Syn Flood Attack Detection and Type Distinguishing Mechanism Based on Counting Bloom Filter

Tomáš Halagan, Tomáš Kováčik(✉), Peter Trúchly, and Andrej Binder

Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava, Ilkovičova 2, 842 16 Bratislava, Slovakia
{tomas.halagan, tomas.kovacik, peter.truchly, andrej.binder}@stuba.sk

Abstract. Presented work focuses onto proposal, implementation and evaluation of the new method for detection and type identification of SYN flood (DoS) attacks. The method allows distinguishing type of detected SYN flood attacks – random, subnet or fixed. Based on Counting Bloom filter, the attack detection and identification algorithm is proposed, implemented and evaluated in KaTaLyzer network traffic monitoring tool. Proof of correctness of the approach for TCP SYN flood attack detection and type identification is provided – both in practical and theoretical manners. In practice, new module for KaTaLyzer is implemented and TCP attacks are detected, identified and network administrator is notified about them in real-time.

Keywords: DoS detection · DoS identification · Counting Bloom Filter · TCP · SYN · Flood attack · Network security

1 Introduction

Internet allows people to connect with each other in different ways. However, every new functionality, service, new way of communication, new invention designed for the benefit of humanity may pose a potentially exploitable threat which network and systems' administrators need to be aware of.

Computer network security and privacy has a lot of attention, it is currently of very high importance-various detection algorithms or protection mechanisms are implemented on various network layers, network devices and in operating systems (a good example is the widespread use of VPN networks [1] and aims to enhance security in mobile networks [12]). Despite of all mentioned facts, the very important issue remains in information about currently ongoing attack which administrators need to have as soon as possible to take an action. Development of new effective solutions to detect and provide such information is thus open case [3, 4].

Among the most common DoS attacks there are flooding attacks which exploit holes in used network protocols [5]. In our work we focus on proposal of modification

I. Khalil et al. (Eds.): ICT-EurAsia 2015 and CONFENIS 2015, LNCS 9357, pp. 30–39, 2015.
DOI: 10.1007/978-3-319-24315-3_4

of SYN flood attack detection mechanism and its implementation into KaTaLyzer. KaTaLyzer is a network traffic monitoring tool developed at STUBA [6].

Based on existing Counting Bloom Filter (CBF) mechanism, our main contribution described in this paper is modification of method utilizing CBF for attack detection. With the aim of lower memory requirements we use modified CBF structure (one vector) for storing counters of half-open TCP connections [8]. Along the possibility to detect SYN flood attack, our method allows to distinguish the type of the attack. More information about DoS SYN flood attack can be found e.g. in [7, 9].

After detection of ongoing SYN flood attack, network administrator is notified.

TCP SYN flood attacks can be distinguished into:

- Random – spoofed source IP address for each packet is generated randomly
- Subnet – spoofed source IP address is for each packet generated from specific subnet range
- Fixed– several chosen IP addresses are used.

Section 2 of this paper describes Bloom filter data structure for storing data and its modifications. In Sect. 3 description of CBF modification is presented while in Sect. 4 new S-Orthros algorithm is given. Section 5 with evaluation of proposed method is followed by discussion and paper conclusion.

2 Bloom Filter and Its Modification

In the early 70's of the 20th century, H. Burton Bloom [10] introduced new hash-coding methods, which have become the cradle of a new approach to storing data into a data structure, later called the Bloom filter. His efficient structure provides a way to reduce space required for storing data, at the cost of false-positive members. As described in [11], Bloom Filter data structure is widely used in today's internet, viruses, worms and network intruders which cause service damages with enormous economic impact. In our approach it will be modified and used for storing data about attacking IP addresses.

2.1 Bloom Filter Algorithm

Mathematics behind the Bloom Filter data structure is following:
consider a set of m elements, in our case a set of m IP addresses

$$\mathrm{IP} = \{\mathrm{ip}_1;\ \mathrm{ip}_2;\ \mathrm{ip}_3;\ \ldots \mathrm{ip}_m\}$$

The set will be after application of Bloom filter described by vector V which is n bits long, it is initially set to n zeros:

$$\mathrm{V} = (\mathrm{v}_1,\ \mathrm{v}_2, \ldots \mathrm{v}_n) = (0, 0, \ldots 0)$$

Consider k independent hash functions which are used by the Bloom filter to generate k hash values from each element from the IP set

$$K_i = h_i\left(ip_j\right), 1 \leq i \leq k, 1 \leq j \leq m$$

The hash functions output values are integers $K_i \in \{1, \ldots, n\}$ and represent index in the vector *V*. If *x-th* hash function h_x, $1 \leq x \leq k$, applied to one member of set *IP*, $ip_j \in IP$, $1 \leq j \leq m$ results in value K_x, i.e. $K_x = h_x(ip_j)$, K-th bit of the *V* vector is set to 1

$$V = (v_1, v_2, \ldots v_k, \ldots v_n) = (v_1, v_2, \ldots 1, \ldots v_n)$$

For each element $ip_i \in IP$, $1 \leq i \leq k$, K-th bit of V vector is set to 1, while $K = h_i(ip_j)$, for each $1 \leq j \leq m$, $1 \leq i \leq k$. This way *k* hash functions applied to m members of IP set change *k* bits in vector V (true, if the K index is always different). If two or more hash functions result in the same index K, the bit in vector V is not changed more than once, it is set to 1 once.

To find out whether an IP address *H* was not a member of set IP, the hash functions are applied to it and appropriate bits in vector V are checked. If even one of these bits is set to 0, *H* was not a member of IP set. Due to overlapping possibility of setting bits by hash functions to 1, Bloom filter does not provide reverse information, i.e. whether the H was a member of IP set.

According to relations among the hash functions and overlapping of their results, bits in *V* vector can be set to 1 multiple times. However, only the first setting of the bit to 1 changes the value of the bit.

2.2 Counting Bloom Filter

Assume a situation in which the elements of IP set change periodically and thus they are being inserted to and deleted from the data structure. Inserting elements is a simple process which has been described above. However, during deletion of an element from Bloom Filter data structure, we need to set the corresponding bits to zero. It is possible that this operation will affect bits which were set to 1 by hash function also for different element of the IP set. In this situation the Bloom filter no longer provides correct representation of the elements of the IP set. This problem has been solved in [2] which outlined new data structure called Counting Bloom Filter (CBF). In this structure, bits in vector V are replaced by long integers which are used as counters and each hash function has its own vector of these counters. If we want to save a track of element ip_x in the data structure, each counter corresponding value of independent hash functions will be incremented. During deletion of an element appropriate counters are decremented.

2.3 Independent Hash Functions

Finding well designed set of hash functions is important for the correct storage and distribution of elements in the CBF data structure.Performance of the hash functions is also important. In a comparative study performed by Chen and Yeung in [13], there are independent hash functions designed which have low probability of collisions. 32-bit IP address is used as a key for the hash functions.

The hash functions are defined as follows:

$$\mathrm{hi(IP)} = (\mathrm{IP} + \mathrm{IP\ mod\ pi})\ \mathrm{mod\ n}, 1 \leq \mathrm{i} \leq \mathrm{k}$$

where mod denotes the modulus operation, n is the row length of the hash table, and pi is a prime number less than n.

Following table from work of Chen and Yeung shows comparison of the proposed hash function with other known hash functions [13] (Table 1).

In our work our examination resulted in setting variables of the above mentioned function as follows: n = 1024, k = 4

Table 1. Tested hash functions [13]

Hash function	Consumed time (s)	Number of collisions
Our function	0.187	0
Robert's 32-bit function	0.188	3838
Robert's 96-bit function	0.250	0
Cruth's function	0.031	977
Hybrid function	0.328	0

3 New Proposal of Modified CBF (MCBF)

The biggest disadvantages of CBF data structure compared to Bloom Filter are:

- more memory space is needed to store data: consider k independent hash functions which require one row (vector) of counters per function. In this case $k = r$, where r is number of needed rows of counters. Next, consider n as number of counters in a row. The complexity of the space needed for stored elements in the data structure CBF can be expressed as $n*r$.
- possible overflow of counters may pose a risk especially with increased savings of elements.

Simplification of the CBF data structure is one of our contributions in this paper. Compared to CBF, we propose to use only 1 vector of counters (as in BF) where independent hash functions increment values while saving an IP address from which half-open TCP connection is initiated (SYN packet is received). The counters are decremented when a connection is fully opened from the IP address (ACK is received) (Fig. 1).

As in our attack detection approach the data structure is cleared periodically after defined time interval (see following chapter), one vector of long integer counters is

sufficient for storing IP addresses of half-open connections by incrementing and decrementing the counters. It is designed to fit the proposed solutions to detect SYN flood attack and these will be described in later section describing S-Orthros detection algorithm.

4 Detection Module S-Orthros

Our method for detecting SYN Flood attack uses MCBF data structure. Modification of the CBF data structure resulted into simplification and clarification of the solution and also the method itself. The intention is therefore the evaluation of the conditions and thus attack detection and evaluation in a given constant time interval.

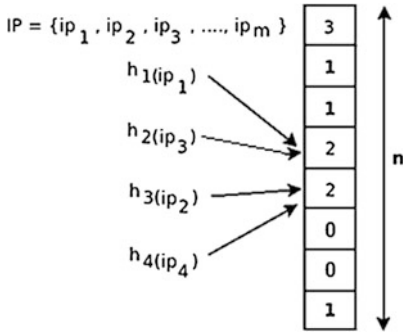

Fig. 1. Storing data in MBCF data structure using 4 hash functions and counters

4.1 S-Orthros in Nutshell

Consider the case where the detection algorithm cooperates with a measuring tool which is used to capture and analyse network traffic in real time. At the beginning, continuous process of capturing network traffic statistical data is started and runs as continuous process.

After a defined time interval (set by administrator, usually 1 min), process analysing captured network traffic data is started. In this process, also data important for S-Orthros detection algorithm are saved - source and destination IP addresses of SYN and ACK packets. These IP addresses are saved using chosen hash functions to the MCBF structure. It consists of two tables containing n long integer counters (2*1024*4B).

The first table is used to store source IP addresses, the second table stores destination IP addresses. During the analysis, the detection algorithm S-Orthros collects information about initiated connections (i.e. SYN packets) and confirmations of the connections (i.e. ACK packets). If the analysis detects a SYN packet, the MCBF data structures are incremented counters in both tables are incremented according to results of hash functions applied to IP addresses. In case of receipt of ACK confirming previous SYN packet, counters in data structures in both tables are decremented. If

there is no flood attack, the TCP handshakes are correct (i.e. number of SYN and ACK packets are the same) and data structures remain empty.

In case of a flood attack, values in MCBF structure are rising fast. Threshold of acceptable half-open connections has been according to experiences (e.g. settings of CISCO routers [19] or different operating systems) set to 50. Attack detection process checks the number of half-open connections against the threshold and alerts administrator.

The data stored in the MCBF can be analysed and distribution of values can show type of SYN flood attack - fixed, random, subnet (see following chapter).

5 Evaluation

For theoretical evaluation of our type of attack-distinguishing approach, we implemented the algorithm in spreadsheet table processor Microsoft Excel. Regarding the practical method, we used generated SYN flood DoS attacks which were detected by our new SYN flood attack-detection module implemented in KaTaLyzer. Detected attacks have proven correctness and functionality of the implemented detection algorithm.

5.1 Theoretical Evaluation

Using MCBF we are able to simulate different variations of SYN flood attacks. We have simulated 3 types Random SYN flood, Subnet SYN flood, Fixed SYN flood.

To obtain input data for simulations in Excel, it is necessary to define (in case of Fixed attack) and generate (in case of Random and Subnet attacks) IP addresses from which the hash functions calculate their values. These values are stored in the MCBF data structure.

For our simulations, an IP address is represented by numeric representation of 32 bit number. For instance, well-known address 192.168.0.1 has been calculated as follows:

$$1 * 256^0 + 0 * 256^1 + 168 * 256^2 + 192 * 256^3 = 3232235521$$

For Random and Subnet SYN flood attacks we have generated 250 000 IP addresses in Excel (they represent number of half-open connections). We chose following ranges for particular attacks:

- Random – *RANDBETWEEN(1; 4294967295)* – covers whole range of IPs
- Subnet – *RANDBETWEEN(3232235776; 3232236031)* - covers IP addresses from 192.168.1.0 to 192.168.1.255

Both of these attacks can be distinguished thanks to the typical arrangement of IP addresses stored in the used data structure. This arrangement of IP addresses has been verified on sample of 250.000 IP addresses with proposed hash functions, which are for Excel defined as follows:

MOD((D1 + MOD(D1;307));1024),
where D1 represent cell containing IP address, 307 is prime number and 1024 is length of vector.

The results of simulations are shown below in Figs. 2 and 3. As we can observe, numbers of half-open connections in MCBF during simulated Random attack have equal distribution – small amounts of half-open connections are measured for each IP address. On the other hand, Fixed attack is typical with high numbers of half-open connections for specific IP addresses. For Subnet attack it is typical that chosen range of IP addresses have relatively high amount of half-open connections.

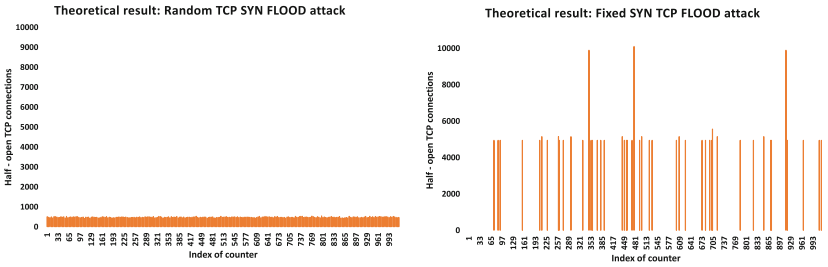

Fig. 2. Theoretical results - Random and Fixed TCP SYN Flood attacks

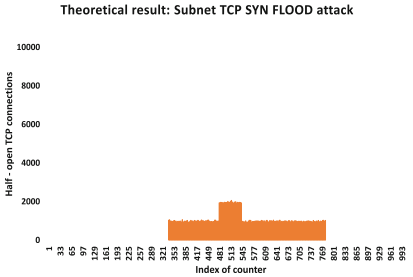

Fig. 3. Theoretical results - Subnet TCP SYN Flood attack

5.2 Practical Evaluation

There are number of tools for generating DoS attacks. Project Neptune is used to generate SYN Flood attack, it can continuously send TCP SYN packets at a rate 248 SYN packets per second [14]. For our purposes we tested hping3 tool [15] sending TCP SYN packets on port 443 to target host and sending TCP SYN packets with the ACK flag set to target host. We also used Letdown tool [16] and Ev1syn [17].

To verify the basic functionality of our proposed and into KaTaLyzer implemented detection method, a TCP SYN Flood attack has been executed using Ev1syn. As expected, S-Orthros module was able to correctly detect even weak SYN Flood attack (121 half-open connections).

Fixed Attack Identification. Fixed TCP SYN flood attack was generated from 4 IP addresses. They can be identified in the graph shown in Fig. 4 as 4 tipping points. S-Orthros module stored information about each IP address in the data structure of MCBF 4 times with 4 independent hash functions.

As expected, attack generated on real network had similar characteristics as the theoretical run of the attack – compare Figs. 2 and 4. Minor measured variations are caused by a variety of regular communications which were captured together with the attack on the server – monitored network is connected to regular network.

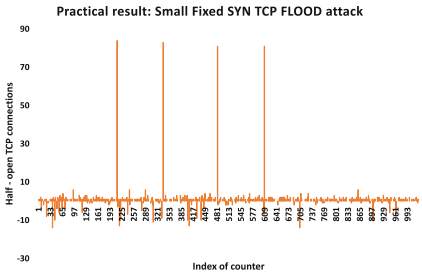

Fig. 4. Practical results – Fixed TCP SYN flood attack

Random Attack Identification. In the next step we evaluated detection of Random SYN Flood attack. The test also evaluated stability of the implemented module, as well as the stability of the measurement tool KaTaLyzer during high load. All tools used for attack generation randomly generated the source IP address for each packet, the destination IP address was the address of measuring server.

For more detailed comparison of theoretical and practical approaches we closely investigated theoretical and measured attacks (see Fig. 5). It is necessary to take into account that the algorithm used to generate random IP addresses in Excel and the algorithm used to generate random spoofed IP addresses implemented in the attack generator give us different but similar input data. Nonetheless, a way of storing data in the MCBF data structure should be retained and therefore the results from both graphs in Fig. 5 show uniformly stored data.

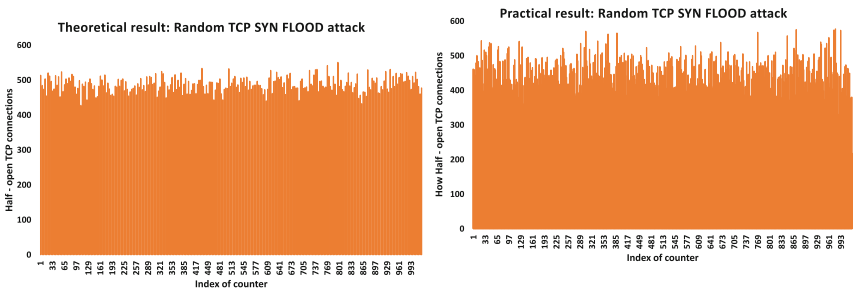

Fig. 5. Detailed Random TCP SYN flood attack - theoretical and practical results

6 Discussion

Security and protection against DoS attacks can be addressed at different levels. It is even possible to avoid such threats in operating system by simple firewall settings [18].

Nevertheless, we can still find unsecured systems and security holes through which the attack can be successfully performed. Network traffic measuring tool KaTaLyzer, which runs 24/7, allows not only to analyze and to save the network traffic statistical data. Thanks to the new implemented attack detection module it provides an additional level of network protection. Obtained results show us, that network administrator obtains almost immediate notification about ongoing SYN flood attack, thanks to which he can take the necessary steps to mitigate or eliminate the ongoing attack. The type of attack can be also distinguished.

Correctness, completeness and functionality of the proposed detection algorithms are confirmed by the results obtained by practical methods and theoretical methods described in this paper. Comparison of theoretical and practical results show similar statistical characteristics of generated attacks. We generated TCP SYN flood attack for several hours so that we not only verify the stability of the implemented module, but also the entire measuring tool KaTaLyzer in which the module was added.

7 Conclusion

We proposed fast and memory-effective method for SYN Flood DoS attack detection and type identification. It is based on modification of Counting Bloom Filter, multiple vectors of counters are replaced by one vector. Appropriate counter in the vector is incremented by new half-open TCP connection and decremented by successful TCP connection establishment. Large number of half-open TCP connections evokes SYN flood attack in progress. Without an attack, the counters remain empty.

Through modified Counting Bloom Filter, we are able to distinguish three main TCP SYN flood attacks (random, fixed, subnet) which may significantly help the network administrator to mitigate or avert the ongoing attack.

The new method has been implemented into new S-Orthros module for network monitoring tool KaTaLyzer. After detection and identification of SYN Flood DoS attack, the module informs network administrator.

Detection method has been verified by theoretical and practical methods for random, subnet and fixed SYN flood DoS attack.

Acknowledgement. This work is a result of the Research and Development Operational Program for the projects Support of Center of Excellence for Smart Technologies, Systems and Services, ITMS 26240120005 and for the projects Support of Center of Excellence for Smart Technologies, Systems and Services II, ITMS 26240120029, co-funded by ERDF. It is also a part of APVV-0258-12, VEGA 1/0708/13 and KEGA 047STU-4/2013. It is also part of Katalyzer project katalyzer.sk and initiative ngnlab.eu.

References

1. Kotuliak, I., Rybár, P., Trúchly, P.: Performance comparison of IPsec and TLS based VPN technologies. In: ICETA 2011: 9th IEEE International Conference on Emerging eLearning Technologies and Applications, October 27–28, 2011, Stará Lesná, The High Tatras, Slovakia, pp. 217–221. IEEE, Piscataway (2011). ISBN 978-1-4577-0050-7
2. Fan, L., et al.: Summary cache: A scalable wide-area web cache sharing protocol. IEEE/ACM Trans. Netw. **8**(3), 281–293 (2000)
3. Kambhampati, V. et al.: A taxonomy of capabilities based DDoS defense architectures. In: 9th IEEE/ACS International Conference on Computer Systems and Applications (AICCSA), pp. 157–164 (2011)
4. Rejimol Robinson, R.R, Thomas, C.: Evaluation of mitigation methods for distributed denial of service attacks. In: 7th IEEE Conference on Industrial Electronics and Applications (ICIEA), pp. 713–718 (2012)
5. Habib, A., Roy, D.: Steps to defend against DoS attacks. In: 12th International Conference on Computers and Information Technology, ICCIT 2009, pp. 614–619 (2009)
6. Network monitoring tool Katalyzer. http://www.katalyzer.sk/
7. CERT Advisory CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks, September 1996. http://www.cert.org/advisories/CA-1996-21.html
8. IETF RFC 793.: Transmission control protocol, September 1981. http://www.ietf.org/rfc/rfc793.txt
9. CERT Advisory CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks [CA-96.21] CERT, September 1996. http://www.cert.org/advisories/CA-1996-21.html
10. Bloom, Burton H.: Space/Time trade-offs in hash coding with allowable errors. Commun. ACM **13**(7), 422–426 (1970)
11. Tabataba, F.S., Hashemi, M.R.: Improving false positive in Bloom filter. In: 19th Iranian Conference on Electrical Engineering (ICEE), p. 1. IEEE (2011)
12. Nagy, M., Kotuliak, I.: Enhancing security in mobile data networks through end user and core network cooperation. In: MoMM 2013: The 11th International Conference on Advances in Mobile Computing and Multimedia, Vienna, Austria, pp. 253–259. ACM, New York (2013). ISBN: 978-1-4503-2106-8
13. Yeung, D., Chen, W.: Throttling spoofed syn flooding traffic at the source. Telecommunication Systems **33**(3), 47–65 (2006)
14. Cardinal, S.: Use offense to inform defense. Find flaws before the bad guys do. SANS Institute 2000 – 2012, 31 August 2014. http://pen-testing.sans.org/resources/papers/gcih/neptunec-birth-syn-flood-attacks-102303
15. Hping – Active Network Security Tool, 31 August 2014. http://www.hping.org/
16. Acri, E.: Complemento Howto (2011). http://complemento.sourceforge.net/howto/
17. Ev1Syn - A SYN Flood with Random Spoofed Source Address. http://gopherproxy.meulie.net/sdf.org/0/users/wisdomc0/code_c/ev1syn.c. Accessed 31 August 2014
18. Brouer, J.: Mitigate TCP SYN Flood Attacks with Red Hat Enterprise Linux 7 Beta, 11 April 2014. http://rhelblog.redhat.com/2014/04/11/mitigate-tcp-syn-flood-attacks-with-red-hat-enterprise-linux-7-beta/
19. Davis, P.T.: Securing and Controling CISCO Routers. CRC Press, Boca Raton (2002)