Journal of Intelligent & Fuzzy Systems 35 (2018) 5983–5991
DOI:10.3233/JIFS-169839
IOS Press

An efficient detection of TCP Syn flood attacks with spoofed IP addresses

Youngjun Bae[a], Intae Kim[a] and Seong Oun Hwang[b,*]
[a]*Department of Electronics and Computer Engineering, Hongik University, Korea*
[b]*Department of Software and Communications Engineering, Hongik University, Korea*

Abstract. TCP SYN flood attack has been one of representative DDoS attack in computer security history. To cope with this, a number of researches have been done, but they have a high false detection rate and are hard to be applicable in network address translation environment which is very common in the real world. To address these problems, we propose an efficient scheme to cope with SYN flood attacks with spoofed IP addresses. Compared to the existing approaches, it achieves the lowest false positive rate of 0.0003% at maximum and detects false IP packets at an earlier point of time, which serve to reduce the impacts of DDoS attacks significantly.

Keywords: SYN flood attack, DDoS attack, bloom filter

1. Introduction

When a server and a client establish Transmission Control Protocol (TCP) communication connection on the Internet, they go through 3-way handshake process [1, 2]. However, when a large volume of SYN packets are sent to the server side during the three way handshake, SYN flooding attacks [3, 4] can occur which is one of DDoS (Distributed Denial of Service) attacks depleting the resources of the server.

To exploit the TCP vulnerabilities, an attacker repeatedly transmits the SYN packets which originally serve to initiate a connection from a client to a server. When a SYN packet has arrived at the server side, the server normally responds a corresponding SYN/ACK packet to the client. However, when the client does not respond the server with the subsequent ACK packet corresponding to the SYN/ACK packet, which usually happens when the client IP address is faked by the attackers, the server waits for the response from the client during a specified time in "half open" state. As SYN packets are piled up at the server side, the server cannot accept SYN packets any more. When SYN flooding attacks occur with spoofed IP addresses, it is very hard to determine at the server side whether the packets are from legitimate clients or attackers. Once accurate detection is made, we can cope well with DDoS attacks at their initial phases. Therefore, as an effective countermeasure to DDoS attacks, it is very important to detect spoofed IP addresses.

Until now, a number of researches have been done to detect such attacks. Hop count filtering [5, 6] and TTL analysis [3, 6] attempt to infer hop counts of incoming packets by observing the proper TTL value on a given network. In the work such as packet marking detection [7–10] and bogon filter detection [11], a routers divides tracing data into a number of packet's headers which are collected and used to trace back to the source. Threshold algorithms [12, 13] are based on the observation result that a normal user cannot send out at most 3-4 requests per a second. They initially set a threshold value, and regard a client as

*Corresponding author. Seong Oun Hwang, Department of Software and Communications Engineering, Hongik University, Korea. E-mail: sohwang@hongik.ac.kr.

an attacker when its connection requests exceed the threshold value. In addition, additional measure such as SYN cookies [12] has been done.

In the above, however, the false detection rate is very high and not applicable in network address translation (NAT) environment which is very common in most business networks. Practically, it is not easy to apply router-based detection algorithms [8–10] to most routers in the real world. These router-based techniques require a high storage capacity of the log table on routers and involve false positive (due to digest collision on the log table) as well as false negative (caused when refreshing logged data) problems. Threshold-based approaches [12, 13] deal well with DDoS attacks with the same source using the accumulated source information, but cannot detect the ones with randomly generated IP addresses. Another problem with [13] is, when the threshold value is wrongly set, it could falsely regard a normal connection as abnormal.

The proposed scheme is designed to achieve the following properties: (1) It achieves almost zero/adjustable false positive (i.e., reducible to zero false positive by adjusting parameters) and zero false negative rates in IP spoofed packet detection; (2) It is more efficient than the existing approaches in the sense that it can detect false IP packets at an earlier point of time (i.e., as shown in Fig. 1, existing approaches wait for ACK packets from the client and then take the response measures later, while as soon as our approach detects forged SYN packets from the client, it responds accordingly), which significantly results in reducing the amount of target packets to be checked; (3) It is able to effectively cope with the real world network settings such as NAT; (4) Storage and search speed requirements for the detection server are good enough to be supported by the current server capability.

2. Proposed system

2.1. Overview

As shown in Fig. 2, the proposed system consists of two agents: the client agent and the server agent. The client agent inserts an authentication data into SYN packets [1, 2] which are usually used to signal its intention to connect to the server. The authentication data as shown in Fig. 3 consists of MAC address, IP address and keyed hash value. The keyed hash value is generated by applying the XOR

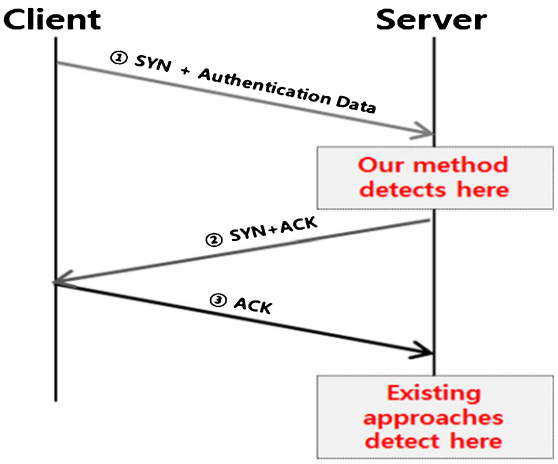

Fig. 1. TCP three-way handshake.

operation of a key and the MD5 hash function on the input data which consists of MAC address, IP address and counter at the client agent's side. The reason why it uses a keyed hash function is to prevent the brute force attacks. In this process, two random value is used as the input and key, respectively. Therefore, the attacker cannot deduce the next right keyed hash value from previous values. The client agent inserts the authentication data in a segmented way across both IP and TCP header fields.

As shown in Fig. 4, we use Differentiated Services Field (TOS) (1byte), Identification Field (2bytes) in the IP header; Acknowledgement Number Field (4bytes), Urgent pointer Field (2bytes) in the TCP header.

Since aforementioned data fields are rarely used in the real world and used only when the agents establish connections, they do not cause any standard-conforming problems. Particularly, they do not cause any impacts on the higher application layer such as HTTP, Telnet, and FTP.

When there exists no authentication data inside the SYN packets, the server agent redirects the connection request to the installation Web page, which allows the client (or user) to install the client agent.

When installing the client agent, the server agent assigns a random counter value and a random key to the client agent and stores the client's MAC address, the counter value and the random key on its mapping table.

When the client and server agents successfully establish a connection, they update their counter values in a synchronized way by incrementing the counter value by one.

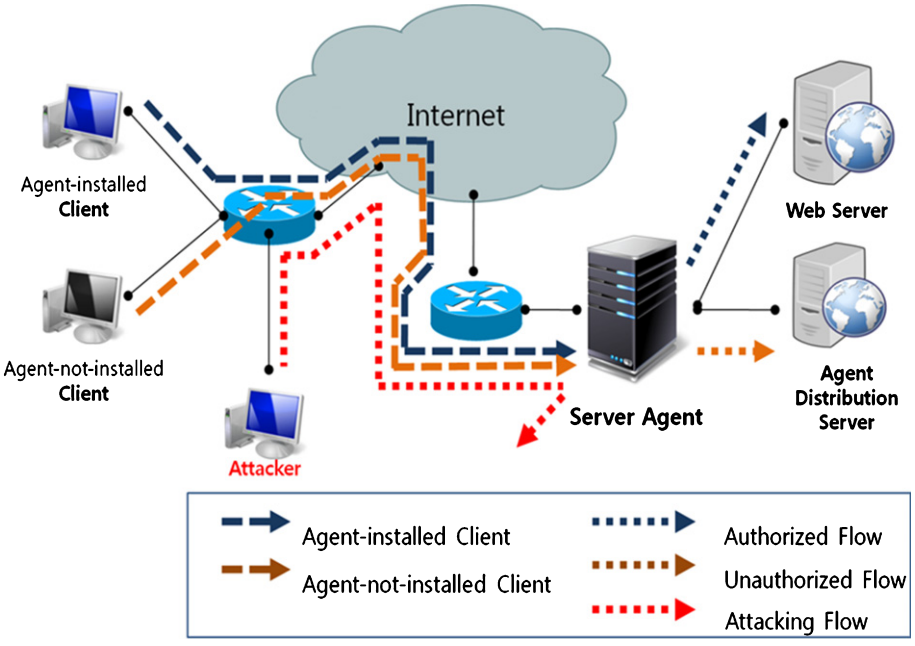

Fig. 2. Configuration of the proposed system.

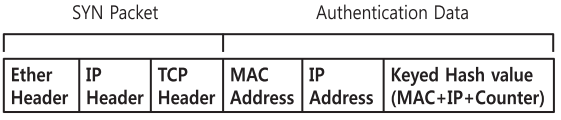

Fig. 3. Format of SYN packet including authentication.

Through retrieving and analyzing the authentication data inside the SYN packets received from the client agent, the server agent determines where the requested connections come from legitimate users or attackers.

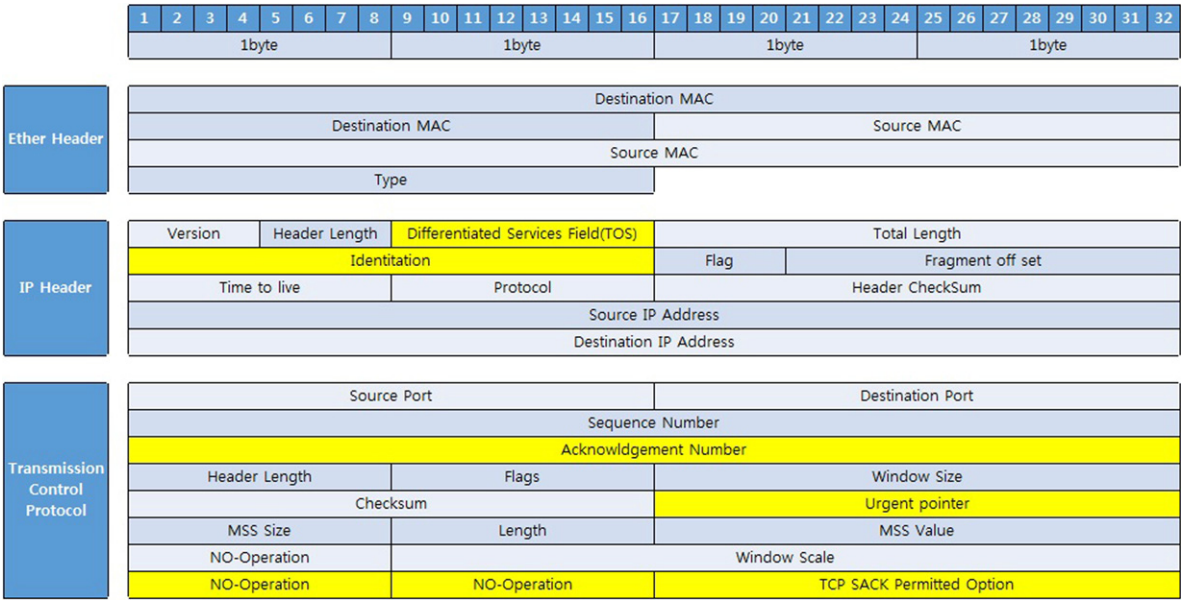

Fig. 4. Bright colored fields in both IP and TCP headers are used to insert an authentication data.

2.2. Hashing scheme and mapping table

A Bloom filter is a space-efficient probabilistic data structure that is widely used to test membership. A Bloom filter registers hash function values of elements and compare a hash function value of an element to query in the bit array. In the paper, we use a Bloom filter whose bit size is 16,777,216 bits requiring 2 MBs. Note that 2MB data cannot be inserted into a SYN packet. It also causes significant network traffic and low processing speed, which degrades significantly the performance of the system. Therefore, it is not a good idea to apply the ordinary Bloom filter to insert the authentication data into packets. To address this issue, instead of using the Bloom filter as it is, we use a new variant of the Bloom filter by applying three hash functions to the outputs of the Bloom filter to get three 24 bit outputs, which are XORed to get the final 24 bit output. Instead of inserting bit vectors (or arrays) used in the Bloom filter into a packet, we convert them into hexadecimal code values which serve as indexes to the entry of the mapping table at the server side. The final 24 bit output can have a maximum integer value of 16,777,215, which just requires 3 byte space, small enough to be transmitted as stored in the empty space of a SYN packet. This is a great advantage because we can insert authentication data without increasing the packet size. The reason why we select bitwise-XOR operation instead of other operations such as AND, OR is that this operation has an effect of distributing the resulting data evenly which leads to comparatively reduce the number of duplicates. By taking the proposed method, we can reduce the communication overhead from 128 bits to 24 bits. Lastly, it is completed by operating keyed hash function to use a key.

The server agent maintains a mapping table which contains entries of MAC address (6 bytes), counter value (3 bytes) and key (2 bytes) of the client agent which was once successfully connected. The counter value of 3 bytes ranging from 0 to 16,777,215 and the key of 2 bytes serves to make the keyed hash value of the authentication data inserted by the client agent random. The server agent computes the authentication data of the client agent by using these values and compares it with the one retrieved from the SYN packet. If the verification is successful, it increments the counter by one, which makes the authentication data randomly updated whenever the connection is established.

To access the mapping table in a form of DB may lead to a slow processing at the server side.

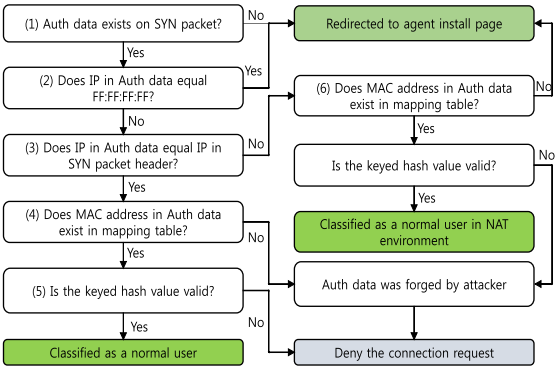

Fig. 5. Packet detection flow by the server agent.

To speed-up the access of the mapping table, the server agent stores the mapping table in a DB when it shutdowns and reloads it into a memory when it starts. When loading the mapping table into a memory, we need 11 bytes per client. Therefore, we need 1 Giga byte memory for 97,612,893 clients, which is not a big one when considering the recent computer specifications.

2.3. Packet detection

The server agent monitors the received packets and determine whether they are normal or abnormal by checking the SYN packets.

Packet detection is done by applying the algorithm as shown in Fig. 5:

(1) The server agent checks if there exists an authentication data inside the SYN packet. If not, it forwards the connection request to the agent installation Web page.
(2) If the IP address of the authentication data is equal to FF:FF:FF:FF, it presumes that the NIC device was changed and forwards the connection request to the agent installation Web page. If yes, it forwards the connection request to the agent installation Web page.
(3) Otherwise at step (2), it compares the IP address in the IP header of the SYN packet with the one in the authentication data.
(4) When the IP addresses are equal at step (3), it checks whether the MAC address in the authentication data exists in the mapping table.
(5) When the same MAC address exists at step (4), it determines whether the connection request comes from a normal user by verifying the keyed hash value of the authentication data

```
PacketClassify
    Get SYN packer;
    If SYN packet does not have authentication data field
        Classify it as Type 1: Agent-not-installed client;
    End if
    If SYN packet has authentication data field
        If IP in authentication data field == FF:FF:FF:FF
            Classify it as Type 1: Agent-not-installed
    client;
        End
        If MAC exists in the server mapping table
      sip := source IP in IP header;
      sip':= IP in authentication data field;
      bf:= computed keyed hash value on MAC, and IP in
          authentication data field, and counter and key in
          the server mapping table;
      bf':= keyed hash value in SYN packet;
      If sip==sip' and bf == bf'
          classify it as Type 2: Agent-installed client;
      Else if sip != sip' and bf == bf'
          classify it as Type 3: NAT:Agent-installed client;
      Else
          classify it as Type 4: Forged packet;
      End if
    Else
      Classify it as Type 1: Agent-not-installed client;
    End if
  End if
```

Fig. 6. Packet classification algorithm by the server agent.

based on both the IP address in the authentication data, the counter value and the key in the mapping table.

(6) When the IP addresses are not equal at step (3), it presumes that the connection request comes from a user in NAT environment and checks whether the MAC address in the authentication data exists in the mapping table. When the same MAC address exists, it verifies the keyed hash value of the authentication data based on both the IP address in the authentication data, the counter value and the key in the mapping table. If the verification is successful, it determines that the connection request comes from a normal client in NAT environment. Otherwise it determines that the authentication data was forged by attackers.

The server agent classifies the incoming packets into the following four types and processes them accordingly as shown in Fig. 6:

- Type 1: A client which does not install the client agent.
- Type 2: A client which installs the client agent and turns out to be a normal user after the verification of the authentication data
- Type 3: A client which installs the client agent and turns out to be a normal user behind an NAT server after the verification of the authentication data
- Type 4: An attacking client by forging the authentication data

In NAT-installed local network, a number of hosts share a single IP address to access the public network. In face of the foreseeable global IP address space exhaustion, NAT was increasingly used in enterprise network environment.

In face of the foreseeable global IP address space exhaustion, NAT was increasingly used in enterprise network environment. In security viewpoint, however, NAT makes it difficult to trace back to the attackers because all of the private internal IP addresses are viewed as one public IP address. To get around this problem, we propose the following packet detection flow in NAT environment:

(1) When the IP address does not match the IP address in the IP header, it determines the connection request comes either from a user behind an NAT server or from an attacker.
(2) It searches the entries in the mapping table for the counter value whose corresponding MAC address is equal to the MAC address inside the authentication data.
(3) It computes a MD5 hash value using the MAC address, IP address inside the authentication data and the counter value obtained from the mapping table. Then it is completed by operating keyed hash function to use a key.
(4) When the hash value inside the authentication data and the computed one at step (3) are equal, it determines that the connection request comes from a normal user in an NAT environment. Otherwise it determines that the connection request comes from an attacker.

2.4. *Security analysis and performance evaluation*

We first analyze the security of the proposed system and then present the maximum probability of false positive rate through our simulation.

Theorem 1. *The attacker cannot make an accurate estimation of the valid hash value in the authentication field (hash value prediction attack).*

Proof: Packet sniffing is possible on the same local network or on a higher network group. Attackers can figure out that a client agent inserts its MAC and IP addresses by analyzing the sniffed packets.

Table 1
Mapping table on server's agent

Bit Size	No. of hash functions	Max. of false positive	Min. of false positive
100	2	1.0111	0.6987
	3	1.1121	0.5741
	4	7.7387	0.6071
	5	1.0313	0.5051
	6	1.2221	0.5388
1,000	2	0.1078	0.0886
	3	0.1238	0.0787
	4	3.2668	0.0368
	5	0.1139	0.0686
	6	0.1134	0.0878
10,000	2	0.0133	0.0126
	3	0.0125	0.012
	4	0.9586	0.0011
	5	0.0131	0.0023
	6	0.0127	0.0029
100,000	2	0.0026	0.0001
	3	0.0023	0.0001
	4	0.3901	0.0001
	5	0.0023	0.0001
	6	0.0027	0.0001
1,000,000	2	0.0009	0.0001
	3	0.0009	0.0001
	4	0.1764	0.0001
	5	0.0009	0.0001
	6	0.0009	0.0001

In the proposed system, however, it is very hard for an attacker to compute the exact hash values because the hash values are computed on every connection using the counter value which is known to and maintained by both the client and server agents only and not inserted directly into the packet. In addition, since the counter value is incremented by one upon the connection is established, the hash values of the authentication data change per every connection. Therefore, the probability that an attacker successfully obtains exactly the same hash values as the one inside the authentication data is $1/2^{24} = 1/16777216$, which is low enough for an attacker to launch an attack successfully.

Result 1. The maximum false positive rate of the proposed method is 0.0003%.

The proposed scheme eliminates false negatives, but introduces some false positives which can be tuned to be very small. To find out the false positive rate, we did the following simulation as shown in Table 1.

To generate the final hash values, we first generated MD5 hash values by randomly increasing both MAC and IP addresses of the client agent. Then, those hash values are fed to our own Bloom filter program, which outputs 1,000,000 hash values of 24 bits for each case with the number of hash functions of 2, 3, 4, 5, 6 and bit sizes of 100, 1000, 10000, 100000, and 1000000. Figure 7 and Table 2 shows that the probability of false positive has no relationship with the number of hash functions, but has some relationship with the size of the Bloom filter.

Setting the bit size of the Bloom filter to 16,777,215 and the number of hash functions to 3, we generated 1,000,000 client information and performed bitwise-XOR operation on the 1,000,000 resulting data set. In this case, the probability that the hash value generated from one client information and the one generated from 1,000,000 client information are identical is 3 out of 1,000,000. It means the maximum false positive rate is 0.0003%. As shown in Fig. 8, we can see that up to 3 duplicates happen per each hash value.

3. Experimental study

3.1. Experimental environment

Unlike the previous simulation whose purpose is to get the maximum value of false positive rates, this experimentation is to find out how many false detections happen per each type. We constructed a small-sized test bed which has a standard structure in the Internet. The test bed consists of 20 nodes PCs (clients and servers) with Intel P4 930 3 GHz, 2 G RAM and FreeBSD 6.2, six routers with Cisco 1800 Series, Cisco 1900 Series, Cisco 2800 Series and switches with Cisco Catalyst 2960 Series, Cisco Catalyst 3560 Series.

As shown in Fig. 2, an Agent-installed Client sends SYN packets containing the authentication data, but an Agent-not-installed Client sends normal SYN packets (i.e., one without the authentication data) to Packet Detection Server. Clients belonging to NAT Group sends SYN packets with internal private IP addresses and the NAT server replaces the private IP addresses with pre-configured public IP addresses. Attacker sends SYN packets with forged authentication data to Packet Detection Server.

We conducted our experiment of packets transmission and detection by classifying the clients into the aforementioned four types. Type 1, Type 2, and Type 3 packets are generated by normal clients and Type 4 packets are forged packets by attackers, who use randomly generated IP addresses and hash values with MAC address registered in the mapping table of the server. From this, we could confirm that there exists

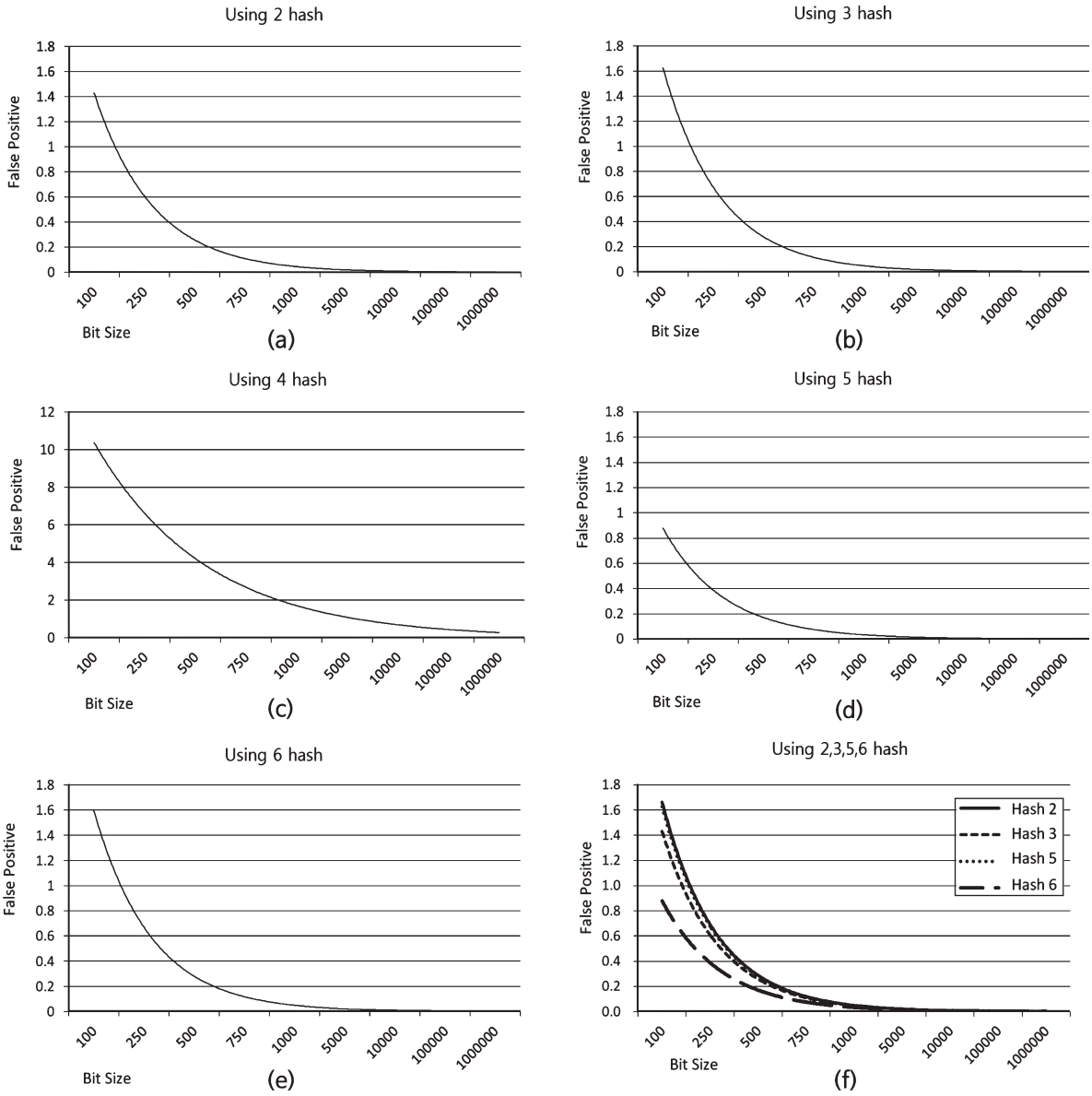

Fig. 7. False positive rates under different configurations.

Table 2
Transmission result according to packet type

Packet type	No. of transmitted packets	Transmission result							
		1	2	3	4	5	6	7	8
TYPE1	1,000,000	1,000,000	1,000,000	1,000,000	1,000,000	1,000,000	1,000,000	1,000,000	1,000,000
TYPE2	1,000,000	1,000,000	1,000,000	1,000,000	1,000,000	1,000,000	1,000,000	1,000,000	1,000,000
TYPE3	1,000,000	1,000,001	1,000,000	1,000,002	1,000,000	1,000,000	1,000,000	1,000,000	1,000,001
TYPE4	1,000,000	999,999	1,000,000	999,998	1,000,000	1,000,000	1,000,000	1,000,000	999,999
No. of false detection		1	0	2	0	0	0	0	1
Probability of false detection		0.0001%	0%	0.0002%	0%	0%	0%	0%	0.0001%

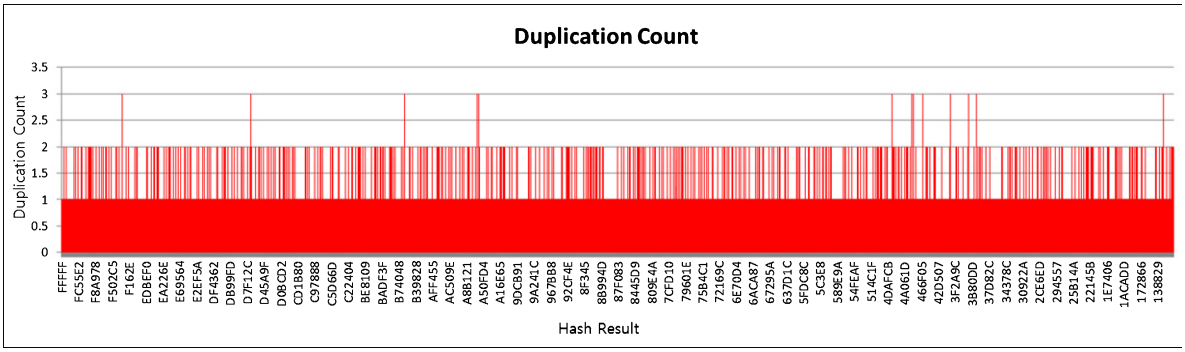

Fig. 8. Occurrence of duplication in the proposal.

a chance of determining forged packets as the ones from NAT environment, although it is very low.

3.2. False positive and false negative rates

Table 2 shows that packets of Type 1 and 2 are correctly classified, but, those of Type 3 and 4 are incorrectly classified, where one packet of Type 4 is classified as Type 3. It is due to the fact that one forged packet of Type 4 was classified as normal packet of Type 3 because the hash value inside the authentication data in the packet matches the hash value computed over both MAC and IP addresses in the packet and a counter value in the corresponding entry of the mapping table. The probability of false positive was found to range from 0.0001% to 0.0002%. The probability can be made as small as desired as follows: Note that the size of the authentication data is 13 bytes (MAC address: 6 bytes, IP address: 4 bytes, hash value: 3 byte). If we increase the size of the hash value of the authentication data from 3 bytes up to 13 bytes, then the probability can be reduced to 0%.

Since MTU of Ethernet is 1500 bytes, the maximum size of a packet is 1500 bytes. The size of basic form of SYN packets is 66 bytes: 20 (TCP header) + 12 (TCP option) + 20 (IP header) + 14 (Ethernet header) (excepting Presemble (8bytes) + CRC (4bytes)). Because we add 20 bytes to the basic SYN packet (3 bytes among 23 bytes can use the Option field in the Ethernet header), which results in 86 bytes, there is no problem in transmitting the augmented packets. However, there is a possibility that a network device on ingress border drops off the augmented packets because their sizes are different from those of normal SYN packets.

Table 3 shows that the false detection rate of the proposed method is significantly low compared to the existing methods.

Table 3
Comparison of error rates

Method	Error rate	Features
BASE	N/A	Error rate increases when the information stored in BASE filter is wrong.
Hop count filtering	10%	Error rate is large because detection is made based on the average of network addresses.
FDPM	0.1%	Detection is made based on the information inserted by neighboring routers to the source of the packets.
Proposal	0.0003%	Accurate detection is made based on the inserted authentication data

4. Discussions

The proposed system can be efficiently applicable in the real world, for example, DDoS Cyber-Shelter model [14] in a cyber control center, which identifies targets network or services to be protected, and installs the proposed systems at the target system beforehand. We note that a cyber control center usually maintains large bandwidth network equipments and strong capacity servers enough to resist against DDoS attacks. And when an indication/symptom of attacks is probable, for example, the proposed system can be activated to response. In this way, our authentication-based agent model can be operated in a cost-effective way.

Although the scope of the proposed scheme is limited to SYN flood attacks with spoofed IP addresses, it can be helpful in reducing the impacts of attacks from when combined with the above DDoS Cyber-Shelter model: The server checks whether the incoming packets contain authentication data fields or not. If not, it simply redirects the packets to the installation server deployed in a cyber control center. Note that normal

users with authentication data can get access to the service even during the attacks are made.

5. Conclusion

In the paper, we proposed an efficient scheme to cope with SYN flood attacks with spoofed IP addresses. Compared to the existing approaches, it achieves the lowest false positive rate of 0.0003% at maximum and detects false IP packets at an earlier point of time, which serve to reduce the impacts of DDoS attacks significantly. The system also makes packet sniffing/analysis attack difficult by using a randomly generated hash value inside the SYN packets which are hard to guess by the attacker. The propose scheme also can be applicable in NAT network environment, where it is intrinsically hard to detect spoofed IP addresses.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2017R1A2B4001 801). This work (Grants No. C0564519) was also supported by Business for Cooperative R&D between Industry, Academy, and Research Institute funded Korea Small and Medium Business Administration in 2017.

References

[1] R.S. Tomlinson, Selecting Sequence Numbers, INWG Protocol Note 2, IFIP Working Group 6.1, August 1974. Also in Proceedings of the ACM SIGCOMM/SIGOPS Interprocess Communications Workshop, (Santa Monica, CA, March 24–25, 1975), and ACM Operating Systems.

[2] J. Postel, RFC792–Darpa Internet Program Protocol Specification, 1981.

[3] CERT Advisory CA-1996-21 TCP-SYN Flooding and IP Spoofing Attacks, 2000.

[4] H.-R. Tang, C. Xu, X.-G. Luo and J.-Q. Ouyang, Traceback-Based Bloomfilter IPS in Defending SYN Flooding Attack, Wireless Communications, Networking and Mobile Computing, 2009, pp. 1-6.

[5] H. Wang, C. Jin and K.G. Shin, Defense against spoofed IP traffic using hop-count filtering, *IEEE/ACM Trans Networking* **15**(1) (2007).

[6] S.J. Templeton and K.E. Levitt, detecting spoofed packet, the DARPA Information Survivability Conference and Exposition (DISCEX'03), 2003.

[7] Y. Xiang and W. Zhou, Flexible Deterministic Packet Marking - An ip traceback system to find the real source of attacks, *IEEE Trans. Parallel and Distributed Systems* **20**(4) (2009), 567–580.

[8] A. Yaar, A. Perrig and D. Song, New packet marking and filtering mechanisms for DDoS and IP spoofing defense, *IEEE Journal on Selected Areas in Communications* **24**(10) (2006).

[9] D. Li, Pei Chen Optimized hash lookup for bloom filter based packet routing, Network-Based Information Systems (NBiS), *16th International Conference on*, 2013.

[10] H. Lee, M. Kwon, G. Hasker and A. Perrig, BASE an incrementally deployable mechanism for viable IP spoofing prevention, *Proceedings of the 2nd ACM symposium on Information, computer and communications security*, 2007, Singapore.

[11] J. Arnold, O. Maennel, A. Flavel, J. McMahon and M. Roughan, Quantitative analysis of incorrectly configured bogon filter detection, ATNAC 2008. Australasian, 2008, pp. 10-15.

[12] H. Wang, D. Zhang and K.G. Shin, Detecting SYN flooding attacks, *Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, 2002.

[13] R.M. Mutebi and I.A. Rai, An integrated victim-based approach against IP packet flooding denial of service, *International Journal of Computing and ICT Research* **4**(1) (2010), 70-80.

[14] http://www.boho.or.kr/webprotect/cyberShelters/cyberShelters.do, April 2016.