University of Plymouth

PEARL **https://pearl.plymouth.ac.uk**

Faculty of Science and Engineering School of Engineering, Computing and Mathematics

2015-09-16

FHSD: An improved IP spoof detection method for web DDoS attacks

Shiaeles, SN

http://hdl.handle.net/10026.1/12693

10.1093/comjnl/bxu007
Computer Journal

All content in PEARL is protected by copyright law. Author manuscripts are made available in accordance with publisher policies. Please cite only the published version using the details provided on the item record or document. In the absence of an open licence (e.g. Creative Commons), permissions for further reuse of content should be sought from the publisher or author.

© The British Computer Society 2014. All rights reserved.
For Permissions, please email: journals.permissions@oup.com
Advance Access publication on 21 February 2014
doi:10.1093/comjnl/bxu007

FHSD: An Improved IP Spoof Detection Method for Web DDoS Attacks

STAVROS N. SHIAELES[1] AND MARIA PAPADAKI[2]*

[1]*Department of Electrical and Computer Engineering, Democritus University of Thrace, Building A, ECE, Kimmeria Campus, Xanthi 67100, Greece*
[2]*Centre for Security, Communications and Networks Research (CSCAN), School of Computing and Mathematics, Plymouth University, Drake Circus, Plymouth PL4 8AA, UK*
**Corresponding author: maria.papadaki@plymouth.ac.uk*

Distributed denial of service (DDoS) attacks represent a significant threat for companies, affecting them on a regular basis, as reported in the 2013 Information Security Breaches Survey (Technical Report. http://www.pwc.co.uk/assets/pdf/cyber-security-2013-technical-report.pdf.). The most common target is web services, the downtime of which could lead to significant monetary costs and loss of reputation. IP spoofing is often used in DDoS attacks not only to protect the identity of offending bots but also to overcome IP-based filtering controls. This paper aims to propose a new multi-layer IP Spoofing detection mechanism, called fuzzy hybrid spoofing detector (FHSD), which is based on source MAC address, hop count, GeoIP, OS passive fingerprinting and web browser user agent. The hop count algorithm has been optimized to limit the need for continuous traceroute requests, by querying the subnet IP Address and GeoIP information instead of individual IP addresses. FHSD uses fuzzy empirical rules and fuzzy largest of maximum operator to identify offensive IPs and mitigate offending traffic. The proposed system was developed and tested against the BoNeSi DDoS emulator with encouraging results in terms of detection and performance. Specifically, FHSD analysed 10 000 packets, and correctly identified 99.99% of spoofed traffic in <5 s. It also reduced the need for traceroute requests by 97%.

Keywords: distributed denial of service attack; network anomaly; anomaly detection; hop counting; fingerprinting; spoofing detection; user agent; HCF; IP2HC mapping

Received 26 April 2013; revised 8 December 2013
Handling editor: George Loukas

1. INTRODUCTION

Denial of service (DoS) attacks continue to become more common and affect organisations on a regular basis [1]. As reported in the 2013 Information Security Breaches Survey, 39% of large businesses (up from 30% last year), and 23% of small businesses (up from 15% last year) have been affected in the last year [1, 2]. Based on the same survey, 19% of respondents reported a frequency of at least one DoS incident per week, whereas a smaller minority of 4% experienced hundreds of such attacks every day. The cost of a distributed DoS (DDoS) attack is substantial enough to necessitate the need for detection and mitigation, as according to [3], more than half of respondents (65%) experienced average costs per incident to be up to $10K per hour. A further 35% reported cost of over $10K per hour, and a combined 34% experienced loss of over $50K per hour. The direct monetary cost is of course not the only impact of DDoS, as affected companies could suffer from long-term effects, such as loss of reputation, loss of revenue, poor customer experience and eventually even job losses. According to a research by the Yankee Group, a mid-size enterprise with an annual revenue of $10 million would lose an additional $20 000 (0.02% of revenue) in the longer term [3].

In order to work towards detecting and mitigating DDoS attacks, it is important to understand their characteristics first. According to [4], the most common target is unprotected websites (86%), but they also tend to affect DNS (70%), email (31%), IP telephony (17%) and even IRC (9%) services. The most common attack vector for web services is HTTP GET (76%), followed by more sophisticated tools such as LOIC, HOIC, XOIC, PyLoris, Slowloris, Apache Killer and SlowPost [3]. As a result, the focus of this work is web-based

Downloaded from https://academic.oup.com/comjnl/article-abstract/58/4/892/336014 by University of Plymouth user on 30 October 2018

DDoS attacks, which are defined as DDoS attacks targeting web-related ports, such as 80/tcp, 443/tcp, etc.

A common defence mechanism against DDoS attacks is to block the offending source IPs. However, attacks have evolved to employ IP Spoofing, mainly as a way to defeat such mechanisms [5]. Also, as [6] reveals, bots often utilize random spoofing, subnet spoofing or fixed spoofing in DDoS attacks to hide their identity and make mitigating DDoS attack harder. Although ingress and egress filtering can help significantly towards minimizing the problem, the potential for IP spoofing still exists [7]. According to the MIT Spoofer Project, which provides an aggregate view of ingress, egress filtering and IP spoofing on the Internet, 23% of autonomous systems and 16.8% of IP addresses are spoofable; this means that an estimated 560 million out of 3.32 billion IP addresses can still be spoofed [8].

As such, the aim of this paper is to propose an IP spoofing detection model for web-based DDoS attacks. The proposed work is an extension of our previous research [9], which proposed a DDoS detection mechanism based on fuzzy estimator on the mean time between network events. The inability to distinguish spoofed IPs and remove false positives generated by spoofed traffic was a limitation of our method and hence it is the topic that the present paper aims to expound. The following outline presents the main topics that are covered in this paper. Section 2 provides a critical review of existing research in the detection of IP spoofing and specifies how the proposed approach differs from the reviewed methods. Section 3 presents the main concepts behind the fuzzy hybrid spoofing detector (FHSD) model. Section 4 then proceeds to describe the prototype implementation and the experimental design that was adopted to test FHSD. Section 5 presents the experimental results from our work, and Section 6 discusses the significance of these results. Section 7 identifies the limitations of the proposed method, and is followed by Section 8, which concludes the paper and discusses future work in the area.

2. RELATED WORK

A considerable amount of literature has been published on identifying spoofed IPs in DDoS attacks. These methods can be divided into two categories: router based and host based [7]. The main difference between these is that the former needs routers software modification, whereas the latter can run on an end host as a program.

Pi and StackPi [5, 10] is a router-based approach, which introduces a new packet marking mechanism where a fingerprint is embedded in each packet to identify the path it takes through the Internet. Following a similar approach, Refs [11] and [12] have tried to detect spoofed IPs at the source network based on their arrival rate threshold and at a victim network by marking spoof packets based on the IP source arrival rate using their respective TTL value. Using cryptographic techniques to encrypt hop count and router to maintain the hop count to IP address tables, Ref. [13] has also tried to defend against spoof IPs in a DDoS attack. In addition, a novel defence mechanism was proposed by [14]; this new mechanism makes use of the edge routers that connect end hosts to the Internet to store and detect whether the outgoing SYN, ACK or incoming SYN/ACK segment is valid. This is accomplished by maintaining a mapping table of the outgoing SYN segments and incoming SYN/ACK segments and by establishing the destination and source IP address database. All these ideas are really interesting and promising, but they are difficult to implement in real life, as they require modifications of networking infrastructure on a global scale.

Host-based approaches have also attracted significant interest by research communities. Wang *et al.* [15] were the first to propose a novel hop count-based filter (HCF) in the end system that builds an accurate IP-to-hop count (IP2HC) mapping table. The initial IP2HC was created using traceroute and was from actual hop-count distributions. Based on the IP2HC table, they compared the arriving TTL values to identify spoofed IPs. For example, if the arriving TTL was 60, the assumption would be that the initial TTL was 64, and the source IP was 4 hops away. A selection of concurrent traffic from different networks, but with exactly the same arriving TTL, would indicate a higher probability of spoofed traffic. Similarly, if the traceroute results reveal different hop count, this would also suggest spoofed traffic. They included a secure mechanism to update the IP2HC mapping table, and eventually protect it against poisoning attacks as well as take into account changes in dynamic network conditions. Although HCF was a significant first step, it had some limitations. First, it used strict TTL values, without margins for error, which made it prone to false positives and false negatives [16]. Also it did not check the OS of the source IP to validate the assumed initial TTL value. Continuing the example above, where the assumed initial TTL was 64 (the default initial TTL for Linux), it would be beneficial if the O/S of the packet was determined to validate the result. Furthermore, the method is memory and network intensive, which lowers performance as well as its resistance to a DDoS attack. DHCF is an improved version of HCF, as it adopts a distributed model and has the advantage of overcoming the problems of exhausting network bandwidth and host resources at a single location [17]. However, it would be worth investigating whether alternative approaches with less memory and network intensive designs could potentially alleviate the problem. A probabilistic model was proposed by Swain and Sahoo, who managed to reduce the computation and memory requirements of HCF, but they still have the low detection problems of the initial method [18].

Wu and Chen [19] moved beyond the IP layer to improve detection of IP spoofing by adopting a multi-layer approach. They used HCF to block the majority of spoofed traffic and then a SYN Proxy Firewall on transmission layer to filter TCP half-open connections. The last step was to limit application layer DDoS traffic that uses legitimate HTTP requests.

Downloaded from https://academic.oup.com/comjnl/article-abstract/58/4/892/336014 by University of Plymouth user on 30 October 2018

The three-layer inspection manages to improve detection, but the paper does not specify how legitimate HTTP requests are distinguished from malicious ones. Also, the inherent limitations of HCF were not addressed. Zhang *et al.* [16] have also adopted a multi-layer approach, by using an improved version of HCF, SYN cookies and a SYN proxy. The new method is called hop count proxy (HCP) and it overcomes HCF's problem of strict TTL values by applying a wider TTL threshold. Also, a SYN proxy and SYN cookies are used to filter out malicious TCP half-open connections. HCP regularly updates the IP2HC mapping table, when not under attack. Another drawback of HCP is that it has some issues with machines behind NAT boxes, resulting in faulty results. Moreover, O/S information is not used to validate the arriving TTL, which increases the risk for false negatives. Finally, the method is limited to the network and transport layers only, and not to the application layer; hence, it is more suitable as a SYN attack DDoS mitigation method.

Apart from adopting multi-layer approaches, Covarrubias-Rodriguez *et al.* [20] have tried to improve detection by using fuzzy logic along with HCF to setup a flexible threshold of decision. Their method will modify the routing table every time there is a change in hop count (HC) tables. However, the problems associated with HCF are still present.

To overcome the problems of router implementation, the proposed method focuses on end host systems. It also adopts a multi-layer approach, by focusing on the link-layer, network, transport and application layers, which have shown improved detection results. The novel contribution of this paper is that it explores the extent to which additional metrics, such as source MAC address, OS information, GeoIP or web browser header information (user agent) can help improve detection of IP Spoofing. Finally, the proposed research also attempts to optimize performance, to allow the detection system to operate in DDoS attack conditions.

3. FUZZY HYBRID SPOOFING DETECTOR CONCEPTUAL MODEL

The proposed FHSD adopts a multi-layer approach to provide an efficient IP spoofing detection mechanism that is able to run under attack conditions. Therefore, the proposed approach needs to meet the following operational requirements:

(i) Multi-layer approach based on source MAC address, hop count, passive OS fingerprinting, HTTP user agent, and HTTP request method.
(ii) Improve detection by cross checking hop count with passive OS fingerprinting results and HTTP user Agent.
(iii) Minimize network and resource requirements for repeated traceroute queries by considering GeoIP and subnet address, rather than queries for single IP addresses.
(iv) Take into account changing network conditions and incomplete results by adopting flexible TTL values, along with GeoIP and subnet information for hop counting.

The proposed hybrid multi-layer approach considers as input a large selection of metrics, such as source MAC address, hop count, passive OS fingerprinting, HTTP user agent, and HTTP request method. The rationale for selecting source MAC address stems from [21], which recognises the potential of pairing MAC and IP addresses to control IP spoofing. Therefore, the proposed work aims to test this hypothesis. Still, it is recognised that the applicability of having the sender's MAC address is rather limited, as it can only apply to local network addresses. Hence it is imperative that the proposed method is able to function regardless of whether the sender MAC Addresses are available. The reason behind using passive OS fingerprinting, and HTTP user agent is to allow cross-checking of hop-count and HTTP user agent with passive OS fingerprinting to lower false positives and false negatives. Changes in user agent requests and user request methods (POST, GET) are also considered to signify illegitimate HTTP traffic. This is based on the assumption that legitimate HTTP requests will have lower variability than abnormal traffic [22]. Finally, calculating hop count is influenced by previous work on HCF and HCP (as discussed in Section 2). In this case, the hop count method is optimized to reduce the number of slow and sometimes incomplete traceroute queries, by looking up Class C subnet addresses, rather than individual IP addresses. Also, GeoIP information provides an extra dimension on the geographical location of a subnet. The hop count method also adopts flexible TTL values, to take into account the changing network conditions.

Figure 1 depicts the network flow diagram of the proposed model. According to Fig. 1, the FHSD receives web traffic

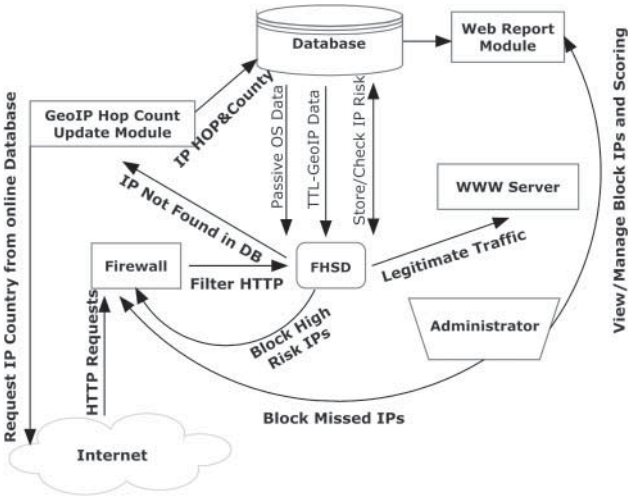


FIGURE 1. Network flow datagram of our proposed method.

Downloaded from https://academic.oup.com/comjnl/article-abstract/58/4/892/336014 by University of Plymouth user on 30 October 2018

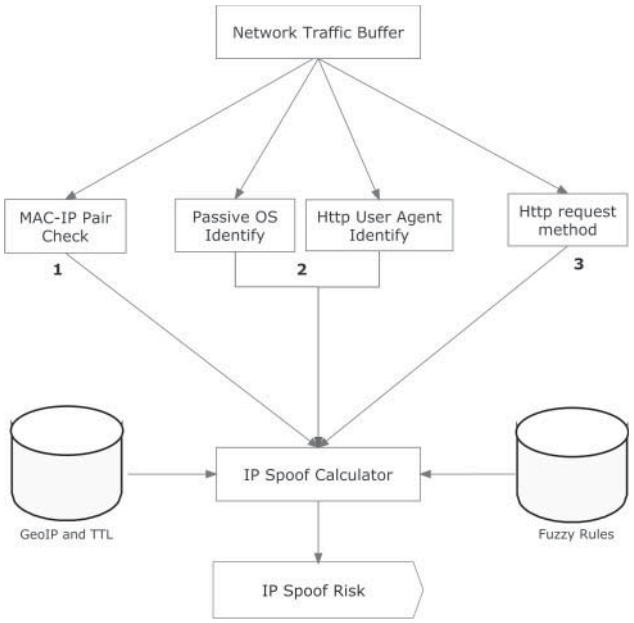


FIGURE 2. FHSD module steps.

for inspection from the firewall. FHSD then retrieves hop count information from the GeoIP Hop Count Update module, which is responsible for the estimation of hop count and GeoIP information. It initially checks whether there is an existing entry in the database for either the IP address or the Class C subnet, before initiating a GeoIP hop count query on the Internet. Once an answer is provided, the database is updated and the relevant information is passed to FHSD, which in turn calculates the IP risk for each IP address. The IP risk is saved in the database, and it is used to distinguish legitimate traffic. When the IP risk is HIGH, FHSD automatically assigns a firewall rule to reject traffic from this IP address, whereas legitimate traffic is allowed to progress to the web server. FHSD can be configured via a Web Report module, which provides configuration and logging functionality. The network administrator is able to monitor the results of the FHSD scoring using the Web Report module. They can also issue blocking commands directly to the firewall, e.g. when FHSD misses malicious spoofed IPs that need to be blocked.

Figure 2 illustrates the core modules of FHSD, where connection flows are buffered before they are passed for analysis. FHSD passes data to the analysis modules; either once the number of connections exceeds a certain threshold or after a specified amount of time elapses. Both metrics can be configurable, and in the present paper, a threshold of 10,000 connections and a time threshold of 2 s is assumed (please see further justification about this threshold in Section 6). The buffer extracts the following data from raw traffic: (a) IP source, (b) source MAC address, (c) IP TTL, (d) HTTP user agent and (e) HTTP request method.

TABLE 1. Operating systems TTL values.

Operating system	TCP	UDP	ICMP
Linux	64	64	255
FreeBSD	64	64	255
Mac OS X	64	64	255
Solaris	255	255	255
Windows 95/98/ME	32	32	255
Windows XP,7,8, 2003, 2008	128	128	255

Once buffer data are passed for analysis, three simultaneous processes start. The first process starts with MAC address and IP pairing. This process checks data according to the list of MAC address of local systems, to detect compromised hosts in the local network that act as zombies. The second process uses passive OS fingerprints and compares them with the operating system information that is retrieved from the user agent string. If the two values are equal, the result is set to 0; otherwise, it is 1 until the IP is changed. Next, the comparison continues through the TTL. Table 1 shows the default initial TTL values of operating systems that were considered, according to the results from the second process [23]. After initial TTL is set, the program checks for IP Hops. If it finds the hops for the particular IP, it uses it to find the difference between initial TTL and hop count. If the results are incomplete, it uses the subnet address instead or the country and city, and considers TTL boundaries of ± 2, as per Refs [16] and [24]. This calculated TTL is compared with the TTL value reported in the network data to detect inconsistencies and count the number of times that they change. The variability of TTL in a normal session is usually very low, where the TTL value largely stays unchanged, or sometimes moves up or down to 1 or 2 hops. Finally, the third process counts user agent changes and frequency of user request methods (POST, GET). Then, the results are collected and passed from a fuzzy rule set, as depicted in Fig. 3. For the input membership function, the triangular membership along with trapezoid function is chosen (Fig. 4). These were chosen due to their extensive use in similar research involving fuzzy logic and intrusion detection. The most notable example is that of [20], which used triangular membership to mitigate distributed DoS. Gomez and Dipankar [25] also used triangular membership for network anomaly detection. Similarly, Ref. [26] used triangular membership as well to identify port scanning, denial of service attacks, backdoors and Trojan horse attacks. It is important to note the possibility that other membership functions could be equally suitable, or yield even better results. Selecting the most efficient function is out of the scope of the proposed work, and it could in fact be a topic for exploration in the future. Primary aim of this paper is to explore the benefits and basic concepts of an improved IP spoofing detection mechanism, and in this context, we opted for well-tried and tested membership functions that work.

Downloaded from https://academic.oup.com/comjnl/article-abstract/58/4/892/336014 by University of Plymouth user on 30 October 2018

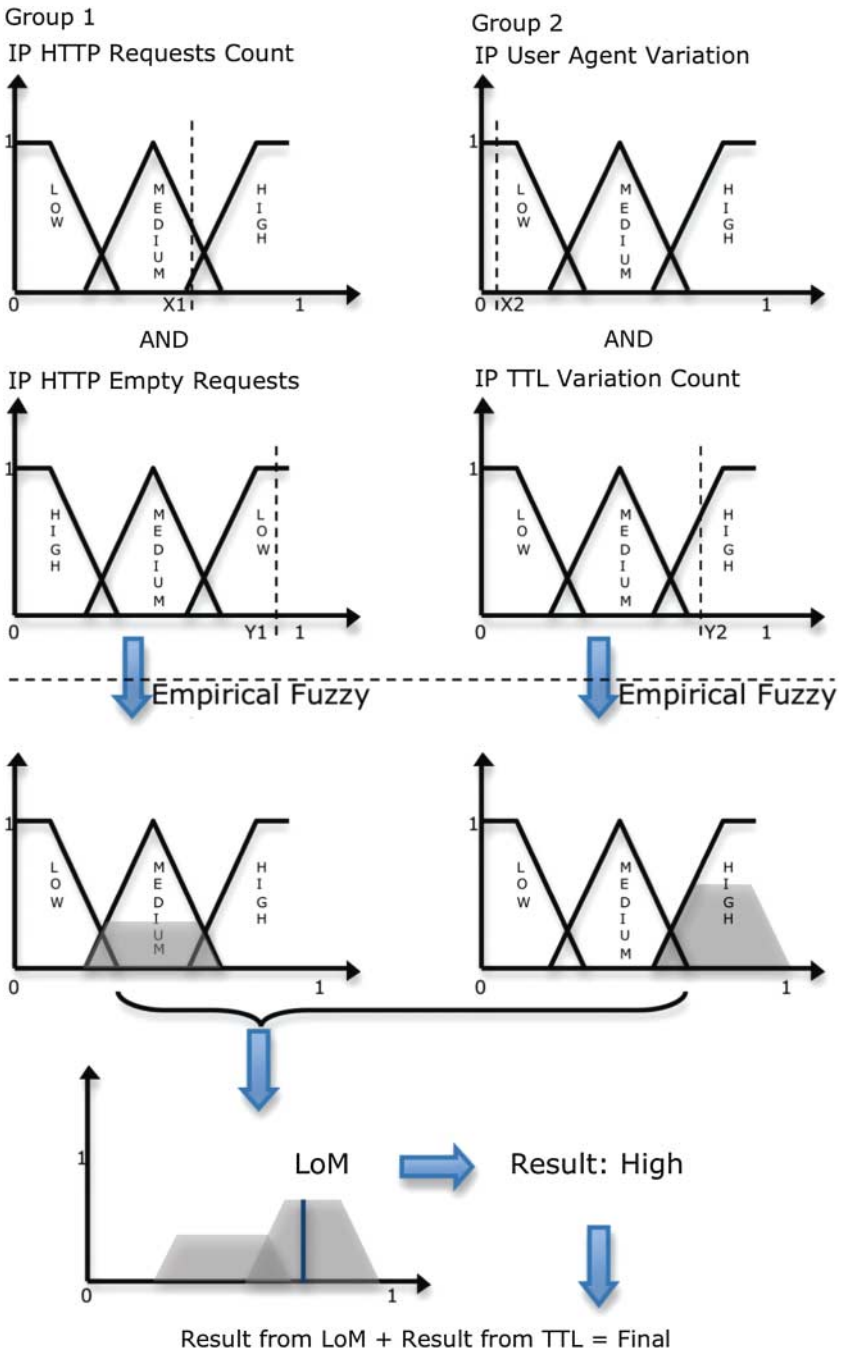


FIGURE 3. Fuzzy with empirical rules method used.

The inputs were defined on a domain interval of 0–1. Each domain, except TTL result and p0f result that are Boolean, was divided into three regions of low, medium and high as shown in Fig. 3, with the values given in Table 2. Note that Table 2 values can be changed according to the needs of the domain or the dataset. All input domains are normalized to the same input range. The rules of the fuzzy system were constructed with the fuzzy input set. Fuzzy rules are written using empirical network administrator experience. For the output, these rules are combined with largest of maximum (LoM) operator.

To best understand these empirical rules, an IP attack example is shown below.

Fuzzy IP HTTP requests count Number = IP HTTP requests count/TOTAL IP COUNTS,

Downloaded from https://academic.oup.com/comjnl/article-abstract/58/4/892/336014 by University of Plymouth user on 30 October 2018

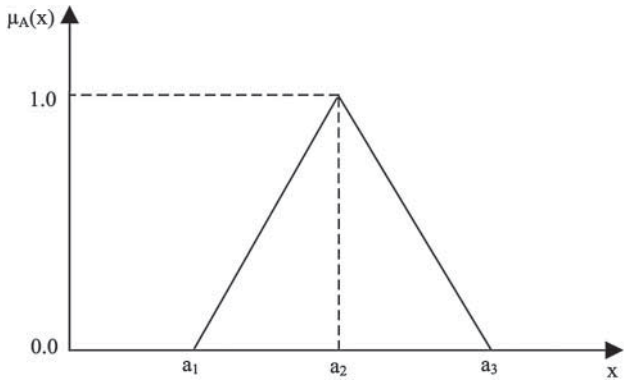


FIGURE 4. Fuzzy triangular membership function.

TABLE 2. Range of input.

Linguistic variable	Fuzzy number
Low	0,0.1,0.2
Medium	0.16,0.3,0.4
High	0.36,0.7,1

Fuzzy IP HTTP empty requests count Number = IP HTTP empty requests count/TOTAL IP COUNTS,

Fuzzy User Agent variation count Number = User Agent variation count/TOTAL IP COUNTS

Fuzzy IP TTL variation count Number = IP TTL variation count/TOTAL IP COUNTS.

The result of each variable is a number. This number is checked in the triangular membership function to find the risk that is belonging. Then these results are passing from two rules:

Rule 1:

IF (IP HTTP requests count == Low) AND (IP HTTP empty requests count == Medium)

THEN "IP RISK" == Medium

Rule 2:

IF (IP user agent variation count == Medium) AND (IP TTL variation count == High)

THEN "IP RISK" == High

The result of the two rules is passed to LoM operator that will report the crisp number of the output, using also triangular membership function. The crisp number of the output can be used with other systems that we develop in order to compare the results and have a more clear output of IP risk. In this system, if the LoM is in the high area, the output is marked as High. After that, the output result of IP risk is weighted with the TTL binary variable, which takes two values: 0 if it is OK according to hop count and 1 if not. All this combination produces the final IP risk. If the TTL is equal to 1, then this is also High, so in combination with the High from the LoM, it will report the system as High in the final IP risk.

TABLE 3. Group 1 empirical fuzzy if–then rules.

IP HTTP request/ IP HTTP empty request	Low	Medium	High
Low	Low	Medium	High
Medium	Low	Medium	High
High	Low	Medium	High

TABLE 4. Group 2 empirical fuzzy if–then rules.

IP User agent variation count/ IP TTL variation count	Low	Medium	High
Low	Low	Medium	High
Medium	Medium	Medium	High
High	High	High	High

TABLE 5. Final result fuzzy if–then rules.

IP LoM result/ IP TTL status	Low	Medium	High
0	Low	Low	Medium
1	Medium	High	High

The empirical fuzzy rules used in our model are shown in Tables 3–5 while Fig. 3 depicts/outlines a detailed representation of the fuzzy rules procedure.

4. A PROTOTYPE IMPLEMENTATION OF FHSD AND EXPERIMENTAL DESIGN

Based on the conceptual model presented in Section 3, the paper proceeds to present a prototype implementation of FHSD and the experimental design that was used to investigate its detection efficiency. The prototype implementation uses binary files instead of a database in order to store data. This was in the interest of time and simplicity. Extending FHSD to use a database would be feasible, and could speed up the result process even further. Nonetheless, even with the use of binary files, the results process is already fast enough. The computational time for 10 000 packets in an Intel Quad Core Machine with 8 GB RAM and 1TB 7200-rpm hard disk was <5 s. Therefore, using binary files was deemed suitable for a proof of concept tool.

The FHSD prototype prepossesses pcap files with tshark and it exports values in CSV format. Such values include IP source, source MAC address, TTL, user agent and request method. Then the collected web traffic for the 10 000 IPs, which correspond

Downloaded from https://academic.oup.com/comjnl/article-abstract/58/4/892/336014 by University of Plymouth user on 30 October 2018

to ∼1 or 2 s of traffic, is passed from p0f v3.0 to identify the OS per IP. The result of p0f is passed to FHSD along with traceroute data, pre-processed GeoIP data, and the tshark file. As it shows in Fig. 2, MAC address and IP pairing is initially checked against the list of local MAC addresses and then data are sorted per IP and the IPs are checked against p0f exported file and user agent. If the two values are equal, the p0f flag is set to 0. Otherwise, the p0f flag gets the value of 1 until the IP is changed. Next, the comparison continues through the TTL using the user agent string to setup the initial TTL of operating system and Table 1. After the initial TTL is set, the program checks for IP hops in the traceroute and GeoIP file. If it finds the hops for the particular IP, it uses this value to find the difference between initial TTL and hop count. If the result is incomplete, it uses the Class C subnet address to find the difference with ±2 boundaries. This value is compared with the TTL value from the network TCP stream, and if different, it counts the number of times the TTL changes. Similarly, FHSD also counts user agent changes and user requests (POST, GET). Then the results are passed to a fuzzy ruleset, using Mamdani Method [27] (Fig. 4) and it outputs the IP risk score.

As part of the experimental evaluation, FHSD is tested against normal and illegitimate web traffic. The DDoS tool BoNeSi [28], a network traffic generator for different protocol types, was used to generate the illegitimate traffic. It has the ability to control with various parameters the attributes of the created packets and connections as, for example, send rate, payload size or even all attributes can be randomized. Also in HTTP mode attack, it behaves as a real Botnet. This is also the reason behind the choice of BoNeSi, as it can emulate real bot behaviour. According to SecurityTube, BoNeSi has been used to simulate large Botnets on an AMD Opteron with 2 GHz, generating up to 150 000 packets per second [29]. BoNesi was also successful against the state-of-the-art commercial DDoS mitigation systems, which it managed to either crash or bypass detection [28]. Hence, BoNeSi was used in this case as an alternative to botnets, as a way to overcome the practical difficulty and ethical problems of obtaining or renting real bot software.

BoNeSi HTTP request attack was used against an Apache 2.2.20 web server, which hosts PHP dynamic web pages. In order to make the HTTP requests more realistic, 45/24 IP subnet ranges (listed in Table 6) and 10 different user agents along with operating system (listed in Table 7) were used. BoNeSi then produced spoof IPs within the IP range of each subnet. For example, the first IP subnet triggered BoNeSi to start sending requests from random IPs within the range of 1.2.3.1–1.2.3.254. So the total number of distinct spoof IPs that could reach the web server would be 11385 (the product of 45 subnets and 253 IPs per subnet). Also, the TTL values and source ports of the attack IPs were generated randomly, in an attempt to make the spoof data more realistic. As for the selected sample of User Agent strings that are shown in Table 7, it was obtained from [30].

Although the word ‘Mozilla’ appears in all entries, these actually represent a wide selection of browsers, such as Internet Explorer, Opera, Safari and Chrome, and not just Mozilla

TABLE 6. List of spoof IP subnets used in BoNeSi.

BoNeSi spoof IP subnet used		
0.1.125.174/24	0.1.91.98/24	0.10.138.194/24
0.10.180.83/24	0.100.194.86/24	0.100.4.147/24
0.101.118.61/24	0.101.253.178/24	0.101.79.119/24
76.92.199.150/24	76.93.12.254/24	76.94.211.44/24
76.94.27.31/24	76.94.67.128/24	76.96.122.8/24
76.98.67.241/24	76.99.14.245/24	77.10.210.127/24
77.101.139.127/24	77.101.185.177/24	77.103.220.1/24
77.104.169.154/24	77.105.240.217/24	77.106.168.16/24
77.177.67.106/24	77.178.90.218/24	77.26.237.147/24
77.26.242.166/24	77.27.51.26/24	77.29.51.117/24
77.29.96.223/24	99.95.56.17/24	100.12.130.16/24
100.212.131.16/24	100.212.132.16/24	100.212.133.16/24
100.212.134.16/24	100.212.135.16/24	100.212.136.16/24
100.212.137.16/24	100.212.138.16/24	100.212.139.16/24
100.212.140.16/24	100.212.141.16/24	100.212.142.16/24

TABLE 7. List of user agent strings used in BoNeSi.

User Agent used in BoNeSi import file
Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; WOW64; Trident/6.0)
Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; Trident/6.0)
Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; Trident/5.0)
Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; Trident/4.0; InfoPath.2; SV1; .NET CLR 2.0.50727; WOW64)
Mozilla/5.0 (compatible; MSIE 10.0; Macintosh; Intel Mac OS X 10_7_3; Trident/6.0)
Mozilla/4.0 (compatible; MSIE 10.0; Windows NT 6.1; Trident/5.0)
Mozilla/1.22 (compatible; MSIE 10.0; Windows 3.1)
Mozilla/5.0 (Windows; U; MSIE 9.0; Windows NT 9.0; en-US)
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.17 (KHTML, like Gecko) Chrome/24.0.1312.60 Safari/537.17

Downloaded from https://academic.oup.com/comjnl/article-abstract/58/4/892/336014 by University of Plymouth user on 30 October 2018

browsers. According to [30], most browsers include the string 'Mozilla' in their user agent string.

A pseudocode of the implementation is shown below:

```
   P = SortPacketsPerIP();
FOR each packet in P
        IP = GetIPfromPacket(P);
        OP = CheckOperatingSystem(P);
        Browser = CheckBrowser(P);
        UserAgentCount = CountUserAgentChanges(P);
        TTL = CheckTTL(IP);
        If (TTL found in database)
             TTLVALUE=TTL
        Else
        TTLVALUE=GEOIP_LOOKUP_WITH_SUBNET_
CHECK(IP);
        IF (TTLVALUE found)
        AddtoDatabase(IP);
        Return TTLVALUE;
        Else
        Mark As Unknown;
              Traceroute(IP) in the background
          AddtoDatabase(IP);
        END IF
        END IF
        CountPG = Count Post and Get Requests(P);
        CountTTLVar = Count_TTL_Changes(P);
END FOR
FinalResult_Per_IP = Summarize_All_Values();
```

The experiments considered four datasets: one dataset with only legitimate users' traffic; the DARPA LLDOS Inside 1.0 dataset; and two datasets with legitimate users traffic along with BoNeSi spoof DDoS attack traffic. The first dataset was legitimate users traffic and was exported from a busy job seeking website used also in [9]. It contained 30 000 network packets over a period of 4 min and 157 unique IP addresses. The second dataset was an attack dataset and was exported using a virtual machine as web server and another one as attacker with BoNeSi. The two machines resided on the same host and the web server machine could be accessed from the Internet. The dataset contained 180000 network packets over a period of 3 min, and it involved 15 legitimate IPs and 2546 spoof IPs. BoNeSi generated around 115 000 packets of spoofed HTTP traffic, using addresses within a set of IP subnets, as shown in Table 6. This option was enabled with the max-bot flag.

The third dataset was also an attack dataset and was exported from the job seeking website used in [9]. The dataset contained 1 600 000 network packets over a period of 4 min. During the capture of legitimate users sessions on this website, a DDoS attack was launched from two different locations using BoNeSi. BoNeSi was configured to use a list of spoof IP addresses, which are shown in Table 6. The max-bot flag was not used in BoNeSi, in this dataset. For user agents, Table 7 was used. BoNeSi generated around 1 550 000 packets of attacking traffic involving 170 distinct source IPs, where the 45 were the attack IPs of Table 6.

The last and forth dataset was DARPA LLDOS Inside 1.0 dataset Inside [31]. This dataset contained 649 787 packets over a period of 3 h 14 min. The HTTP sessions in this dataset are limited.

5. RESULTS

Initially, the DARPA LLDOS 1.0 Inside data set was used [31]. According to DARPA LLDOS 1.0 scenario, an attacker compromises three machines inside the local network. These hosts are mil with IP 172.16.115.20, pascal with IP 172.16.112.50 and locke with IP 172.16.112.10. Using all the three compromised hosts and spoof IPs, the attacker attacks victim IP 131.84.1.31 for 5 s. Our program identifies this attack in the first stage, using MAC address pairing, so the second stage was not needed. Also the second stage was not possible to be used in DARPA because it does not contain web traffic. Specifically, user agents are missing from many IPs.

The second test was done using the dataset from the two virtual machines on the same host. According to this scenario, the attacker machine had BoNeSi installed in order to spoof IPs and attack the second machine's web server. Also this experiment helped to identify the spoofing IPs from MAC addresses that were changing.

Next step was to test the third and fourth datasets that were more realistic and that could happen in live situations. The third dataset dealt with attacking a job seeking website (also used in [9]) from two geographically different locations using BoNeSi with spoofed IPs. Our method successfully found all the spoof IPs in the second stage because the first stage of MAC filter cannot be used in Internet traffic.

Lastly, the fourth dataset was legitimate data from the job seeking website as well. In this scenario, the success rate was 99.99%. There were some minor misclassifications, false positive values set as Medium that should have been set as Low. There were no IPs classified in the High state, which is a reasonable expectation given that the dataset was legitimate user data.

Figure 5 shows the number of attack packets arriving over time, whereas Fig. 6 depicts the number of normal packets arriving over time. Both figures show a different pattern for normal vs. attacking traffic. Specifically, the volume of distinct attacking IPs is much higher, than normal IPs.

Figure 7 is a screenshot of the prototype, showing the outcome of the IP risk classification, using the first and second stages.

6. DISCUSSION

The DARPA DDoS dataset is based upon DDoS attacks from compromised hosts in the Local LAN. Also the attack is not

Downloaded from https://academic.oup.com/comjnl/article-abstract/58/4/892/336014 by University of Plymouth user on 30 October 2018

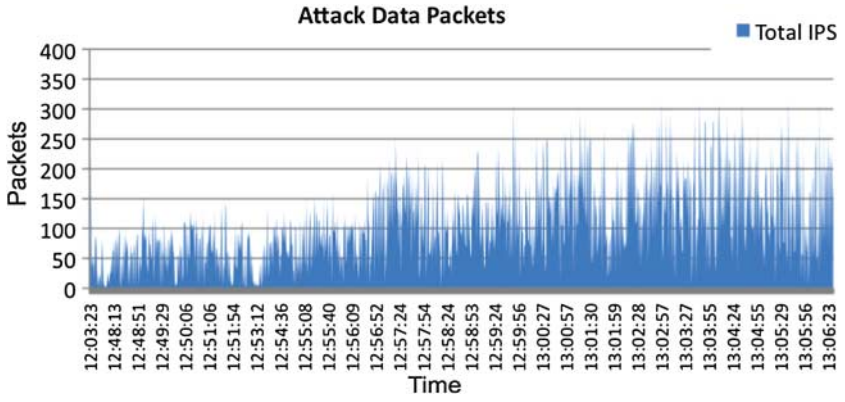


FIGURE 5. Attack data packets per time.

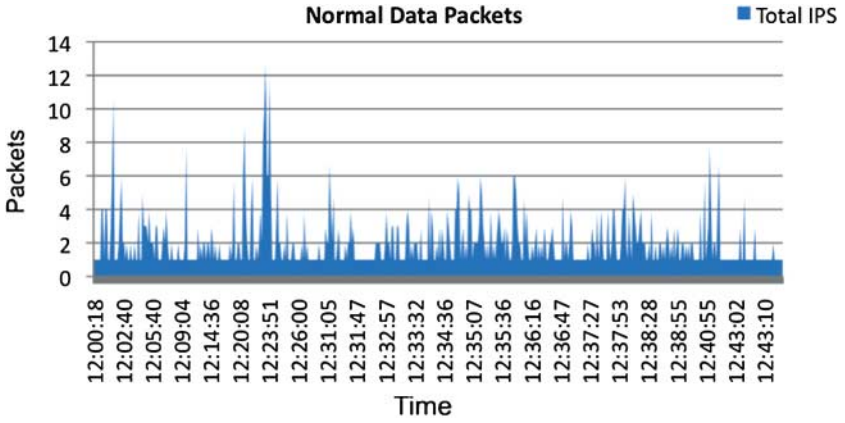


FIGURE 6. Normal data packets per time.

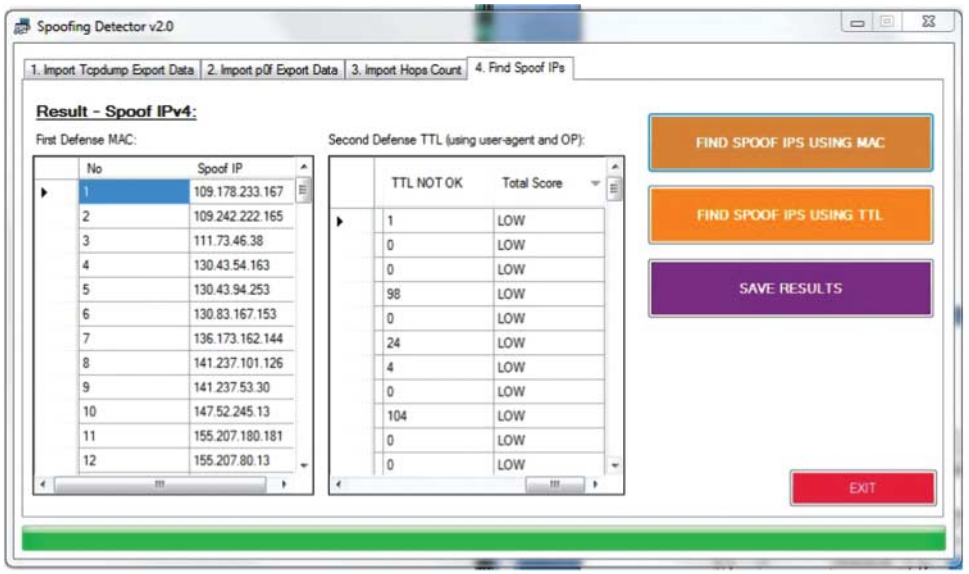


FIGURE 7. Program results.

Downloaded from https://academic.oup.com/comjnl/article-abstract/58/4/892/336014 by University of Plymouth user on 30 October 2018

specific for web server so there was not much information about the user agent and some other features that are needed for our method to find the offensive IPs in the second stage of check. Moreover, the IP and MAC pairing is changing during the DDoS attack using the spoof IPs and having this information in the dataset makes it easier to find spoof IPs. In a real DDoS attack against a web server, the MAC address of the attacker would not be available at the victim side. In the victim site one would only see the MAC address of the router that forwards the packets. As a result, the DARPA DDoS dataset was not considered appropriate to export correct results for the proposed method. What is more, the second dataset allowed us to successfully identify the spoof IPs with two ways: First with the MAC-IP pair changes and secondly using the hop counting, TTL and user agent filtering method. The third dataset was a real DDoS scenario. The aim was to collect data and analyse them to see if the proposed method was effective. Using a hop counting table for some of the spoofing IPs, not all of them, geographical locations and OS fingerprinting techniques used by p0f in comparison with user agent, the proposed method showed encouraging results by identifying 99.99% of spoof IPs. Similar results were produced in the fourth dataset that was live data capture using tcpdump from the job seeking website. This particular dataset did not have attack IPs, and our method corresponded correctly to this scenario, but with a few false positives in the state of Medium score. The reason for this false positive was the use of proxy server in the settings of the user browser that visited our web site; the initial TTL was 64, which is the initial value of a Linux operating system, but the user agent reported Windows operating system, which has initial TTL of 128. Thus, the system reports it as anomaly, which is correct.

FHSD provides improved results in comparison to HCF and other approaches. The additional metrics, such as HTTP request method, user agent and IP TTL value change, proved to be particularly valuable in accurate classifications, without introducing significant overhead. This is evident by the reasonable system performance. A major factor contributing towards a robust solution was the optimization of hop count queries by introducing the GeoIP and subnet TTL. By reducing the need for repeated traceroute requests, the number of traceroute queries was 45 out of 2000, which is approximately a 97% reduction in comparison to HCF, which is a significant improvement of network usage.

Figure 8 shows a comparison between FHSD and HCF, based on detection rate and false positive rate. The detection rate for spoof IPs in FHSD is 100% even though we have some false positive IPs in the rate of 2%. The CPU usage was between 37 and 52%. According to [32], the corresponding figures for HCF are 90% detection rate and 8% false positive rate. It should be noted that the results from [32] are based on a different dataset; therefore, it is not possible to perform a direct comparison of the two methods. Similarly, other alternative methods to HCF base their findings on private datasets, making a direct comparison to FHSD impossible. Wu and Chen [19] shows the most promising results with their three-layer approach using SYN proxy, reporting 98.93% detection rate. No performance data were published though in their work.

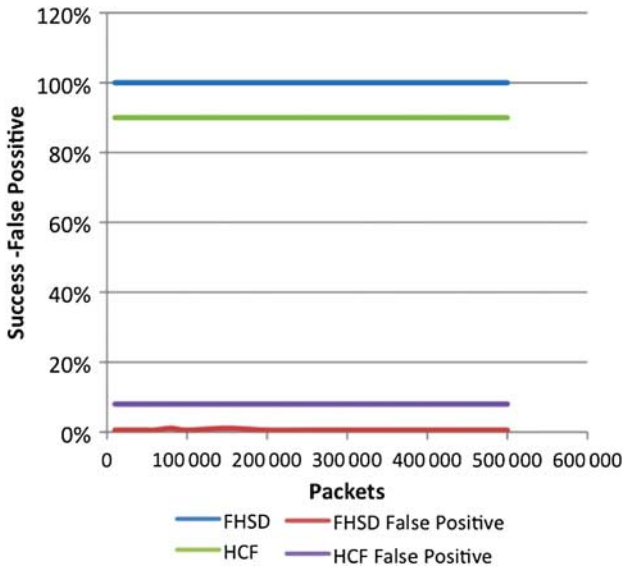


FIGURE 8. FHSD and HCF comparison based on detection rate and false positive rate.

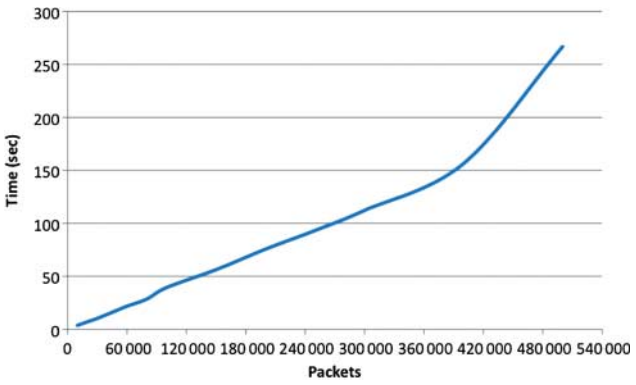


FIGURE 9. Computational time per number of packets.

In terms of performance, the proposed prototype uses a threshold of 10 000 packets or 2 s. This was based on the threshold that was used in our previous work in [9]. The same threshold was chosen again to enable a smoother integration of the two systems in the future. Thus, it is important to note the possibility that other thresholds could yield similar results. In fact, looking at the computational time based on the number of packets our developed system had to process at a time, as shown in Fig. 9, it is possible to estimate that a threshold of 360 K packets or 2 min (120 s) would produce comparable performance results. It should be noted that the FHSD prototype uses CSV files to calculate spoof IPs, and the test was performed

Downloaded from https://academic.oup.com/comjnl/article-abstract/58/4/892/336014 by University of Plymouth user on 30 October 2018

IP_OF_COUNTRY	IP_SUBNET	IP_COUNTRY_NAME	IP_CITY_NAME	HOPS	COMPLETED	LAST_HOP_ENDED	CLOSES_TTL
0.1.125.174	0.1.125.0	UNKNOWN	UNKNOWN	30	NO	29	29
0.101.79.119	0.101.79.0	UNKNOWN	UNKNOWN	30	NO	29	29
76.92.199.150	76.92.199.0	UNITED STATES	KANSAS	30	NO	16	16
76.93.12.254	76.93.12.0	UNITED STATES	CALIFORNIA	30	NO	19	20
76.94.211.44	76.94.211.0	UNITED STATES	CALIFORNIA	30	NO	19	20
76.94.27.31	76.94.27.0	UNITED STATES	CALIFORNIA	30	NO	27	20
76.94.67.128	76.94.67.0	UNITED STATES	CALIFORNIA	20	YES	20	20
76.96.122.8	76.96.122.0	UNITED STATES	CONNECTICUT	30	NO	15	15
76.98.67.241	76.98.67.0	UNITED STATES	NEW JERSEY	30	NO	15	15
76.99.14.245	76.99.14.0	UNITED STATES	PENNSYLVANIA	30	NO	20	20
77.10.210.127	77.10.210.0	GERMANY	BREMEN	16	YES	16	16
77.101.139.127	77.101.139.0	UNITED KINGDOM	ENGLAND	30	NO	13	16
77.101.185.177	77.101.185.0	UNITED KINGDOM	ENGLAND	14	YES	14	14
77.103.220.1	77.103.220.0	UNITED KINGDOM	ENGLAND	16	YES	16	16
77.104.169.154	77.104.169.0	ROMANIA	BUCURESTI	30	NO	13	13
77.105.240.217	77.105.240.0	SWEDEN	KRONOBERGS LAN	30	NO	19	19

FIGURE 10. Traceroute pre-process file.

on a Intel Quad Core Machine with 8 GB RAM and 1 TB 7200-rpm hard disk. It is not known how these results would vary if the implementation was done using database or if dedicated hardware like GPU or FPGA was used.

7. LIMITATIONS

The proposed method uses hop counting, geographical location, user agent and passive OS fingerprinting. This means that FHSD needs to maintain a database with correct TTL values from most IPs of the Internet with country and city. The use of information such as the subnet and geographical location of the IP helps to shrink a little the area of IPs. But for better results, a good database with IP hops should be maintained. Also, the passive OS fingerprinting and user agent database should be updated with new operating system signatures and the user agent new browsers respectively. All these data can be updated daily or when needed by a new proposed method or even use already proposed methods like SYN proxy [16].

In the current developed application, data are stored in files instead of a database. Our intent was to test the efficiency of our proposed method and not its speed. Nonetheless, file-parsing techniques were used to optimize performance and produce results to appear in seconds. To test our scenarios, some IPs using traceroute and GeoIP were pre-processed and stored in a file. An example of the process file is shown in Fig. 10. As seen in Fig. 10, in some cases the traceroute did not lead to the end IP (see column Completed). In these cases, the system checks the subnet, and if the IP is in the same subnet with another that is completed, it takes this value in the field (CLOSES_TTL); if not then it checks the GeoIP using country and city and if it finds the IP that the traceroute completed and is in the same country and city it takes the higher value. In a different case, it takes the value of the LAST_HOP_ENDED, which is the last reply from the traceroute. This could be avoided if a good database is kept with correct values from the subnets for more accuracy and not giving false positives.

8. FUTURE WORK: CONCLUSIONS

The result from the proposed method shows that it is effective in identifying spoof IPs, with detection rate almost 100%. Future work could include the validation of FHSD with flash crowds and whether it can discriminate them from spoof IPs. This was not the main aim of the paper and hence it is yet to be tested. Similarly, further work could investigate the implementation of FHSD for IPv6 and how it performs in IPv6 traffic.

ACKNOWLEDGEMENTS

The authors are indebted to Assistant Professor Vasilios Katos for his suggestion to use BoNeSi DDoS emulator for the experiments.

REFERENCES

[1] PwC (2013) 2013 Information Security Breaches Survey: Technical Report. http://www.pwc.co.uk/assets/pdf/cyber-security-2013-technical-report.pdf (accessed January 23, 2014).

[2] Loukas, G. and Öke, G. (2010) Protection against denial of service attacks: a survey. *Comput. J.*, **53**, 1020–1037.

[3] Neustar (2012) DDoS Survey: Q1 2012: When Businesses Go Dark. http://www.neustar.biz/enterprise/docs/whitepapers/ddos-protection/neustar-insights-ddos-attack-survey-q1-2012.pdf (accessed January 23, 2014).

Downloaded from https://academic.oup.com/comjnl/article-abstract/58/4/892/336014 by University of Plymouth user on 30 October 2018

[4] Techdata. (2011) Worldwide Infrastructure Security Report, Arbor Networks 2011 Volume VII. http://www.techdata.com/arbornetworks/files/Arbor%20Security%20Report%202012.pdf (accessed January 23, 2014).

[5] Yaar, A., Perrig, A. and Song S. (2003) Pi: A Path Identification Mechanism to Defend Against DDoS Attacks. *Proc. 2003 IEEE Symp. on Security and Privacy*, Berkeley, CA, USA, May 11–14, pp. 93–107. IEEE Computer Society.

[6] Thing, V., Sloman, M. and Dulay, N. (2007) A Survey of Bots Used for Distributed Denial of Service Attacks. *Proc. IFIP TC-11 22nd Int. Information Security conference (SEC 2007)*, Sandton, South Africa, May 14–16, pp. 229–240. Springer.

[7] Ehrenkranz, T. and Li, J. (2009) On the state of IP spoofing defense. *ACM Trans. Internet Technol.*, **9**, Article 6:1–29.

[8] MIT (2013). Spoofer Project: Stats. http://spoofer.cmand.org/summary.php (accessed January 23, 2014).

[9] Shiaeles, S.N., Katos, V., Karakos, A.S. and Papadopoulos, B.K. (2012) Real time DDoS detection using fuzzy estimators. *Comput. Secur.* **31**,782–790.

[10] Yaar, A., Perrig, A. and Song, D. (2006) StackPi: new packet marking and filtering mechanisms for DDoS and IP spoofing defense. *IEEE J. Sel. Areas Commun.*, **24**, 1853–1863.

[11] Ali, K., Zulkernine, M. and Hassanein, H. (2007) Packet Filtering Based on Source Router Marking and Hop-Count. *Proc. 32nd IEEE Conf. on Local Computer Networks (LCN 2007)*, Dublin, Ireland, October 15–18, pp. 1061–1068. IEEE Computer Society CPS, Los Alamitos, CA.

[12] Lee, F.Y. and Shieh, S. (2005) Defending against spoofed DDoS attacks with path fingerprint. *Comput. Secur.*, **24**, 571–586.

[13] KrishnaKumar, B., Kumar, P.K. and Sukanesh, R. (2010) Hop Count Based Packet Processing Approach to Counter DDoS Attacks. *Proc. 2010 Int. Conf. on Recent Trends in Information*, Telecommunication, and Computing (ITC 2010), Kochi Kerala, India, March 12–13, pp. 271–273. IEEE Computer Society CPS, Los Alamitos, CA.

[14] Wei, G., Gu, Y. and Ling, Y. (2008) An Early Stage Detecting Method against SYN Flooding Attack. *Proc. 2008 Int. Symp. on Computer Science and its Applications (CSA-08)*, Hobart, Australia, October 13–15, pp. 263–268. IEEE.

[15] Wang, H., Jin, C. and Shin, K. G. (2007) Defense against spoofed IP traffic using hop-count filtering. *IEEE/ACM Trans. Netw. (TON)*, **15**, 40–53.

[16] Zhang, F., Geng, J., Qin, Z. and Zhou, M. (2007) Detecting the DDoS Attacks Based on SYN Proxy and Hop-Count Filter. *Proc. Int. Conf. on Communications, Circuits and Systems (ICCCAS 2007)*, July 11–13, pp. 457–461. IEEE.

[17] Wang, X., Li, M. and Li, M. (2009) A Scheme of Distributed Hop-Count Filtering of Traffic. *Proc. of IET Int. Commun. Conf. on Wireless Mobile and Computing (CCWMC 2009)*, Shanghai, China, December 7–9, pp. 516–521. IET, Stevenage, Herefordshire.

[18] Swain, B.R. and Sahoo, B. (2009) Mitigating DDoS attack and Saving Computational Time using a Probabilistic Approach and HCF Method. *Proc. 2009 IEEE Int. Advance Computing Conference (IACC 2009)*, Thapar University Patiala, India, March 6–7, pp. 1170–1172. IEEE.

[19] Wu, Z. and Chen, Z. (2006) A Three-Layer Defense Mechanism Based on Web Servers Against Distributed Denial of Service Attacks. *Proc. 1st Int. Conf. on Communications and Networking in China (ChinaCom'06)*, Beijing, China, October 25–27, pp. 1–5. IEEE Explore.

[20] Covarrubias-Rodriguez, J.C., Parra-Briones, A. and Arturo-Nolazco, J. (2007) FLF4DoS. Dynamic DDoS Mitigation based on TTL field using fuzzy logic. *Proc. 17th Int. Conf. on Electronics, Communications and Computers (CONIELECOMP'07)*, Cholula Puebla, Mexico, February 26–28, pp. 12–12. IEEE Computer Society CPS, Los Alamitos, CA.

[21] Dumbare, S.S., Patil, P. and Bhanarkar, P. (2012) Survey on Defenses Techniques Used For Controlling IP Spoofing. *Int. J. Eng. Res. Technol.*, **1**. www.ijert.org.

[22] Kandula, S., Katabi, D., Jacob, M. and Berger, A. (2005) Botz-4-sale: Surviving Organized DDoS Attacks that Mimic Flash Crowds. *Proc. 2nd Conf. on Symp. on Networked Systems Design & Implementation-(NSDI'05)*, Boston, MA, USA, May 2–4, Vol. 2, pp. 287–300. USENIX Association Berkeley, CA.

[23] Lloyd, G. (2012) The Need for Hacker Identification and Attribution. http://genelloyd.com/publications.html.

[24] Technical Report 070529A (2007). Dynamics of the IP Time To Live Field in Internet Traffic Flows. Centre for Advanced Internet Architectures, Swinburne University of Technology, Melbourne, Australia.

[25] Gomez, J. and Dipankar D. (2002) Evolving Fuzzy Classifiers for Intrusion Detection. *Proc. 2002 IEEE Workshop on Information Assurance*, Vol. 6, No. 3. IEEE Computer Press, United States Military Academy, West Point, New York, USA.

[26] Dickerson, J.E., Juslin, J., Koukousoula, O. and Dickerson J.A. (2001) Fuzzy Intrusion Detection. *Ifsa World Congress and 20th Nafips Int. Conf.*, 2001, joint 9th, Vol. 3, IEEE, USA.

[27] UPM (no date) Mamdani's Method. http://www.dma.fi.upm.es/java/fuzzy/fuzzyinf/mamdani3_en.htm (accessed January 23, 2014).

[28] Bonesi (2008). BoNeSi—the DDoS Botnet Simulator. http://code.google.com/p/BoNeSi (accessed January 23, 2014).

[29] SecurityTube Tools (2012) BoNeSi. http://www.securitytube-tools.net/index.php@title=BoNeSi.html (accessed January 23, 2014).

[30] UserAgentString.com (no date) All User Agent Strings. http://www.useragentstring.com/pages/All/ (accessed January 23, 2014).

[31] MIT (2000) MIT Lincoln Laboratory Scenario (DDoS) 1.0. http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data/2000/LLS_DDOS_1.0.html (accessed January 23, 2014).

[32] Jin, C., Wang, H. and Shin, K.G. (2003) Hop-Count Filtering: An Effective Defense Against Spoofed DDoS Traffic. *Proc. 10th ACM Conf. on Computer and Communications Security*, Washington, DC, USA, October 27–30, pp. 30–41. ACM, New York.

Downloaded from https://academic.oup.com/comjnl/article-abstract/58/4/892/336014 by University of Plymouth user on 30 October 2018