Computer Networks 108 (2016) 199–209

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

A distributed filtering mechanism against DDoS attacks: ScoreForCore

Kübra Kalkan*, Fatih Alagöz

Computer Engineering Department, Bogazici University, Istanbul, Turkey

ARTICLE INFO

Article history:
Received 1 December 2015
Revised 23 August 2016
Accepted 24 August 2016
Available online 25 August 2016

Keywords:
DDoS
DoS
Filter
Internet security
Intrusion detection and prevention systems
Traffic filtering

ABSTRACT

Traffic filtering is an essential technique that is used as a prevention mechanism against network attacks. This paper presents a proactive and collaborative filtering based defense mechanism against Distributed Denial of Service (DDoS) attacks. Proactivity provides prevention of attacks before it spreads whereas collaboration enables getting knowledge about different points of the network and deciding filters together. The proposed model called ScoreForCore is a statistical mechanism that is inspired from another proactive but individual model. The most distinctive property of our model is the selection of the most appropriate attributes during current attack traffic. We compared our results with the existing model. Our results suggest that the success of system's behavior on legal and attack packets are increased considerably. In addition, most of the attack packets are stonewalled near the source of the attack.

1. Introduction

Distributed Denial of Service (DDoS) attacks are generated by flooding a system from several machines. Its aim is to occupy all available services and prevent innocent users accessing resources. Since attackers are also acting as innocent users and attack packets do not have any malicious part, it is not easy to tackle with this problem. In order to avoid these attacks, several defense mechanisms are proposed. These mechanisms broadly include the idea of detection, prevention and countermeasures.

Traffic filtering is a method that is widely utilized as a prevention mechanism. A filter is essentially a rule which permits or prevents a packet to enter the system [1]. They are generally installed on routers since they allow or block packets before they enter a domain. These mechanisms are vital since they intercept an attack posed to give harm to a large number of machines. In our previous work, we classified filtering mechanisms according to their feature of collaboration and response time.

- **Collaboration based classification**: In some circumstances, machines or nodes need to cooperate in order to learn and decide about filters. This type of filtering is called *cooperative filtering* whereas others are called *individual filtering*.
- **Response time based classification**: Filtering mechanisms can also be classified according to the point-in-time of reaction. Filtering defense mechanisms can be active before or after DDoS attack starts. From the filtering perspective, we can have proactive and reactive mechanisms.

According to our analysis, several mechanisms have been proposed which are reactive + cooperative, reactive + individual and proactive + individual. However, according to the best of our knowledge, there are a few works that are both cooperative and proactive in the literature. Proactive and cooperative filtering provides preventive and collaborative mechanisms. If it is possible to deploy this mechanism through the network, it gives an opportunity to block a DDoS attack near the source before it expands. Also, it is inherently more accurate than individual mechanisms since it decides on filters with more visibility and knowledge about the network. However, these scenarios are more challenging since cooperation should be accepted by all peers while no attack is active yet. Therefore, we think that this topic deserves more research. Thus, in this work, we propose a cooperative and proactive model called ScoreforCore that is inspired by PacketScore [2] mechanism.

PacketScore [2] is a statistical filtering mechanism wherein each packet is analyzed according to its attribute values and then scores are calculated according to them. A packet is announced as *legitimate* if its score value is under a dynamic threshold when they are compared with a baseline profile. This baseline profile is generated based on Bayesian Theorem [3]. In previous works, this type of comparison was applied in detection of DDoS attacks, however PacketScore is the first method that utilizes this comparison for real-time packet filtering against such attacks. It is an individual filtering mechanism since it performs analysis and determines filters on its own. Also, it is proactive since it blocks packets according to a scoring approach that is always active. This filtering

* Corresponding author.
E-mail addresses: kubra.kalkan@boun.edu.tr, kubrakalkan@sabanciuniv.edu (K. Kalkan), fatih.alagoz@boun.edu.tr (F. Alagöz).

http://dx.doi.org/10.1016/j.comnet.2016.08.023
1389-1286/

mechanism can differentiate legitimate and attack packets via statistical analysis. Therefore, it can deal with new DDoS attack types. Moreover, it works well for non-spoofed attacks since it does not solely utilize source address attribute, but also other attributes helping to find attack packets. However, it has performance drawbacks. They suggest that when the number of attributes enrolled in scoring increases, their model's ability of filtering increases. But unfortunately, increase in scoring attributes results in huge tables. Actually, the main problem of this model is the lack of appropriate attribute selection for each attack type. In most of the attacks, attack packets have some common properties. For this reason, each type of attack can be detected with specific attributes. Since PacketScore does not have attribute selection property in profile generation, it generates profiles with predetermined fixed attributes. When they increase the number of attributes, probability of enrollment of these attributes in determining several types of attacks increase. However, when the number of attributes increases, profile size increases enormously and it needs huge amount of memory. In order to solve this problem, our model ScoreForCore provides appropriate attribute selection property for current attack traffic. This property is provided by collaboration in ScoreForCore. In our model, PacketScore's advantages are utilized whereas drawbacks are omitted. In order to show these improvements, we evaluate both PacketScore and ScoreForCore.

The structure of this paper is as follows. Section 2 discusses related work whereas Section 3 presents an overview of ScoreForCore including motivation and design details. Section 4 describes our simulation and dataset details. Section 5 demonstrates performance evaluation and Section 6 provides discussion. Finally, Section 7 concludes the paper.

2. Related work

As it is mentioned in Section 3, filtering based mechanisms against DDoS attacks can be classified according to their response time and collaboration properties. There are several works which are individual and proactive. The proposed mechanism in [4] is an individual statistical model that utilizes Statistical Segregation Method (SSM). This mechanism samples the flows in an attack free period, then compares these samples with the ones in an attack period. They also sort them according to the mean values. Then, attack flows are segregated from legitimate ones. Another model, PacketScore scheme [2] is a statistical filtering mechanism wherein each packet is analyzed according to its attribute values and scores are calculated according to these values. A packet is announced as legitimate if its value is above a dynamic threshold. Dynamic threshold is determined according to a baseline profile that is generated by considering Bayesian Theorem [3]. Another model, confidence based filtering [5] is also an individual proactive filtering mechanism similar to PacketScore. They collect legitimate packets in a non attack period to generate attribute pair nominal profiles. They calculate confidence values for packets in order to decide its legitimacy. They utilized some fixed thresholds to use less memory. This model considers all combinations of pair attributes for confidence calculation. However, since they do not choose appropriate attribute selection for current attack traffic, irrelevant pairs decrease accuracy.

A number of significant approaches have already been proposed for reactive+ cooperative filtering mechanisms. In [6], Mahajan et al. propose a scheme called Pushback which rate-limits the aggregated traffic from congested router to upstream counterparts. In this scheme, congestion is detected locally at router level. The congestion signature is determined and translated to a router filter. According to the amount of congestion, an appropriate rate limit is determined locally. Then, the congested router asks upstream routers to rate-limit the traffic. This Pushback operation is propagated through the upstream routers. This is the first scheme that proposes collaborative strategy against DDoS attacks. This scheme is effective if the attacker traffic traverses a different path from legitimate traffic. Besides, it does not block all traffic from an attacker. It only limits attack traffic to a fair share. Another model, Probabilistic Filter Scheduling (PFS) [1] defines a filter technique which deals with not only filtering but also its scheduling and how to find best locations for filters. A filter router probabilistically writes its own IP address to IP header of packets. Victim collects all markings and constructs marking values to request a filter. A filter router which receives several filter requests applies a filter scheduling policy and chooses best k filters. Then, it sends these filters to upstream routers. As soon as the attack stops, the scores of corresponding filters are decreased and these filters are revocated from filter routers. In addition, it requires a security agreement between routers in order to accept filters.

All these techniques are collaborative or proactive but not both. According to the best of our knowledge there are a few works that are both cooperative and proactive. Secure Overlay Services [7] is one of them. In this model, users' packets are firstly authenticated by Secure Overlay Access Points. Then they send them to the beacon nodes. These nodes are the only nodes that know the place and address of the Secure Servlets. Beacon nodes send packets to Secure Servlets and they forward packets to the destination. This model's weak aspect is the vulnerability that is arose from finding Secure Servlets. As far as cryptographic operations are not utilized in this model, it is easy to reach the IP address of Secure Servlets. If an attacker reaches Secure Servlets, DDoS attack packets can easily be sent to these routers directly. Another collaborative and proactive model is FireCol [8]. This mechanism proposes to form a virtual protection rings around the hosts. These virtual rings are determined according to the distances from IPSs in terms of hops. According to the level of threat, communication is triggered in horizontal or vertical ring levels. Their model tries to block the attack traffic as close as to the attack sources to protect the network.

3. Overview of ScoreForCore

ScoreForCore is a novel filtering mechanism that is preventive and collaborative. In the following subsections we review the underlying motivation and our design issues.

3.1. Motivation

In our previous work, we have analyzed and classified several filtering mechanisms. Our classification is based on reaction time and collaboration. A mechanism can be reactive or proactive in terms of reaction time where as it can be individual or cooperative in terms of collaboration feature. Individual reactive filtering is active after DDoS attack detection and it provides easy deployment. If we need a mechanism which should be active only a small amount of time, since we cannot tolerate extra burden and we cannot deploy it through the network, then this type will be preferable. Hop Count Filtering [9] provides such properties. On the other hand, cooperative reactive filtering provides collaborative decision after DDoS attack is detected. Pushback [6], Active Internet Traffic Filtering (AITF) [10], StopIt [11], Probabilistic Filter Scheduling (PFS) [1] and Adaptive Probabilistic Filter Scheduling (APFS) [12] are active after DDoS attack is detected and filters are propagated through communicated machines. Individual and proactive filtering provides easy deployment and quick intervention which interferes DDoS attacks before it impairs the network operation. However, since it is always active, it results in extra burden on performance. Ingress/Eggress Filtering [13], Route Based Packet Filtering (RDPF) [14], Source Address Validity Enforcement(SAVE) [15],

Table 1
System terms and parameters.

Term	Explanation
SingleNP	A single attribute profile in an attack free period
SingleCP	A single attribute current profile in an attack period
OwnPair	Random attribute pair owned by the router
OwnPairList	A list consists of *OwnPairs*
OPNP	Own pair nominal profile created by considering *OwnPair*
OPNPList	A list consists of *OPNPs*
ScorePair	The pair that is used for generating current pair profiles
ScorePCP	Current pair profile used for score operations
ScorePNP	Nominal pair profile used for score operations
SuspiciousPair	The attribute pair with the most probable signs for current attack
SPNP	Suspicious pair nominal profile created by considering *SuspiciousPair*
A, B	Attribute A and B
Dev_A	Maximum deviation for attribute A
$npcp_{an}$	The number of packets with $A = a_n$ in current profile
$npnp_{an}$	The number of packets with $A = a_n$ in nominal profile
dev_{an}	The deviation for attribute value $A = a_n$
$A = a_p$	Attribute A with a value in packet p
$B = b_p$	Attribute B with b value in packet p
$ScorePNP_{(A=a_p,B=b_p)}$	The number of packets in a nominal profile that have the property of a_p for attribute A and b_p for attribute B
$ScorePCP_{(A=a_p,B=b_p,\ldots)}$	The number of packets in a current profile that have the property of a_p for attribute A and b_p for attribute B
TPNP	The total number of packets in a nominal profile
TPCP	The total number of packets in a current profile
S_p	Score value of packet p
ScoreList	A list that contains scores of packets
Th	Threshold score value for packet discarding
ϕ	Acceptable traffic
ψ	Total current incoming traffic

PacketScore [2] are some examples of individual and proactive filtering. Cooperative proactive filtering provides collaborative and preventive mechanisms. It provides an ability of forestalling the attack packets before they expand. Also, it gives more accurate decisions since they create filters cooperatively with more knowledge of the network. Nevertheless, according to the best of our knowledge there are a few mechanisms that are both cooperative and proactive in the literature. It can be challenging since cooperation of all peers are needed while there is no attack. However, if it is possible to make it efficiently and increase the accuracy significantly, peers will cooperate willingly. For this reason, we think that this type of filtering should be analyzed in details and more models should be proposed. Our model ScoreForCore is a cooperative and proactive model that increases accuracy considerably.

3.2. Design issues

As we deal with distributed systems while we are proposing a cooperative filtering mechanism, we decided to work on an existing and promising individual+proactive model called PacketScore [2]. This model suggests a scoring method based on Bayesian Theorem [3]. Each packet's score is calculated by considering its attributes. If its score is less than a threshold it is regarded as attack packet and discarded, otherwise it is forwarded. In our model, we utilize the same scoring method. Nevertheless, in our model routers create profiles with the most appropriate attributes for current attack which results in much more accuracy. In that sense, we need to consider the following issues: profiling, comparison, collaboration, score calculation, threshold calculation and selective discarding. These issues are explained in the following subsections. The system parameters that are utilized in these sections are shown in Table 1.

3.2.1. Profiling

In ScoreForCore, our aim is to calculate a score for each packet and decide if it is legal or not by considering the score value. It is calculated by utilizing nominal profile and current profile of the traffic. The nominal profiles are created by counting the number of packets in a non attack period, whereas the current profiles are created during the attack period. While these profiles are generated, single and pair attributes are considered in order to keep storage requirement minimal. Following properties are considered as attributes: *IP address, port number, protocol type, packet size, TTL value* and *TCP flag*.

In order to have all combinations of single and pair attributes (for 6 attributes), each router needs to generate 6 single nominal profiles, 6 single current profiles and 15 pair nominal profiles and 15 pair current profiles. This will need too much storage. (Storage requirement analysis is provided in Section 5.4). Then, as far as our router cannot have all the attribute pairs, we need to choose some of the attributes in order to generate the profiles. In PacketScore [1] scheme, they chose predetermined attributes for score calculation. However, the predetermined attributes cannot always detect all the attacks. Because, the packets of different attack types have different properties and these properties can be determined by choosing the most appropriate attributes. For instance, TCP SYN Flood attack packets have the properties as ProtocolType = TCP and TCPFlag = 40, whereas DNS attack packets have properties as ProtocolType = DNS and destinationPort = 53. In order to detect TCP attack packets, we need (ProtocolType, TCPFlag) attribute pair whereas to detect DNS attack packets we need (ProtocolType, destinationPort). Thus, in our model we propose to choose the most appropriate pair for the current attack. As far as, it is a dynamic model, we cannot determine the attributes beforehand. Thus, in ScoreForCore each router chooses its attribute pair randomly and then during the attack, after the most appropriate pair is determined, the router tries to reach the profile of the determined attributes from its neighbors. Randomly chosen two attribute pairs named as *OwnPairs* to generate *OwnPairList*. At the end of this period, each router has single nominal profiles *SingleNP* for each attribute and its own pair nominal profiles *OPNPs* created considering *OwnPairList*. An example of a nominal profile *OPNP* for a pair of attributes is illustrated in Table 2.

In an attack period, when a congestion is detected, packet based analysis are activated. For each attribute, current number of packets is calculated and single current profiles *SingleCPs* are generated.

Table 2
Pair nominal profile (*OPNP*) example.

TTL value	Destination port number	Number of packets
48	25	125
48	53	175
48	80	19
50	80	42
...	...	...

Table 3
Single nominal ($SingleNP_{protocol}$) and single current profiles $SingleCP_{protocol}$ of protocol attribute.

Protocol name	Number of packets	Protocol name	Number of packets
TCP	85	TCP	185
DNS	10	DNS	17
UDP	4	UDP	8
NTP	1	NTP	1
...	...	...	...

Table 4
Single nominal ($SingleNP_{packetSize}$) and single current profiles $SingleCP_{packetSize}$ of packet size attribute.

PacketSize	Number of packets	Packet size	Number of packets
66	38	66	42
60	29	60	35
83	3	83	12
159	1	159	7
90	3	90	3
...	...	...	...

3.2.2. Comparison

In order to calculate the score of a packet, current pair profile *ScorePCP* and nominal pair profile *ScorePNP* are needed. In a router, the pair that will be utilized in current profiling is called *ScorePair* whereas the current profile that is generated from *ScorePair* is called *ScorePCP*. The router should determine *ScorePair* in order to generate *ScorePCP*. The router's aim is to determine *ScorePair* as the most appropriate attribute pair for current attack. For this aim, the router compares single nominal profiles *SingleNP*s with single current profiles *SingleCP*s that are generated in the previous step. The examples of *SingleNP* and *SingleCP*s of protocol type and packet size attributes are illustrated in Tables 3 and 4.

In this step, the main aim is to determine the two specific attributes that have the most deviation from nominal profiles. Maximum deviation Dev_A of an attribute A can be determined by finding the maximum deviation of all values for A. For instance an attribute A takes$\{a_1, a_2, a_3, \ldots, a_n\}$ values, then initially all deviations should be determined for all values and maximum of these deviations can be found as follows

$Dev_A = MAX\{dev_{a1}, dev_{a2}, dev_{a3}, \ldots, dev_{an}\}$. dev_{an} is the difference between the number of packets with $A = a_n$ in current profile, $npcp_{an}$, and the number of packets with $A = a_n$ in nominal profile, $npnp_{an}$. It can be formulated as $dev_{an} = npcp_{an} - npnp_{an}$.

After maximum deviation Dev_A is determined for each attribute, then the attribute that have the largest deviation is chosen as

$Dev_{MAX1} = MAX\{Dev_A, Dev_B, Dev_C, \ldots, Dev_M\}$ in which M is the number of attributes, in our case it is 6. Subsequently, second largest deviation is chosen as

$Dev_{MAX2} = MAX_2\{Dev_A, Dev_B, Dev_C, \ldots, Dev_M\}$. And lastly, the router generates Suspicious Pair from the attributes that gives Dev_{MAX1} and Dev_{MAX2}. This pair is chosen as the most probable signs for ongoing attacks. Thus, it is called as *SuspiciousPair*.

An example of *SingleNP* and *SingleCP* of an attribute protocol type is illustrated in Table 3. The most deviation of protocol attribute is provided by TCP value and $Dev_{Protocol} = 100$. Similarly, Table 4 shows single profiles of packet size. The most deviation of packet size is provided by 83 and $Dev_{PacketSize} = 9$. All remaining attributes are analyzed in a similar way and all deviations as Dev_{ttl}, $Dev_{portNumber}$, $Dev_{IPnumber}$ and $Dev_{TCPFlag}$ are determined. Then, the two attributes that have the most deviations regarded as *SuspiciousPair*. The router should find a way to access Suspicious Pair nominal profile named as *SPNP* if its *OwnPairList* does not contain the *SuspiciousPair*.

3.2.3. Collaboration

After *SuspiciousPair* is determined by the router, it checks if it has this pair. If, this is the case, then it uses its *OwnPair* as *ScorePair* and *OPNP* as *ScorePNP*. Also it generates *ScorePCP* according to this pair. If the node does not have the nominal profile of *SuspiciousPair*, it needs to communicate with its neighbors. It will ask its neighbors for *SuspiciousPair*'s nominal profile *SPNP*. The owner of the queried pair sends *SPNP* to the router, and then it is used as *ScorePNP*. Also, the router determines *ScorePair* as *SuspiciousPair*. Otherwise, the neighbors ask their neighbors. In other words, second hop routers are utilized in order to access *SPNP* of *SuspiciousPair*. If second hop neighbors does not have this pair, then third hop routers are utilized. If *SuspiciousPair* cannot be accessed even in third hop neighbors then the router have to use its *OPNP* as *ScorePNP* and determine *ScorePair* as its *OwnPair*. A sample broadcast messaging is illustrated in Fig. 1. Router R's *SuspiciousPair* is owned by a third hop neighbor. These operations on Router R is expressed in the following pseudo code.

Ensure: *SuspiciousPair* is determined by R
  if $SuspiciousPair \in OwnPairList_R$ **then**
    $ScorePair_R = OwnPair_R$ and $ScorePNP_R = OPNP_R$
  else if $SuspiciousPair \in OwnPairList_{NR}$ for all $NR \in N$ **then**
    $ScorePair_R = SuspiciousPair$ and $ScorePNP_R = OPNP_{NR}$
  else if $SuspiciousPair \in OwnPairList_{NR'}$ for all $NR' \in N'$ **then**
    $ScorePair_R = SuspiciousPair$ and $ScorePNP_R = OPNP_{NR'}$
  else if $SuspiciousPair \in OwnPairList_{NR''}$ for all $NR'' \in N''$ **then**
    $ScorePair_R = SuspiciousPair$ and $ScorePNP_R = OPNP_{NR''}$
  else
    $ScorePair_R = OwnPair_R$ and $ScorePNP_R = OPNP_R$
  end if

In this pseudocode, subscript in each definition shows the router that owns this property. For instance $ScorePair_R$, $OwnPair_R$, $ScorePNP_R$ are all belong to Router R. $OPNP_R$, $OPNP_{NR}$, $OPNP_{NR'}$ and $OPNP_{NR''}$ are the own pair nominal profiles of Router R, NR, NR' and NR''. Similarly, $OwnPair_R$, $OwnPair_{NR}$, $OwnPair_{NR'}$ and $OwnPair_{NR''}$ are the own pairs of R, NR, NR' and NR''. NR is a router in R's neighbor list N whereas NR' is a router in second hop neighbor list N'. Similarly, NR'' is a router in third hop neighbor list N''.

At first glance, collaboration of the neighbors appears like pointless, since it gives a communication burden for neighbor routers. But indeed, it is a win win negotiation, since attack traffic will be blocked near the source and neighbors' resources will be protected. For this reason, neighbors also want to collaborate with the router in trouble, in order to protect themselves from prospective risk.

After *ScorePair* is determined by collaboration, the router starts to generate its current pair profile *ScorePCP* according to *ScorePair*.

3.2.4. Score calculation

After the collaboration period, the routers have *ScorePNP* and *ScorePCP* tables according to *ScorePair*. Each packet's score is calculated considering *ScorePNP*'s corresponding value. If *ScorePair* is determined as A and B, then packet p with the attributes $A = a_p$

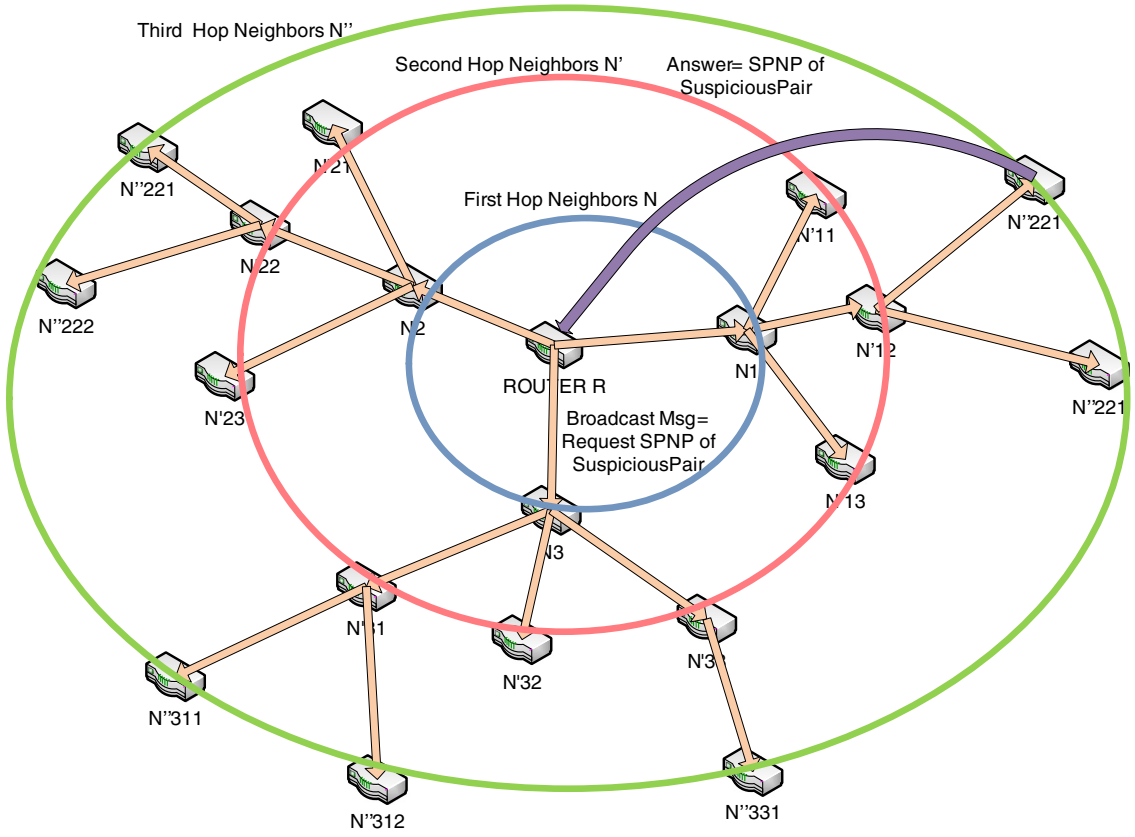

Fig. 1. Broadcasting.

and $B = b_p$ will have the score S_p as follows:

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

In (1), $ScorePCP_{(A=a_p,B=b_p)}$ corresponds to the number of packets in current profile that have the property of a_p for attribute A and b_p for attribute B. $TPCP$ is the total number of packets in current profile. Similarly, $ScorePNP_{(A=a_p,B=b_p)}$ is the number of packets in the nominal profile that have the property of a_p for attribute A and b_p for attribute B. $TPNP$ is the total number of packets in nominal profile.

3.2.5. Threshold calculation

The score of a packet needs to be compared with a threshold Th. All scores are stored in a *ScoreList* and the threshold value Th is determined according to the cumulative distribution of scores by using load shedding algorithm [16]. It is shown as symbolically $CDF(Th) = \Phi$ whereas Φ is the ratio of traffic that should be dropped. The fraction of traffic permitted to pass is $1 - \Phi = \frac{\phi}{\psi}$ whereas ϕ acceptable traffic and ψ is the total current incoming traffic.

3.2.6. Selective discarding

Each packet's score value is compared with the threshold. If it exceeds the threshold, this packet is supposed to be malicious and discarded. Otherwise, it is forwarded to the destination.

To sum up, all these operations are demonstrated in Fig. 2. In this figure, each router has three OPNPs and *SuspiciousPair* is requested from until third hop neighbors. All the operations in each router can be itemized as follows:

- In an attack free period, *SingleNP*s are created for each attribute whereas *OPNP*s are generated by considering randomly chosen *OwnPair*s.
- After a congestion is detected, *SingleCP*s are generated for each attribute.
- *SingleCP*s and *SingleNP*s are compared for each attribute and two attributes that have the most deviation from nominal profiles are determined as *SuspiciousPair*.
- The router itself S, its neighbors N , second hop neighbors N' and third hop neighbors N'' are queried for the *SuspiciousPair*. If any of the routers in neighbor list have the queried pair, then it sends the corresponding pair nominal profile to the requester router. By this way, *ScorePair* and *ScorePNP* are determined by collaboration.
- *ScorePCP* is generated according to the *ScorePair*.
- Each packet's score is calculated by considering *ScorePCP* and *ScorePNP*.
- If the score is under the threshold, it is forwarded otherwise it is discarded.

4. Dataset and simulation details

4.1. Dataset

In our work, we need a real data of the Internet traffic. We use a real dataset from MAWI Working Group Traffic Archive [17]. MAWILab works on traffic measurement analysis in long-term on global Internet. It was started in 2002 and it is still collecting data from the Internet. The part of the data that we have used in our simulations are collected on Jan 12, 2014. This data is utilized to generate nominal profile.

4.2. Simulation environment

We simulated our model ScoreForCore and existing model PacketScore [2], in order to compare their performances. The simula-

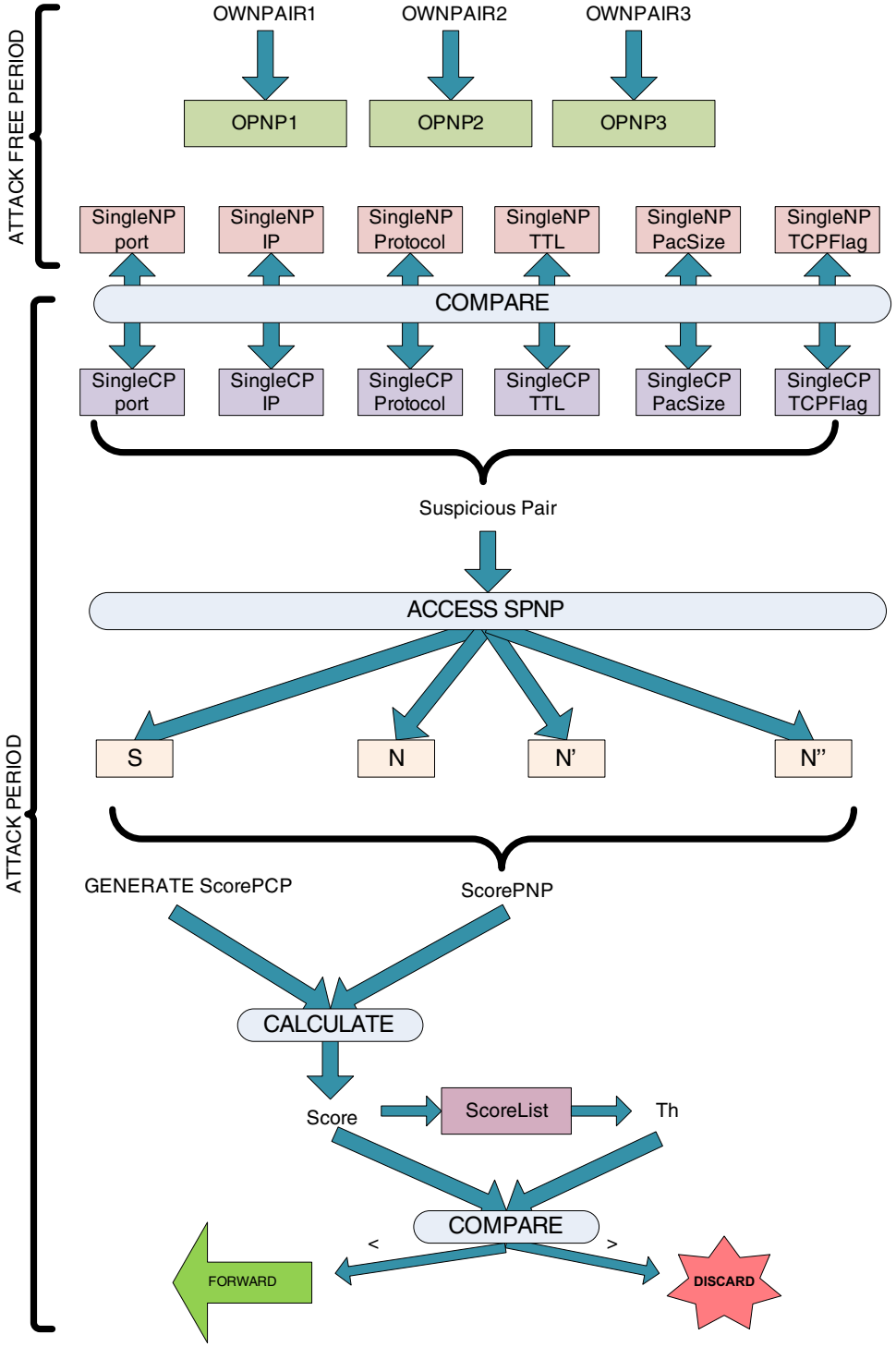

Fig. 2. Operations in ScoreForCore.

tion programs are written in C++. Test environment is a 3.3GHz Intel Core i5 processor with 4GB memory.

4.3. Network generation

In order to perform our simulation, we need to generate a network that is made up of routers. We have utilized a real topology that shows the backbones in United States [18]. It is demonstrated in Fig. 3. It is made up of 18 nodes and average number of neighbors for each node is three.

4.4. Attack generation

According to the network topology in Fig. 3, attack sources and the victim can be scattered. Suppose that backbone router at Hartford is exposed to a DDoS attack. Botnets are attacking from Seattle, St. Louis, Hawaii, San Jose and Atlanta. Other backbone routers are also sending some legitimate traffic to this victim. Legitimate routers generate legitimate traffic that have similar properties and similar amount of traffic to nominal profile packets. Nominal profile packets are created according to the dataset. Attack nodes gen-

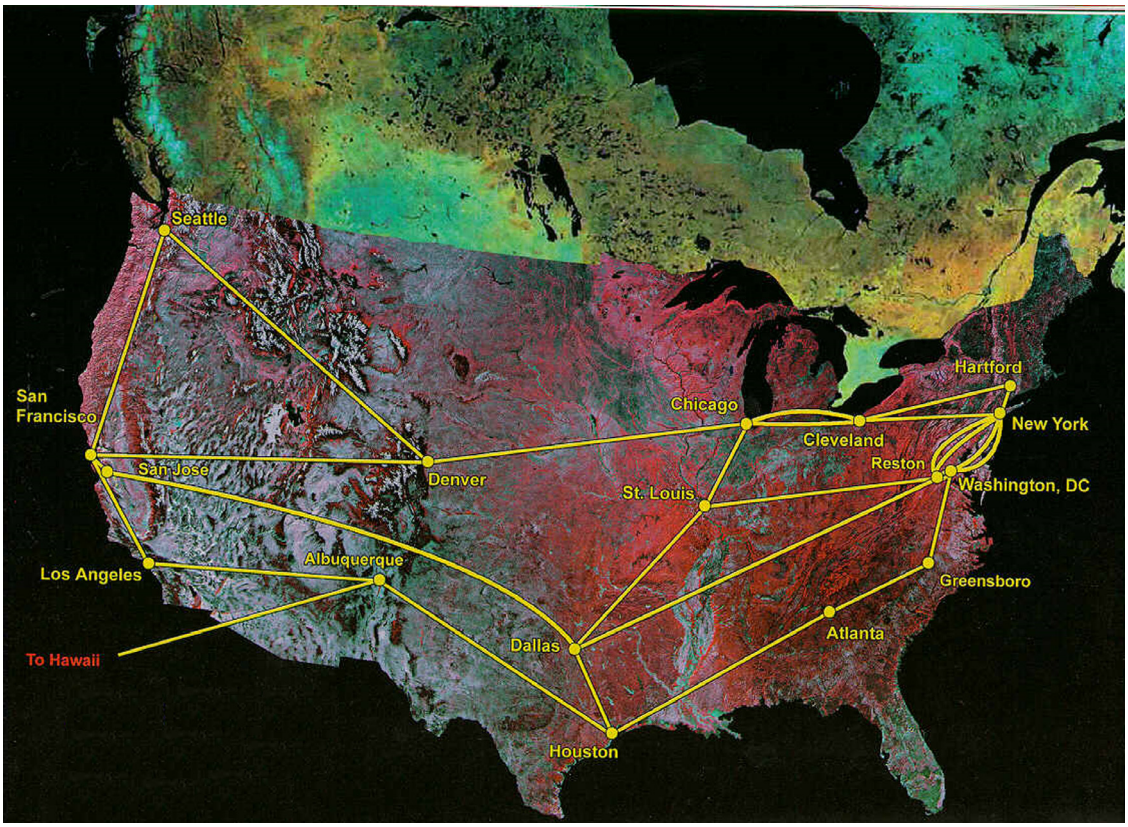

Fig. 3. Network topology.

erate both legitimate and attack traffic. In PacketScore scheme [2], they made comparison in terms of time based vs. packet based period determination. Their results suggest that packet number based intervals are more suitable. Because time-based window may not allow creating meaningful profiles since it may not have as many as packets. In our simulations, for each period, 5000 legal, 50,000 attack packets are generated. We perform attacks by creating new packets that are similar to non-attack period's traffic. We simulate the following attacks:

- Generic attack: All attributes are selected randomly in their ranges.
- *TCP SYN flood attack*: The protocol type of an attack packet is TCP and TCP flag is set to 40. Other attributes are randomized.
- *SQL slammer worm attack*: The protocol type of all attack packets is UDP and destination port is set to 1434. Also, packet size is between 371 and 400 bytes. Other attributes are randomized.
- *DNS amplification attack*: The protocol type of attack packets is DNS and destination port is set to 53. Also, attack packet size is 60 bytes. Other attributes are randomized.
- *NTP attack*: The protocol type of attack packets is NTP and destination port is set to 123. Also, attack packet size is 90bytes. Other attributes are randomized.

5. Performance evaluation

In this section we evaluate the performance of ScoreForCore. Firstly, we explain our performance metrics, secondly analyze attribute distribution in the network topology, then compare the results of ScoreForCore and PacketScore. At the end, technical and empirical storage analysis are provided.

5.1. Performance metrics

In pattern recognition and information retrieval, true positive (TP), true negative (TN), false positive (FP) and false negative (FN) based binary classification metrics 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, FN is the number of legal packets that are falsely discarded whereas FP is the number of attack packets that are falsely forwarded to the destination. Following metrics calculated via these parameters are utilized in performance measurement:

- Precision (PN): What percentage of the forwarded packets were legal?

$$PN = \frac{TP}{TP + FP} \tag{2}$$

- Recal (RL)l: What percentage of the legal packets were forwarded to the destination?

$$RL = \frac{TP}{TP + FN} \tag{3}$$

- True negative rate (TNR): What percentage of the attack packets were dropped?

$$TNR = \frac{TN}{TN + FP} \tag{4}$$

- Negative predictive value (NPV): What percentage of the dropped packets were attack packets?

$$NPV = \frac{TN}{TN + FN} \tag{5}$$

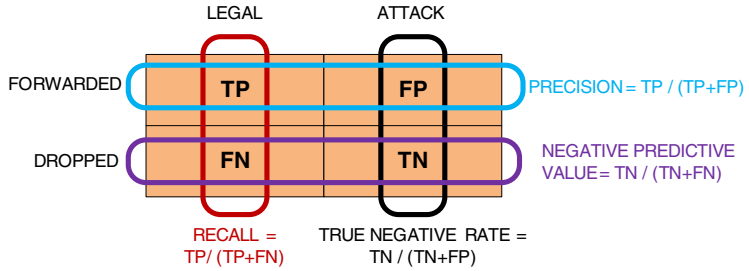

Fig. 4. Metrics.

Table 5
Precision, recall and F-measure results.

Attack types	Model	PL	RL	FM
Generic	PS:	0.59	0.35	0.44
attack	SFC:	0.72	0.94	0.81
TCP-SYN flood	PS:	0.73	0.49	0.59
attack	SFC:	1	0.99	0.99
SQL slammer	PS:	0.98	0.86	0.92
worm attack	SFC:	1	1	1
DNS amplification	PS:	0.84	0.55	0.67
attack	SFC:	1	0.98	0.99
NTP	PS:	0.99	0.82	0.90
attack	SFC:	1	0.99	0.99

- What percentage of the decisions were correct?

$$AY = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

- 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:

$$FM = \frac{2 \times PN \times RL}{PN + RL} \tag{7}$$

- F-measure complement (FMC): As opposed to F-measure, this is dealing with system's success in attack packets. This metric combines true negative rate and negative predictive value. It is the harmonic mean of these metrics. (Fig. 4)

$$FMC = \frac{2 * TNR * NPV}{TNR + NPV} \tag{8}$$

- Attack prevention efficiency (APE): Attack packets are occupying network in vain. Especially in DDoS attacks, since number of packets are huge, they prevent the system to work efficiently. If they are stonewalled early in a network, efficiency of the network will increase. Thus, attack prevention efficiency measures how early the network can get rid of the attack packets. It is formally illustrated as in (9). In this equation, AP is the total number of attack packets, whereas dis_i shows the discard hop of the attack packet i and p_i shows the length of the path for the attack packet i from the source to the destination.

$$APE = 1 - \frac{\sum_{i=1}^{AP} \frac{dis_i}{p_i}}{AP} \tag{9}$$

5.2. Results

In this section, we demonstrate our performance results according to the metrics explained above. In our simulations, PacketScore and ScoreForCore models are both run on all routers in the network. In the following tables, SFC stands for ScoreForCore model whereas PS stands for PackstScore model. Table 5 shows precision, recall and F-measure results for all attack types. Table 6 illustrates true negative rate, negative predictive value and F-measure complement results. Table 7 demonstrates accuracy and attack prevention efficiency values for PacketScore and ScoreForCore.

Table 6
True negative rate, negative predictive value and F-measure complement results.

Attack types	Model	TNR	NPV	FMC
Generic	PS:	0.84	0.67	0.75
attack	SFC:	0.76	0.95	0.85
TCP-SYN flood	PS:	0.89	0.73	0.80
attack	SFC:	1	0.99	0.99
SQL slammer	PS:	0.98	0.92	0.95
worm attack	SFC:	1	1	1
DNS amplification	PS:	0.93	0.76	0.84
attack	SFC:	1	0.98	0.99
NTP	PS:	0.99	0.89	0.94
attack	SFC:	1	0.99	0.99

Table 7
Accuracy and attack prevention efficiency results.

Attack types	Model	AY	APE
	PS:	0.61	0.41
Generic attack	SFC:	0.83	0.65
	PS:	0.73	0.84
TCP-SYN flood attack	SFC:	0.99	0.97
	PS:	0.93	0.78
SQL slammer worm attack	SFC:	1	0.97
	PS:	0.78	0.84
DNS amplification attack	SFC:	0.99	0.97
	PS:	0.92	0.84
NTP attack	SFC:	0.99	0.97

F-measure value shows the success of the system's behavior on legal packets. According to Table 5, for all attack types, ScoreForCore's F-measure result is much better than PacketScore's F-measure result. Success of the system's behavior on legal packets can be interpreted as follows: If there is an attack, PacketScore is mostly inclined to punish legal packets in addition to attack packets, whereas ScoreForCore lets legal packets to reach to destination in massive attack environment. If we analyze values in Table 5, we can remark that SFC gives perfect results for the known attacks. Even for the unknown attack, named as generic attack, SFC overwhelmingly beats PS since SFC chooses the most appropriate attribute whereas PS uses random attributes for analysis.

Table 6 illustrates true negative rate, negative predictive value and F-measure complement results for all attack types. F-measure complement value shows the success of the system's behavior on attack packets. For all attack types, ScoreForCore's F-measure complement value is better than PacketScore's F-measure complement. For known attacks SFC gives perfect results whereas for the unknown attack it manages to give 85% success on attack packets.

In Table 7, accuracy and attack prevention efficiency results are demonstrated. Accuracy values show the system's accurate deci-

Table 8
Communication overhead, memory requirement, accuracy and attack prevention efficiency analysis.

	MEM = 1	MEM = 2	MEM = 3
HOP=1	AY: 0.625 APE: 0.292	AY: 0.764 APE: 0.491	AY: 0.822 APE: 0.586
HOP=2	AY: 0.746 APE: 0.335	AY: 0.801 APE: 0.546	AY: 0.835 APE: 0.682
HOP=3	AY: 0.813 APE: 0.599	AY: 0.821 APE: 0.634	AY: 0.836 APE: 0.685

sions on all packets. As it is obvious, for all attack types ScoreForCore outperforms PacketScore. Since the most appropriate attributes are considered in ScoreForCore, accurate decision number of the system increased. Accuracy is perfect for known attacks in SFC, whereas it performs 83% success for an unknown attack. In addition, attack prevention efficiency results for SFC is perfect for known attacks which means that almost all of the attack packets are stonewalled near the source of the attack.

5.3. Memory requirement, communication overhead, accuracy and attack prevention efficiency analysis

Since our model suggest to select the most appropriate attribute pair, it is important to reach the *SuspiciousPair* during an attack traffic. This is strictly related to the number of hops that the *SuspiciousPair* can be requested and the number of *OPNP*s for each router. If the number of requested hop increases, the probability of reaching desired pair will also increase. Similarly, if the number of nominal profiles increases for each router, the probability of reaching desired pair will also increase. If this probability increases, accuracy increases. Attack prevention efficiency will also increase, since accurate decisions of the routers near the source of the attack will result in early drop of the attack packets. Clearly, increase in the number of hops results in additional communication overhead, whereas increase in the number of nominal profiles results in additional memory requirement. It is obvious that there is a relation between memory requirement, communication overhead and accuracy and attack prevention efficiency. We also analyzed this relationship in our simulations. We perform our analysis for the generic attack. Its results are shown in the following table.

This table quantifies how much accuracy and attack prevention efficiency increase when the number of hop and OPNP increase. The results suggest for a range from 0.29 to 0.69 for attack prevention efficiency and a range from 0.63 to 0.83 for accuracy. The network administrator can select the settings according to the requirements of the network (Table 8).

5.4. Storage analysis

Since several profiles are generated in ScoreForCore, it is important to make analysis of its storage requirement. Theoretical and empirical analysis are provided in the following subsections.

5.4.1. Theoretical

Storage requirement *SR* for a nominal profile can be formalized as follows: $SR = ES * NE$ in which *ES* is the size of an entry, *NE* is the number of entries. Also *ES* can be formalized as $ES = (NA + 1)X32$ bits. *NA* demonstrates the number of attributes in a profile. An entry in a single attribute profile consists of an attribute value and the number of packets that have these attributes. Thus for each entry 32-bit is used for the attribute value whereas the other 32-bit is used for the corresponding number of packets. Then, each entry in a single attribute profile needs 64 bits in other words $ES = 64$. Similarly, in pair attribute profile each entry needs $ES = 96$ bits. In order to find a theoretical bound for *SR*, all packets are supposed to have different values for each packet, then the number of entries are equal to $NE = NP^{NA}$ whereas *NP* is the number of packets. Then, for a single attribute profile storage requirement is $SR = 64 * NP$ bits whereas it is $SR = 96 * NP^2$ bits for a pair attribute profile. If the number of packets is considered as 5000, a single attribute profile needs 40KB, pair attribute profile needs 300MB. These values are far more than real storage since the number of entries are regarded as maximal.

5.4.2. Empirical

In order to show more realistic values, empirical results are provided in this section. The real dataset that we have utilized in our simulations is analyzed in terms of number of different values for each attribute. These values are displayed in Table 9:

According to these values, the largest single attribute nominal profile needs $SR = 64 \times 324$ which means approximately 2.5 KB whereas the largest pair attribute profile requires $SR = 96 \times 324 \times 80$ in other words 311 KBs. These values are much less than the theoretical results and can be affordable easily by the core routers. In our model SFC, routers need to reach the most appropriate attribute pair for score calculation. In order to increase the probability of access to *SuspiciousPair*, nominal profile tables for routers can be increased. Each table increase corresponds to 311 KB memory consumption. In our case, we have utilized three tables for each router.

6. Discussion

Packet marking or communication techniques can be utilized to provide collaboration. Initially, we focused on packet marking techniques and generate a cumulative scoring model. However, we noticed that for core routers the cost of marking packets is a considerable burden, since each packet requires checksum recalculations. In addition, cumulative scoring does not always give accurate results since irrelevant attributes are considered and they mislead the statistical analysis. Then, we decided to focus on finding the most appropriate attribute pair and utilize communication for providing collaboration.

In ScoreForCore, each router needs to access the most appropriate attribute pair. However, this condition depends on the number of neighbors for each router in the topology. In order to increase the probability of access to the desired pair, request hop and number of attribute pair for a router can be increased. Increasing request hop results in increase in communication overhead whereas increasing number of attribute pair for a router requires more storage space. Thus, ScoreForCore has memory and communication overhead issues that needs to be considered. As it is mentioned in previous section, since pair attributes are utilized instead of tuples, storage requirements is in KBs which can be afforded easily by the core routers. Besides, it does not give much burden on communication since additional hop means an additional broadcast message and an answer which is a piece of cake for a router.

Another issue is the need of different profiles for different times of day. This is also analyzed in [2]. Their analysis suggests that the traffic profiles are most similar among the traffic at the same day, at the same time. In addition, a traffic profile is still very similar for a different time or day. They compared morning and evening profiles and they suggest that it may be enough to keep one profile. But still, it is possible to compare the stability of the profiles and if there is too much difference, it may be better to keep more than one profile for different times of a day and then routers will exchange these profiles according to the current time. However, it is obvious that maintaining different profiles per site and per time of day would increase managerial overhead. Thus, there is a trade off between more accurate profiling and managerial overhead. In

Table 9
The number of different values for each attribute.

	SourceIP	DestPort	Protocol type	TCP flag	TTL	Packet size
The number of different Values	324	80	7	8	59	67

our simulations, we suppose that profiles are very similar during the day and one profile is utilized for nominal traffic.

7. Conclusion

In this work, we propose a proactive and cooperative filtering model against DDoS attacks. This type of filtering provides an ability of preventing the attack packets before they expand and gives more accurate decisions since they create filters together with more knowledge of the network. The proposed model ScoreForCore is a statistical based model that utilizes several attributes. It can protect against botnets since not only IP address but also several attributes are considered. It can also protect against unknown attacks since it makes statistical analysis and compares the current traffic with the nominal profile. The most distinctive property of this model is the selection of the most appropriate attributes for current attack traffic. This provides considerable improvement in accuracy. We evaluate ScoreForCore according to the precision, recall, f-measure, f-measure complement, accuracy and attack prevention efficiency metrics. The results suggest that it gives perfect accuracy and attack prevention efficiency for several known attacks. Also, almost all known attack traffic is stonewalled near the source of the attack. Besides, it performs 80% accuracy for an unknown attack called as *generic attack*. Our results are good enough to encourage other researchers to work on proactive and collaborative filtering mechanisms against DDoS attacks.

As a future work, we plan to make experiments in a real testbed. This will make our results more realistic. In addition, secure communication protocols can be provided for collaboration of backbones. Besides, choosing the most appropriate attribute strategy can be applied for other statistical defence mechanisms. In addition, this collaborative model can be applied to emerging Software Defined Networking (SDN) technology. As it is stated in [19], SDN has a logical centralized controller that has network-wide knowledge of the system and can analyze the traffic patterns easily. Also, it can update forwarding rules dynamically. For these reasons, SDN brings new opportunities to defeat against DDoS attacks in cloud computing environment. Besides, according to [20,21], a cooperative defense mechanism can be very effective against DDoS attacks.

References

[1] D. Seo, H. Lee, A. Perrig, PFS: probabilistic filter scheduling against distributed denial-of-service attacks, in: IEEE 36th Conference on Local Computer Networks (LCN), 2011, pp. 9–17.
[2] Y. Kim, W.C. Lau, M.C. Chuah, H.J. Chao, PacketScore: a statistics-based packet filtering scheme against distributed denial-of-service attacks, IEEE Trans. Dependable Secure Comput. 3 (2) (2006) 141.
[3] D.S. Sivia, Data Analysis, A Bayesian Tutorial, Oxford, 1996.
[4] J. Udhayan, T. Hamsapriya, Statistical segregation method to minimize the false detections during DDoS attacks, Int. J. Network Secur. 13 (3) (2011) 152–160.
[5] Q. Chen, W. Lin, W. Dou, 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, 2011, pp. 427–434.
[6] R. Mahajan, S.M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, S. Shenker, Controlling high bandwidth aggregates in the network, ACM SIGCOMM Comput. Commun. Rev. 32 (3) (2002) 62–73.
[7] A.D. Keromytis, V. Misra, D. Rubenstein, SOS: an architecture for mitigating DDoS attacks, IEEE J. Sel. Areas Commun, 22 (1) (2004) 176–188.
[8] J. François, I. Aib, R. Boutaba, FireCol: a collaborative protection network for the detection of flooding DDoS attacks, IEEE/ACM Trans. Networking (TON) 20 (6) (2012) 1828–1841.
[9] C. Jin, H. Wang, K.G. Shin, Hop-count filtering: an effective defense against spoofed DDoS traffic, in: Proceedings of the 10th ACM Conference on Computer and Communications Security, CCS '03, 2003, pp. 30–41.
[10] K. Argyraki, D.R. Cheriton, Active internet traffic filtering: real-time response to denial-of-service attacks, in: Proceedings of the Annual Conference on USENIX Annual Technical Conference, ATEC '05, USENIX Association, Berkeley, CA, USA, 2005, p. 10.
[11] X. Liu, X. Yang, Y. Lu, To filter or to authorize: network-layer DoS defense against multimillion-node botnets, ACM SIGCOMM 2008, 2008.
[12] D. Seo, H. Lee, A. Perrig, APFS: adaptive probabilistic filter scheduling against distributed denial-of-service attacks, Comput. Secur. 39 (2013) 366–385.
[13] P. Ferguson, D. Senie, Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing, RFC 2827, RFC Editor, 2000.
[14] K. Park, H. Lee, On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law internets, SIGCOMM Comput. Commun. Rev. 31 (4) (2001) 15–26, doi:10.1145/964723.383061.
[15] J. Li, J. Mirkovic, M. Wang, P. Reiher, L. Zhang, SAVE: source address validity enforcement protocol, in: IEEE INFOCOM 2002, 2001, pp. 1557–1566.
[16] S. Kasera, J. Pinheiro, C. Loader, M. Karaul, A. Hari, T. LaPorta, Fast and robust signaling overload control, in: Ninth International Conference on Network Protocols, 2001, IEEE, 2001, pp. 323–331.
[17] R. Fontugne, P. Borgnat, P. Abry, K. Fukuda, MAWILab: combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking, ACM CoNEXT '10, Philadelphia, PA, 2010.
[18] The internet topology zoo, (http://www.topology-zoo.org/, note = 2016,).
[19] Q. Yan, R. Yu, Q. Gong, J. Li, Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: a survey, some research issues, and challenges, Commun. Surv. Tutorials, IEEE PP (99) (2015) 1, doi:10.1109/COMST.2015.2487361.
[20] T. Chin, X. Mountrouidou, X. Li, K. Xiong, An SDN-supported collaborative approach for DDo flooding detection and containment, in: Military Communications Conference, MILCOM 2015-2015 IEEE, IEEE, 2015, pp. 659–664.
[21] Q. Yan, et al., Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges, IEEE Communications Surveys & Tutorials 18 (1) (2016) 602–622.

Kübra Kalkan received her MS and BS degrees in Computer Science and Engineering department from Sabanci University, Istanbul, Turkey in 2011 and 2009, respectively. Currently she is a Ph.D. candidate in computer engineering, and is working as a researcher at Telematics Research Center (TAM) of Bogazici University. She is also working as a teaching assistant at Istanbul Medeniyet University. Her current research interests include network security, computer networks and wireless networks.

Fatih Alagöz is a professor in the Department of Computer Engineering, Bogazici University. He received the B.Sc. degree in Electrical Engineering from Middle East Technical University, Turkey, in 1992, and M.Sc. and Ph.D. degrees in Electrical Engineering from The George Washington University, USA, in 1995 and 2000, respectively. His current research interests are in the areas of wireless/mobile/satellite networks. He has contributed/managed ten research projects for the US Army of Intelligence Center, Naval Research Laboratory, UAE Research Fund, Turkish Scientific Research Council, State Planning Organization of Turkey, BAP, etc. He has published more than 150 scholarly papers in selected journals and conferences.