Computers and Electrical Engineering 39 (2013) 2252–2262

Contents lists available at SciVerse ScienceDirect

Computers and Electrical Engineering

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

Detecting denial of service by modelling web-server behaviour[☆]

Luis Campo Giralte[*], Cristina Conde, Isaac Martin de Diego, Enrique Cabello

Computer Architecture and Technology Department, Universidad Rey Juan Carlos, C/Tulipan s/n, 28933 Madrid, Spain

A R T I C L E I N F O

Article history:
Available online 3 August 2012

A B S T R A C T

The article presented here discusses a system which characterizes HTTP traffic and discriminates between legitimate and other kinds of HTTP traffic, such as those generated by Botnets or distributed denial of service (DDoS) tools. The system presented in this paper uses three analyses that are sequentially applied to the traffic flow to detect abnormal users.

Combining statistical methods as well as analysis of HTTP request paths and the access time to the different resources in the web server, we have labelled abnormal users in real traffic flow. First, we have tested our prototype in real traffic from a multi-site web server detecting all abnormal users, such as an illegitimate audit of the server, Google bot and web-crawlers. In a second experiment, the most common DDoS attacks were introduced in the real traffic flow. As a result, all suspicious users were detected and labelled.

1. Introduction

Distributed denial of service (DDoS) attacks have increased in recent years. All web servers can be targeted for such attacks. One important issue of these DDoS attacks is that they are more difficult to detect than the traditional denial of service (DoS) ones. A survey from Arbor Networks [1], a security-engineering firm, indicates that DDoS attacks continue to grow in both scale and sophistication. The largest observed attack reached around 40 Gbps in 2008, a 67% growth over the previous year.

In April 2009, there were multiple attacks which overloaded DNS cache servers by issuing large quantities of DNS queries that return large responses [2]. In May 2009, a DDoS attack on DNS servers in China caused extensive connection failures for several hours [3]. Additionally, a DoS attack tool, called Slowloris [4,5], that affects several HTTP servers was released, and information suggests it was used in DDoS attacks in Iran [6].

On March 20, 2010, a criminal organization registered a series of malicious domains signalling the birth of a new Botnet; one designed to offer a commercial service for delivering Distributed Denial of Service (DDoS) attacks against any desired target [7]. This publicly available service, hosted in China, is available for lease to anyone willing to establish an account on-line, input the domain(s) they wish to attack, and pay for the service.

Los Angeles-based Web host Media Temple (http://www.mediatemple.net) was struck on May 25, 2010, by the largest distributed denial-of-service attack against its name-servers in the company's history. According to the company's reports [8], the attack, which started around 3:50pm PDT, was the first of its kind to affect Media Temple services, resulting in an interruption of service spanning 2 h and 5 min, and affecting such big name clients as Adobe, ABC, Sony, NBC, Time, Volkswagen, and Starbucks.

☆ Reviews processed and recommended for publication to Editor-in-Chief by Guest Editor Prof. Gregorio Martinez.
* Corresponding author.
E-mail addresses: luis.campo.giralte@urjc.es (L.C. Giralte), cristina.conde@urjc.es (C. Conde), isaac.martin@urjc.es (I.M. de Diego), enrique.cabello@urjc.es (E. Cabello).

0045-7906/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.compeleceng.2012.07.004

On November 28, 2010, WikiLeaks was hit with a mass distributed denial of service attack, [9,10] the organization said on its official Twitter account. In this sort of attack, many computers generally flood a server with requests or use other techniques to prevent the server from displaying a Web page.

Arbor Networks reported that after the attack started, WikiLeaks redirected traffic to its Cablegate site from a Swedish hosting provider to the mirror sites in France and the US, which provide exact copies. In response to these attacks and to the companies which do not support WikiLeaks, on December 1, 2010 [11,12] the Anonymous activist group said it brought down the websites using distributed denial of service (DDoS) attacks in reprisal for MasterCard ceasing to process donations for WikiLeaks. The whistleblower site is in the process of releasing hundreds of thousands of US government diplomatic cables. The tool used to carry out these attacks was Loic (Low orbit ion cannon)[13].

The system presented in this paper is based on the combination of several techniques in order to reduce the false positives of the detection approach and also to improve the performance in real-time systems. The main contributions of this research are: Firstly, normal user behaviour is described in a statistical way. Secondly, the use of a graph to model the several paths of the web server as well as the different costs of navigating through the server is proposed. And finally, the frequency of HTTP operations carried on the web server are considered. With all these techniques our system is able to detect the most used DDoS tools of the current state such as Loic, Slowloris and Siege [14] that have been used on recent attacks.

Currently, the majority of DDoS attacks are performed using TCP (90–94%) [15] and the predominant protocol on the Internet is HTTP. Our system not only takes into account this information, but also analyses and characterizes HTTP on a multi-site server in order to discriminate normal users from abnormal behaviour ones.

This research presents the implementation of a detection system capable of detecting DDoS, Bots [16], and other kinds of attacks. The system is implemented in Python language so it is fully portable to other platforms. Our tests have been carried out on Linux to evaluate the system performance with a high ammount of data from a real web server.

The rest of the paper is organized as follows. In Section 2 we report the related work and classify the most interesting techniques. Section 3 is devoted to an overview of the different techniques used to characterize the users network behaviour and how the system works. Section 4 discusses the technical results. Finally, conclusions are reached in Section 5.

2. Related work

Over recent years, several schemes have been proposed to detect flooding DDoS attacks (see, for instance [17–20]).

Mirkovic [21] detect attacks based on metrics that measure the Quality of Service (QoS) experienced by the end users during the attack. The proposed metrics consider QoS requirements for a range of applications and map them into measurable traffic parameters with acceptable thresholds. The problem of the proposed model is that the metrics could be easy simulated in order to avoid the detection.

Cavrilis [22] propose a detector based on the reaction differences between human and machine users in the presence of decoy hyperlinks on web pages on the server side. Hyperlinks not shown by navigators are followed by machine-based users. Their experimental results derived from real websites give the extremely low false positive rate of 0.0421%, although their system needs to update the server web pages and be more resilient to timer attacks which use well-known links from the web server. Their major weaknesses are the management of these links on the web-servers wherever the number of webpages are high.

Seufert [23] explore the effectiveness of machine learning techniques in developing automatic defenses against DDoS attacks. A data collection and traffic filtering framework is developed and later used to explore the potential of artificial neural networks in the defense against DDoS attacks. They just make emulations not with real traffic and don't give results of how many flows and users can be handled.

Chonka [24] use the theory of network self-similarity to differentiate DDoS flooding attack traffic from legitimate self-similar traffic in the network. They observed that DDoS traffic causes a strange attractor to develop in the pattern of network traffic, and developed a novel prediction algorithm that detected DDoS attacks packets and gave some certainty of when the attack started. Their proposal lacks proper measures of the performance of the algorithm, and the complexity of it gives us the impression that the implementation could be a tedious work.

Li [25] utilize energy distribution based on wavelet analysis to detect DDoS attack traffic. Wavelet analysis is able to capture complex temporal correlations across multiple time scales with very low computational complexity. Wavelet analysis has a set of parameters to be tuned in order to be effective, and this parameter optimization is not an easy task.

Danner [26] propose an algorithm for detecting attacks on the Tor network. Tor is currently one of the more popular systems for anonymizing real-time communications on the Internet. It is a network of virtual tunnels that allows people and groups to improve their privacy and security on the Internet. It also enables software developers to create new communication tools with built-in privacy features. Tor provides the foundation for a range of applications that allow organizations and individuals to share information over public networks without compromising their privacy. As they claim, their attack it is easy to detect it and a real attack from this network is improbable.

Wang [27] propose an architecture called Patricia where edge networks cooperate to prevent misbehaving sources from flooding traffic in both control and data channels. The architecture has three major features. First, it provides a cooperative mechanism for edge networks to share information about suspected users. Second, it employs an edge-network endorsement procedure. A source node must obtain an endorsement from its own network and an authorization from the destination. Third, they switch a host to a mode that requires active regulation of its incoming traffic only when necessary. The main drawback of their proposal is that the attack could happen from inside the network. Furthermore their protocol could be distorted in order to stop the detection.

Yatagai [28] propose two detection algorithms to detect HTTP-GET flooding attacks, one focused on the browsing order of pages and the other on the correlation of browsing time with page information size. It lacks in that although some attacks combine the TCP/IP and application layers. Their algorithm take into account only the application layer.

Siaterlis [29] propose the use of multiple metrics to detect flooding attacks at the edge and classify them as incoming or outgoing attacks with an Artificial Neural Network. They follow a supervised learning approach. By using network statistics as an input signal, they train a multilayer feed forward network to classify normal and attack states in UDP traffic. Their metrics are easy to implement and their solution could be affected by the Slashdot effect [30].

Xie [31] shows a detector based on the hidden semi-Markov model. They describe the dynamics of Access Matrix to detect the attacks. The entropy of document popularity fitting to the model is used to detect the potential application-layer DDoS attacks. They show numerical results on real Web traffic but their solution lacks in the use of the TCP/IP layer. In addition, their data set was collected in 1999, which seems quite old for a current DDoS attack.

Lin [32] analyse massive web traces in order to categorize the new tendencies of the HTTP traffic. They provide an automatic method to analyse the composition of modern HTTP and to evaluate the possible performance implication of modern HTTP traffic on existing web servers.

Zhu [33] characterize the uncacheable HTTP traffic. Their research offers the following observations: a considerable portion of uncacheable HTTP content is cacheable; domination of network latency factors motivate the movement of functionality for uncacheable HTTP content generation to the edge of the network; prefetching for personalized Web content is promising because of the concentrated popularity of clients and servers; convinced by the observed P2P request popularity, they believe that content-based caching is significant to the ever-increasing P2P traffic; exploiting URL-alias is a promising direction to improve cacheability of uncacheable content.

As a summary of this section, we describe on Table 1 a classification of the most relevant techniques for detect DDoS attacks. They haas: (a) L4 Metrics, use the metrics from TCP/IP or UDP/IP as bit-rate, packets per flow, relations between SYN and SYN/ACK, and so on. (b) Distributed, have some kind of distributed architecture as collaborative nodes. (c) Machine Learning, have some supervised or no supervised methods for the detection. (e) Frequencies, stands for the use of frequency analysis of the time of the packets, wavelets and Fourier analysis. (f) L7 metrics, use the application layer in order to detect suspicious behaviour and finally and (g) Neural and Markov, that techniques related to artificial neural networks and Markov chains.

Our proposed system has been included in Table 1 (labelled as Campo) and has the following characteristics. It is distributed, so it could be placed on any network node or even on the same web server with negligible consumption. Our system is completely independent of the web-traffic server, so it is capable of adapting to several web environments, such as content delivery networks (CDNs), static content (images, web-pages), and so on. Previous works have resulted in that DoS detection methods operating solely under on layer are not effective against all attack classes targeting the application. Our model uses all possible statistics information of the implied layers.

3. System description

3.1. System overview

The system presented in this paper is designed to detect abnormal user behaviour. The goal of the system is to detect a DDoS attack to a HTTP server. To detect an attack three consecutive analyses (we have called it: statistics, HTTP graph, and HTTP path caches) are applied to the Internet flows. When one of these analyses detects an abnormal behaviour, the user is labelled as suspicious and could be notified to an external mitigation system. If one analysis indicates that a user is suspicious, the remaining analyses are not applied. The first analysis is a statistical one for every flow user, taking into account the

Table 1
Classification of the state of the art methods according to the considered techniques.

Author	L4 metrics	Distributed	Machine learning	Frequencies	L7 metrics	Neural and Markov
Mirkovic [21]	*					
Cavrilis [22]					*	
Seufert [23]	*		*			*
Chonka [24]		*				*
Li [25]	*			*		
Danner [26]		*				
Wang [27]	*	*				
Yatagai [28]					*	
Siaterlis [29]	*					*
Xie [31]					*	*
Campo	*	*			*	

global number of users. In this analysis general measurements of L4 (TCP/IP) and L7 are considered. Lets say: the number of packets per flow, flows per user, request per user, and so one, are compared against the average value of these measurements considering also their standard deviation. If a high difference between a user value and the average exists, we label this user as suspicious.

The second one is an analysis of the HTTP flow. For each flow the system checks the different requests and time between each request. Let's say, the HTTP path of the flow is considered. All legitimated HTTP paths are previously stored in a cache memory with the minimum time between two consecutive requests. The flow path is compared against this cache, so if the sequence of requests does not match any stored path or the time is smaller than a minimum, the user is labelled as suspicious. We will call this technique HTTP graph cache analysis.

The third method is the frequent HTTP paths. This method counts the number of times that one HTTP-path is repeated in a flow or user. If a path is repeated with a frequency higher than a threshold, we could assume that the flow could be generated by a bot and, as a consequence, the user will be labelled as suspicious. We will call this frequent HTTP path cache analysis.

3.2. Detailed system description

Fig. 1 shows the proposed algorithm with its main parts: the 'update mode' and the 'detection mode', described in the next sections.

The first operation performed by the method, involves checking IP address. Only selected IP can update the values of the parameters used in the analysis (update mode). These IP addresses correspond to computers that are placed near the server (in the same network), to obtain the fastest response times and the lowest latency possible. If the IP has been authorized, the HTTP request of the flow will update the statistics of the system, such as average gets per flow, and post per flow, as well as updating the graph costs with all the possible paths of the web server. Our system requires that the values for the parameters in the statistical analysis, graph and frequent HTTP path caches were initialized and updated. This values can only be changed by one specific kind of users (ones who are undoubtedly administrators or developers).

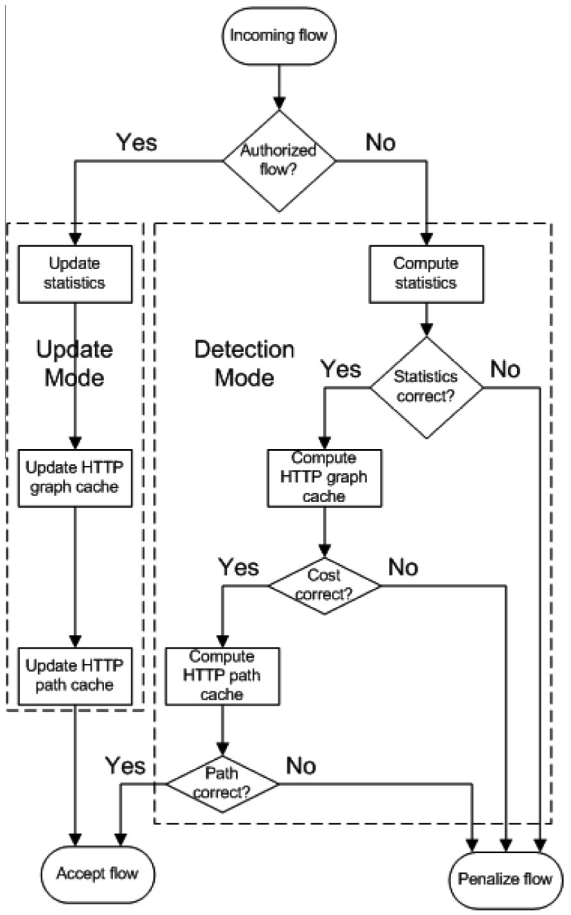

Fig. 1. Diagram of the proposed system. Both detection mode and update model are detailed.

If a normal (not an authorized user to change the parameters) web user is connected, the system changes to detection mode. First, the flow statistics are compared against the global ones so the flow is penalized when their statistic values are very different from the normal ones (see Section 3.3.1 for details). If the flow statistics do not exceed the global ones the request flow is checked on a graph, where every edge contains the minimum time between the different requests (see Figs. 3 and 4) as detailed in Section 3.3.2. If a flow is faster than the results in the cache, a time buffer is increased, and if this buffer achieves a threshold level the flow is penalized. In order to avoid attacks based on repetitions, and also to avoid false negatives due to the Slashdot effect, a system that checks the frequent paths of the user have been carried out (see Section 3.3.3). This is called the path cache, that tracks the frequent paths the user takes.

3.3. Analyses of the flows (detection mode)

Once the values of the data are initialized in the update mode, the system will analyse all incoming users. All the flows generated by each web user were processed in “detection mode” in the sequential application of the three analyses presented before and detailed in this section.

3.3.1. Statistics analysis

The following statistical values are computed for each incoming user: number of get requests, gets mean, standard deviation of gets, mean of flows per user, standard deviation of flows per user, posts mean and standard deviation of posts, flows per minute per user, request per minute per user and so on.

The combination of TCP/IP and HTTP statistics gives more robustness to our model, noticing that most authors only take into account one of the statistic layers. The correlation between a high number of flows created and using the same URI access by a user, could be detected by using the combination of these statistics.

By using global statistics to determine web server behaviour we can discard the users whose behaviour is completely different from the normal ones.

3.3.2. The HTTP graph cache analysis

The HTTP graph cache is a memory zone where the Uniform Resource Locator (URL) hashes of the web server are stored. In update mode, these URLs can be loaded from pcap files, a file that contains network packets, or interactively by authorized users. The following information is also stored: the hash identifier of the URL and the time interval between each URL request of the flow. As shown in Fig. 2 time delay between two request (from T0 to T4) is obtained from the network delays (from T1 to T0 and T3 to T2) and the operations carry out on the server (T2 to T1) and in the user navigator (T4 to T3). Since authorized users are in the same network than the server, this time interval will be the minimum one.

When a normal user arrives, the system checks the time interval between each URL request. Then, if the time interval between the requests in the flow is smaller than in the cache, we can say that the flow is faster than the cache time. This is virtually impossible, because outside the network the latency is higher than in the internal network where the authorized users made the access.

Two simplified flow graphs are shown in Fig. 3. In order to clarify the figure, the URL hashes are labelled as node number. In these graphs all time intervals are computed in seconds. Notice that some of them are retransmissions, for example node2,

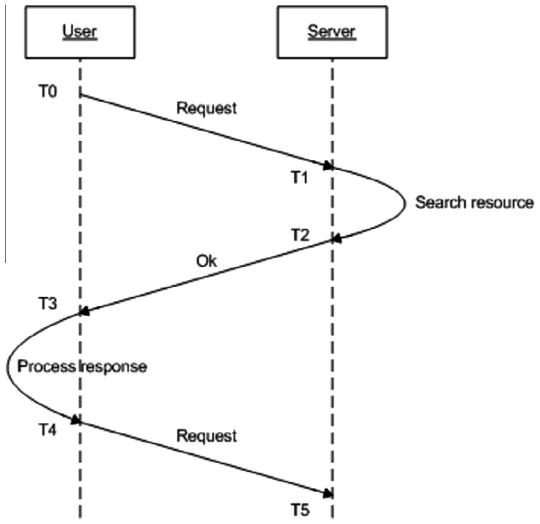

Fig. 2. Sequential description of an HTTP request and the corresponding response.

node7, node8, node9 and node12. Time of retransmission is also labelled in Fig. 3. This information is required for HTTP graph cache analysis.

As shown in Fig. 4, the complexity of a real HTTP graph is quite high due to the large number of nodes and links the graph have. In our experiments more than 6000 nodes appear, so just for clarification the figure only takes into account less than one minute of active flows to understand how the HTTP graph is stored in memory.

3.3.3. The HTTP path cache analysis

To avoid attacks based on timers, where the same access pattern is constantly repeated, we develop a frequent path detector which involves checking the requests of the complete flow in order to avoid the same access pattern to the web server. One of the DDoS tools which makes use of timer attacks is Slowloris [4]. However, most of the DDoS tools repeat the same request in the flows unless a mechanism for modifying the access pattern is included in the tool, as is the case with Siege.

Slowloris has been involved in several attacks [6], with disastrous results. As we can see in Fig. 5, the number of repeated paths per second is higher than that in the web server shown in Fig. 7. The same effect can be observed in Fig. 6 with Slowloris and even with Loic.

3.4. Attack detection

With the previous analysis techniques the system is able to detect the following DoS attacks at the application layer:

- Random walking (http://<target>/random_path. This technique involves randomly accessing different web pages or resources, resulting in high CPU consumption and high hits of false cache. Our system is capable of detecting this case checking the pattern access to the HTTP graph cache.

- Login access (http://<target>/login.php). The goal of this technique is to provoke a high rate of access to the database. The proposed system could detect this case observing the repetitions of requests to the same resources. Therefore statistical analysis will detect the attacks based on this technique.

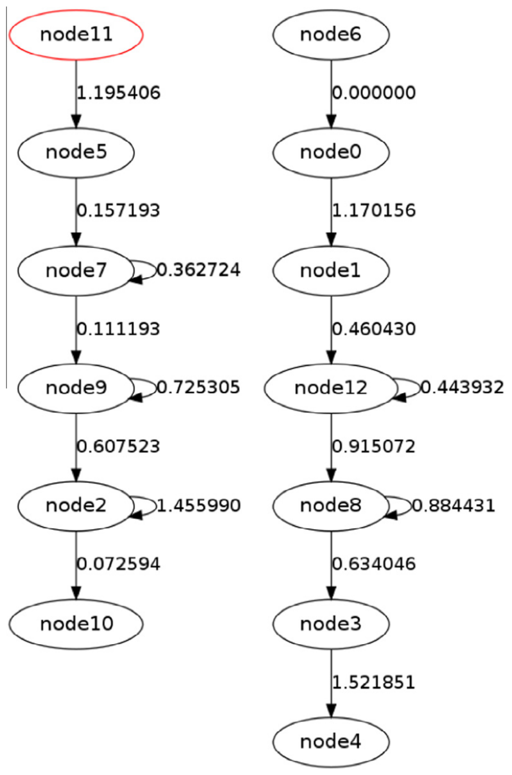

Fig. 3. Example of a HTTP graph cache. Sport website index request. The graph node is shown and numbers in arrows represents the time delay (in milliseconds) for the operation.

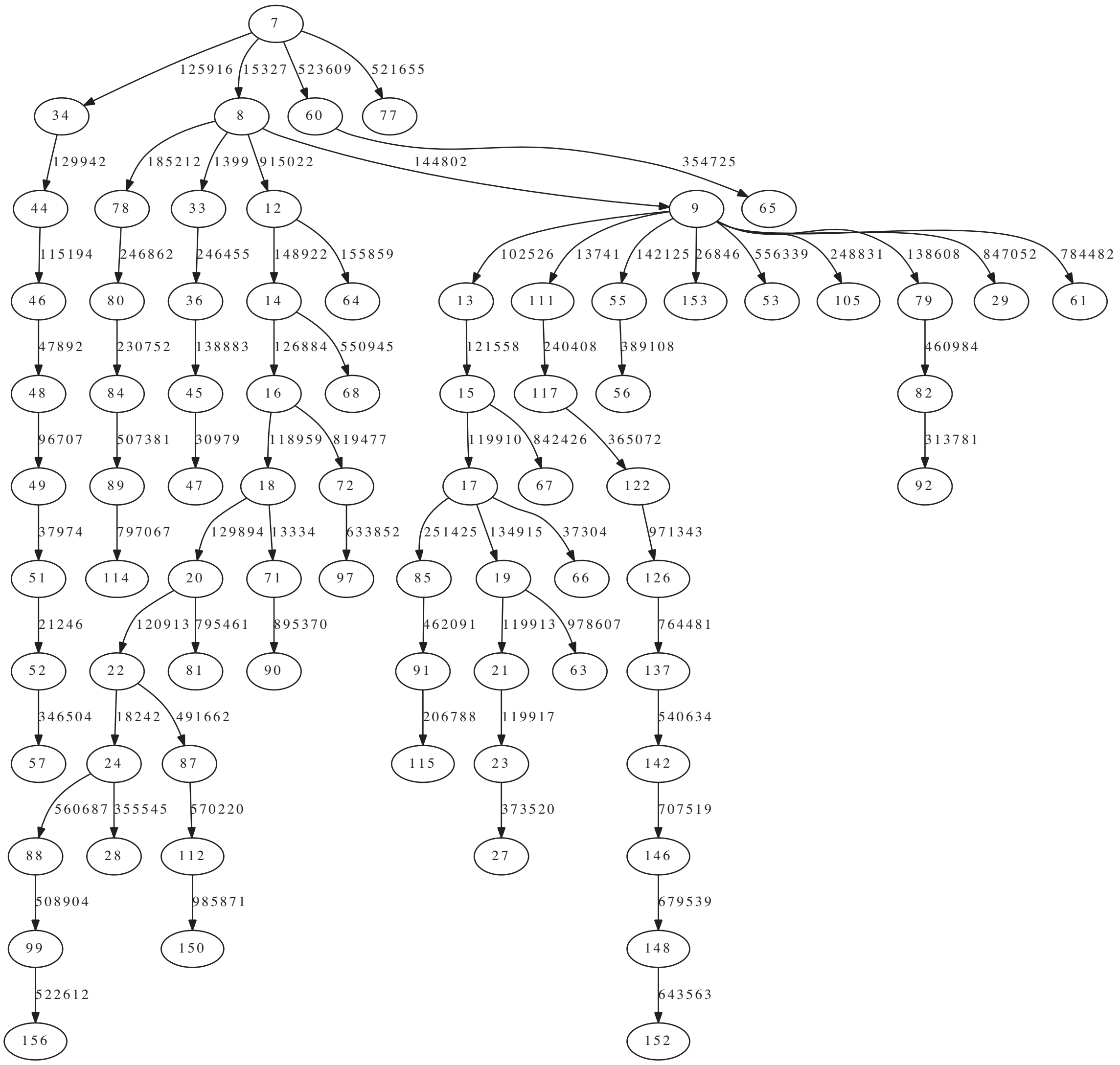

Fig. 4. One minute real HTTP graph access

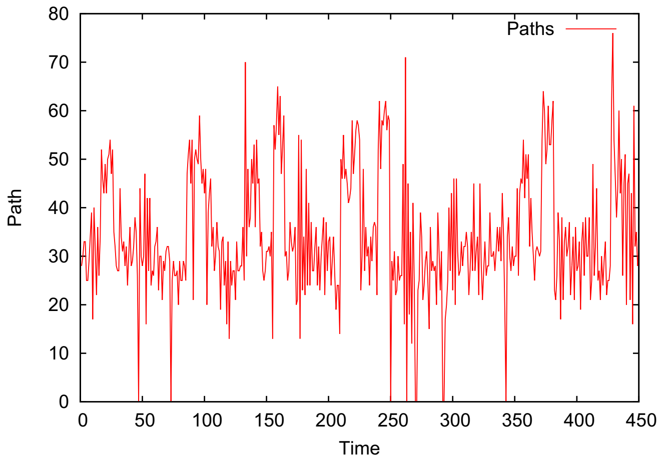

Fig. 5. Frequency of HTTP Paths in a Siege attack

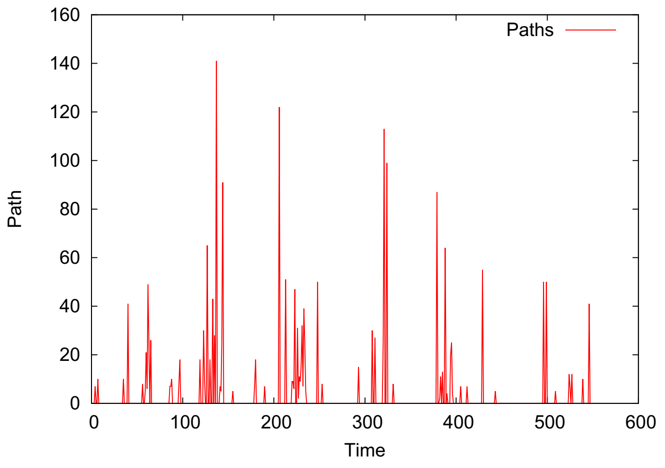

Fig. 6. Frequency of HTTP Paths in a Slowloris attack

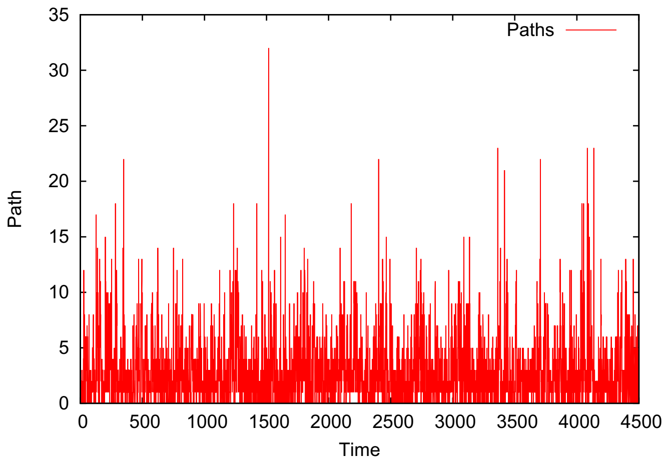

Fig. 7. Frequency HTTP Paths in the web server with no attack

- Search access (http://<target>/search.php). The main objective here is to provoke a high rate of scripts which try to access the information on the web pages or even on the database. This usually provokes a high CPU consumption in the web server. This case is detected by the HTTP graph cache or the HTTP path cache.

- POST operations. These operations can often affect the RAM Memory because the uploading file is buffered into the memory. The POST operations are also related to Pulsing attacks [34–36], which use the TCP retransmission mechanism in order to trigger a DoS in the POST flows. The presented system combines the statistics at network layer and surveillance in the access to specific resources to detect it. This kind of attacks are clearly detected by the combination of the different analyses, since one of them alone could not be able to detect it.

4. Experiment description and results

For testing, we have used real traces from a public web server which hosts two different domains. These traces have been captured on different days of the same month. The web server was made up of an Apache 2.0 with PHP support on an Intel core duo 2 GHz with 4 GBs of memory. In this experiment more than 290,000 HTTP flows corresponding to 14,000 different users has been analysed.

The first part of the experiment consists on the modelling of the web traffic (as described in Section 4.1). The second part is to test our algorithm with different attacks. To do this, abnormal users have been introduced to the stored traffic. The most usual DDoS attacks have been tested and, as a result, our system has been able to detect and label as suspicious all the induced users (as described in Section 4.2).

4.1. Web-server cache traffic model

The traffic used for our purposes comes from a public Spanish web server. The web server runs an Apache software with PHP support. The traces have been taken on different days (see Table 2) of the week, two of which had normal activity and a third one which had high activity (during a weekend). The table also contains the consumption of the system. Notice that the memory consumption and the CPU are negligible if we compare it with the duration of the sample. Table 2 shows that the selected web server is continuously visited by a great number of users, generating a large number of flows.

Table 2
Web server activity and algorithm consumption.

Sample	Duration	Users	Flows	Requests	RAM memory	CPU
Normal day	7 h	2317	42807	63582	332 Mbytes	20 segs
Normal day	7 h	1676	33253	52172	332 Mbytes	21 segs
High activity day	24 h	10939	217410	288728	339 Mbytes	1 and 3 min

Table 3
Global statistics.

Avg flows	Std flows	Avg gets	Std gets	Avg posts	Std posts
9	52	30	75	1	0

Table 4
Global time statistics.

Avg flows/min	Std flows/min	Avg request/min	Std request/min
7.45	24.9	1.52	5.26

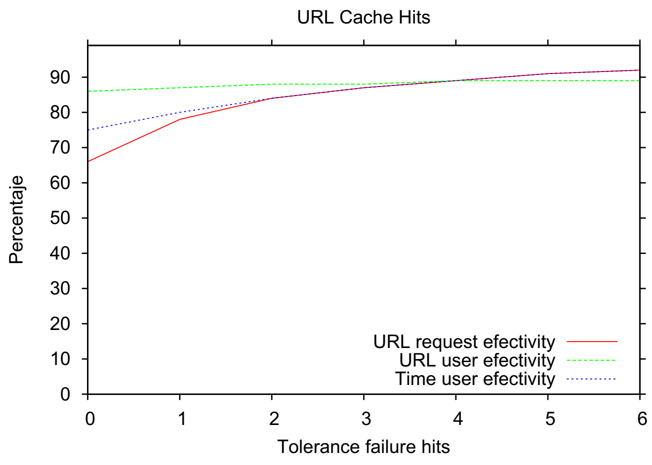

Fig. 8. HTTP caches effectiveness

Table 5
Model vs attacks.

Attack tool	1 Stage (statistics)	2 Stage (graph cache)	3 Stage (path cache)
Slowloris	Detected		
Siege	Detected		
Siege slow random page	Non detected	Detected	
Siege slow repeat page	Non detected	Detected	
Siege slow repeat path	Non detected	Non detected	Detected
Loic fast	Detected		
Loic slow	Detected		

The first analysis to be performed is based on user statistics. The values of the statistics considered in this test are shown in Tables 3 and 4. In this analysis not only the mean value is considered, but also the standard deviation is taken into account. If standard deviation in one data is high, big differences between user and mean values are allowed without raising an alarm. If this deviation is small, also small variations can raise an alarm. There is a special case when a user tries to download a hard resource, such as a large file. Some navigators or even download managers open several flows to perform the operation. Our system is capable of detecting this kind of behaviour increasing the limit of flows per user within a resource.

The effectiveness of the HTTP graph and path cache (see Fig. 8) has been tested. A cross validation procedure has been done. One day has been selected for training the system (to obtain the values of the parameters in update mode) and the other two days for testing (these data are used in detection mode). This procedure has been repeated three times, changing the training day and also testing days. The number of failure hits of the graph and path cache are stored for each flow. If this number is higher than a threshold, the user is labelled as a suspicious.

As shown in Fig. 8, when the threshold (Tolerance failure hits) increases the effectiveness of the HTTP cache grows up to 90%. When this percentage is reached, the increment of the threshold has no effect. In our case, we could select this number as an optimal threshold (the value in our experiment is set to 4). This means that 90% of the users of one day has the same graph and path than in the other days (with a tolerance of 4 failure hits per flow). So, even for a real web server whose web contents are dynamically updated and changed frequently, we could maintain these caches with no so frequently updates. If we would like to increase the effectiveness of the caches, more frequent cache updates are required. In our experiment, the selected threshold was enough for our purposes, even considering that training and testing were done in different days.

In this part of the experiment, our initial estimation considered that the traces from the web server were free of attacks. However, our system discovered three abnormal users. In the statistical analysis our system discovered an illegal auditory to the web server carried out by using the Acunetix framework [37]. Also by checking different paths the Google Bot [38] and some web-crawlers were also detected. Therefore the effectiveness of the method were validated.

4.2. Comparing model vs attacks

In this section traces from Siege, Slowloris and Loic had been injected into the system in order to check the different parts of the algorithm. As shown in Table 5, the four generated cases were detected by using statistics (they are labelled as Slowloris, Siege, Loic fast and slow in the first column). Therefore in these cases the statistical analysis was crucial to detect all of them. To continue the experiment, the request frequency in the Siege slow cases were reduced in order to pass the statistical phase and to test the other parts of the algorithm. These are labelled as Siege slow random page, repeat page and repeat path. With these cases the most interesting attack techniques are covered in our experiment.

Table 5 it is shown that all induced attacks have been detected by our system. Since the analyses are applied sequentially, if one of them detects an abnormal user the process is finished. All induced attacks are labelled in the table. We can observe that from the seven considered attacks, four of them are detected in an early analysis (using statistical information only). Two attacks are detected in the next analysis, and only one is detected in a later stage. As a result, the system is able to detect all possible attacks in an optimal way.

5. Conclusions and future work

The system proposed in this paper has focused on the combined use of graphs, statistics, and frequent analysis of HTTP request paths to characterize users behaviour in a multi-site web server. Combining different statistics from network and application layers that helps to cover the most common DoS attacks. Our model has been capable of autonomously detects bots, such as the one found in Google, and security scanners like Acunetix which are used to audit the web. Also, our model is able to detect top DDoS tools as well as new ones.

The proposed system has the following characteristics. First, it can be implemented in a distributed way, in this case the system can support big DDoS attacks that could saturate CPU resources. Second, the system is completely independent of the content of the web server, so it is capable of adapting to several web server environments. Finally, the integration with mitigation systems its easy due to our system notifying the alerts to the Linux common log system.

One extension to our system could be the use of several models which depend on contextual information like the time of the day or the day of the week. By using this extension the system will reduce the number of false positives and minimizes the Slashdot effect [30] on the web-site. This extension will provide further research for using HTTP models according to the contents of the web-site and also for using time as a variable of the system.

References

[1] Arbor networks. worldwide infraestructure security report. vol. 4., <http://www.arbornetworks.com/en/research.html>; 2012.
[2] [Dns-operations] lots of queries for txt records?. <https://lists.dns-oarc.net/pipermail/dns-operations/2009-April/003779.html>; 2012.
[3] Dns attack downs internet in parts of china. <://www.networkworld.com/news/2009/052109-dns-attack-downs-internet-in.html>; 2012.
[4] Slowloris http dos. <http://ha.ckers.org/slowloris/>; 2012.
[5] Mitigating slowloris. <http://www.cert.org/blogs/certcc/2009/07/slowloris_vs_your_webserver.html>; 2012.
[6] Slowloris and iranian ddos attacks. <http://isc.sans.edu/diary.html?storyid=6622>; 2012.
[7] The imddos botnet: Discovery and analysis. <http://www.damballa.com/downloads/r_pubs/Damballa_Report_IMDDOS.pdf>; 2012.

[8] Media temple network unavailability. <http://status.mediatemple.net/weblog/category/system-incidents/1319-mt-media-temple-network-unavailability/>; 2010.
[9] Wikileaks using amazon servers after attack. <http://blogs.wsj.com/digits/2010/11/29/wikileaks-using-amazon-servers-after-attack/>; 2012.
[10] Wikileaks and the failure of cyberattacks as censorship. <http://blogs.forbes.com/andygreenberg/2010/11/28/wikileaks-and-the-failure-of-cyberattacks-as-censorship/>; 2012.
[11] Operation payback attacks target mastercard and paypal sites to avenge wikileaks. http://thelede.blogs.nytimes.com/2010/12/08/operation-payback-targets-mastercard-and-paypal-sites-to-avenge-wikileaks/; 2012.
[12] Operation payback. <http://en.wikipedia.org/wiki/Operation_Payback>; 2012.
[13] Low orbit ion cannon. <http://sourceforge.net/projects/loic/>; 2012.
[14] Siege. <http://www.joedog.org/index/siege-home>; 2012.
[15] Moore D, Shannon C, Brown DJ, Voelker GM, Savage S. Inferring internet denial-of-service activity. ACM Trans Comput Sys 2006;24(2):115–39.
[16] Erbacher RF, Cutler A, Banerjee P, Marshall J. A multi-layered approach to botnet detection. In: Arabnia HR, Aissi S, editors. Security and Management. CSREA Press; 2008. p. 301–8.
[17] Chen Y, Hwang K, Ku W-S. Collaborative detection of ddos attacks over multiple network domains. IEEE Trans Parallel Distrib Sys 2007;18(12):1649–62.
[18] Lakhina A, Crovella M, Diot C. Mining anomalies using traffic feature distributions. SIGCOMM Comput Commun Rev 2005;35(4):217–28.
[19] Hwang K, Cai M, Chen Y, Qin M. Hybrid intrusion detection with weighted signature generation over anomalous internet episodes. IEEE Trans Dependable Secur Comput 2007;4(1):41–55.
[20] Liu X, Yang X, Lu Y. To filter or to authorize: network-layer dos defense against multimillion-node botnets. In: SIGCOMM '08: Proceedings of the ACM SIGCOMM 2008 Conference on Data communication. New York, NY, USA: ACM; 2008. p. 195–206.
[21] Mirkovic J, Hussain A, Wilson B, Fahmy S, Reiher P, Thomas R, et al. Towards user-centric metrics for denial-of-service measurement. In: ExpCS '07: Proceedings of the 2007 workshop on Experimental computer science. New York, NY, USA: ACM; 2007. p. 8.
[22] Gavrilis D, Chatzis IS, Dermatas E. Detection of web denial-of-service attacks using decoy hyperlinks.
[23] Seufert S, O'Brien D. Machine learning for automatic defence against distributed denial of service attacks. In: ICC, IEEE; 2007. p. 1217–22.
[24] Chonka A, Singh J, Zhou W. Chaos theory based detection against network mimicking ddos attacks. Comm Lett 2009;13(9):717–9.
[25] Li L, Lee G. Ddos attack detection and wavelets. Telecommun Sys 2005;28(3-4):435–51.
[26] Danner N, Krizanc D, Liberatore M. Financial cryptography and data security. Berlin, Heidelberg: Springer-Verlag; 2009. p. 273–284 [chapter detecting denial of service attacks in tor].
[27] Wang L, Wu Q, Luong DD. Engaging edge networks in preventing and mitigating undesirable network traffic. In: NPSEC '07: Proceedings of the 2007 3rd IEEE Workshop on Secure Network Protocols. Washington, DC, USA: IEEE Computer Society; 2007. p. 1–6.
[28] Yatagai I, Sasase. Detection of http-get flood attack based on analysis of page access behavior. In: Proceedings IEEE Pacific RIM Conference on Communications, Computers, and Signal Processing, IEEE; 2007. p. 232–35.
[29] Siaterlis C, Maglaris V. Detecting incoming and outgoing ddos attacks at the edge using a single set of network characteristics. In: Proceedings of the 10th IEEE Symposium on Computers and Communications. Washington, DC, USA: IEEE Computer Society; 2005. p. 469–75.
[30] Lampe C, Johnston E. Follow the (slash) dot: effects of feedback on new members in an online community. In: Pendergast M, Schmidt K, Mark G, Ackerman M, editors. GROUP. ACM; 2005. p. 11–20.
[31] Xie Y, Yu S-Z. Monitoring the application-layer ddos attacks for popular websites. IEEE/ACM Trans Netw 2009;17(1):15–25.
[32] Lin X, Quan L, Wu H. An automatic scheme to categorize user sessions in modern http traffic. In: GLOBECOM, IEEE; 2008. p. 1485–90.
[33] Zhu Z, Mao Y, Shi W. Workload characterization of uncacheable http content. In: Proceedings of International Conference on Web Engineering, Munich, LNCS 3140. Springer; 2004. p. 391-39 [Caching over the Entire User-to-Data Path, 2003].
[34] Luo X. Optimizing the pulsing denial-of-service attacks. In: Proceedings of the 2005 International Conference on Dependable Systems and Networks, DSN '05. Washington, DC, USA: IEEE Computer Society; 2005. p. 582–91.
[35] Luo X, Chan EWW, Chang RKC. Detecting pulsing denial-of-service attacks with nondeterministic attack intervals. EURASIP J Adv Signal Process 2009;2009:8:1–8:13.
[36] Dong K, Yang S, Wang S. Analysis of low-rate tcp dos attack against fast tcp. In: Intelligent Systems Design and Applications, International Conference on 2006;3:86–91.
[37] Acunetix web application security, <http://www.acunetix.com/>; 2012.
[38] Google's official googlebot, <http://www.google.com/bot.html>; 2012.

Luis Campo Giralte received a MS in Computer Engineering from the Pontifical University of Salamanca (Madrid, Spain). Since 2000 he has been working for several International Company's developing security and network products. Since 2004 he is Professor of Computer Security in several Universities and has been working towards a Ph.D. His research interests include network and computer security, polymorphic and metamorphic exploits, covert channels, denial of service, secure protocols and network applications.

Cristina Conde is professor with the Face Recognition and Artificial Vision (FRAV) group at the University Rey Juan Carlos. She received the ME Physics degree from University Complutense, Madrid, Spain, in 1999, and the Ph.D. degree in Computer Science from the University Rey Juan Carlos, Madrid, Spain, in 2006. During 2000–2001 she was researching assistant at the University of Salamanca. Her primary research interests include Computer Vision, Image Processing, Biometrics, Pattern Recognition, Intelligent Transportation Systems and Bio-inspired computer Systems.

Isaac Martin de Diego is an associate professor with the FRAV group, University Rey Juan Carlos since 2006. He received a Ph.D. in Mathematical Engineering and a Bsc in Statistics, University Carlos III of Madrid. He was teaching assistant at the University Carlos III of Madrid, during 1999–2005. During 1994–1999 he was researching assistant at the University of Valladolid and University Autonoma of Barcelona. His research interests include Pattern Recognition, Representation and Combination of Information, and Data Mining.

Enrique Cabello received BS in Physics (Electronics) from the University of Salamanca and the Ph.D. degree from the Polytechnic University of Madrid. In 1990, he joined as assistant professor the University of Salamanca. From 1998 he is at the University Rey Juan Carlos. Since 2001 he is the Coordinator of the FRAV research group. His research interests include computer architecture, image and video analysis, pattern recognition and machine learning.