarrow U; \forall a \in PKT_s, x = [a]_R \\
UoP(a) = f_1(x) \\
Precede : PKT_s \longrightarrow PKT_s; \forall a \in PKT_s \\
Precede(a) = \{b \in [a]_R | 0 \leq TS(b) \leq TS(a)\} \\
MPS = \{(u, mfs, sz, ts, n^{th}) | \forall a \in PKT_s; u = UoP(a), \\
mfs = MFS([a]_R), sz = Size(a), ts = TS(a), \\
n^{th} = |Precede(a)|\}.
\end{aligned} \tag{3}$$

u and mfs determine the user and the flow associated with the packet and SZ stores the size of the packet. TS indicates the packet received time at the middle-box. n^{th} determines the number of packets in the flow. Additionally, the following parameters have been considered in this phase:

a) Flows Per Second (FpS): We define FpS as the number of new flows created from a specific source which is calculated for each user u. As stated earlier each flow is identified by five attributes: source IP, destination IP, source port, destination port, and transport layer protocol (TCP or UDP). Flows have the same source IP address belonging to the same user. So, for the same user u, a different value for each of the other four attributes is considered as a new flow. The FpS value is related to the behavior of each user. Specifically, FpS for user u is calculated as follows:

$$\begin{aligned}
FpS(u)^t &= \frac{\sum_{j=T_1(t-1)}^{j=T_1(t)} \sum_{f=1}^{n} |A_0|}{T_1(t) - T_1(t-1)} \\
A_0 &= \{(a_1, a_2, a_3, a_4, a_5) \in MPS | a_1 = u, \\
a_2 &= MFS(f), a_4 = j, a_5 = 1\}.
\end{aligned} \tag{4}$$

where n (in the $\sum$ phrase) is the cardinal number of F (a partition of PKT_s under R). For user u, we count every packet whose $a_5 = 1$ in the $T_1(t^{th})$ time interval, which indicates a new flow establishment. Using Eqn. 4, all new flows for each user are calculated in the $T_1(t^{th})$ time interval.

b) Packets Per Second (PpS): PpS is the number of transmitted packets, whose a_4 is in the $T_1(t^{th})$ time interval, calculated per second for each user u. We consider PpS as an important indicator of user behavior. $PpS(u)^{(t)}$ is calculated as follows:

$$\begin{aligned}
PpS(u)^t &= \frac{\sum_{j=T_1(t-1)}^{j=T_1(t)} \sum_{f=1}^{n} |A_1|}{T_1(t) - T_1(t-1)} \\
A_1 &= \{(a_1, a_2, a_3, a_4, a_5) \in MPS | a_1 = u, \\
a_2 &= MFS(f), a_4 = j\}.
\end{aligned} \tag{5}$$

In other words, $PpS(u)^{(t)}$ can be calculated as the difference of the sequence number of the last packet in each flow from the smallest value of this parameter in each time interval. We use $f_7(x)$ to get the last packet number of every flow.

$$\begin{aligned}
PpS(u)^t &= \frac{\sum_{f \in UF(u)} (f_7(f)) - A_2}{T_1(t) - T_1(t-1)} \\
A_2 &= min\{a_5 | \forall (a_1, a_2, a_3, a_4, a_5) \in MPS; \\
a_1 &= u, a_2 = MFS(f), T_1(t-1) < a_4 \leq T_1(t)\} \\
&\quad where\ UF(u)\ is\ the\ number \\
&\quad of\ active\ flows\ for\ "u''.
\end{aligned} \tag{6}$$

c) User Generated Traffic (UGT): Attackers either generate requests at a very high rate (such as TCP flood attacks) or request for large-sized responses (such as DNS amplification flooding attacks), in order to make the server unreachable to benign users by keeping bandwidth busy. We define User Generated Traffic (UGT) as an indicator to consider a user as benign or attacker. UGT is calculated as the summation of the size of all received packets in each time interval.

$$\begin{aligned} UGT(u)^t &= \sum_{i=1}^{|A_3|} A_3 \\ A_3 &= \{a_3 | \forall (a_1, a_2, a_3, a_4, a_5) \\ &\in MPS; a_1 = u, a_2 = MFS(f), \\ T_1(t-1) &< a_4 \leq T_1(t)\}. \end{aligned} \tag{7}$$

d) Concurrent Connections per User (CC): Concurrent connections (CC) means the number of created and active flows at any moment. This parameter has a direct relation to the average of flows duration for each user, ($\overline{du}(u)$). Accordingly, if the number of open flows per user enhances, the flows duration average and the number of concurrent connections will be increased. So, the average of flow duration and concurrent connections per user can be used interchangeably. In slow-rate DDoS, attackers establish some flows and do not terminate them in order to put the victim under pressure by keeping flows alive. ($\overline{du}(u)$) is formally illustrated in Eqn. 8.

$$\begin{aligned} \overline{du}(u)^{(t)} &= \frac{\sum_{f \in UF(u)} (A_4 - f_6(f))}{|UF(u)|} \\ A_4 &= max\{a_4 | \forall (a_1, a_2, a_3, a_4, a_5) \in MPS a_1 = u, \\ a_2 &= MFS(f), T_1(t-1) < a_4 \leq T_1(t)\} \\ &\quad where\ f_6(f)\ is\ the\ time\ of \\ &\quad first\ packet\ in\ a\ flow\ f,\ and \\ &\quad UF(u)\ is\ the\ number\ of \\ &\quad active\ flows\ for\ "u". \end{aligned} \tag{8}$$

To obtain $\overline{du}(u)^{(t)}$, we calculate the difference between the maximum time of packets in each flow and $f_6(f)$. A flow is inactive if either it has been terminated (through FIN or RESET) or it has not received any packet for 300 seconds. In the cases, where a flow contains no packet so far, $T_1(t)$ is used as the max phrase in Eqn. 8.

The value of the four aforementioned parameters is calculated in the time interval of the first phase (T_1). Specifically, every 30 seconds these values are collected, and ranking of them are done at the end of the time interval. Normalization of the above parameters can improve the accuracy of classification for profiling. A special value can be considered for each parameter per Gbps of bandwidth. These values for different applications and networks are different. For example, in [31] for each 1Gbps bandwidth, 5000 new sessions per second are considered. Furthermore, other parameters can also be calculated using properties of the network himself and eventually, considering dedicated bandwidth per each user. Our goal here is to rank users by mixing the user behavior parameters based on user activity (according to the previous traffic behavior of the user, type of traffic, and list of used applications). We can compute multiple rankings based on different applications for users to improve accuracy, but here we compute one user ranking list for the sake of simplicity.

2) Statistical User Ranking: We perform a statistical user ranking using *User Legitimate Probability* (ULP) which is a combination of the four proposed parameters. The concept of ULP is adopted from the CLP scoring mechanism [32] that uses the Bayesian theorem for ranking packets. CLP indicates the probability of being legitimate for every packet by comparing the values of the input packet properties with the baseline profile. In ULP, we define the probability of a user being legal using parameters related to her flows. ULP is computed based on each user inter-flow correlation using the FpS, PpS, UGT, and CC parameters. FpS, PpS, and UGT support detection of high-rate DDoS attacks significantly, while the CC parameter contributes toward the detection of slow-rate DDoS attacks.

User scoring using ULP versus packet scoring using CLP has a lower processing overhead of policy enforcement following the attack detection. Furthermore, due to the existence of control packets in network traffic, packet scoring is faced with many challenges in selecting suitable features. For each user u, $ULP(u)$ is formally yielded in Eqn. 9.

$$\begin{aligned} ULP(u) &= P(user\ u\ is\ legitimate | u'FpS \\ parameter &= fs_u, u'PpS\ \ parameter = ps_u, u'CC \\ parameter &= cc_u, u'UGT\ \ parameter = ug_u). \end{aligned} \tag{9}$$

According to the proposed method in [32] and considering the introduced parameters, user legitimate probability can be rewritten as follows:

$$\begin{aligned} ULP(u) &= \frac{P((u = legitimate) \bigcap (A_5))}{P(A_5)} \\ A_5 &= (FpS{=}fs_u, PpS{=}ps_u, CC{=}cc_u, UGT{=}ug_u). \end{aligned} \tag{10}$$

In Eqn. 10, the numerator is the ratio of the class size of legal users ($Cnt_B(fs_u, ps_u, cc_u, ug_u)$) whose behavioral parameters are $\{fs_u, ps_u, cc_u, ug_u\}$ among all users (N_U). The denominator is the ratio of the class size ($Cnt_U(fs_u, ps_u, cc_u, ug_u)$) whose behavioral parameters are $\{fs_u, ps_u, cc_u, ug_u\}$ among all users. To formalize the concept of ULP, we define some notations. (N_U) indicates the total number of users ($|U|$) which is the sum of attackers (N_A) and benign (N_B) users ($N_U = N_B + N_A$). In order to classify the users based on their behavioral parameters, we assume that $FpS = \{fs_1, fs_2, fs_3, \ldots, fs_w\}$ is the set of all the quantized values for the FpS parameter. In other words, fs_i represents the i^{th} quantized value for the FpS parameter. Nevertheless, the quantized fs_u is a member of FpS set. Also, the number of users whose FpS parameter is equal to fs_i, is represented by the symbol $Cnt(FpS = fs_i)$. Similarly, the set $\{ps_1, ps_2, ps_3, \ldots, ps_x\}$, $\{cc_1, cc_2, cc_3, \ldots, cc_y\}$, and $\{ug_1, ug_2, ug_3, \ldots, ug_z\}$ are defined for the PpS, CC, and UGT parameters.

$$\begin{aligned} N_j &= Cnt_j(X{=}x_1){+}Cnt_j(X{=}x_2){+}Cnt_j(X{=}x_3){+}\ldots \\ where\ j &= \{U, B, A\}\ and\ X{=}\{FpS, PpS, CC, UGT\}. \end{aligned}$$

For example, the total number of users is equal to:

$$N_U = \sum_{i=1}^{w} Cnt_U(FpS = fs_i) = \sum_{k=1}^{x} Cnt_U(PpS = ps_k) = \sum_{l=1}^{y} Cnt_U(CC = cc_l) = \sum_{o=1}^{z} Cnt_U(UGT = ug_o).$$

In order to specify different classes of users, we joint all possible values for the parameters. Hence, the number of users whose FpS, PpS, CC, and UGT parameters are equal to fs_i, ps_k, cc_l, and ug_o, is indicated by the symbol $Cnt(FpS = fs_i, PpS = ps_k, CC = cc_l, UGT = ug_o)$ and it can be defined for all the users, attackers, and benign users separately.

$$N_U = \sum_{i=1}^{w}\sum_{k=1}^{x}\sum_{l=1}^{y}\sum_{o=1}^{z} Cnt_U(A_6)$$
$$N_A = \sum_{i=1}^{w}\sum_{k=1}^{x}\sum_{l=1}^{y}\sum_{o=1}^{z} Cnt_A(A_6)$$
$$N_B = \sum_{i=1}^{w}\sum_{k=1}^{x}\sum_{l=1}^{y}\sum_{o=1}^{z} Cnt_B(A_6)$$
$$A_6 = (FpS = fs_i, PpS = ps_k, CC = cc_l, UGT = ug_o).$$

Nevertheless, we can rewrite Eqn. 10 as:

$$ULP(u) = \frac{Cnt_B(A_7)/N_U}{Cnt_U(A_7)/N_U}$$
$$A_7 = (FpS{=}fs_u, PpS{=}ps_u, CC{=}cc_u, UGT{=}ug_u).$$

We multiply the numerator by $\frac{N_B}{N_B}$ and the denominator by $\frac{N_U}{N_U}$. Hence, $ULP(u)$ is rewritten as follows:

$$ULP(u) = \frac{N_B * Cnt_B(A_7)/N_B}{N_U * Cnt_U(A_7)/N_U}. \tag{11}$$

In Eqn. 11, the phrase $((Cnt_j(A_7)/N_j), where j = \{U, B\})$, is defined as the user ratio that demonstrates the probability of the relevant users with mentioned parameter values among all users (N_U) as well as attackers (N_A) and benign users (N_B) separately. Similar to [32], we use logarithm calculations and delete constants for simplicity in computing $ULP(u)$.

$$\begin{aligned} ULP(u) &\cong log\frac{N_B * Cnt_B(A_7)/N_B}{N_U * Cnt_U(A_7)/N_U} \\ &= log\frac{N_B}{N_U} + log\frac{Cnt_B(A_7)/N_B}{Cnt_U(A_7)/N_U} \\ &\cong log\frac{Cnt_B(A_7)}{N_B} - log\frac{Cnt_U(A_7)}{N_U}. \end{aligned}$$

One of the main points in the Bayesian theorem is the independence of the selected parameters. In this paper, this is achieved due to considering users' inter-flow relationships and the independence of the declared parameters. Therefore:

$$\begin{aligned} ULP(u) =& \sum_X log\frac{Cnt_B(X = x_u)}{N_B} - log\frac{Cnt_U(X{=}x_u)}{N_U} \\ & where(X = x_u) \\ =& \{(FpS{=}fs_u, PpS{=}ps_u, CC{=}cc_u, UGT{=}ug_u)\}. \end{aligned} \tag{12}$$

Algorithm 1 The First Phase Algorithm.

Input: Network Traffic
Output: Users_Ranked_List[]

Begin
while True **do**
  Construct packet, meta-flow and meta-pkt structure (PKT_s, MFS, and MPS)
  Collect All Flows in Network for $T_1(t)$ period
  for u = 0 to |U| **do**
    Calculate performance parameters:
    $FpS(u)^t = \frac{\sum_{j=T_1(t-1)}^{j=T_1(t)}\sum_{f=1}^{n} A_0}{T_1(t)-T_1(t-1)}$
    $PpS(u)^t = \frac{\sum_{j=T_1(t-1)}^{j=T_1(t)}\sum_{f=1}^{n} |A_1|}{T_1(t)-T_1(t-1)}$
    $UGT(u)^t = \sum_{i=1}^{|A_3|} A_3$
    $\overline{du}(u)^{(t)} = \frac{\sum_{f\in UF(u)}(A_4 - f_6(f))}{|UF(u)|}$
    Compute $ULP(u)^{(t)} \cong \sum_X log\frac{Cnt_U^{T_1(0)}(X=x_u)}{N_U^{T_1(0)}} - log\frac{Cnt_U^{T_1(t)}(X=x_u)}{N_U^{T_1(t)}}$
    Users_Ranked_List.append($ULP(u)^{(t)}$)
  end for
  Sort Users_Ranked_List[]
  Send Users_Ranked_List[] to the Third Phase
end while
End

The earlier phrase in $\sum_X(.)$ demonstrates the proportion of the legitimate users with mentioned parameter values among all legitimate users. In the attack free time duration ($T_1(0)$), it can be assumed the proportion of the relevant users with mentioned parameters values among all users, named normal profile. The later phrase demonstrates the same ratio for the $T_1(t^{th})$ time interval when the victim is under attack. So, we can rewrite Eqn. 12 as:

$$\begin{aligned} &ULP(u)^{(t)} \\ &\cong \sum_X log\frac{Cnt_U^{T_1(0)}(X = x_u)}{N_U^{T_1(0)}} - log\frac{Cnt_U^{T_1(t)}(X = x_u)}{N_U^{T_1(t)}}. \end{aligned} \tag{13}$$

It is worth mentioning that the quantized value of each fs_u, ps_u, cc_u, ug_u is a member of the possible values set of that parameter; $fs_u \in \{fs_1, fs_2, fs_3, \ldots, fs_w\}$, $ps_u \in \{ps_1, ps_2, ps_3, \ldots, ps_x\}$, $cc_u \in \{cc_1, cc_2, cc_3, \ldots, cc_y\}$, and $ug_u \in \{ug_1, ug_2, ug_3, \ldots, ug_z\}$. ULP is computed for each user at the end of the first phase time interval ($T_1(t)$). The normal profile can vary depending on how users use different services in one day which creates several normal profiles for different times. This aspect is out of scope for our work despite being an interesting research direction.

B. The Second Phase: Detecting Abnormal Situation at the Server-Side

The second phase of our defense mechanism which is done at the server-side (victim), works based on the parameters that

demonstrate the effect of the victim resources consumption on the packet processing queue. Processing parameters of this phase are:

- **Average delay of packet processing** ($\overline{Dly}$) ($\overline{Dly}$) is the average delay in processing the packets from users (legal users or bots) which are in the server processing queue. This parameter is the timespan between receiving a packet to the server and sending the response to the user. Technically, to estimate delay, we calculate for each flow f, the timespan between receiving the first packet of request $ptime_1^f$ in the server and the time of sending the last packet of response from the server $ptime_n^f$. This timespan is divided by the number of packets in the flow to derive the average packet processing delay, as demonstrated in Eqn. 14.

$$\overline{d^{[f]}} = \frac{ptime_n^f - ptime_1^f}{n}. \tag{14}$$

  Now, ($\overline{Dly}(t)$) is calculated as the average packet processing overhead of the server in $T_2(t)$ time interval ($t = n * T_2$). Supposing m as the number of flows, Eqn. 15 is used to obtain ($\overline{Dly}(t)$) as the average of $\overline{d^{[f]}}$ for all the flows. Whenever the server is under attack, ($\overline{Dly}$) is significantly increased and if its value exceeds the threshold (α), the server will announce an abnormal situation to the middle-box.

$$\overline{Dly}(t) = \frac{\sum_{f=1}^{m} \overline{d^{[f]}(t)}}{m}. \tag{15}$$

- **Jitter in packet processing (Jtr):** Jitter is the difference between the delays of consequent packets processing. We define Jtr as the difference between the average latency of packet processing of the previous time interval $T_2(t-1)$ and that of the current one $T_2(t)$, as illustrated in Eqn. 16. This parameter is also increased in the case of attack and when exceeds the threshold (β) the alarm should be sent to the middle-box.

$$Jrt(t) = \overline{Dly}(t) - \overline{Dly}(t-1). \tag{16}$$

- **The number of dropped packets per second (Drp):** Drp is the number of dropped packets belonging to different flows in a time interval $T_2(t)$ which is the result of overflowing of the processing queues in the server. Considering m as the number of flows and Dr_t^f as the number of dropped packets of a flow f in the time interval t, $Drp(t)$ is calculated as in Eqn. 17. Likewise, in the case of attack, $Drp(t)$ exceeds the threshold (λ) and an alarm is sent to the middle-box.

$$Drp(t) = \sum_{f=1}^{m} Dr_t^f. \tag{17}$$

For simplicity, the number of dropped packets can be obtained from *pcap_stat* from the pcap library in the OS kernel. To express a unique concept of victim overhead, we need to combine three processing parameters. Due to the different types of the mentioned scalar quantity parameters, a numeric value should be considered for each of them. Hence we can rewrite Eqn. 15, 16, and 17 as:

$$\begin{aligned}
\overline{Dly^n}(t) &= Func_1(\overline{Dly}(t)) \\
Func_1 : R^+ &\longrightarrow [0,1],\ y = F(x) \\
0 &\leq Func_1(\overline{Dly}(t)) \leq 1 \\
Jrt^n(t) &= Func_2(Jrt(t)) \\
Func_2 : R^+ &\longrightarrow [0,1],\ y = G(x) \\
0 &\leq Func_2(Jrt(t)) \leq 1 \\
Drp^n(t) &= Func_3(Drp(t)) \\
Func_3 : R^+ &\longrightarrow [0,1],\ y = H(x) \\
0 &\leq Func_3(Drp(t)) \leq 1.
\end{aligned}$$

α, β, and λ are the expected thresholds for delay, jitter, and dropped packets. If the value of the processing parameters exceeds these thresholds (α, β, and λ), the $Func_1(.)$, $Func_2(.)$, and $Func_3(.)$ values will be greater than zero. Depending on the type of service, each of the processing parameters has a different level of importance in the network. For example; packet delay has a strong impact on the TCP mechanism as it impresses on congestion window size; jitter influences the user experience in streaming services such as call conference in social network application; packet drop is used by TCP congestion control mechanism for network congestion detection. Accordingly, $Func_1(.)$, $Func_2(.)$, and $Func_3(.)$ must be fine-tuned to clarify the effect of processing parameters on the type of services.

The value of the three aforementioned parameters is calculated within the second phase time interval ($T_2 = 60s$). So, every 60 seconds, such parameters are used to classify if the server state is normal or abnormal. A careful observation should be made to set the appropriate thresholds considering the peak/un-peak state of traffic. When an abnormal condition is identified, an alarm is issued to the next phase to perform an appropriate action.

C. The Third Phase: Dynamic Policy Enforcement and Continuous Assessment

Using the approaches introduced in the previous phases, users are ranked based on their behavior in the middle-box and abnormal situations can be detected at the server-side. The purpose of the third phase is to scrutinize the victim load and, if needed, apply the policy of users who have behaved suspiciously.

The level of ranked users (outcome of the first phase) being affected by policy depends on the severity of the alarm received from the second phase. Using the packet processing parameters, we estimate the load status of the victim as in Eqn. 18 for each time interval $T_3(t)$, where $\frac{t}{T_2} = \lfloor \frac{t}{T_2} \rfloor$.

$$\begin{aligned}
Load(t) &= \theta * A_8 + (1-\theta) * Load(t-1) \\
A_8 &= \overline{Dly}^n(t') + Jtr^n(t') + Drp^n(t').
\end{aligned} \tag{18}$$

where $t' = \lfloor \frac{t}{T_2} \rfloor * T_2 = T_2(\lfloor \frac{t}{T_2} \rfloor)$ and $\theta \in [0,1]$ is a ratio which represents the trade-off between the victim prior load and its current load; the greater θ shows an immediate reaction. The maximum possible value for the sum of three parameters

Algorithm 2 The Second Phase Algorithm.

Input: The Statistics of Victim Kernel Space OR Network Traffic

Output: Attack Situation Alarm and Intensity

Begin
while True **do**
  $\overline{Dly^n}(t) = 0, \quad Jrt^n(t) = 0, \quad Drp^n(t) = 0$
  Situation = 'Normal'
  Collect all processing parameter at server-side for $T_2(t)$ period
  for f=1 to m **do** % m is the number of flows.
    $\overline{d^{[f]}} = \frac{ptime_n^f - ptime_1^f}{n}$
  end for
  $\overline{Dly}(t) = \frac{\sum_{f=1}^{m} \overline{d^{[f]}(t)}}{m}$
  $Jrt(t) = \overline{Dly}(t) - \overline{Dly}(t-1)$
  $Drp(t) = \sum_{f=1}^{m} Dr_t^f$
  if $\overline{Dly}(t) \succeq \alpha$ **then**
    $\overline{Dly^n}(t) = Func_1(\overline{Dly}(t))$
    Situation = 'Attack'
  end if
  if $Jrt(t) \succeq \beta$ **then**
    $Jrt^n(t) = Func_2(Jrt(t))$
    Situation = 'Attack'
  end if
  if $Drp(t) \succeq \lambda$ **then**
    $Drp^n(t) = Func_3(Drp(t))$
    Situation = 'Attack'
  end if
  if Situation == 'Attack' **then**
    Send Alarm and $(\overline{Dly^n}(t), Jrt^n(t), Drp^n(t))$ to the Third Phase
  else
    Send Attack-free to the Third Phase % To stop the policy enforcement.
  end if
end while
End

$(\overline{Dly}^n(t') + Jtr^n(t') + Drp^n(t'))$ must be 1, hence under the same conditions, the maximum possible value for each of them can be 1/3.

To enhance the defense mechanism in detecting the actual attackers, we assess the behavior of suspicious users after policy enforcement. After enforcing policy, their behavior reveals (with high certainty) the real identity of users. According to the TCP mechanism, it is expected from a benign user when the packets of her flow are dropped (for whatever reason such as congestion) and does not receive any acknowledgment, the user has to retransmit the sent packets using a lower bit-rate. Also for applications using the UDP protocol, when the flow becomes monologue and the user does not receive any packet from the server, they stop sending traffic. However, a bot, regardless of dropping its packets, not only continues sending with the same rate, but also attempts to establish new flows or reconnect to resume its attack on the server (victim). After policy enforcement, the user behavior varies based on the enforced actions. Table II illustrates the details of policy actions and user reactions for different DDoS attacks scenario that affect users' behavioral parameters. After receiving an attack alarm from the second phase and enforcing policy on suspicious users in the third phase, if a blocked user does not reduce her rate or tries to establish new connections, she will be identified as a bot. Otherwise, if the number of packets received from the blocked user decreases or is being sent in current flows with less rate, the user is mistakenly considered as an attacker and the enforced policy should be lifted.

In this phase, we use a time interval of 15 seconds, (T_3 = 15sec) and every 15 seconds, we assess the behavior of users after enforcing the relevant policy. The value of (T_3) should be adjusted precisely, as excessive value for this time interval leads to more penalties for mistaken detection of benign users, and underestimating the value sacrifices the accuracy of the mechanism to detect benign and bot users.

Algorithm 3 The Third Phase Algorithm.

Input: $Users\_Ranked\_List$, Attack Situations Alarm $(\overline{Dly^n}(t), Jrt^n(t), Drp^n(t))$

Output: Policy Enforcement Rules

Begin
$Load(0) = 0$
while True **do**
  if Situation=="attack" **then**
    Receive $Users\_Ranked\_Lists$ from the First Phase
    $Load(t) = \theta * A_8 + (1-\theta) * Load(t-1)$
    Generate Appropriate Rules for $Load(t)\%$ from $Users\_Ranked\_Lists$
  end if
  After $T_3(t)$ sec of blocking suspicious users
  for i = 1 to Len($Load(t)\%$ from $Users\_Ranked\_Lists$) **do**
    if Deacreased_rate(User[i]) == False **then**
      Continue
    else
      Remove User[i] Policy Enforcement Rules
    end if
  end for
end while
End

IV. Experimental Evaluation

In this section, we demonstrate our evaluation results according to the parameters explained above. We first evaluate RAD through CICDDoS2019 and compare it with four well-known DDoS detection algorithms. Then we extend our analysis in order to show the effect of the number of classes on the detection metrics.

We select such values for the time intervals that have proven appropriate and have been used in the literature [34]. For example, the best classification point has been reached, where the detection time interval was 60 seconds [28]. It was also explained that the reason behind selecting a smaller value

TABLE II

POLICY ACTIONS AND USER REACTIONS FOR DIFFERENT DDOS ATTACKS SCENARIO; DEC STANDS FOR THE USER DECREASES TRAFFIC RATE AFTER THE POLICY ENFORCEMENT, INC STANDS FOR THE OPPOSITE MEANING, AND N STANDS FOR NO CHANGE IN THE TRAFFIC RATE

protocols	*User Behavior Parame-ter*	*User Type*	*Action Policy*	*User Behavior after Policy Enforcement*	*Description*
TCP	*FpS*	*Benign*	*TCP RESET*	*DEC*	*When the benign user receives TCP RESET, she reduces her traffic rate.*
		Benign	*FIN*	*DEC*	*When the benign user receives FIN, she reduces her traffic rate.*
		Benign	*DROP*	*N*	*No change in the traffic rate happens.*
		Attacker	*TCP RESET*	*N/INC*	*When the attacker receives TCP RESET, she does not reduce her traffic rate (reconnect prior session or create a new connection).*
		Attacker	*FIN*	*N/INC*	*When the attacker receives FIN, she does not reduces her traffic rate (reconnect prior session or create a new connection).*
		Attacker	*DROP*	*N/INC*	*When the attacker traffic drops, she does not reduces her traffic rate (reconnect prior session or create a new connection).*
	PpS	*Benign*	*TCP RESET*	*DEC*	*When the benign user receives TCP RESET, she reduces her packet sending rate.*
		Benign	*FIN*	*DEC*	*When the benign user receives FIN, she reduces her packet sending rate.*
		Benign	*DROP*	*N/INC*	*No change in the traffic rate happens.*
		Attacker	*TCP RESET*	*N/INC*	*When the attacker receives TCP RESET, she does not reduce her packet sending rate (retransmit prior packets).*
		Attacker	*FIN*	*N/INC*	*When the attacker receives FIN, she does not reduces her packet sending rate (retransmit prior packets).*
		Attacker	*DROP*	*N/INC*	*When the attacker traffic drops, she does not reduces her packet sending rate (retransmit prior packets).*
	UGT	*Benign*	*TCP RESET*	*DEC*	*The benign user stops working after receiving the TCP RESET.*
		Benign	*FIN*	*DEC*	*The benign user stops working after receiving the FIN.*
		Benign	*DROP*	*DEC*	*The benign user stops working after not receiving the response.*
		Attacker	*TCP RESET*	*N/INC*	*Since the attacker does not wait for the response, she does not change her behavior. She usually tries to disrupt the server by generating massive traffic and making changes WINSIZE metric.*
		Attacker	*FIN*	*N/INC*	*Since the attacker does not wait for the response, she does not change her behavior. She usually tries to disrupt the server by generating massive traffic and making changes WINSIZE metric.*
		Attacker	*DROP*	*N/INC*	*Since the attacker does not wait for the response, she does not change her behavior. She usually tries to disrupt the server by generating massive traffic and making changes WINSIZE metric.*
	CC	*Benign*	*TCP RESET*	*DEC*	*When the benign user receives TCP RESET, she reduces her traffic rate, hence her CC parameters decreases.*
		Benign	*FIN*	*DEC*	*When the benign user receives FIN, she reduces her traffic rate, hence her CC parameters decreases.*
		Benign	*DROP*	*N*	*No change in the traffic rate happens*
		Attacker	*TCP RESET*	*N/INC*	*The attacker holds previous sessions regardless of the server status.*
		Attacker	*FIN*	*N/INC*	*The attacker holds previous sessions regardless of the server status.*
		Attacker	*DROP*	*N/INC*	*The attacker holds previous sessions regardless of the server status.*
UDP	*Policy enforcement verification is not possible for traffic that uses UDP protocol. It may be considered for applications that use UDP protocol, when the flow becomes monologue and the user does not receive any packet from the server, they stop sending traffic. It should be noted that 80% of attack traffic is related to TCP [33]. This aspect is out of scope for our work despite being an interesting research direction.*				

for (T_3), compared to others is to decrease the negative effect of probable false detection of users. In order to enhance intelligibility, the timely interactions between the three phases of our mechanism are illustrated in Fig. 2. The aspects of how the different time intervals influence the performance of the mechanism, open up some avenues for future work such as improving the complexity of the solution and affecting the detection metrics. However, this is beyond the scope of our work.

We have simulated our mechanism and have compared its performance metrics with existing models in [35]. The simulation programs have been written in python. The test environment was run on a PC that had 8 GB RAM and Intel Core i5-3317U 1.7GHz CPU with 4 logical processors.

A. The Evaluation of RAD Through CICDDoS2019

To evaluate *RAD* performance, we have used CICDDoS2019 dataset which contains benign users traffic and the most up-to-date common DDoS attacks. Authors in [35] used the B-Profile system to create the profile of user interactions behavior and generate naturalistic benign background traffic in the proposed testbed. For this dataset, they built the abstract behavior of 25 users based on HTTP, HTTPS, FTP, SSH, and email protocols. CICDDoS2019 contains a true real-world PCAP and metadata (CSV) files. CSV files include the results of the network traffic analysis using CICFlowMeter-V3; flows are labeled based on their timestamp, source IPs, destination IPs, source ports, destination ports, and protocols. Four common machine learning algorithms namely ID3, Random Forest (RF), Naïve Bayes, and logistic regression have been used in order to detect DDoS attacks.

The ID3 uses the entropy concept in order to find the best attributes, create the decision tree, and split the dataset recursively. Based on the result in [35], ID3 took a few minutes to be trained and classify the testing set. Random Forest combines decision tree and ensemble learning. In this machine learning algorithm, the forest has a collection of trees with controlled variance (using the low number of parameters and resistance to over-fitting) and the result of classification

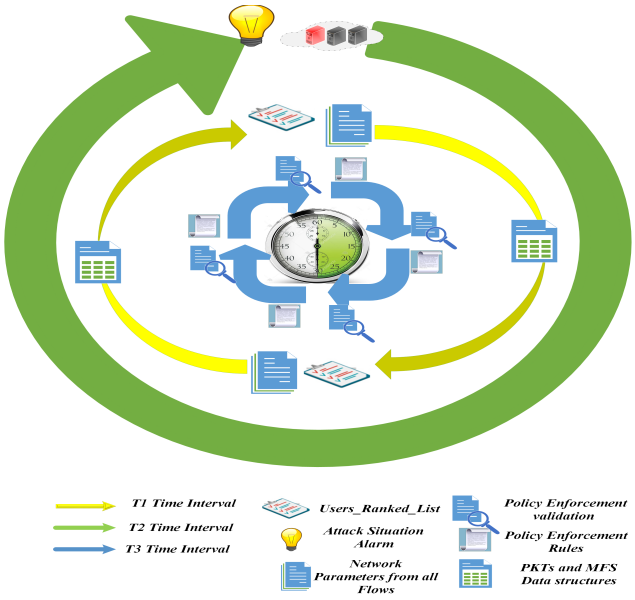


Fig. 2. Timely interactions between the three phases in RAD.

can be decided by majority voting or weighted voting. The result in [35] shows that a Random Forest with 100 trees took more than 15 hours for the same process. Naïve Bayes is an online algorithm and a probabilistic classifier. It is based on Bayes Theorem with strong independence assumptions between features (not a true assumption in many problems). Its training can be completed in linear time. Multinomial logistic regression is a predictive analysis like other regression analyses and can describe data and explain the relationship between features and classes. It took more than 2 days to be trained and classify the testing set. Authors in [35] used five-fold cross-validation for their experiments. We have used the results in [35] directly to compare with the results of RAD.

While using CICDDoS2019, we consider True Positive (TP) as the user traffic flows correctly detected for attacker traffic; False Positive (FP) as the normal user traffic flows incorrectly detected for attacker traffic; False Negative (FN) as attacker traffic flows incorrectly detected for normal user traffic at the first phase or being enforced no policy at the third phase.

Moreover, the following metrics are utilized for performance measurement:

- Precision (Pr): Pr is the fraction of flows correctly classified as attack flows (TP) corresponds to all the flows classified as attack flows ($TP + FP$).

$$Precision = \frac{TP}{(TP + FP)}$$

- Recall (Rc): Rc is the fraction of flows correctly classified as attack flows (TP), corresponds to all the actual attack flows, regardless of being detected ($TP + FN$).

$$Recall = \frac{TP}{(TP + FN)}$$

- F-Measure (F_1): F_1 combines precision and recall into a single measure.

$$F_1 - Measure = \frac{(2 * Precision * Recall)}{(Precision + Recall)}.$$

User ranking is done in the first phase based on the combination of FpS, PpS, UGT, and CC. Fig. 3a depicts the transfer ratio distribution over time. On the other hand, the ratio of packets transferred over time is illustrated in Fig. 3b. Similar graphs are shown in Fig. 3c and Fig. 3d for FpS and CC. The spikes in the graphs indicate the potential of attack traffic that affects the above parameters.

For the second phase parameters, $Func_1(.)$, $Func_2(.)$, $Func_3(.)$, and also the value of the processing parameters thresholds (α, β, and λ) are fine-tuned as illustrated below:

$$Func_1(\overline{Dly}(t)) = \begin{cases} 0: & \overline{Dly}(t) \leq 15ms \\ 1/3: & \text{otherwise} \end{cases}$$

$$Func_2(Jrt(t)) = \begin{cases} 0: & Jrt(t) \leq 10 \\ 1/3: & \text{otherwise} \end{cases}$$

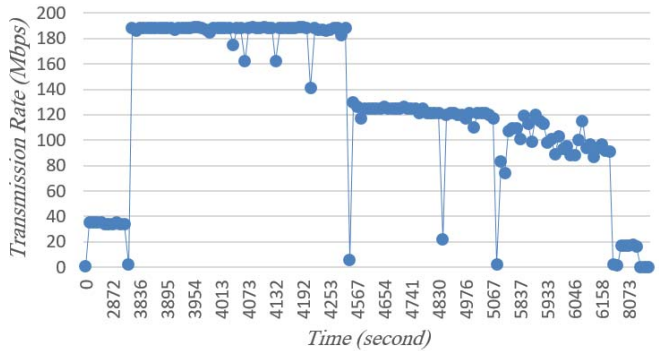
$$Func_3(Drp(t)) = \begin{cases} 0: & Drp(t) = 0 \\ 1/3: & \text{otherwise} \end{cases}$$

Since the detection of abnormal situations in RAD is based on the target availability and on the other hand there is no actual server for all the attacks existing in CICDDoS2019 in our simulation scenario, the attack status notification in the second phase is considered using some HTTP requests and their response time. From this data, we determine the load status at the server-side. The HTTP protocol includes status codes that we can use for a particular service that is still available [33]. For this purpose, we use the Http-Ping tool [36] to measure the delay values continuously. According to the capacity of the system in the simulation environment and the results of the Http-Ping tool in an attack-free period, the delay, jitter, and drop threshold values in this simulation are considered equal to $\alpha = 15$ms, $\beta = 10$, and $\lambda = 0$. Due to the number of users in this dataset, the value of the θ in the third phase is considered 0.2 in our simulation.

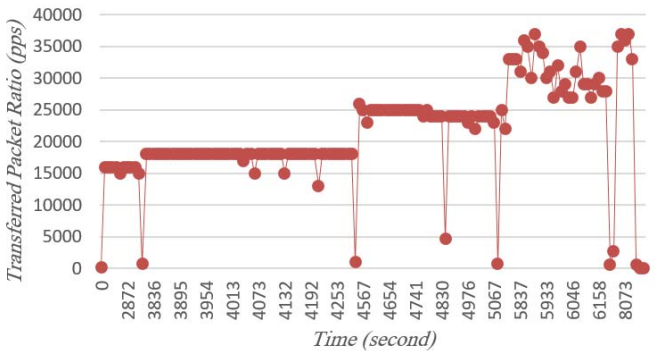
Fig. 4 presents the detection performance of the four mentioned mechanisms v.s. RAD in two modes (two phases and three phases). As demonstrated, RAD has better values for Rc and $F_1 - Measure$ and the third highest in terms of Pr even when the third phase is relaxed compared to the other mechanisms. Having the third phase in RAD results in a higher Pr as well.

In RAD, counteracting the attack is done at the user level (instead of flow/packet level). Having a user identified as an attacker, all of her traffic will be enforced against appropriate policy. The coverage of all the attacker traffic in policy enforcement results in a higher TP, lower FN, and eventually, a higher Rc metric. On the other hand, if a legitimate user is mistakenly marked as an attacker, policy enforcing on all her traffic leads to higher FP and so lower Pr.

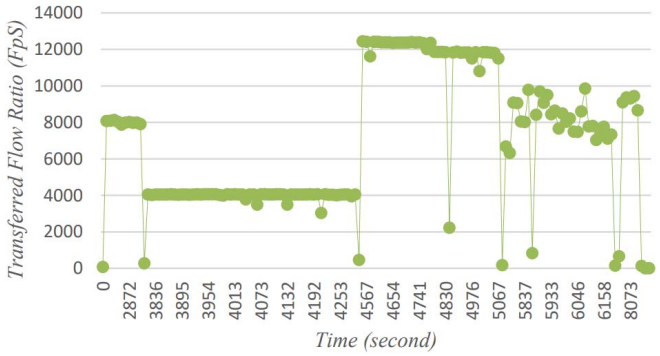
Most of the FP detection in RAD has been originated from the TCP traffic in CICDDoS2019. By re-examining user behavior and also considering different TCP back-off



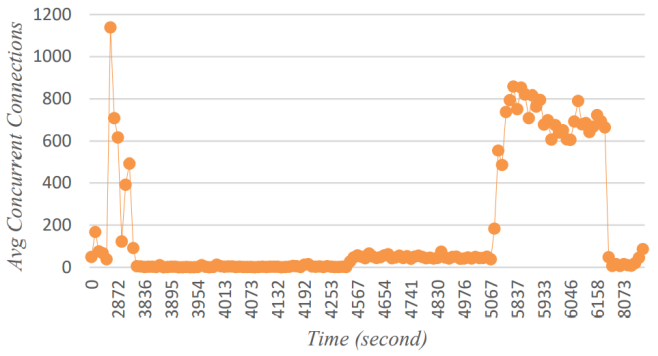
(a) The ratio of traffic volume transferred.



(b) The ratio of packet transferred.



(c) The ratio of flow transferred.



(d) The number of concurrent connections transferred.

Fig. 3. An overview of the status of the parameters in the dataset.

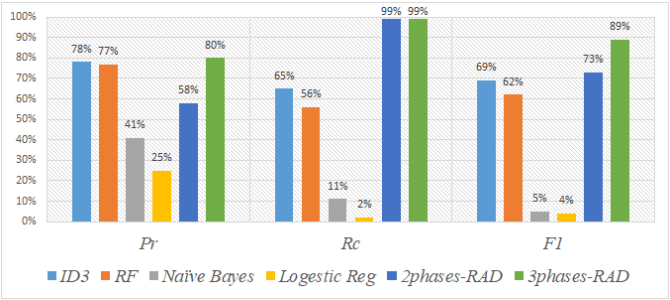


Fig. 4. Comparison of detection performance with 2Phases-RAD and 3phases-RAD.

mechanisms in the third phase leads to the reduction of 1/3 flows mistakenly detected as attack flows. As a result, FP is reduced and Pr is increased up to 80%, as illustrated in Fig. 4.

B. The Effects of the Number of Classes on the Detection Metrics

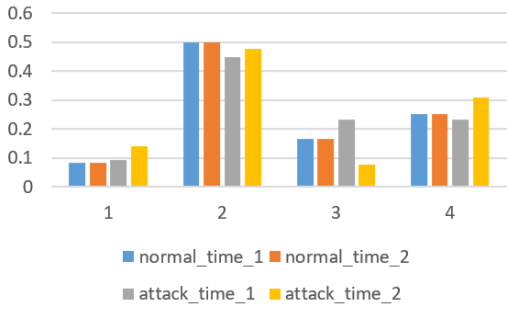
One of the influential parameters that affect detection metrics in our proposed solution is the number of user classes. To show the effect of the number of classes on the detection metrics and normal and attack traffic on calculations of the first phase, we have designed a scenario on a local network. In this scenario, we have 192 normal users in 12 different IP ranges. All the users will send their requests to the server at an almost monotonous rate. In our mathematical operation, PpS and FpS have been calculated in two different snapshots of normal time and two different snapshots of attack time. The result of our observations is being represented on four different charts shown in Fig. 5 and Fig. 6. Blue and orange columns in these diagrams show the percent of packets in the first and second snapshot of normal time respectively.

As you can see, percentages of PpS and FpS are very similar in both snapshots of normal time. Although a quite difference can be seen when we have more ranges. If the value of a column equals zero, it will mean that there has been no user with the specified amount of PpS or FpS.

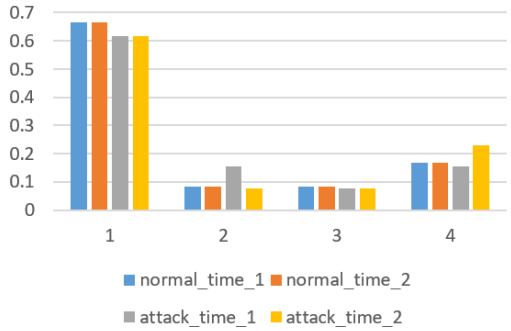
In the next step, a TCP flood attack has been created with 16 attackers. The percentage of PpS and FpS has been gathered and all results have been recalculated again. Yellow and gray columns on charts show four snapshots of all the calculations on FpS and PpS with two different snapshots of attack. In the attack snapshots, all the computations have resulted from the effect of benign users beside attackers. The fluctuation of results after the occurrence of the attack is clear in our plots.

Our plots show that the number of ranges impacts directly on the final result of our scores. The first influence of the number of ranges is on the intensity of changes in scores. For example, in Fig. 5a and Fig. 5b where we have four ranges, columns of normal snapshots are equal. But in 10 ranges as shown in Fig. 6a and Fig. 6b, there are some small differences even in the normal time results. Although these differences are tiny, if our system is very sensitive against any changes and detects any simple modification as a DDoS attack, these conversions will be dangerous. By way of explanation, more ranges would have a negative effect on FP in our mechanism.

On the other hand, when we have fewer ranges, any change in the ranges would happen more slowly and the calculation of every column is more stable. For example, in the column of the

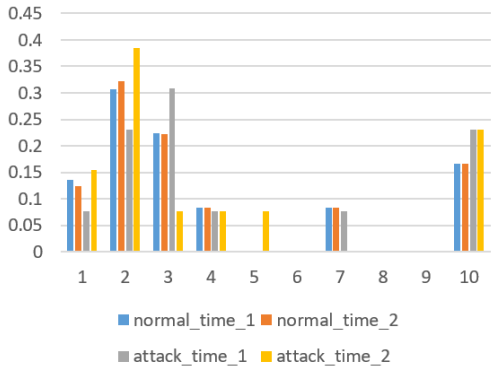


(a) Percentage of FpS for 4 specified ranges.

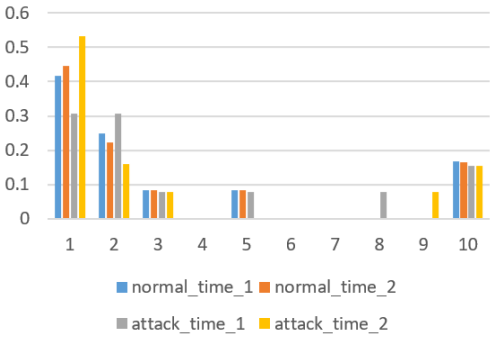


(b) Percentage of PpS for 4 specified ranges.

Fig. 5. Percentage of FpS and PpS for 4 specified ranges in four different time snapshots.



(a) Percentage of FpS for 10 specified ranges.



(b) Percentage of PpS for 10 specified ranges.

Fig. 6. Percentage of FpS and PpS for 10 specified ranges in four different time snapshots.

first snapshot of the attack in Fig. 5a, we can see that only the third range has perceptible alteration and the changes of other ranges, especially the first and fourth columns, are not very sensible. This will increase the probability of false negative in our mechanism. In the attack time, there could be either benign users or attackers in the same class. So if we see an anomaly in a class and detect these users as attackers, benign users in that class will be treated as attackers. In this case, if there would be policy enforcement against attackers, benign users will have behaved like attackers and, the availability of some of the benign users will be denied. The more ranges we use, the fewer users in every class would be. Hence if we enforce a range, fewer benign users will be affected. In other words, having more ranges will reduce FP of policy enforcement for our users and it would be less probable for a benign user to be treated as an attacker.

V. Conclusions and Future Work

In this work, RAD, a statistical mechanism, is proposed to detect and mitigate DDoS attacks using a 3-phase defense mechanism. The first phase ranks users based on behavior parameters and is implemented in the middle-box devices such as firewalls. In the second phase, at the server-side, some processing parameters such as dropped packets are monitored and when these parameters exceed a specific threshold, an alarm is sent to the third phase. In the third phase, after receiving the user ranked list from the first phase and an abnormal state alarm from the second phase, middle-boxes block the already identified suspicious users using relevant policy. Through continuous assessment of the blocked users behavior, the third phase in the middle-box tries to validate the mechanism of attacker detection. Our evaluation shows RAD outperforms counter the generated attacks in CICDDoS2019 with a precision of 58% if we use only the first and the second phases. However, if the third phase is used, this metric will reach 80%. The recall metric in both evaluations was about 99%, indicating that RAD can detect most of the attackers flows. Accordingly, we conclude that, having a cooperative defense mechanism being deployed in different locations and correlating their events not only improves the accuracy of detecting attacks but also increases the coverage variety of DDoS attacks.

This research opens up three avenues for future work. Firstly, using different machine learning techniques and creating profiles of different users' behavior can be more accurate. This can also help to detect suspicious users with more accuracy, by extracting better features and specifying more realistic thresholds for clustering. Secondly, DPI systems can be employed to identify the granularity of each application or protocol per user and better characterize abnormal behavior based on application or protocol. In other words, we can specify application fingerprints per user. This will further increase accuracy and decrease false positives. Thirdly, RAD can be deployed in Software Defined Networking (SDN) to utilize the concept of the control plane for the first and second phases and update rules dynamically in the data plane for the third phase.

ACKNOWLEDGMENT

The authors would like to express their very great appreciation to Ahmad Jalali for his valuable discussions. They also would like to offer their special thanks to Dr. Mohsen Rohani and Amir Mahdi Sadeghzadeh for their valuable reviews and feedbacks.

REFERENCES

[1] *The Kaspersky Quarterly Ddos Attack Report*. Accessed: Jun. 15, 2021. [Online]. Available: https://usa.kaspersky.com/about/press-releases
[2] S. Cook. (2021). *DDoS Attack Statistics and Facts for 2018–2021*. [Online]. Available: https://www.comparitech.com/blog/information-security/ddos-statistics-f%acts/
[3] R. Biswas, S. Kim, and J. Wu, "Sampling rate distribution for flow monitoring and DDoS detection in datacenter," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 2524–2534, 2021.
[4] K. Singh, P. Singh, and K. Kumar, "User behavior analytics-based classification of application layer HTTP-GET flood attacks," *J. Netw. Comput. Appl.*, vol. 112, pp. 97–114, Jun. 2018.
[5] M. D. Mauro, G. Galatro, and A. Liotta, "Experimental review of neural-based approaches for network intrusion management," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 4, pp. 2480–2495, Dec. 2020.
[6] Ö. Kasim, "An efficient and robust deep learning based network anomaly detection against distributed denial of service attacks," *Comput. Netw.*, vol. 180, Oct. 2020, Art. no. 107390.
[7] G. Somani, M. S. Gaur, D. Sanghi, M. Conti, and R. Buyya, "DDoS attacks in cloud computing: Issues, taxonomy, and future directions," *Comput. Commun.*, vol. 107, pp. 30–48, Jul. 2017.
[8] A. Praseed and P. S. Thilagam, "DDoS attacks at the application layer: Challenges and research perspectives for safeguarding web applications," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 661–685, 1st Quart., 2019.
[9] M. E. Ahmed, S. Ullah, and H. Kim, "Statistical application fingerprinting for DDoS attack mitigation," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 6, pp. 1471–1484, Jun. 2019.
[10] A. Wang, W. Chang, S. Chen, and A. Mohaisen, "Delving into internet DDoS attacks by botnets: Characterization and analysis," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2843–2855, Dec. 2018.
[11] W. Chang, A. Mohaisen, A. Wang, and S. Chen, "Measuring botnets in the wild: Some new trends," in *Proc. 10th ACM Symp. Inf., Comput. Commun. Secur.*, Apr. 2015, pp. 645–650.
[12] A. Wang, A. Mohaisen, W. Chang, and S. Chen, "Measuring and analyzing trends in recent distributed denial of service attacks," in *Proc. Int. Workshop Inf. Secur. Appl.*, Cham, Switzerland: Springer, 2016, pp. 15–28.
[13] J. David and C. Thomas, "DDoS attack detection using fast entropy approach on flow-based network traffic," *Proc.-Proc. Comput. Sci.*, vol. 50, pp. 30–36, Dec. 2015.
[14] X. Qin, T. Xu, and C. Wang, "DDoS attack detection using flow entropy and clustering technique," in *Proc. 11th Int. Conf. Comput. Intell. Secur. (CIS)*, Dec. 2015, pp. 412–415.
[15] K. Kalkan, L. Altay, G. Gür, and F. Alagöz, "JESS: Joint entropy-based DDoS defense scheme in SDN," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2358–2372, Oct. 2018.
[16] K. Kalkan, G. Gur, and F. Alagoz, "SDNScore: A statistical defense mechanism against DDoS attacks in SDN environment," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2017, pp. 669–675.
[17] K. Kalkan and F. Alagöz, "A distributed filtering mechanism against DDoS attacks: ScoreForCore," *Comput. Netw.*, vol. 108, pp. 199–209, Oct. 2016.
[18] M. Cirillo, M. D. Mauro, V. Matta, and M. Tambasco, "Botnet identification in DDoS attacks with multiple emulation dictionaries," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 3554–3569, 2021.
[19] E. Aminanto and K. Kim, "Deep learning in intrusion detection system: An overview," in *Proc. Int. Res. Conf. Eng. Technol. (IRCET)*, 2016, pp. 1–12.
[20] Y. Imamverdiyev and F. Abdullayeva, "Deep learning method for denial of service attack detection based on restricted Boltzmann machine," *Big Data*, vol. 6, no. 2, pp. 159–169, 2018.
[21] K. Kim and M. E. Aminanto, "Deep learning in intrusion detection perspective: Overview and further challenges," in *Proc. Int. Workshop Big Data Inf. Secur. (IWBIS)*, Sep. 2017, pp. 5–10.
[22] K. P. Reddy, S. Kodati, M. Swetha, M. Parimala, and S. Velliangiri, "A hybrid neural network architecture for early detection of DDOS attacks using deep learning models," in *Proc. 2nd Int. Conf. Smart Electron. Commun. (ICOSEC)*, Oct. 2021, pp. 323–327.
[23] Y. Wei, J. Jang-Jaccard, F. Sabrina, A. Singh, W. Xu, and S. Camtepe, "AE-MLP: A hybrid deep learning approach for DDoS detection and classification," *IEEE Access*, vol. 9, pp. 146810–146821, 2021.
[24] A. Procopiou, N. Komninos, and C. Douligeris, "ForChaos: Real time application DDoS detection using forecasting and chaos theory in smart home IoT network," *Wireless Commun. Mobile Comput.*, vol. 2019, pp. 1–14, Feb. 2019.
[25] P. Redekar and M. Chatterjee, "Hybrid technique for DDoS attack detection," *Int. J. Comput. Sci. Inf. Technol.*, vol. 8, pp. 377–379, Jan. 2017.
[26] 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.
[27] S. Khan, A. Gani, A. W. A. Wahab, and P. K. Singh, "Feature selection of denial-of-service attacks using entropy and granular computing," *Arabian J. Sci. Eng.*, vol. 43, no. 2, pp. 499–508, Feb. 2018.
[28] X. Ma and Y. Chen, "DDoS detection method based on chaos analysis of network traffic entropy," *IEEE Commun. Lett.*, vol. 18, no. 1, pp. 114–117, Jan. 2014.
[29] K. Singh, P. Singh, and K. Kumar, "Application layer HTTP-GET flood DDoS attacks: Research landscape and challenges," *Comput. Secur.*, vol. 65, pp. 344–372, Mar. 2017.
[30] M. Noferesti and R. Jalili, "ACoPE: An adaptive semi-supervised learning approach for complex-policy enforcement in high-bandwidth networks," *Comput. Netw.*, vol. 166, Jan. 2020, Art. no. 106943.
[31] *NGFW Test Methodology*. Accessed: Dec. 3, 2018. [Online]. Available: https://research.nss-labs.com/library/methodologies/1097ngfw-test-methodology%-v9.0/ and https://www.tecnozero.com/wp-content/uploads/2019/08/nss-labs-test-methodology-v9.0.pdf
[32] Y. Kim, W. C. Lau, M. C. Chuah, and H. J. Chao, "PacketScore: A statistics-based packet filtering scheme against distributed denial-of-service attacks," *IEEE Trans. Depend. Secure Comput.*, vol. 3, no. 2, pp. 141–155, Apr. 2006.
[33] T. Lukaseder, "Security in high-bandwidth networks," Ph.D. dissertation, Open Access Repository Univ. Ulm, Ulm, Germany, 2020, doi: 10.18725/OPARU-33154.
[34] A. Koay, A. Chen, I. Welch, and W. K. G. Seah, "A new multi classifier system using entropy-based features in DDoS attack detection," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2018, pp. 162–167.
[35] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," in *Proc. Int. Carnahan Conf. Secur. Technol. (ICCST)*, Oct. 2019, pp. 1–8.
[36] *A Small Http Requests Tool*. Accessed: Jun. 15, 2022. [Online]. Available: https://github.com/Killeroo/HttpPing/

Mosayeb Hajimaghsoodi received the B.Sc. degree in software engineering from Shahid Beheshti University, Tehran, Iran, in 2012, and the M.Sc. degree in computer engineering from the Amirkabir University of Technology in 2014. He is currently pursuing the Ph.D. degree with the Department of Computer Engineering, Sharif University of Technology. His research interests include security in high-bandwidth networks, intrusion detection systems, DDoS attacks, and security in software defined networks.

Rasool Jalili received the B.Sc. degree in computer science from the Ferdowsi University of Mashhad in 1985, the M.Sc. degree in computer engineering from the Sharif University of Technology, Tehran, Iran, in 1989, and the Ph.D. degree in computer science from The University of Sydney, Australia, in 1995. Then, he joined the Department of Computer Engineering, Sharif University of Technology, in 1995. He has published more than 150 papers in computer security and pervasive computing in international journals and conferences proceedings. He is currently an Associate Professor with the Sharif University of Technology. His research interests include access control, vulnerability analysis, and database security, which he conducts at his Data Network Security Laboratory (DNSL, dnsl.sharif.ir).