Application layer HTTP-GET flood DDoS attacks: Research landscape and challenges

Article *in* Computers & Security · October 2016

DOI: 10.1016/j.cose.2016.10.005

CITATIONS
27

READS
1,493

3 authors:

Karanpreet Singh
National Institute of Technology Jalandhar
10 PUBLICATIONS **60** CITATIONS
SEE PROFILE

Paramvir Singh
University of Auckland
36 PUBLICATIONS **126** CITATIONS
SEE PROFILE

Krishan Kumar Saluja
University Institute of Engineering & Technology, Panjab University, Chndiagarh
120 PUBLICATIONS **958** CITATIONS
SEE PROFILE

Application Layer HTTP-GET Flood DDoS Attacks: Research Landscape and Challenges

A B S T R A C T

Application layer Distributed Denial of Service (DDoS) attacks have empowered conventional flooding based DDoS with more subtle attacking methods that pose an ever-increasing challenge to the availability of Internet based web services. These attacks hold the potential to cause similar damaging effects as their lower layer counterparts using relatively fewer attacking assets. Being the dominant part of the Internet, HTTP is the prime target of GET flooding attacks, a common practice followed among various application layer DDoS attacks. With the presence of new and improved attack programs, identifying these attacks always seems convoluted. A swift rise in the frequency of these attacks has led to a favorable shift in interest among researchers. Over the recent years, a significant research contribution has been dedicated toward devising new techniques for countering HTTP-GET flood DDoS attacks. In this paper, we conduct a survey of such research contributions following a well-defined systematic process. A total of 63 primary studies published before August 2015 were selected from six different electronic databases following a careful scrutinizing process. We formulated four research questions that capture various aspects of the identified primary studies. These aspects include detection attributes, datasets, software tools, attack strategies, and underlying modeling methods. The field background required to understand the evolution of HTTP-GET flood DDoS attacks is also presented. The aim of this systematic survey is to gain insights into the current research on the detection of these attacks by comprehensively analyzing the selected primary studies to answer a predefined set of research questions. This survey also discusses various challenges that need to be addressed, and acquaints readers with recommendations for possible future research directions.

Keywords: systematic survey, application layer DDoS attacks, denial of service, HTTP-GET flood, sophisticated attacks.

1. Introduction

Internet was originally designed to serve the basic requirement of data transfer between systems with a very little focus toward the underneath security concerns. Hence, it remains vulnerable to a variety of attacks, either direct or indirect, posing threat to both the service providers and their users. Among many possible attacks, Denial of Service (DoS) attacks are one of the eminent threats to the Internet. As the name suggests, these attacks are launched with an intention

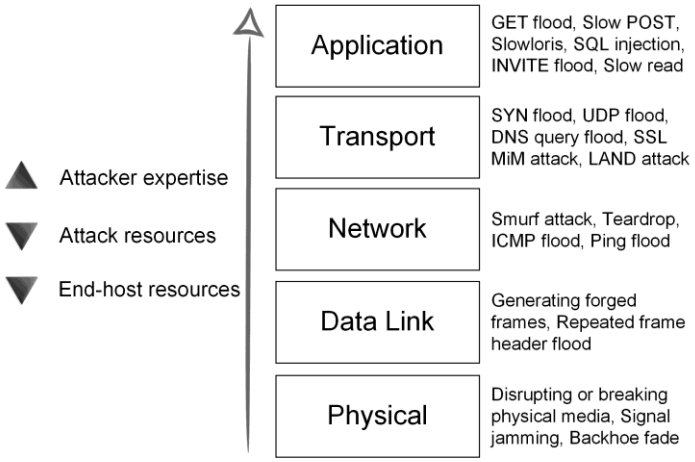

Fig. 1 - DoS attacks at different layers of network stack.

to forbid service access to the legitimate users, in contrast to some traditional network attacks where the intention is to steal and misuse the confidential data. A commonly exercised approach to accomplish this is to flood the target with a large number of unsolicited packets [1]. To further intensify the attack, an improved attack model (Distributed DoS flood) was introduced wherein a large number of differently located bots (compromised systems) are simultaneously triggered by the attacker to produce an irrepressible traffic stream. Distributed DoS (DDoS) attack starts with an attacker initially forming a network of compromised systems known as a *botnet*. This network is controlled by the attacker hidden behind several layers of bots known as *stepping stones*, with the intent to conceal its identification. The attacker initiates the attack process by disseminating commands to these compromised systems. After receiving the commands, these systems launch the actual attack flow toward the victim in an attempt to overload its bandwidth or other resources. During an attack, the victims' services become unavailable until the attack is either terminated or effectively mitigated. These attacks might sustain for a few seconds or even days, leading to huge financial losses to the service providers. The motives behind such types of attacks are generally linked to business competitions, financial gains and hactivism [2].

Typically, a DDoS attack falls into one of the three categories: volume based attacks, protocol attacks, and application layer attacks [3]. *Volume based attacks*, also known as flooding attacks, direct a large amount of unsolicited traffic toward the victim resulting in the exhaustion of infrastructure-level bandwidth of the victim. Attacks like ICMP flood, UDP flood, etc. fall under this category. Attacks exploiting the vulnerabilities of various protocols present at the network layer are categorized under *protocol attacks*. These include SYN flood, Smurf DDoS attack, etc. Finally, the attacks that abuse the configurations and functionalities of various application layer protocols and services are classified as *application layer attacks*. These include Slowloris, HTTP-GET flood, etc. Every single one of the above discussed attacks are directed toward a particular level of Internet's network stack. Fig. 1 represents some common denial of service attacks on these different levels.

1.1. Motivation

Many industries and organizations are capitalizing on the power of the Internet to provide their customers with an abundance of online services. These web services include banking, education, entertainment, electronic mailing, voice over IP and more. Most of these services make use of one or more application layer (top-most layer of network stack)

protocols like HTTP, SMTP, DNS, NTP, etc. In view of this fact, the attackers have advanced to a higher level of sophistication by relocating their attack targets to the application layer. Such attacks, inherent to their characteristics, are generally termed as application layer DDoS attacks. As fortifying every facet of an application is nearly impossible, these attacks concentrate on individual susceptible areas instead of unleashing a massive traffic flood toward the victim. It is evident that in this modern era, no system can be perceived as an absolute secure system. Hence, with the increasing number of online applications being offered these days, the likelihood of such attacks is also increasing. Incessant evolution and dynamic behavior of application layer DDoS attacks make them more formidable in comparison to the lower layer attacks. There has been almost 60 percent increase in application layer DDoS attacks from Q1-2014 to Q1-2015 [4].

An application layer DDoS attack is more or less concentrated on the unique vulnerabilities of application layer protocols. Vulnerability may relate to application functions, configurations, features, etc. The attacker initially disguises as a legitimate user and acquires access to the victim's resources, with an aim to furtively assault the running services. There are many different methods exercised by the attackers for exploiting the vulnerabilities present at the application layer. One of those methods is HTTP-GET flood attacks which, according to a report by Akamai [5], were very much prevalent among attackers' communities in the last quarter of the year 2014. It functions by letting the army of bots to throng the server with GET requests in order to overwork the latter's resources. A major strength of this attack is its ability to evade network and transport layer security mechanisms. In the absence of a suitable protection mechanism, almost every web server present on the Internet is exposed to such attacks. In the view of that, this survey makes an effort to comprehensively explore the state-of-the-art defense solutions against HTTP-GET flood DDoS attacks.

1.2. Contributions

The major focus of an HTTP-GET flood DDoS attack is toward contriving attack traffic that closely simulates legitimacy of a human user. Thereby it becomes harder for a victim to differentiate between legitimate and attack traffic [S20]. A mere detection of attacks is not the only objective here. Instead eradicating the attack effect by characterizing each user is considered a more important task. An extensive ongoing research aims at delivering solutions to the problem of mitigating HTTP-GET DDoS attacks. Devising effective and real-time defense mechanisms is need of the hour as these harmful attacks are rising each year [6].

The survey performed in this paper incorporates a 'systematic' approach to establish a substantial and complete database of the state-of-art literature focusing on detection of HTTP-GET flood DDoS attacks. A systematic approach uses a well-defined methodology to identify, analyze and interpret all available evidence related to a specific research question in a way that is unbiased and (to a degree) repeatable [7]. Only high quality studies were selected for the final survey based on a quality assessment performed by the authors. The research questions formulated in this work are answered by analyzing the data extracted from a final set of primary studies (also referred as PRs). The major contributions of this survey can be summarized as follows:

- Coverage of all high quality works in the field of detection of HTTP-GET flood DDoS attacks using a systematic approach.
- Taxonomy of the potential attack strategies and in-depth examination of detection attributes.

- Exploration of approaches and modeling methods followed by various detection techniques.
- Mining and categorization of the traffic datasets and various software tools hired by the selected primary studies.
- Suggesting promising future research directions through a careful analysis of various limitations and challenges.

The content of this paper is structured as follows. We discuss the related work in Section 2. A detailed background on HTTP-GET flood DDoS attacks is presented in Section 3. Section 4 explains the survey protocol designed for this study. Section 5 provides mapping results of the selected primary studies along with the answers to the research questions. Various limitations and challenges in the field are discussed in Section 6. Section 7 outlines the validity threats to this survey. Finally, Section 8 concludes this paper.

2. Related work

After comprehensively exploring the available literature, we found a small number of surveys dealing with defense related works against HTTP-GET flood DDoS attacks. Vadlamani [8], in his master's thesis, surveyed some well-known detection studies against HTTP-GET flood DDoS attacks. In spite of analysing the pros and cons of various detection studies, it failed to provide an in-depth and exhaustive coverage of the current literature. Moreover, the survey was limited to only thirteen detection studies. Wong *et al.* [9] discussed a number of attacks linked to network layer, transport layer and application layer DDoS in a very limited scope. Only four detection studies against HTTP-GET flood attack were investigated. Most recently, Mantas *et al.* [10] offered a taxonomy focusing on broadly categorizing different application layer DDoS attacks. It examined working of various application layer DDoS attacks along with HTTP-GET flood attacks. This work does not considered the attack strategies followed by an attacker to instigate an HTTP-GET flood attack. Moreover, the detection approaches to defend against these attacks have not been explored.

Nonetheless, there are a number of high quality surveys already available that address DDoS attacks primarily at network and transport layers. For instance, Peng *et al.* [11] provided an exhaustive overview of several lower layer DDoS attacks along with the mechanisms used to defend against these attacks. It investigated the strengths and weaknesses of various defense mechanisms to provide the researchers with avenues for the future research. Sachdeva *et al.* [12] evaluated various DDoS defense techniques based on some key metrics like deployment, robustness, security, etc. Zargar *et al.* [13] initially proposed a taxonomy of defense mechanisms against flooding based DDoS attacks. Various defense mechanisms are then investigated based on the proposed classifications, which also include studies on detection of HTTP-GET flood attacks as well. However, the survey was confined to around fifteen detection studies against HTTP-GET flood attacks.

Our work differs from the above mentioned existing surveys in that we followed a systematic approach to conduct an exhaustive survey and to provide in-depth details of literature focusing on detection techniques against HTTP-GET flood DDoS attacks. A systematic approach for surveying the literature initially gained popularity in the field of medicine, public health, etc. In computer science domain, such surveys are common in the field of software engineering ([14], [15]). A number of surveys also exist in the area of cloud computing ([16]–[18]). In a recent work [19], we conducted a systematic review on IP traceback schemes that covered various approaches, functional classes, and evaluation metrics of IP traceback techniques along with challenges and possible future avenues.

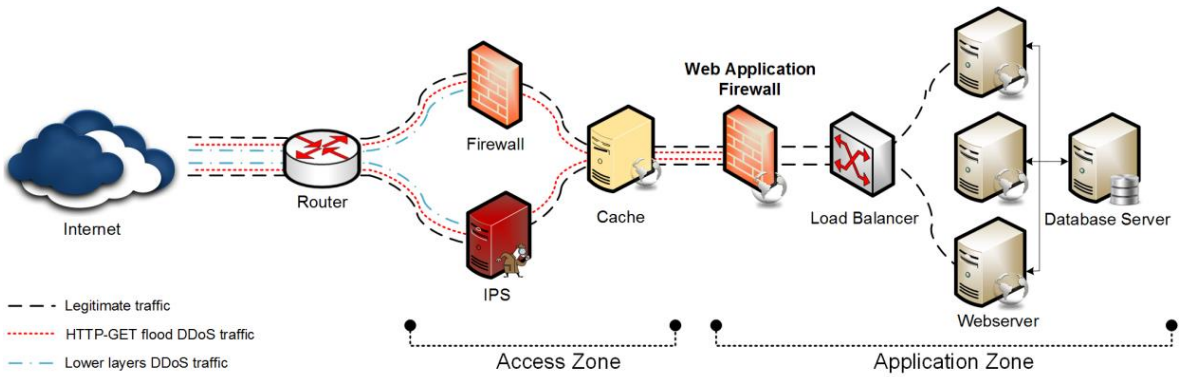

Fig. 2 - A typical scenario of a server site receiving legitimate and attack traffic over the Internet.

3. HTTP-GET Flood DDoS attacks

The Internet is highly dominated by the presence of the World Wide Web (WWW) [20], a system of interlinked documents known as web pages, that allows a user to access a diverse assortment of online information and services. WWW uses HTTP as its underlying protocol that manages the logic behind the transmission of web pages from one system to another. Almost every high-end commercial or non-commercial organization owns a website to provide the customer with uninterrupted availability of a variety of services. The total number of websites active globally will soon cross one billion mark [21]. Considering the expanding base of online services, the attacks targeting these services are also envisaged to rise. Among all application layer protocols, HTTP is the most targeted protocol due to its wide-range integration with online services [22]. Moreover, attackers are conscious that HTTP traffic is not blocked by any security equipment by default. Since HTTP-GET flood attack is one of the most common types of application layer DDoS attacks, the term 'application layer DDoS attacks' is often interchangeably used with the HTTP-GET flood attacks in literature ([S20], [S27], [S44], [S48], [S50] and [S59]). This systematic survey focuses on covering the available literature on detection of HTTP-GET flood attacks.

Lower layer flooding attacks do not require the attackers to have any special skills as compared to the more demanding application layer attacks, which require an attacker to be highly proficient in targeting specific vulnerable areas. HTTP-GET flood DDoS attacks differ from lower layer flooding attacks in the following ways [S50]:

- *Valid TCP connection*. The bots themselves complete a successful TCP 3-way handshake before actually launching an HTTP-GET flood attack. The bots continuously exchange data over these connections with the server.
- *Legitimate IP address.* Because the attackers use legitimate IP addresses, the anomaly detection schemes based on identifying spoofed IP addresses are not entirely successful. Even blacklisting of IP addresses proves to be an unproductive solution.
- *Mimicking humans*. The bots tend to mimic the access behaviors of the legitimate users, due to which it becomes infeasible to filter the attack traffic without proactively apprehending the source as a human or a bot.
- *Evade lower layer detection.* Lower layers of the network stack do not witness any significant anomalies when under an application level attack. As a result, many lower layer detection mechanisms are not able to extract enough information necessary to identify the presence of application layer attacks.

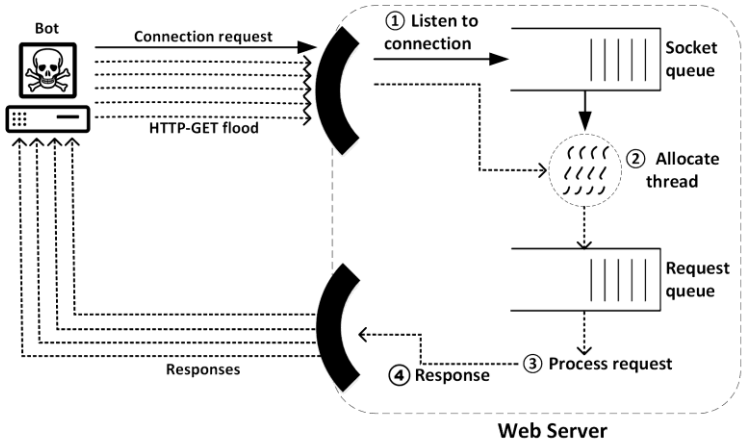

Fig. 3 – HTTP-GET flood DDoS attack against a typical web server architecture.

A contemporaneous struggle over the years between security vendors and attackers has led to smarter protection mechanisms and increasingly more complex attack routines. Fig. 2 illustrates in what manner the HTTP-GET flood attack traffic passes through lower layer security equipment. The *access zone* consists of various network and transport layer equipment such as a router, firewall, Intrusion Prevention System (IPS), etc., whereas the *application zone* contains resources like web and database servers. The traffic to the internal network is initially routed to a firewall, IPS or other security equipment. The requested resources are delivered to the user if present on web cache, otherwise the load balancer forwards the request to the web server. The legitimate traffic is inspected and allowed to pass through the network, whereas lower layer attack traffic is blocked at the access zone. Due to the legitimate procedure used for communication, HTTP-GET flood attack traffic is able to breach security premises protected using network and transport layer defense mechanisms. To secure the application services on the Internet from such attacks, the security firms provide Web Application Firewalls (WAFs) whose absence enables the HTTP-GET flood attack traffic to effortlessly reach the target server. WAFs can be either installed as supplementary plugins on the existing server or deployed as standalone appliance to exclusively scrutinize the application level communications between the server and clients. A number of security procedures are implemented on WAF in order to defend a web server from various application layer attacks. This survey explores those security procedures that have been proposed by the researchers in form of individual primary studies.

3.1. Susceptible server architecture

The inherent operational design of a server makes it vulnerable to the GET flood attacks. A typical web server architecture is illustrated in Fig. 3. A client initiates a connection process by sending a request to the server. The socket queue holds all the connection requests until dedicated threads are allocated to handle those requests. A client sends service requests (GET requests) to the server only after a TCP connection is established. All these service requests are accumulated in the request queue where the scheduler subsequently processes and responds to individual requests.

Just as any other legitimate user, a bot also initially establishes an authentic connection for communicating with the server. It then sends a large number of HTTP GET requests to the server and awaits responses much like a legitimate user. During an HTTP-GET flood attack, the requests from bots quickly get accumulated in the request queue, which

leads to dropping of subsequent incoming requests sent by the legitimate clients. A key challenge for the server here is to classify the originators of these requests as bots or humans. To exacerbate the situation, bots even try to mimic a legitimate user's access behavior that, if successful, defies the logic behind various modern-day attack detection techniques. The server keeps processing and responding to the requests received from bots as it considers them legitimate. Attacker keeps the web server engaged in continuous processing by generating requests at an above normal rate, thereby degrading the overall service quality delivered to the legitimate users.

3.2. Humans and bots

The attack requests sent to a server are generated using various compromised machines whose respective owners may be unaware that their machines are being exploited (as a bot) for orchestrating an attack. Therefore, it becomes vital for the receiving end to arbitrate the incoming traffic and suppress requests belonging to such malicious machines. The foremost challenge in handling HTTP-GET flood DDoS attacks is indeed restricting bots from accessing the web server by all possible means.

The bots can be grouped into good or bad bots based on their integral characteristics. The former are the software programs such as spammers, scrapers, etc. that do not intend to harm or disrupt the functionalities of web services. The latter on the other hand are actually responsible to carry out multiple attacks. Surprisingly, the bots contribute a higher percentage of total traffic received by a website as compared to the traffic received from humans [23]. The detection mechanisms distinguish between the legitimate users and bad (attacking) bots by the means of traits associated with them. Some actions or features usually associated with the legitimate users are as follows:

- use bookmarks to open web pages;
- navigate to a web page through search engines;
- make use of hyperlinks to navigate among web pages;
- generate legible mouse click and scroll events on web pages;
- likely to request for popular web pages;
- hardly ever repeat their web access patterns;
- rely on legitimate web browsers to connect to the server;
- exhibit widely dispersed geographical distribution.

Some of the few properties related to attacking bots are as follows:

- infected by the presence of malwares;
- await attacker's command before initiating an attack;
- provide false identity by faking user-agent strings;
- almost similar access patterns among all attacking bots;
- access behavior more often than not deviates from legitimate users;
- have a tendency to re-iterate their access patterns;
- concentrated geographical distribution at various locations.

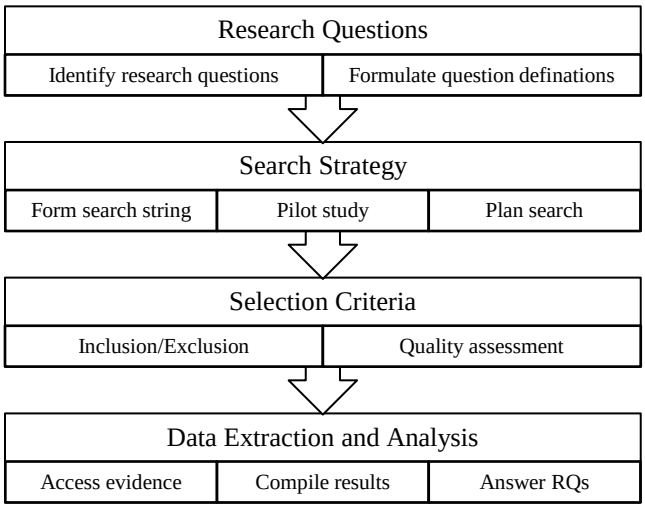

Fig. 4 - Flow chart of systematic survey process.

Various techniques such as JavaScript test, cookie test, etc., that were initially considered effective in handling HTTP-GET flood DDoS attacks, now seem ineffective due to the emergence of more superior bots. A modern bot is well capable of handling and working with technologies like cookies, JavaScript, etc., which previously only browsers used to support. Attackers even pervert real browsers on a vulnerable machine and force it to send requests to the server. This allows the attacker to deceive detection setups into believing the request originator bot as a legitimate human visitor.

The attacker makes bots generate legitimate appearing communication traffic that introduces detection complexities especially when trying to tackle application layer attacks from lower layers. A careful examination of individual user behavior is required to identify the presence of such bots. This makes revealing HTTP-GET flood DDoS attack traces from an aggregate traffic stream a strenuous task.

In general, detection studies under HTTP-GET flood DDoS attacks can be divided into two broad classes. The first class focuses on differentiating flash events from HTTP-GET flood DDoS attacks based on aggregated traffic flow analysis ([S37] and [S49]). Such studies however fail to identify the abnormal behavior of individual users. The second class comprises detection studies that investigate the access behavior at user level to enable the detection of HTTP-GET flood DDoS attacks. Apart from these, there are studies ([S2], [S15], [S34], [S39], [S58], and [S63]) that primarily identify bots present in a server's user base. Researchers have been continuously exploring bot behaviors in order to enable the apprehension of human features that cannot be easily mimicked.

4. Survey protocol

The system of methods followed by systematic literature surveys assists in gaining an extensive understanding of the problem at hand. Emanated from the field of medicine, systematic surveying is considered to be a reliable research method [24]. It provides an effective means to gather and apprehend the literature pertaining to the problem definition. It is deemed as an efficient method to recognize any research gaps and identify avenues for future research work. Following the specifications proposed by Kitchenham *et al*. [24], we have undertaken a systematic approach to perform an exhaustive survey of the available literature related to the detection of HTTP-GET flood DDoS attacks.

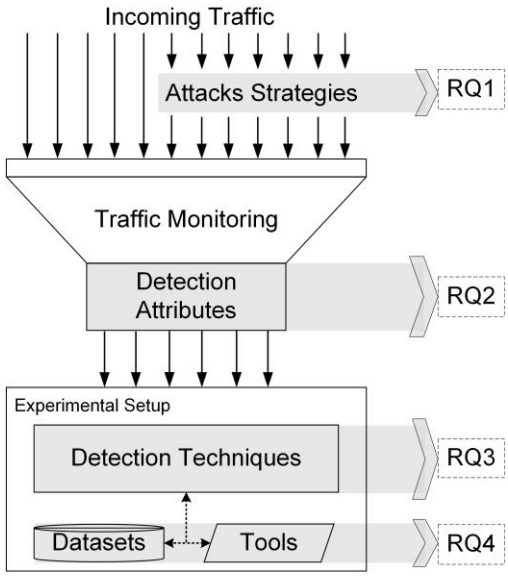

Fig. 5 - Various facets addressed by the research questions.

The validity of the survey protocol was verified by conducting a pilot study before getting down to the actual protocol execution. The conceptual view of our survey protocol is shown in Fig. 4. The foremost step of a systematic survey is to define research questions based on which the search string is formulated. This search string is then used to conduct a comprehensive literature search that will form the basis of the answers to the research questions. Various steps involved in conducting this systematic survey are detailed in the following subsections. The outcome of this study would help highlight various challenges related to the field, thereby motivating the researchers to perform further investigations.

In the following subsections, the details of survey protocol phases related to research questions, search strategy, pilot study, inclusion/exclusion criteria, reference checking, quality assessment, data extraction, and categorization method are explored.

4.1. Research questions

The aim of this systematic survey is to perform an in-depth investigation into the literature available on detection techniques for HTTP-GET flood DDoS attacks. To attain this goal, we formulate four research questions that are answered by analyzing the data extracted from the list of final qualified studies. These research questions map to the different stages of generic attack detection process as shown in Fig. 5. The research questions addressed in this paper are given below.

RQ1. What are the different attack strategies that attackers employ to launch HTTP-GET flood DDoS attacks?

RQ2. Which user and traffic attributes have been utilized in literature for the detection of HTTP-GET flood DDoS attacks?

RQ3. What are the various approaches and modeling methods that form the basis of detection techniques available in literature?

RQ4. What kinds of datasets and software tools have been used in the evaluation of various attack detection techniques?

In this survey, the focus is on holistically defining and answering the research questions, and examining the literature on detection of HTTP-GET flood DDoS attacks from multiple perspectives. These attacks can be launched with varying semantics in their efforts to outwit various detection mechanisms. In RQ1, we explore such different attack strategies that have been investigated in the literature. RQ2 attempts to identify both the elementary as well as derived detection attributes. The classification of detection techniques based on the approaches followed and the modeling methods used, is covered in RQ3. There are a wide range of datasets that have been used in the evaluation of these detection techniques. Also, the evaluation of a technique requires the use of various software tools for supporting a variety of experimental setups. RQ4 aims at exploring both the datasets and software tools that have marked their presence in the existing literature.

4.2. Search strategy

A systematic survey is initiated with a search of electronic libraries to collect the relevant literature. Being a crucial point of the survey process, formulating an effective search strategy is considered as a critical pre-requisite. In this work, an automatic search was performed in two different phases. Search Phase 1 comprised an examination of five digital libraries namely *ACM Digital Library*, *IEEE Xplore*, *ScienceDirect*, *Springer*, and *Wiley*. Search Phase 2 was assisted by scholarly search engine *Google Scholar*. The inclusion of *Google Scholar* as a source aided in building a stronger base of primary studies and avoided missing any relevant studies. The search was restricted to the article title, abstract, and meta-data in *ACM Digital Library*, *IEEE Xplore*, *ScienceDirect*, and *Wiley*. Due to the absence of such customization options, the execution of the search query on *Springer* and *Google Scholar* yielded 2041 results. Stemming of search keywords was supported by all electronic databases that helped in finalizing a shorter search query. The generic search string that was used with little modifications suiting different libraries is:

((application layer **OR** *layer 7)* **AND** *(dos* **OR** *ddos* **OR** *denial of service))* **OR** *((HTTP * flood)* **AND** *attack)*

The results obtained by executing this search query in selected digital libraries were narrowed down to the relevant fields wherever possible by using 'filtering' options. Fig. 6 represents the execution flow of the survey process along with the number of studies resulting at each stage.

4.3. Pilot study

A pilot study was performed before conducting the actual process of data collection to refine the search method. A total of 20 articles were selected for this purpose from a set of pre-collected articles stored in our database. These comprised 10 most cited and 10 most relevant articles (published between the year 2013 and 2015) as decided on by all three authors. Following it, a pilot search was carried out on *IEEE Xplore* targeting the studies published between the year 2013 and 2015. The resulting entries were passed through further stages of the survey protocol. The articles resulting from this process were then compared with the 20 selected articles. 14 out of all 20 nominated articles (70 percent) were found to be consistent with the pilot search results, which validates our search string and the survey process.

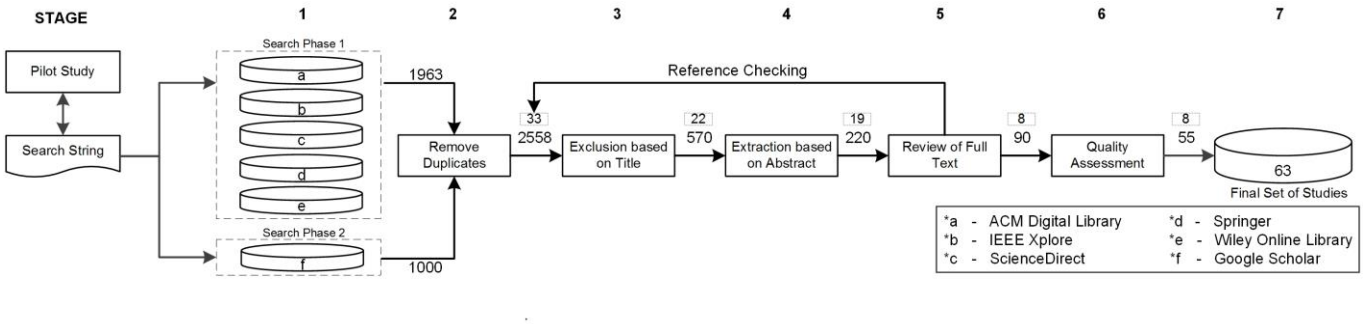

Fig. 6 - Flow illustration of systematic survey process.

4.4. Study selection

In the study selection process, inclusion and exclusion criteria were applied to filter out any insignificant articles with respect to the defined research questions. Hence only those studies that could possibly answer the research questions were added to the final list of primary studies. The study selection criteria were collectively designed taking into consideration the defined research questions. In addition to the above-defined criteria, the studies listed in Table A2 were also filtered out, which were either identical to any existing studies or extended as a new and mature study published elsewhere. Only the articles that were published as extended versions of their respective previous works were selected. The studies with ambiguous exclusion decisions were retained for analysis in subsequent stages. Search Phase 1 returned 1963 entries.

Google Scholar fetched 1000 entries in Search Phase 2, making it a total of 2963 studies in Stage 1. In Stage 2, 405 duplicate entries were removed from the list of total entries obtained from Stage 1. This was followed by the elimination of results based on their titles (excluded 1988 studies), abstracts (excluded 350 studies), and full texts (excluded 130 studies) respectively in subsequent stages. Finally, a total of 90 studies were extracted after Stage 5. Table 1 shows the count of results obtained from the selected digital sources at different stages of the search and filtering process. The inclusion and exclusion criteria used in the selection process are defined below.

The inclusion criteria checklist is as follows:

- all studies that provide a novel approach for detection or mitigation of HTTP GET flood DDoS attacks;
- studies that deal with differentiating a flash event from HTTP-GET flood DDoS attacks (flash crowd attack);
- studies that comply with research questions;
- studies that along with the detection of any other attacks have also considered detecting HTTP floods;
- studies that are closely related but vary in one or more important parameters were included as individual primary studies.

The exclusion criteria checklist is as follows:

- studies not in the English language;
- works that do not deal with HTTP flood based DDoS attacks;

- works that have been extended and published as a new article elsewhere;
- tutorials, editorials, covers, news, interviews, surveys, simulation studies, and summaries of workshops and symposiums;
- works not outlining the adequate amount of information;
- studies dedicated to lower layer low-rate attacks.

4.5. Reference checking

The references of 90 studies obtained from Stage 5 were also examined to avoid omitting any relevant work. The resulted list of 33 studies was passed back to Stage 3 for relevance assessment based on title and abstract. The studies not complying with study selection criteria were removed. After the full text analysis using inclusion and exclusion criteria, 11 studies were removed. Finally, a total of 8 studies were obtained through reference checking.

4.6. Quality assessment

A quality assessment check was performed in Stage 6 to extract only the high quality works out of 98 studies passing Stage 5. To assess the quality of each study, all authors followed the same predefined quality checklist. The quality assessment of a study was conducted by assigning every checkpoint with a score value after carefully analyzing the relevant credentials of that particular study. The average of these score values assigned by all three authors was then calculated. A study that scored higher than 0.5 was included in the final list. 55 studies from the original list and 8 studies from the list obtained through reference checking qualified for inclusion in the final list of 63 primary studies. 35 other studies were eliminated in this phase. The quality checklist comprises following checkpoints:

- Are the results of practical significance?
- Is the approach followed novel?
- Does the paper highlight implementation details?
- Is the method used to evaluate the work appropriate?
- Is the content adequate to support the research?
- Are the results explicitly stated?
- Is the study design in agreement with the research questions?
- Are conclusions drawn appropriately?
- Does the work utilize any datasets for evaluating the proposed technique?

4.7. Data extraction

To address each research question, the required data were extracted by reviewing the complete article text. A pre-designed data extraction form was initially filled with the detailed information extracted from every study. It was then refined by the other two authors keeping the research questions in mind. At the end, a total of 63 data extraction forms (spread-sheets) were prepared to facilitate the definition of responses to the research questions. A preliminary data extraction and categorization of primary studies according to year, country, and venue was performed by the first author.

Table 1 – Source-wise distribution of PRs at various survey stages.

Source	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 6
ACM Digital Library	249	173	61	22	12	7
IEEE Xplore	90	90	60	49	38	21
ScienceDirect	550	537	76	17	9	7
Springer	1041	962	141	25	11	9
Wiley	33	33	5	1	1	1
Google Scholar	1000	763	227	106	19	10
Reference Checking	-	33	22	19	8	8
Total	2963	2591	592	239	98	63

4.8. Categorization method

The classifications of primary studies performed under different research questions were independently carried out by all three authors. We followed a 3-phase method in order to put together a consistent list of categories during the course of this survey. The first phase independently carried out by the authors begins with writing down all the possible keywords to which the given input set may fall under. The common keywords among authors were selected for further exploration. Secondly, the input set was grouped based on these keywords. The process is repeated until a coherent category structure is obtained. Finally, the lack of congruence among authors while finalizing classifications of a primary study was resolved thereafter by reaching a consensus after diligent discussions.

5. Results and discussions

This section divides into two subsections providing various results that were derived from the final set of primary studies. Firstly, the mapping results of primary studies are presented. These results represent the distribution of studies according to publication type, year, and venue supported by illustrations in the form of graphs and tables. Besides this, the distribution of primary studies with regards to the contributing countries is also presented. The detailed answers to the research questions are discussed in the latter subsection. Each of the selected primary studies is assigned a unique identifier as [S*X*], where *X* ranges from 1 to 63. Table A1 lists the mapping that is used to refer a specific primary study in the following subsections.

5.1. Overview of Studies

An increase in the frequency of application layer DDoS attacks in recent years has attracted many researchers as evident from a steady increase in the number of publications from the year 2007 to 2013 (see Fig. 7). As our search phases were conducted in August 2015, the number of publications is low for the year 2015. The primary studies were mapped to countries based on the authors' affiliations as mentioned in the article metadata. It was decided that there must be at least one author (per study), affiliated to a particular country, for attributing a contribution to that country. For each study, two or more authors affiliated to the same country were considered as replicated entries and hence counted as a single

contribution. For example, if there are five authors of an article, two authors from country ‘A’ and other three from country ‘B’, then it contributes as a single instance toward each of these countries. A significant research contribution in this area has been sourced from China followed by the USA, as depicted in Fig. 8. According to a report [41], China and the USA were the countries facing the largest number of DDoS attacks in the first quarter of year 2015. Another report [42] revealed that in the second quarter of year 2015, nearly 15 percent of all application layer DDoS attack traffic emerged from China, followed by Vietnam and the USA. This necessitates continuous research efforts, especially from these countries, toward defending application layer DDoS attacks in future.

Table 2 represents the distribution of primary studies mapped to their publication types. Most of the selected primary studies (46 percent) were published in journals. The distribution of primary studies over some of the most renowned publication venues is shown in Table 3. These 63 primary studies were either published or presented across 55 different publication venues. Since the studies are highly distributed among venues, there are no channels that can be considered as the major sources of primary studies. Only the venues that contribute two or more number of studies to the final set of primary studies are listed in Table 3. ‘Others’ represents the venues that contribute a single primary study.

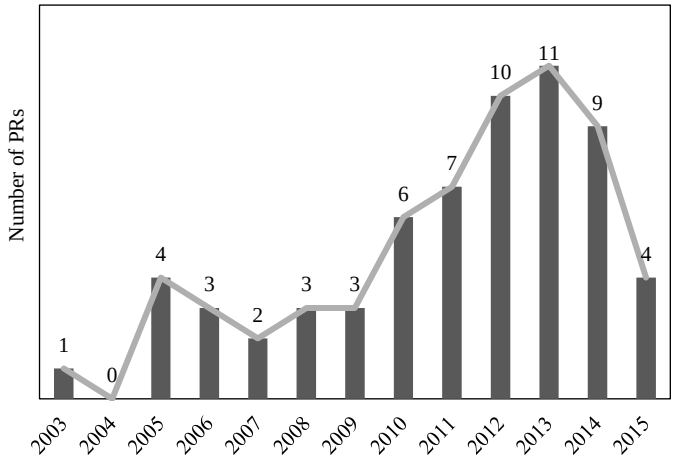

Fig. 7 - Distribution of PRs according to the publication year.

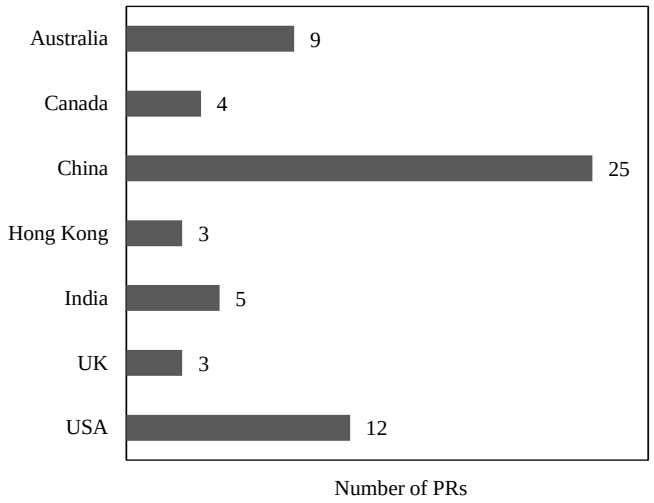

Fig. 8 - Distribution of PRs according to contributing country.

Table 2 - Distribution of PRs according to the publication type.

Publication type	No. of PRs	Percentage
Conference	27	42.9%
Journal	29	46.0%
Symposium	4	6.3%
Workshop	3	4.8%

5.2. Discussion on research questions

A total of 63 primary studies were comprehensively analyzed for the extraction of relevant data. In this subsection, the data extracted from these primary studies are used to address research questions defined in the previous section.

RQ1. What are the different attack strategies that attackers employ to launch HTTP-GET flood DDoS attacks?

In an attempt to evade attack detection, the attacker attempts to closely mimic the behavior of human users. The attacker varies its attacking strategies based on rate, access pattern, etc. to launch an HTTP-GET flood DDoS attack. Various different HTTP-GET flood DDoS attack strategies have been examined in the literature. These strategies can be broadly classified into two different categories as shown in Fig. 9. Similar attack strategies, that were found annotated with more than one label in different primary studies, were collated and represented with a unique title. The attack strategies represented in Fig. 9 can effectively be mapped to all possible attack profiles examined in the selected primary studies.

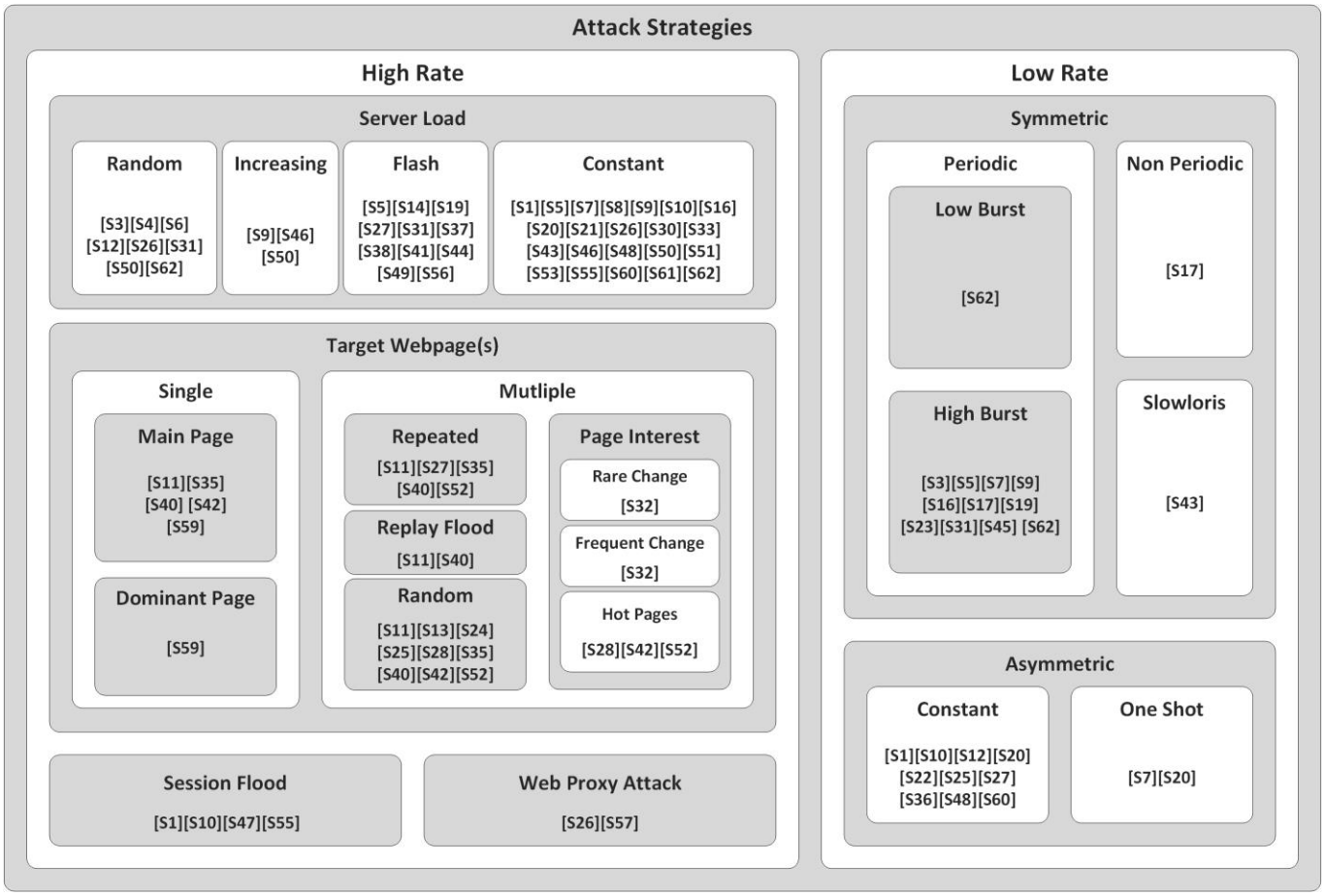

Fig. 9 - HTTP-GET flood DDoS attack strategies.

Table 3 - Distribution of PRs according to the publication venue.

Type	Venue	No. of PRs
Journal	• Computer Communications, Elsevier	3
	• Computer Networks, Elsevier	3
	• Transactions on Networking, IEEE/ACM	2
	• Transactions on Parallel and Distributed Systems, IEEE	2
	Others	19
Conference/ Symposium/ Workshop	• IEEE International Conference on Communications	3
	• International Workshop on Information Security Applications	2
	Others	29

Depending on the attack rate, HTTP targeting DDoS attacks can be divided into 'low-rate' and 'high-rate' attacks. It may be noted that low-rate DDoS attacks are behaviorally distinct for lower layers and application layer. In case of the former, attacker keeps sending packets at a slow pace with high amplitude pulses at regular intervals to exploit TCP retransmission timeout policy. This periodic burst of traffic builds up sudden congestion in the network resulting in packet losses. It forces the sender to increment its retransmission timeout thereby affecting various TCP flows. Unlike low-rate lower layer attacks that target router queues, the attacks in the latter case overwhelm the application queues of the server by sending request bursts at regular intervals. Only those studies that are associated with the detection of the low-rate application layer attacks are considered in this systematic survey. Various possible attack strategies explored in this survey are described below.

1. High Rate. The bots attack with their full capacity due to which the actual server load reaches its breaking point or sometimes overshoots it. High-rate attack strategy can be divided into two subclasses, high request rate and high session rate.

1.1. Server Load. The server resources run out as the bots keep on disseminating requests at an aggressive pace. These requests quickly use up the buffer capacity of the request queue leading to dropping of legitimate requests.

1.1.1. Random. A continuous burst of requests close to the server threshold is generated. The overall request rate keeps on ranking above or just below the server threshold.

1.1.2. Increasing. Starting from a low value, the overall request rate keeps on rising at a slow pace. This attack strategy is quite difficult to detect, as there is no sudden change of traffic rate at any moment during the attack.

1.1.3. Flash. Also known as *burst attacks*, these attacks mimic flash events making it hard for the victim to discriminate such attacks from the genuine flash events. The request rate is beyond the server's tolerable limits.

1.1.4. Constant. Attacker initially selects a constant rate based on which the bots send requests to the server. All the bots follow same request rate that does not change over time. The majority of primary studies have focused their research on countering this type of attack strategy.

1.2. Target Webpage(s). To deceive the detection mechanisms that are based on user access patterns, the attackers have exercised some attack strategies that try to imitate legitimate users. Bemusing the server, the bots access web pages

in a manner that closely mimics human users. Depending on whether single or multiple web page(s) accessed, this attack strategy can be classified into single URL or multiple URL.

1.2.1. Single. The bots repeatedly access a single web page rather than accessing multiple web pages from the collection of web pages belonging to a particular website. This attack strategy is not commonly employed by the attackers because its detection is quite straightforward.

1.2.1.1. Main Page. A single web page is repeatedly requested by the bots. The requested web page is usually the home page of a website.

1.2.1.2. Dominant Page. The server is bombarded with continuous requests for the same web page that is currently of greater interest among legitimate users. Recognizing such an attack is however straightforward as they follow a relatively simple attack strategy.

1.2.2. Multiple. Instead of targeting a single web page, multiple web pages are requested by bots during the course of an attack. The access patterns followed by this attack strategy are in close proximity to legitimate access patterns, as humans also tend to access a variety of web pages while surfing a website.

1.2.2.1. Repeated. The same access sequence is repeatedly followed by the bots. Either the sequence is pre-established or bots initially choose it at random. Due to the presence of similar request sequences among different incoming flows, this attack strategy could easily be detected.

1.2.2.2. Replay Flood. The bots try to mimic the behavior of legitimate users but at an inflated rate. Usually, this is achieved by capturing the access patterns of human users and then replaying the same at an accelerated pace.

1.2.2.3. Random. The bots access web pages on a pure random basis irrespective of their categories. All web pages are equally likely to be requested irrespective of whether they belong to the set of hot pages or not. Sometimes in order to make an attack stealthier, bots tend to randomly follow hyperlinks on web pages.

1.2.2.4. Page Interest. Web pages belonging to a website can be grouped according to the type of information they contain. For instance, news related website could group their web pages into categories like entertainment, sports, politics, etc. A user is more likely to follow web pages falling under a few selected categories of his/her interest. However, an attacker may not look for such classifications and request web pages randomly. Depending on the web page request pattern, the attack strategies under this category can be divided into three possible classes.

1.2.2.4.1. Rare Change. The bots send requests for web pages belonging to same category and very rarely switch to another category. Hence, the web pages that belong to a certain group are more likely to be requested.

1.2.2.4.2. Frequent Change. The requests for web pages belonging to different categories are generated. A user swiftly switches across these categories. However, this switching is performed randomly.

1.2.2.4.3. Hot Pages. The bots randomly request for web pages that are already being frequently accessed by the legitimate user. This makes it more difficult for the server to discriminate bots from legitimate users. There is no specific sequence in which these web pages are requested.

1.3. Session Flood. Instead of generating requests, bots tend to create new sessions without waiting for the previous sessions to terminate [S1]. The session rate is generally higher than a normal user. Multiple sessions coexist between a bot-server pair, thus overloading the server.

1.4. Web Proxy Attack. Instead of directly communicating with the victim server, the attacker makes use of web

proxies present on the Internet to act as intermediate entities, which forward the requests and receive relevant responses. Web proxies that completely anonymize its users' identities complicate the process of differentiating bots and humans.

2. Low Rate. In such attack strategies, the request rate of the bots is kept at a considerably low. The rate in some cases matches that of the legitimate users. These are further categorized into symmetric and asymmetric, based on the size of requested workloads.

2.1. Symmetric. These attacks, also known as *shrew floods*, exhibit stealthy behavior by generating periodic pulses of high request rate instead of a continuous burst. These attacks are sometimes also known as *pulsating attacks*.

2.1.1. Periodic. A burst of requests is triggered by bots at specified time intervals. The interval between two bursts is adjusted to keep the overall request rate low.

2.1.1.1. Low. Burst A period burst of requests is generated at regular intervals. The attack rate is high enough to disrupt the server for a while.

2.1.1.2. High Burst. The amplitude of an attack pulse generated in this strategy is lower than the high burst attack strategy. These attacks do not however affect the performance of high-end websites but could definitely deteriorate the functioning of low-end websites.

2.1.2. Non Periodic. The attack peaks are produced but at irregular intervals i.e., without any synchronization among bots. However, the amplitude of a peak is high enough to degrade the server performance.

2.1.3. Slowloris. Originally a software program, it is one of the perilous attack strategies that can torment a server with minimal effort. In this attack strategy, a bot creates multiple connections with the server and continuously sends partial HTTP requests at regular intervals to keep the connection open. The limit of maximum concurrent connections supported by the server is thereby reached quickly.

2.2. Asymmetric. The bots establish connections with the server and keep it busy by continuously sending streams of requests. To raise the attack intensity, bots start requesting web pages that cultivate higher workload on the server resources (CPU cycles or disk usage). Such attacks are classified as *asymmetric* because the download rate is much higher than the upload rate.

2.2.1. Continuous. The bots continuously generate requests for workload intensive content and the server is kept busy in responding to these requests. A single request may initiate multiple operations on the server.

2.2.2. OneShot. Starting with the normal request rates, bots are programmed to simultaneously send only a single high workload request at a specific time instance that chokes the victim server for a short duration. After this, bots revert to their original request rates making it difficult for the victim to identify the offenders.

Fig. 9 also maps the attack strategies to their corresponding primary studies. Here, those studies are not highlighted that are solely responsible to identify spurious bots. The literature has mainly dealt with random rate based and repeated patterns based attacks. High-rate attacks, because of their more powerful detrimental impact, have gained more attention of researchers as compared to less focused low-rate HTTP-GET flood DDoS attacks. The attack strategies classified based on the accessed URL(s) should also be considered as high-rate attacks as large number of requests accessing the web pages in a specific pattern are generated by the bots at a high rate. The problem of attack detection increases in case of bots attempting to mimic a phenomenon called *flash event*. A flash event occurs in reaction to a prominent happening

that induces a large population (legitimate users) toward a small set of web pages (hot pages). The studies that attempt to differentiate flash events from actual attacks have been classified under *Flash (1.1.3)*.

RQ2. Which user and traffic attributes have been utilized in literature for the detection of HTTP-GET flood DDoS attacks?

There is a significant difference between the detection processes for application layer and lower layer DDoS attacks. Mitigating HTTP-GET flood DDoS attacks is a complex task that is highly dependent on identifying the behavior of attacking bots. The detection of HTTP-GET flood attack mainly relies on monitoring the browsing semantics of users whereas lower layer attack detection mainly considers traffic-related characteristics.

A detection attribute is a characteristic value extracted from the coalesced incoming traffic flows of legitimate clients and the attacking bots. These detection attributes, considered either alone or in tandem with other attributes, when brought together with the standard algorithms defined under machine learning, information theory, statistics, etc. facilitate in effectively discriminating the bots from the legitimate user base. A variety of detection attributes have been obtained by the analysis of data extracted from the primary studies. Both lower layer and application layer detection attributes have been used independently as well as in combination for detecting HTTP-GET flood DDoS attacks. The elementary detection attributes along with short descriptions outlining their essence are presented in Table 4. The corresponding primary studies have also been listed. Each of these attributes is assigned a unique identifier. The attributes having high levels of similarities were treated as same during the attribute extraction process. Table 4 also lists the attribute values that a bot or a human typically possesses. The attribute values are more or less different for humans and bots, which allow the detection process to successfully characterize them apart. In Table 4, we provide a qualitative tag for each attribute in terms of *average*, *high*, *valid*, and *invalid*. In many cases, the attribute values either *follow* or *deviate* from the normal range of values as suggested by the legitimate behavioral models. It may be noted that the advanced stealthy bots could however manifest human-like behavior by closely impersonating human users.

The elementary detection attributes are divided into six classes: content, frequency, sequence, size, time, and workload. These classes were decided on after a thorough analysis of the type of information the end server is able to interpret from the captured detection attribute. The primary aim of an attack is to overload the server making it imperative for the attacker to amplify some of the attack request generating parameters. Consequentially, a large portion of these attributes belongs to the ‘frequency’ and ‘time’ classes as shown in Fig. 10.

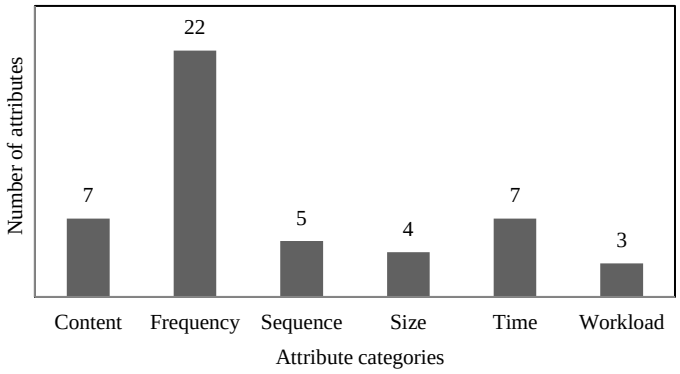

Fig. 10 - Distribution of detection attributes among different categories

Table 4 - Summary of various elementary detection attributes.

ID	Detection attribute	Description	Typical human value	Typical bot value	PRs	Class
A1	Header data	Analysis of HTTP header content like *Accept, User-Agent, Connection, Method,* etc.	Valid	Invalid	[S5]	Content
A2	Hot pages	List of pages highly accessed by users	-	-	[S5][S46][S50]	Content
A3	Out-of-time pages	Web pages that are obsolete	-	-	[S16]	Content
A4	Robot.txt file access	Indicates if user accessed robot.txt file or not	None	High	[S39]	Content
A5	Sequence number of packets*	Sequence number of a TCP segment	-	-	[S38]	Content
A6	Type/Utility of a request	Category of requested web page	-	-	[S16][S45]	Content
A7	User's response/capability	CAPTCHA puzzles, AYAH, JavaScript or cookie handling capability, etc.	Valid	Invalid	[S5][S10][S14][S15] [S49]	Content
A8	Arrival distribution rate	Arrival rate of new users during a specific period	Follow	Deviate	[S16]	Frequency
A9	Consecutive sequential requests	Percentage of web page requests at same depth	Follow	Deviate	[S34][S39]	Frequency
A10	diffReqPercent	Proportion of different category requests	Average	High	[S40]	Frequency
A11	Download rate*	No. of bytes transferred from server to user	Average	High	[S22][S34]	Frequency
A12	Flows per second*	No. of flows a server is receiving in a second	Average	High	[S25]	Frequency
A13	HTML-to-image ratio	Ratio of web page to image requests in a session	Average	High	[S34][S39]	Frequency
A14	HTTP erroneous response	Percentage of HTTP error responses	Average	High	[S34][S39][S40] [S45]	Frequency
A15	Hyperlink fraction click	Fraction of frequently clicked links on web pages	-	-	[S16]	Frequency
A16	No. of bytes in ON period*	Bytes transferred during uptime of a user	Average	High	[S62]	Frequency
A17	No. of connections to same host*	No. of active connections to a single IP address	Average	High	[S30]	Frequency
A18	No. of TCP packets*	No. of total no. of TCP packets received	Average	High	[S37][S55]	Frequency
A19	Packet rate* (Count/Average)	No. of packets received at server in a time window	Average	High	[S22]	Frequency
A20	Packets per flow*	No. of packets per flow	Average	High	[S25][S37]	Frequency
A21	Percentage of PDF/PS requests	Percentage of PDF/PS file requests in a session	Average	High	[S34][S39]	Frequency
A22	Repetitive pages	Web pages that are being requested more frequently	Average	High	[S16]	Frequency
A23	Request rate (Web page or object)	No. of requests or sub-requests in a given time window calculated usually for a user in a session (Count/Max/Average)	Average	High	[S2][S4][S7][S16] [S17][S18][S19] [S22][S25][S27] [S31][S34][S38] [S39][S41][S45] [S46][S47][S49] [S54][S55][S56] [S57] [S61][S62]	Frequency

Table 4 – (continued)

ID	Detection attribute	Description	Typical human value	Typical bot value	PRs	Class
A24	Requests of type HEAD	Percentage of HTTP HEAD requests in a session	Average	High	[S34][S39]	Frequency
A25	Requests with unassigned referrers	Percentage of unassigned referrer field requests	Average	High	[S34][S39]	Frequency
A26	Session rate (Count/Average)	Rate of creation of new sessions	Average	High	[S55]	Frequency
A27	Unsettled packets*	No. of packets that are still needed to be processed	Average	High	[S53]	Frequency
A28	UrlLevelRate	Frequency of change in the website hierarchy level	Follow	Deviate	[S40]	Frequency
A29	xFrequencyTimes	No. of user requests beyond a specific threshold	Average	High	[S40]	Frequency
A30	Hyperlink depth	Depth of a web page from root directory	-	-	[S16]	Sequence
A31	Request sequence (Sequence and length)	Sequence of web pages as requested by a user	Follow	Deviate	[S3][S8][S21][S24] [S25][S28][S29] [S33][S36][S38] [S46][S49][S52] [S57]	Sequence
A32	Sequence of categorical requests	Sequence of the categories to which an accessed web page belongs	Follow	Deviate	[S32]	Sequence
A33	Sequence of request frequency	Sequence of no. of requests received in consecutive time windows	Follow	Deviate	[S11]	Sequence
A34	Sequence of request interval	Sequence of request inter-arrival time in consecutive time windows	Follow	Deviate	[S11]	Sequence
A35	Queue length	Length of queue filled by requests	-	-	[S17]	Size
A36	Queue size	Capacity of a request queue	-	-	[S17]	Size
A37	Response size	Size of the web page requested by a user	Average	High	[S16][S26][S43]	Size
A38	Website/Webpage size	Total no. of objects/pages on a website	-	-	[S2][S4][S46][S54]	Size
A39	Downtime	Time gap after which user reconnects to server	High	Average	[S16]	Time
A40	Packet inter-arrival time*	Interval between successive packets	High	Average	[S30]	Time
A41	Request inter-arrival time or viewing time	Interval between successive requests	High	Average	[S2][S7][S20][S25] [S33][S40][S45] [S47][S55][[S62]	Time
A42	Session duration or uptime	Duration of a user activity	Average	High	[S16][S45]	Time
A43	Session inter-arrival time	Interval between successive requests	High	Average	[S20]	Time
A44	Timestamp of incoming request	Time at which a particular request is received	-	-	[S23][S45][S46] [S47]	Time
A45	Pieces of data to be processed	Resources required for partially completed requests	-	-	[S6]	Workload
A46	Request partially processed	Requests that are fractionally completed	-	-	[S6]	Workload
A47	Server resource consumption	Workload on server for a particular request	-	-	[S10][S20][S48]	Workload

*Lower layer detection attributes

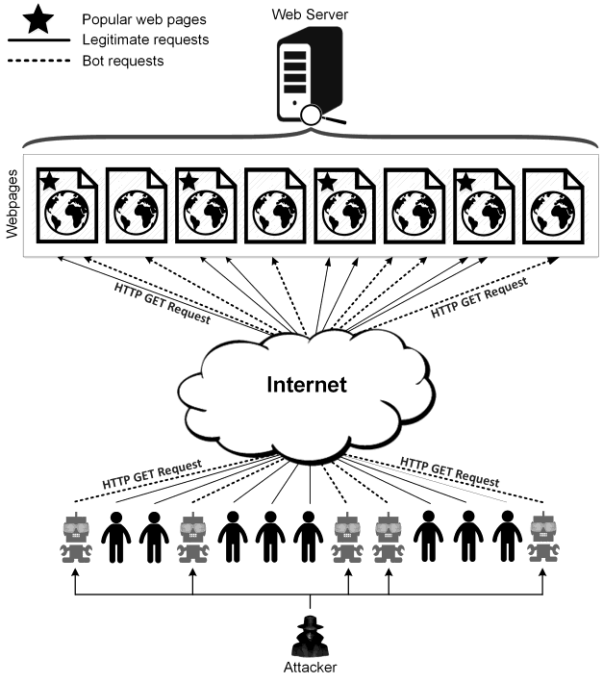

Fig. 11 - HTTP GET flood attack launched by attacker-controlled bots.

Apart from this, the congestion can be forced by producing higher server workloads. The attributes belonging to class 'workload' and 'size' mainly deal with the detection of such asymmetric attack strategies. The class 'content' identifies the type of information a web page is carrying. The sequence in which legitimate users navigate among different web pages is limited to certain possibilities due to the web page interconnections (maintained using hyperlinks). The detection attributes preserving such information belong to the 'sequence' class. This class also constitutes the detection attributes that capture series of successive user actions.

At any moment, there exists a set of certain web pages that are accessed more frequently than the others. Such web pages are known as *hot pages* or *dominant pages*. Although on average, these hot pages constitute about 10 percent of the total number of web pages (in a website), they represent more than 90 percent of the total traffic [S3]. The bots, not conscious of these hot pages, unknowingly requests the server for randomly selected web pages as shown in Fig. 11. As a result, this feature has been extensively utilized to make a distinction between legitimate users and bots. In order to capture such information, the researchers make use of two or more detection attributes to derive composite attributes as listed in Table 5. The identifiers of elementary detection attributes from which these attributes are derived are also listed in the table. The derived attributes listed in Table 5 generally capture users' interest, or in other words the popularity of web pages or objects as accessed by the users.

RQ3. What are the various approaches and modeling methods that form the basis of detection techniques available in literature?

The operational structure of a defense technique can be represented as a generalized framework illustrated in Fig. 12. This framework comprises three main modules namely monitoring, detection and filtering. A user initially sends a

Table 5 - Summary of various derived detection attributes.

Derived attribute	Description	Linked elementary attributes	PRs
Average popularity of all objects	$\frac{\text{Total no. of requests by all users}}{\text{No. of objects}}$	A23, A38	[S11]
Average transition probability	Likelihood of switching between web pages	A31	[S11]
Click average of hot web pages	$\frac{\text{Total no. of requests for all hot pages}}{\text{Total no. of hot pages}}$	A23, A2	[S5]
Entropy of overall requests	$\frac{\text{uri count}}{\text{total count}} * \log\frac{\text{uri count}}{\text{total count}}$	A23	[S5]
Entropy of requests per user	$\frac{\text{Number of requests by user}}{\text{Total no. of requests}} * lbp(\text{user})$	A23	[S56]
Entropy of destination URL	Entropy of destination web pages requested by user	A23	[S27]
Entropy of source IP address	Randomness of IP addresses of users connected to the server	A8	[S27]
Hot-Spot access	$\frac{(I-F)*V_k^t}{\sum(I-F)*V_k^t}$ I = (1, 1, … 1); F = Each page normalized access frequency; V = Data flow vector	A23	[S12]
Individual web page clicks ratio of user Y	$\frac{1}{n}\sum_{t=t_1}^{t_n} 1_{wp_i}(Y_t)$ n = observation sequence length; $1_{wpi}(y_t)$ = 1 if user Y accessed web page wp_i	A23, A31	[S42]
Intensity of the user's interest	$\frac{\text{No. of requests for greatest interest pages}}{\text{Total no. of page requests}}$	A23, A2	[S59]
Mean request interval	$\frac{\text{Sum of Inter} - \text{arrival times of requests}}{\text{Total no. of requests}}$	A41, A23	[S1]
Mean request workload	$\frac{\text{Total workload requested in a session}}{\text{Total no. of requests}}$	A23, A38	[S1]
Normalised re-visitation	$\frac{\text{Average request per user on } i^{th} \text{ document at } t^{th} \text{ time}}{\text{Average request amount per user at } t^{th} \text{ time}}$	A23	[S50][S54]
Overall web page clicks ratio	$\frac{\text{No. of requests for Page}(i)}{\text{Total no. of requests for all pages}}$	A23	[S16][S42]
Page popularity index	$-\log\left(\frac{\text{No. of requests for Page}(i)}{\text{Total no. of requests}}\right)$	A23	[S34]
RequestTimeDistribution	$\frac{\sum_{i=1}^{n-1}[d_i - \mathrm{E}(d)]^2}{n-1}$; $\mathrm{E}(d) = \frac{\sum_{i=1}^{n-1} d_i}{n-1}$ n=number of requests; d=time between two requests	A41, A23	[S40]
Stack distance	Learning access behavior with stack distance model	A23, A31	[S26][S57]
σ(Requested page's depth)	Standard deviation of page depth across all requests	A28	[S34][S39]

connection request to the server. Only after a successful connection establishment, a user is permitted to request for various resources from the server. A large number of users are simultaneously connected to the server. These users are responsible for continuously sending requests to the server. The monitoring module deployed on the server regularly captures the incoming traffic semantics at two levels of abstraction: user level and traffic level. The detection module identifies the presence of an attack based on these monitored attributes. The detection is assisted by various mathematical models that work on real-time and historical web logs of a server. Subsequent to successful attack detection, the attack traffic needs to be filtered in order to minimize its impact on the legitimate users. The filtering module can work at three levels: packet/request level, user level and session level. The filtering module also regulates the admission control policies of the incoming traffic. The majority of defense techniques investigated in this survey can be implemented on WAFs positioned ahead of the web server to interpret and filter out anomalous user traffic.

This research question is answered by exploring various modeling methods and approaches used to detect HTTP-GET flood DDoS attacks. Machine learning, information theory and statistical models are the three leading methods that form the basis of majority of present-day detection techniques. The detection attributes explored in the RQ2 are used in conjunction with these methods to single out the spurious users. Table 6 provides classification of the primary studies based on their underneath modeling method(s). Markov models and clustering are considered as a part of machine learning method. As the implementation of a significant number of detection techniques considers either markov models or clustering, these two qualified as individual categories. Apart from these two methods, all other machine learning based techniques are classified separately. There are a number of studies that do not fall under any of the above-defined classes of methods. Such studies are classified under ‘Others’.

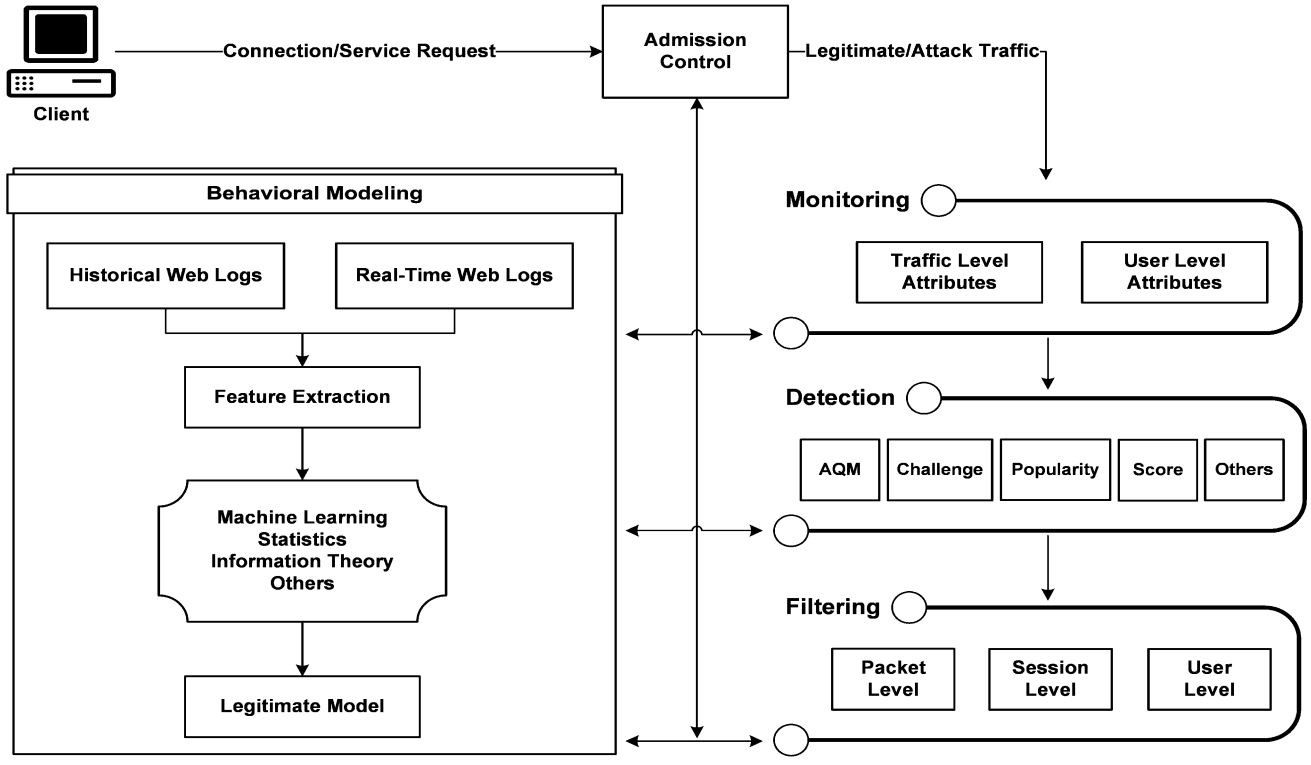

Fig. 12 - A generalized defense framework for HTTP-GET flood DDoS attacks.

Some important approaches followed by various detection techniques are discussed below.

(i) *Admission Control.* Access to the server is regulated by various admission control policies. During the initial phase of connection establishment, the server itself regulates the access or depends on a user action; based on which the final access is granted. There are two main categories under this approach:

- *Active Queue Management (AQM).* There is an active monitoring and regulation of the queue size and its contents. When required to decrease load on the server, contents of the queue are dynamically dropped based on various factors. During an attack, a queue is expected to witness more attack requests than legitimate requests.
- *Challenge*. The defense technique mandates a user to perform an action based on which accesses to the resources are granted. This approach requires the user to react to an event. Such techniques mainly rely on CAPTCHAs (Completely Automated Public Turing test to tell Computers and Humans Apart), AYAHs (Are You a Human), etc. Other techniques may test JavaScript or cookie handling capabilities among users.

(ii) *Normal Access Behavior.* The differences in the access patterns of users and bots are the key to devise detection techniques following this approach. The behavior of malicious bots differs from the legitimate users. The detection techniques cultivate on this difference to identify bots among all users. The two main types under this approach are *score* and *popularity*. The studies that do not belong to any of these two categories are classified as 'Others'.

- *Score*. Every user is assigned with a value, commonly termed as a *score* or *currency*, based on its access behavior. This value represents the credibility of a user being legitimate. The value assigned to each user is used by the request scheduler to regulate the incoming traffic when the server is under attack.
- *Popularity.* At a particular time, there exists a set of web pages that is popular among the server's user base. That is, the users are more interested in visiting these web pages. The legitimate users are supposed to request the active web pages or objects as compared to bots that usually deviate from this behavior. These trends of web page accesses by the users are used for the detection of HTTP-GET flood DDoS attacks.

Table 6 summarizes the classification of primary studies based on the underneath modeling methods and approaches along with the standard algorithms (if any) that have been implemented in those studies. A detection technique could fall under more than one category based on its inherent functionality.

The challenge based techniques have a valuable practical application but failed to grab the researchers' focus due to their inimical nature. However, majority of modern day real world web servers can be seen implementing CAPTCHA based solutions. Machine learning methods have been used to model user's behavioral patterns in a number of primary studies. Depending on the type of website, users have different access patterns that can be captured using various mathematical models. Markov models have been used in a number of studies to capture anomalous access behaviors. Normal access behavior approach has been prevalent among a majority of primary studies, since it can be applied to domains with distinct user behaviors. The detection techniques responsible to distinguish a flash event from an HTTP-GET flood DDoS attack fall short of identifying individual abnormal users.

Table 6 - Various approaches and algorithms.

PRs	Approach					Modeling method							Algorithm(s)
	Admission control		Normal access behavior										
	AQM	Challenge	Score	Popularity	Others	Markov models	Clustering	Other ML techniques	Information theory	Statistics	Others		
[S1]			✓				✓						K-means Clustering
[S2]					✓	✓							Hidden semi Markov Model
[S3]					✓	✓							Hidden semi Markov Model
[S4]				✓			✓		✓				Relative Entropy
[S5]				✓					✓				Flexible Advanced Entropy
[S6]	✓										✓		-
[S7]					✓				✓				Euclidean Distance
[S8]					✓	✓							Probabilistic Hidden Markov Model
[S9]					✓	✓							Hidden semi Markov Model
[S10]		✓									✓		-
[S11]					✓		✓						Sparse Vector Decomposition and Rhythm Matching, L-Kmeans Clustering
[S12]				✓							✓		Dempster–Shafer Theory
[S13]				✓			✓						Hierarchical Clustering
[S14]		✓									✓		-
[S15]		✓									✓		-
[S16]			✓	✓							✓		-
[S17]	✓									✓			Nonparametric CUSUM algorithm
[S18]					✓						✓		-
[S19]					✓				✓	✓			Linear Discriminant Analysis, Pearson's Correlation Coefficient, Shannon's Entropy
[S20]			✓						✓	✓			Kullback Leibler Metric, Residue Factor Metric
[S21]					✓					✓			-
[S22]	✓		✓								✓		-
[S23]	✓										✓		-
[S24]					✓					✓			Wavelets Analysis
[S25]					✓					✓			-
[S26]				✓		✓							Hidden semi Markov Model
[S27]					✓				✓				-
[S28]					✓		✓		✓				Entropy-based Clustering, Jensen Shannon Divergence
[S29]					✓		✓			✓			Entropy-based Clustering, Bayes Factor
[S30]					✓			✓					Adaptive Neuro-Fuzzy Inference System
[S31]					✓					✓			Pattern Disagreement
[S32]					✓		✓		✓				Entropy-based Clustering, Mahalanobis Distance

Table 6 – (continued)

PRs	Approach					Modeling method						Algorithm(s)
	Admission control		Normal access behavior									
	AQM	Challenge	Score	Popularity	Others	Markov models	Clustering	Other ML techniques	Information theory	Statistics	Others	
[S33]					✓					✓		-
[S34]				✓				✓				Self-Organizing Maps, Modified ART-2
[S35]		✓									✓	-
[S36]					✓					✓		Random walk model, Jacobi coefficient
[S37]					✓					✓		Flow Correlation Coefficient
[S38]					✓					✓		Pearson's Correlation Coefficient
[S39]					✓			✓				C4.5, RIPPER, KNN, Bayesian, Bayesian Network, State Vector Machine, Neural Networks
[S40]					✓			✓				Naive Bayes, RBF network, C4.5. 10-fold
[S41]					✓				✓			Fine Correntropy
[S42]				✓						✓		Reversible EWMA
[S43]					✓					✓		Hashing and collisions
[S44]					✓				✓			Relative Entropy, Kullback-Leibler Distance
[S45]					✓			✓				Principal Component Analysis
[S46]				✓		✓	✓					K-means Clustering, Hidden semi Markov Model
[S47]			✓								✓	-
[S48]					✓						✓	-
[S49]		✓									✓	-
[S50]				✓		✓						Hidden semi Markov Model, Principal Component Analysis, Independent Component Analysis
[S51]		✓									✓	-
[S52]					✓			✓				Data Streams with Concept Drift
[S53]			✓								✓	-
[S54]				✓				✓				Naive Bayes, K-Nearest Neighbor
[S55]					✓			✓				Enhanced SVM with String Kernels
[S56]					✓			✓		✓		Adaptive Autoregressive, Kalman Filter, State Vector Machine
[S57]				✓		✓						Hidden semi Markov Model
[S58]		✓									✓	-
[S59]				✓			✓					K-means Clustering
[S60]					✓	✓						Markov Games
[S61]					✓			✓				State Vector Machine
[S62]	✓										✓	-
[S63]		✓									✓	-

RQ4. What kinds of datasets and software tools have been used in the evaluation of various attack detection techniques?

The traffic traces employed for evaluating a defense technique play a significant role in establishing performance standards. These standards can then be used by the other researchers working in the field to compare their proposed techniques with the existing techniques in literature. A large collection of publicly available datasets is commonly used as traffic sources in the evaluation process of various detection techniques. These datasets can further be amalgated with traffic traces generated by different tools in order to design multiple validation scenarios. The datasets that have been used in the selected primary studies can be classified into three main categories namely academic, commercial, and benchmark.

- *Academic*. These datasets include web access logs of university, institution and department level websites. Such datasets cannot be directly downloaded, as they are not publicly available on the Internet. A number of studies ([S3], [S4], [S9], [S34], [S39], [S42], and [S57]) have utilized academic datasets. The traffic traces that are completely synthesized using various traffic generating tools are not classified under this category.
- *Benchmark*. The datasets that are freely available online for collective use are referred to as *benchmark* datasets. The researchers from around the world can easily download these datasets to evaluate and compare their proposed techniques. Due to the lack of real world HTTP-GET flood DDoS attack datasets, the benchmark datasets used as a source of background traffic are generally mixed with the traffic generated using various traffic generator software tools in order to evaluate a detection approach.
- *Commercial*. The web logs of various third-party commercial websites are considered as *commercial* datasets. These datasets also include web logs that are obtained by academicians and researchers from various Internet Service Providers (ISPs). Such datasets are not freely available for individual use. The traffic traces used in studies [S7], [S25], [S26], [S27], [S32], and [S36] can be regarded as commercial datasets.

Table 7 enlists benchmark datasets along with the primary studies that have utilized those datasets. Here, only the benchmark datasets are listed, as academic and commercial datasets are not publicly available. There were 28 studies with missing or insufficient information about the used datasets.

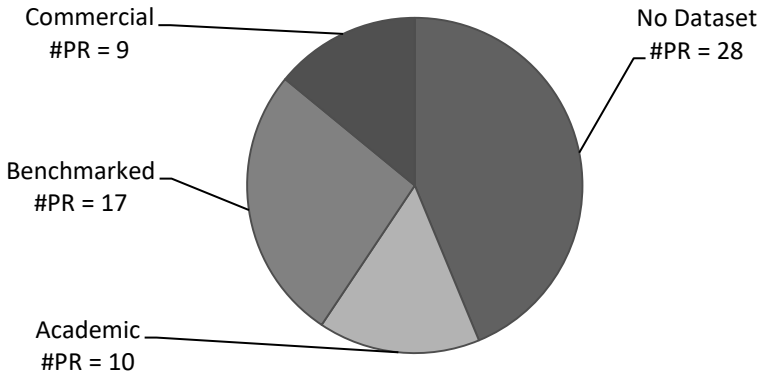

Fig. 13 - Categories of datasets used in selected PRs.

Table 7 - Benchmark datasets explored in PRs.

Dataset	Online Reference	Year	PRs
CAIDA conficker	http://www.caida.org/data/passive/telescope-3days-conficker_dataset.xml	2008	[S30]
CAIDA DDoS	http://www.caida.org/data/passive/ddos-20070804_dataset.xml	2007	[S30]
Clarknet	http://ita.ee.lbl.gov/html/contrib/ ClarkNet-HTTP.html	1995	[S11][S16][S24][S28] [S29][S40]
EPA WWW server	http://ita.ee.lbl.gov/html/contrib/EPA-HTTP.html	1995	[S13]
FIFA world cup	http://ita.ee.lbl.gov/html/contrib/WorldCup.html	1998	[S2][S19][S37][S38][S46] [S49][S50] [S54]
LBNL trace archive	http://ee.lbl.gov/anonymized-traces.html	2003	[S31][S49]
MSNBC.com	https://kdd.ics.uci.edu/databases/msnbc/msnbc.html	1999	[S32]
Mstream	http://www.ll.mit.edu/ideval/data/2000/LLS_ DDOS_1.0. html	1999	[S19][S37][S38]
NASA web server	http://ita.ee.lbl.gov/html /contrib/NASA-HTTP.html	1995	[S24][S28][S29][S49]
UCI KDD cup	https://kdd.ics.uci.edu /databases/kddcup98/kddcup98.html	1998	[S30][S31]
UNINA traffic traces	http://traffic.comics.unina.it/Traces/ttraces.php	2007	[S30]
USC ISI web server	http://www.isi.edu /ant/traces/dataset_list.html	2008	[S49]

Out of the rest, 17 studies used benchmark datasets as traffic sources for evaluating their work. FIFA world cup [48] and Clarknet [49] have been used in the majority of studies. The distribution of primary studies against the above mentioned dataset categories is shown in Fig. 13. The study by Choi *et al.* [S7] is accounted for both academic and commercial categories as it has utilized datasets pertaining to both of the categories. A wide variety of software tools assists the researchers in developing and implementing their proposed techniques. Various software tools used along with the respective online source (web-link) referring to these tools are listed in the selected primary studies are presented in Table 8. The studies associated with the software tools have been specified whose online source were not available. Only the studies that contain sufficient information about the software tools employed to assist the detection techniques have been included in Table 8.

Various software tools employed in the primary studies can be divided into following three categories:

(i) *Validation Tools.* These are the software tools that help in realizing the proposed techniques in practice. Determining the performance of a proposed technique is an essential part of a research work. In the area of computer networks, there are two important validation modes i.e., network simulation or emulation testbeds.

- *Simulators*. NS2 and NS3 are the two most dominant open source network simulators that have been extensively used in the area of networking. Other commercial software tools like MATLAB and OPNET have also been used to evaluate different works. Various machine learning based detection techniques have been validated using the WEKA tool, whereas modeling based techniques generally utilize MATLAB for the same.
- *Emulation testbeds.* Emulation testbeds provide results that closely represent real world environments. PlanetLab and Emulab are the two very well-known testbeds that support a wide range of deployment and customization options for implementing multitude of proposed detection techniques.

Table 8 - Summary of various software tools.

		Software tools	Online Reference	PRs
Validation tools	Simulators	MATLAB	http://in.mathworks.com/products/matlab	[S11][S17][S24][S27][S32][S34][S60]
		NS2	http://www.isi.edu/nsnam/ns	[S10][S23][S26][S50][S57][S62]
		NS3	https://www.nsnam.org	[S42]
		OPNET	http://www.riverbed.com/in/products/ steelcentral/opnet.html	[S44]
		WEKA	http://www.cs.waikato.ac.nz/ml/weka	[S11][S39][S40]
	Emulation testbeds	Emulab	https://www.emulab.net	[S16][S18]
		PlanetLab	https://www.planet-lab.org	[S14][S51][S63]
	Traffic generators	Apache JMeter	http://jmeter.apache.org	[S5][S45]
		Black energy	Refer to [43]	[S7][S61]
		BlueCode.Worm Netsky.Q Trojan_Sientok	Refer to [S16] and [S33]	[S16][S33]
		DDoSim	http:// sourceforge.net/ projects/ddosim	[S31]
		DoSHTTP	http://doshttp.soft32.com	[S57]
		Httperf	http://www.hpl.hp.com/research/linux/ httperf	[S17]
		LOIC	http://sourceforge.net/projects/loic	[S25]
		Metasploit	http:// www .metasploit.com	[S30]
		Netbot	Refer to [44]	[S7][S43]
		Netwag tool	http://ntwag.sourceforge.net	[S30]
		PHARM	http://pharm.ece.wisc.edu	[S48]
		Siege	https://github.com/JoeDog/siege	[S25]
		Sloworis	http://sourceforge .net/projects/slolorisgui	[S25][S43]
		TFN2K	https://github.com/greath/tfn2k	[S30]
		Webstone 2.5	http://www.mindcraft.com/webstone	[S14]
	Server software	Apache	https:// httpd.apache.org	[S5][S18][S25][S48][S63]
		Mathopd	http://www.mathopd.org	[S14]
		Mini-HTTPd	http://acme.com/software/mini_ httpd	[S17]
		Quagga and iproute2	https://wiki.gentoo.org /wiki/Quagga	[S5]
		SFS toolkit	Refer to [45]	[S18]

(ii) Traffic Generators. Due to the lack of available datasets containing traces of HTTP-GET flood DDoS attacks, the researchers make use of various software tools to fabricate such attack workloads (traffic). These software tools generate synthetic traffic traces to be used during technique evaluation. Such software tools available online vary in their capabilities to launch a variety of application layer DDoS attacks.

(iii) Server Software. When working in an emulated environment, the victim that usually is a server can be implemented on a computer machine using specific server software. This paper reports some widely used server

software in the primary studies under consideration. It is observed that the open source server software, Apache HTTP server, has been commonly employed by a number of studies as depicted in Table 8.

6. Limitations and challenges

The growing interest in the field of detection of HTTP-GET flood DDoS attacks is seeing many vital issues and challenges emerging from the recent research. In this section, we discuss various challenges, either partially addressed or pending, faced when working toward countering HTTP-GET flood DDoS attacks.

(i) Mimicking Bots

A large portion of literature makes use of different mathematical models to filter out anomalous users, particularly bots. Such detection techniques rely on training these models with the legitimate user behavior. The users that deviate from the normal range of modeled parameters are usually classified as bots. In a study performed by Yu et al. [S41], it has been observed that bots can even mimic behavioral patterns of legitimate users to an extent that may defy the underlying logic behind the detection of such attacks, thus making most of the present-day solutions look futile.

(ii) NATs and Web Proxies

The detection complications aggravate with users sharing the same set of IP addresses from behind the NAT (Network Address Translation) or proxy servers. Capturing request semantics in the presence of proxies, caches, and NATs may induce deceptive results. An attacker may skillfully deploy its bots behind proxies and NATs such that the actual request semantics of individual users become difficult to record, which in turn impedes the overall detection performance. Whitelisting and blacklisting IP addresses in such cases also become infeasible.

(iii) Unavailability of Attack Datasets

Many studies have exercised their proposed techniques on a decade old datasets. These datasets are not only used as a source of background traffic but also for training mathematical models. However, these datasets contain browsing semantics that do not necessarily reflect the behaviors of present-day users. Most of the researchers have used their university/organization web logs instead of benchmark datasets to evaluate their proposed techniques. As these datasets are collected from different sites, the semantics of browsing behaviors vary drastically. Comparing performances of detection techniques based on these datasets hence become meaningless.

(iv) Lack of Standards defining Attack Strategies

In the absence of any benchmark datasets containing traces of traffic defining real world application layer DDoS attacks, researchers employ various software tools to fabricate attack traffic. The traffic semantics pertaining to different attack strategies generated from these tools vary across primary studies due to the lack of standards governing the parameters used.

(v) State Monitoring Overheads

Some detection approaches introduce exorbitant complexities with increase in website strength (in terms of either

number of web pages or active users). It sometimes becomes infeasible for such detection techniques to work under real-time conditions. Moreover, tracking each user's request semantics could result in inflated detection complexity or reduced scalability. The detection techniques that require tracking user characteristics, like type of request, access sequence, etc., suffer from state monitoring overhead as maintaining such parameters for popular websites with a large number of active users is quite difficult.

(vi) Sophisticated Bots

Bots nowadays are evolving with multiple capabilities. Attackers are constantly reforming the bot designs to support various features that previously only legitimate browsers used to exhibit. Bots are becoming more and more competent against traditional detection mechanisms. The abilities like handling cookies, working with JavaScript codes, etc. are some of the prime bot features these days. The bots that are able to process hyperlinks on a web page can also mimic the request sequences of the legitimate users. Identifying the presence of such bots is a cumbersome task.

(vii) Enraging Graphical Puzzles

One of the widely adopted detection solutions against HTTP-GET flood DDoS attacks is to challenge the user with one or more forms of visual puzzles. These puzzles, may it be CAPTCHAs or AYAHs, are still extensively implemented and considered as the most effective methods to identify bots from a pool of large number of users. Though simple and easy to implement, the legitimate users may get annoyed when required to repeatedly solve such puzzles. Instead, an equally effective detection solution, that works without expressing the underneath operations would be more acceptable to both users and service providers.

(viii) Impracticable Offenses

There are approaches found in literature that follow offensive mechanisms to protect legitimate users from being influenced by HTTP-GET flood DDoS attacks. The underlying assumption is that during an attack, bots would be utilizing their full bandwidth whereas the legitimate users are expected to be left with some free bandwidth as they are communicating at much lower rates. The major downfalls of such mechanisms are that, firstly such assumption does not necessarily hold; and secondly, instead of scaling down the load on the server, insignificant traffic is generated during the process that may result in overloading the whole network.

(ix) Complex Attack Procedures

A detection technique should be well equipped with routines that are capable of dealing with tricky manoeuvres (see Fig. 9) employed by the attackers. Handling such cases requires extensive knowledge of various tactics that an attacker may employ to instigate an attack. The detection proficiency of a technique lies in successfully detecting such potential attack strategies.

(x) Static and Small-scale Websites

Popularity is a widely used phenomenon relying on the fact that at any particular moment there is always a set of web pages that dominates overall website traffic. Legitimate users are more likely to access these web pages in comparison to bots that may randomly access any web pages. This however applies to websites whose content popularity changes

rapidly over time. Other websites (mainly static) that do not belong to such a group become vulnerable if an attacker gets hold of current popular web pages.

(xi) Sensitive to Control Parameters

Various mathematical models used in the traffic characterization process require setting up of different control parameters. Formulating such parameters occasionally requires experience and training as different websites possess different user access characteristics. The performance of a detection technique could drastically plunge due to inexperienced personnel engaged in dealing with such situations.

(xii) Characterizing Legitimate Bots

Online search engines employ a huge set of machines to access various attributes of a website using web crawlers (also known as legitimate bots). The indexing systems then process the crawled web pages to determine their respective ranks. It is critical for an application layer defense mechanism to distinguish legitimate bots from spurious bots in order to avoid any mal-functioning related to indexing and ranking of the websites.

Although, the limitations and challenges discussed above could serve as the areas for future research work, there are some points generalized from the primary studies that are worth mentioning. The first one being the assumption 'sufficient number condition' which states that the number of active or live bots are considerably less than the number of legitimate users connected to a server [S41]. In all the PRs, we found that the detection techniques are evaluated using fewer bots in contrast to the legitimate users, making this assumption implied directly or indirectly. However, in times to come, it might get workable for an attacker to convene a large army of live bots (more than tens of thousands), also known as superbots [46], against which the detection techniques entirely reliant on this assumption will collapse. This can be attributed to the fact that the bots, which were initially enforced to flood a victim with large number of requests, are now allowed to continue at a similar pace as of a legitimate user. As a large portion of literature has exploited the frequency based detection attributes (as depicted from Table 4), the presence of superbots will definitely challenge the effectiveness of these techniques. Also, small-scale web servers are vulnerable even in the absence of superbots since it is possible for an attacker to accumulate few thousands of live bots in the present time.

Another tempting option on the subject of detection is to identify what an individual user accesses. As discussed earlier, the access pattern of a legitimate user is inclined toward hot pages i.e., the currently popular web pages. This concept has gained major attention of the researchers in the recent past. There has been a significant research effort made to represent such information in form of various composite detection attributes (see Table 5). The upcoming research should now focus on reducing the complexity of capturing such composite attributes for each user, since a web server can have an on-going communication with thousands of legitimate clients.

Another key issue arises when the present-day solutions have to deal with users of a campus or an organization networks. As discussed earlier, these users share a common pool of IP addresses. The only mode of discrimination among users behind NAT is the access to the configurations of their systems and software. As the bots can be instructed to send requests with varying user agents and system configurations, it becomes very challenging for the server to efficiently identify them among all other users. Moreover, the detection mechanism deployed at the server end is likely

to treat all users similarly which may lead to possible blacklisting of legitimate machines in that network. Yu *et al.* [S47] attempts to circumvent these issues by proposing a framework to work under such scenarios. However, it too suffers from few limitations. We believe that a potential workaround is to extend the works [S26] and [S57] that deal with web proxy attacks over the situations discussed above.

The investigation of primary studies indicates that considerable research efforts are being made in profiling the user behavior as it is important to reach out to the individual user level to distinguish legitimate from anomalous traffic. The datasets employed for this task lack traces of real world HTTP-GET flood DDoS attacks leaving the researchers with no option but to use the fabricated workloads for the evaluation of their respective detection techniques. A possible solution is that the researchers can publically share the datasets used in their respective studies. The privacy risk arising as a result can effectively be purged by the prior anonymization of datasets. This practice will make it viable for the researchers to establish coherent performance standardisation across prospective defense techniques. Various performance parameters specifically formulated for appraising defense mechanisms against application layer DDoS attack can be found in [47]-[49].

The literature is mainly divided into two sets with one targeting low rate and the other targeting high rate HTTP-GET flood DDoS attacks. Another future research prospect is a solitary solution that is capable of defending a web server against both these attack strategies. Conclusively, the rising attack complexity is pushing the researchers to dig deeper into the behavioral aspects of legitimate users in order to extricate unique features for possible distinction between legitimate and attack users.

7. Threats to validity

Several possible measures, discussed further in this section, were taken into consideration to ensure rationality of this survey. The four types of validity threats associated with this systematic survey [50] are examined below:

(i) Conclusion validity

A systematic survey is said to have reproducibility, an important characteristic, which allows other researchers to procure results analogous to that of the original study. For efficient repetition of this survey by other researchers, the information on search string, inclusion and exclusion criteria, and detailed layout of the systematic approach followed has been explicitly specified in this survey. However, this does not assure similar results across researchers attributing to the difference in their perception when dealing with the study selection criteria. The inclusion and exclusion of studies based on the study selection criteria were performed independently by the first two authors. A considerable effort was put into discussions and feedbacks arising as a result of conflicts to ensure mutual conformity during the study selection stage.

(ii) Construct validity

We followed a well-defined survey protocol for our systematic survey. However, there may be the case that the keywords used to search the electronic databases might have missed some relevant literature. In *Google Scholar*, keywords like ad-hoc networks, wireless sensor networks, grid computing, etc. were used in the search query to omit

irrelevant results from being retrieved by the search query. This might have resulted in over-sighting any relevant studies that contained such keywords. A pilot study was conducted to mitigate the effect of bias caused during the search phases. The results of this pilot study favoured our search process. Reference checking was conducted on the set of studies extracted from Stage 5 to further eliminate such bias. Other two authors were also involved in performing reference checking to generate more reliable results. The study selection phase comprises inclusion/exclusion criteria along with a quality assessment criteria based on a pre-defined checklist. The search output from *Google Scholar* used in Search Phase 2 was restricted to first 1000 results only. Consequently, any relevant results beyond the first 1000 entries could not be extracted for further survey stages. Also, the literature gathering phase of this work was conducted before August 2015 due to which we might have missed some studies that were published after August 2015. There might be a few articles of significant nature that were not able to make it to the final set of primary studies, due to either the language barrier or inaccessibility.

(iii) Internal validity

Interpreting correctly the inclusion and exclusion criteria is important so as not to include unrelated work and at the same time refrain from excluding any significant studies. To enable this, the entries obtained from Stage 4 were investigated twice. The quality assessment was carried out to eliminate studies that do not meet the standards defined for this work. These standards might have restricted us from selecting some prominent works. However, utmost care was taken in characterizing the studies against these standards. Any differences in opinions were mutually settled. Furthermore, a 3-phased process for category resolution must have ensured minimal threat to internal validity.

(iv) External validity

This survey primarily targets the detection techniques that provide defense against HTTP-GET flood DDoS attacks. The application of categories presented in this survey across different domains of DDoS attacks is not feasible since there exist a number of dissimilarities among detection techniques against different attack types. For instance, the classifications of attack strategies discussed in RQ1 entirely contrasts with other DDoS attack types. The majority of detection attributes explored in RQ2 intend to capture behavioural features at individual user level, whereas the lower layer detection attributes usually measures traffic characteristics as a whole. However, the categorization of various detection techniques based on approaches and modelling methods is partially applicable to other detection techniques. Also, there are few limitations ((ii), (iv), (ix) and (x)) that co-exist between literature domains of different attack types.

8. Conclusions

We have conducted a profound systematic survey of the available literature on detection of HTTP-GET flood DDoS attacks aiming to provide the current state-of-the-art, supported by the classification of the research landscape and identification of potential challenges. We have initially provided a brief primer on HTTP-GET flood DDoS attacks that allows the reader to gain a better understanding of the behaviors and operations of such attacks. Thereafter the systematic protocol that was used to conduct this survey is discussed. The research questions defined in this work aim to

comprehensively embrace all potential facets of the selected primary studies. These facets include attack strategies, detection attributes, detection approaches, modeling methods, datasets used, and software tools employed.

Following a systematic approach to acquire the relevant research literature, our search was conducted in two phases that comprised six different electronic databases namely *ACM Digital Library*, *IEEE Xplore*, *ScienceDirect*, *Springer*, *Wiley*, and *Google Scholar*. The search string was validated using a pilot study to ensure comprehensive literature coverage resulting from the execution of search process. In addition, the reference checking stage also consolidated our survey process. Out of 2963 search results, a total of 63 studies qualified our study selection and quality criteria. These 63 studies (PRs) were then thoroughly examined to answer the research questions.

The mapping results of primary studies indicated the dominance of research works, related to the detection of HTTP-GET flood DDoS attacks, conducted by China (as country of author(s) affiliations) followed by the USA and Australia. The final list of primary studies considered by this survey comprised studies that were published before August 2015. Post completing this survey, we conducted a final search of electronic databases (limited to the year 2015) to observe the trend of publications after that period. A quick inspection of resultant search entries in cognizance with the study selection criteria witnessed an expanding research literature on detection against HTTP-GET flood DDoS attacks with a total of 17 primary studies.

We expect that the readers will be able to acquire a detailed understanding of various elements connected to a study on detection against HTTP-GET flood DDoS attacks. Additionally, the limitations and challenges listed in this survey will stimulate researchers to carry out their investigations further into this field. Now we look forward to implement a similar study focusing on other application layer DDoS attacks.

Appendix A

Table A1 - Selected primary studies.

ID	Title	Type	Year	Citation count
[S1]	A detection and offense mechanism to defend against application layer DDoS attacks	Conference	2007	64
[S2]	A detection approach of user behaviors based on HsMM	Conference	2005	16
[S3]	A large-scale hidden semi-markov model for anomaly detection on user browsing behaviors	Journal	2009	156
[S4]	A new relative entropy based app-DDoS detection method	Symposium	2010	22
[S5]	A novel protective framework for defeating HTTP-based denial of service and distributed denial of service attacks	Journal	2015	06
[S6]	A selective defense for application layer DDoS attacks	Conference	2014	09
[S7]	AIGG threshold based HTTP GET flooding attack detection	Workshop	2012	02
[S8]	An effective defense mechanism for detection of DDoS attack on application layer based on hidden markov model	Conference	2012	01
[S9]	An HTTP flooding detection method based on browser behavior	Conference	2006	31
[S10]	An OTP-based mechanism for defending application layer DDoS attacks	Conference	2011	03
[S11]	Application layer DDoS attack detection using cluster with label based on sparse vector decomposition and rhythm matching	Journal	2015	02
[S12]	Application layer DDoS detection model based on data flow aggregation and evaluation	Conference	2012	02
[S13]	Application layer DDoS detection using clustering analysis	Conference	2012	08
[S14]	Botz-4-sale: surviving organized DDoS attacks that mimic flash crowds	Symposium	2005	402
[S15]	Captcha: using hard AI problems for security	Conference	2003	1218
[S16]	Connectionscore: a statistical technique to resist application layer DDoS attacks	Journal	2013	06
[S17]	Countermeasures on application level low-rate denial-of-service attack	Conference	2012	07
[S18]	DDoS defense by offense	Journals	2010	225
[S19]	DDoS discrimination by linear discriminant analysis	Conference	2012	10
[S20]	DDoS-resilient scheduling to counter application layer attacks under imperfect detection	Conference	2006	135
[S21]	Defending application DDoS with constraint random request attacks	Conference	2005	16
[S22]	Defending systems against tilt DDoS attacks	Conference	2011	22
[S23]	Defense techniques for low-rate DoS attacks against application servers	Journal	2010	17
[S24]	Detecting anomalous web browsing via diffusion wavelets	Conference	2010	10
[S25]	Detecting denial of service by modeling web-server behavior	Journal	2013	14
[S26]	Detecting latent attack behavior from aggregated web traffic	Journal	2013	14
[S27]	Detection and defense of application layer DDoS attacks in backbone web traffic	Journal	2014	21
[S28]	Detection of application layer DDoS attack with clustering and likelihood analysis	Workshop	2013	-
[S29]	Detection of application layer DDoS attacks with clustering and bayes factors	Conference	2013	03
[S30]	Detection of Distributed Denial of Service attacks using an ensemble of adaptive and hybrid neuro-fuzzy systems	Journal	2013	47
[S31]	Detection of HTTP flooding attacks in multiple scenarios	Conference	2011	19
[S32]	Detection of HTTP-GET attack with clustering and information theoretic measurements	Symposium	2013	02
[S33]	Detection of HTTP-GET flood attack based on analysis of page access behavior	Conference	2007	66
[S34]	Detection of malicious and non-malicious website visitors using unsupervised neural network learning	Journal	2013	35
[S35]	Detection of web Denial-of-Service attacks using decoy hyperlinks	Conference	2006	05

Table A1 – (continued)

ID	Title	Type	Year	Citation count
[S36]	Detection on application layer DDoS using random walk model	Conference	2014	05
[S37]	Discriminating DDoS attacks from flash crowds using flow correlation coefficient	Journal	2012	102
[S38]	Distributed Denial of Service (DDoS) detection by traffic pattern analysis	Journal	2014	12
[S39]	Feature evaluation for web crawler detection with data mining techniques	Journal	2012	20
[S40]	Feature extraction and construction of application layer DDoS attack based on user behavior	Conference	2014	04
[S41]	Fool me if you can: mimicking attacks and anti-attacks in cyberspace	Journal	2015	10
[S42]	HTTP-soldier: an HTTP-flooding attack detection scheme with the large deviation principle	Journal	2014	-
[S43]	Implementation of GESNIC for web server protection against HTTP GET flooding attacks	Workshop	2012	03
[S44]	Information theory based detection against network behavior mimicking DDoS attacks	Journal	2008	62
[S45]	Low-rate application layer DDoS attacks detection by Principal Component Analysis (PCA) through user browsing behavior	Journal	2013	01
[S46]	Mining web user behaviors to detect application layer DDoS attacks	Journal	2014	-
[S47]	Mitigating application layer distributed denial of service attacks via effective trust management	Journal	2010	26
[S48]	Mitigating application-level denial of service attacks on web servers: a client-transparent approach	Journal	2008	60
[S49]	Modeling human behavior for defense against flash-crowd attacks	Conference	2009	92
[S50]	Monitoring the application layer DDoS attacks for popular websites	Journal	2009	198
[S51]	New approach to mitigating distributed service flooding attacks	Conference	2012	03
[S52]	Next generation application layer DDoS defenses: applying the concepts of outlier detection in data streams with concept drift	Conference	2014	03
[S53]	Overcourt: DDoS mitigation through credit-based traffic segregation and path migration	Journal	2010	16
[S54]	Predicting application layer DDoS attacks using machine learning algorithms	Journal	2014	-
[S55]	Real time detection and classification of DDoS attacks using enhanced SVM with string kernels	Conference	2011	16
[S56]	Real-time detection of application layer DDoS attack using time series analysis	Journal	2013	02
[S57]	Resisting web proxy-based HTTP attacks by temporal and spatial locality behavior	Journal	2013	14
[S58]	Sentinel: hardware-accelerated mitigation of bot-based DDoS attacks	Conference	2008	08
[S59]	Sequence-order-independent network profiling for detecting application layer DDoS attacks	Journal	2011	21
[S60]	Strategy-aware mitigation using markov games for dynamic application layer attacks	Symposium	2015	01
[S61]	Timeslot monitoring model for application layer DDoS attack detection	Conference	2011	04
[S62]	WDA: a web farm distributed denial of service attack attenuator	Journal	2011	12
[S63]	WebSoS: an overlay-based system for protecting web servers from denial of service attacks	Journal	2005	69

Table A2 – Excluded redundant or prior studies.

ID	Title	Type	Year
[51]	A lightweight mechanism to mitigate application layer DDoS attacks	Conference	2009
[52]	Cald: Surviving various application-layer DDoS attacks that mimic flash crowd	Conference	2010
[53]	HTTP-sCAN: Detecting HTTP-flooding attack by modeling multi-features of web browsing behavior from noisy dataset	Conference	2013
[54]	HTTP-sCAN: Detecting HTTP-flooding attack by modeling multi-features of web browsing behavior from noisy web-logs	Journal	2015
[55]	Web DDoS detection schemes based on measuring user's access behavior with large deviation	Conference	2011
[56]	In-network Server-directed Client Authentication and Packet Classification	Conference	2010
[57]	Using graphic turing tests to counter automated DDoS attacks against web servers	Conference	2003
[58]	A middleware system for protecting against application level denial of service attacks	Conference	2006
[59]	Discriminating DDoS attack traffic from flash crowd through packet arrival patterns	Conference	2011
[60]	Tackling application-layer DDoS attacks	Conference	2012
[61]	Detecting Shrew HTTP Flood Attacks for Flash Crowds	Conference	2007
[62]	A novel model for detecting application layer DDoS attacks	Conference	2006
[63]	Online anomaly detection based on web usage mining	Conference	2012
[64]	Unsupervised clustering of web sessions to detect malicious and non-malicious website users	Conference	2011
[65]	Detecting web crawlers from web server access logs with data mining classifiers	Conference	2011
[66]	Application-layer DDoS in dynamic web-domains: Building defenses against next-generation attack behavior	Conference	2014

Acknowledgments

The authors would like to extend their sincere appreciation to Barbara Kitchenham (Keele University, UK) for her valuable feedbacks that helped us in conducting this systematic survey in a more effective manner.

REFERENCES

[1] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 2, pp. 39-53, 2004.

[2] K. Kumar, R. C. Joshi, and K. Singh, "A distributed approach using entropy to detect DDoS attacks in ISP domain," In: *Proc. Int. Conf. Signal Process. Commun. Netw.*, 2007, pp. 331-337.

[3] D. Kostadinov, "Layer Seven DDoS Attacks," InfoSec Institute. [Online]. Available: http://resources.infosecinstitute.com/layer-seven-ddos-attacks/.

[4] "Q1 2015 state of the internet - security report," Akamai, 2015. [Online]. Available: https://www.akamai.com/uk/en/multimedia/documents/akamai-state-of-the-internet-report-q2-2015.pdf.

[5] "Q2 2015 state of the internet - security report," Akamai, 2015. [Online]. Available: https://www.stateoftheinternet.com/resources-web-security-2015-q1-internet-security-report.html.

[6] R. Atias, "An inside look at one of the most complex DDoS attacks to date," Incapsula, 2013. [Online]. Available: https://www.incapsula.com /blog/funded-persistent-multi-vector-ddos-attack.html.

[7] B. A. Kitchenham, “Systematic review in software engineering: Where we are and where we should be going,” In: *Proc. Int. Workshop Evidential Assess. Softw. Technol.*, 2012, pp. 1-2.

[8] N. S. Vadlamani, “A survey on detection and defense of application layer DDoS attacks,” M.S. thesis, Univ. Nevada, Las Vegas, 2013.

[9] F. Wong and C. X. Tan, “A survey of trends in massive DDoS attacks and cloud-based mitigations,” *Int. J. Netw. Security Applicat.*, vol. 6, no. 3, pp. 57-71, 2014.

[10] G. Mantas, N. Stakhanova, H. Gonzalez, H. H. Jazi, and A. A. Ghorbani, “Application-layer denial of service attacks: taxonomy and survey,” *Int. J. Inf. Comput. Security*, vol. 7, no. 2-4, pp. 216-239, 2015.

[11] T. Peng, C. Leckie, and K. Ramamohanarao, “Survey of network-based defense mechanisms countering the DoS and DDoS problems,” *ACM Comput. Surveys*, vol. 39, no. 1, article 3, 2007.

[12] M. Sachdeva, G. Singh, K. Kumar, and K. Singh, “A comprehensive survey of distributed defense techniques against DDoS attacks,” *Int. J. Comput. Sci. Netw. Security*, vol. 9, no. 12, pp. 7-15, 2009.

[13] S. T. Zargar, J. Joshi, and D. Tipper, “A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks,” *Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 2046-2069, 2013.

[14] M. A. Babar and H. Zhang, “Systematic literature reviews in software engineering: Preliminary results from interviews with researchers,” In: *Proc. Int. Symp. Empirical Softw. Eng. Meas.*, 2009, pp. 346-355.

[15] B. Kitchenham, R. Pretorius, D. Budgen, O. Pearl Brereton, M. Turner, M. Niazi, and S. Linkman, “Systematic literature reviews in software engineering - A tertiary study,” *Inform. Softw. Technol.*, vol. 52, no. 8, pp. 792-805, 2010.

[16] A. Abdelmaboud, D. N. A. Jawawi, I. Ghani, A. Elsafi, and B. Kitchenham, “Quality of service approaches in cloud computing: A systematic mapping study,” *J. Syst. Softw.*, vol. 101, pp. 159-179, 2015.

[17] I. Iankoulova and M. Daneva, “Cloud computing security requirements: A systematic review,” In: *Proc. Int. Conf. Research Challenges Inform. Sci.*, 2012, pp. 1-7.

[18] A. Jula, E. Sundararajan, and Z. Othman, “Cloud computing service composition: A systematic literature review,” *Expert Syst. Applicat.*, vol. 41, no. 8, pp. 3809-3824, 2014.

[19] K. Singh, P. Singh, and K. Kumar, “A systematic review of IP traceback schemes for denial of service attacks,” *Comput. Security*, vol. 56, pp. 111-139, 2016.

[20] R. Albert, H. Jeong, and A. Barabási, “Internet: Diameter of the World-Wide Web,” *Nature*, vol. 401, no. 6749, pp. 130-131, 1999.

[21] R. Bauer, “Media (R)evolutions: Internet Live Stats,” 2014. [Online]. Available: http://blogs.worldbank.org/publicsphere/media-revolutions-internet-live-stats.

[22] “Worldwide Infrastructure Security Report Volume IX,” Arbor Networks, 2014. [Online]. Available: http://pages.arbornetworks.com/ rs/arbor/images/WISR2014.pdf.

[23] I. Zeifman, “2014 Global Bot Traffic Report,” Incapsula, 2014. [Online]. Available: https://www.incapsula.com/blog/bot-traffic-report-2014.html.

[24] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin, K. Emam, and J. Rosenberg, "Preliminary guidelines for empirical research in software engineering," *IEEE Trans. Softw. Eng.*, vol. 28, no. 8, pp. 721-734, 2002.

[25] J. Yu, C. Fang, L. Lu, and Z. Li, "A lightweight mechanism to mitigate application layer DDoS attacks," In: *Proc. Int. Conf. Scalable Inform. Syst.*, 2009, pp. 175-191.

[26] S. Wen, W. Jia, W. Zhou, W. Zhou, and C. Xu, "Cald: Surviving various application-layer DDoS attacks that mimic flash crowd," In: *Proc. Int. Conf. Netw. Syst. Security*, 2010, pp. 247-254.

[27] J. Wang, M. Zhang, X. Yang, K. Long, and C. Zhou, "HTTP-sCAN: Detecting HTTP-flooding attack by modeling multi-features of web browsing behavior from noisy dataset," In: *Proc. Asia-Pacific Conf. Commun.*, 2013, pp. 677-682.

[28] J. Wang, M. Zhang, X. Yang, K. Long, and J. Xu, "HTTP-sCAN: Detecting HTTP-flooding attack by modeling multi-features of web browsing behavior from noisy web-logs," *China Commun.*, vol. 12, no. 2, pp. 118-128, 2015.

[29] J. Wang, X. Yang, and K. Long, "Web DDoS detection schemes based on measuring user's access behavior with large deviation," In: *Proc. IEEE Global Telecommun. Conf.*, 2011, pp. 1-5.

[30] M. Jamshed and J. Brustoloni, "In-network Server-directed Client Authentication and Packet Classification," In: *Proc. IEEE Conf. Local Comput. Netw.*, 2010, pp. 328-331.

[31] W. G. Morein, A. Stavrou, D. L. Cook, A. D. Keromytis, V. Misra, and D. Rubenstein, "Using graphic turing tests to counter automated DDoS attacks against web servers," In: *Proc. ACM Conf. Comput. Commun. Security*, 2003, pp. 8-19.

[32] M. Srivatsa, A. Iyengar, J. Yin, and L. Liu, "A middleware system for protecting against application level denial of service attacks," In: *Proc. ACM/IFIP/USENIX Int. Conf. Middleware*, 2006, pp. 260-280.

[33] T. Thapngam, S. Yu, W. Zhou, and G. Beliakov, "Discriminating DDoS attack traffic from flash crowd through packet arrival patterns," In: *Proc. IEEE Conf. Comput. Commun. Workshops*, 2011, pp. 952-957.

[34] H. Beitollahi and G. Deconinck, "Tackling application-layer DDoS attacks," In: *Proc. Int. Conf. Ambient Syst. Netw. Techn.*, 2012, pp. 432-441.

[35] Y. Xie and S.-Z. Yu, "Detecting Shrew HTTP Flood Attacks for Flash Crowds," In: *Proc. Int. Conf. Comput. Sci.*, 2007, pp. 640-647.

[36] Y. Xie and S.-Z. Yu, "A novel model for detecting application layer DDoS attacks," In: *Proc. Int. Multi-Symp. Comput. Computat. Sci.*, 2006, vol. 2, pp. 56-63.

[37] Y. Xie and S. Tang, "Online anomaly detection based on web usage mining," In: *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops PhD Forum*, 2012, pp. 1177-1182.

[38] D. Stevanovic, N. Vlajic, and A. An, "Unsupervised clustering of web sessions to detect malicious and non-malicious website users," In: *Proc. Int. Conf. Ambient Syst. Netw. Techn.*, vol. 5, pp. 123-131, 2011.

[39] D. Stevanovic, A. An, and N. Vlajic, "Detecting web crawlers from web server access logs with data mining classifiers," In: *Proc. Int. Symp. Methodologies Intell. Syst.*, 2011, pp. 483-489.

[40] D. Stevanovic and N. Vlajic, “Application-layer DDoS in dynamic web-domains: Building defenses against next-generation attack behavior,” In: *Proc. IEEE Conf. Commun. Netw. Security*, 2014, pp. 490-491.

[41] “Statistics on botnet-assisted DDoS attacks in Q1 2015,” Kaspersky lab, 2015. [Online]. Available: https://securelist.com/blog/research/ 70071/statistics-on-botnet-assisted-ddos-attacks-in-q1-2015.

[42] I. Zeifman, “Q2 2015 Global DDoS threat landscape,” Incapsula, 2015. [Online]. Available: https://www.incapsula.com/blog/ddos-global-threat-landscape-report-q2-2015.html.

[43] J. Nazario, “BlackEnergy DDoS Bot Analysis,” Arbor Networks, 2007. [Online]. Available: http://atlas-public.ec2.arbor.net/docs/BlackEnergy + DDoS+Bot+Analysis.pdf.

[44] K. S. Han and E. G. Im, “A study on the analysis of netbot and design of detection framework,” In: *Proc. Joint Workshop Inform. Security*, 2008.

[45] D. Mazieres, “A toolkit for user-level file systems,” In: *Proc. USENIX Annu. Techn. Conf.*, 2001.

[46] A. Bhandari, A.L. Sangal, and K. Kumar, “Characterizing flash events and distributed denial-of-service attacks: an empirical investigation,” *Secur. commun. netw.*, vol. 9, no. 13, pp. 2222-2239, 2016.

[47] J. Mirkovic, A. Hussain, B. Wilson, S. Fahmy, P. Reiher, R. Thomas, W. Yao, and S. Schwab, “Towards user-centric metrics for denial-of-service measurement,” In: *Proc. Workshop Experimental Comput. Sci.*, article 8, 2007.

[48] K. Singh, P. Singh, and K. Kumar *(in press)*, “Impact analysis of application layer DDoS attacks on web services: A simulation study,” *Int. J. Intell. Eng. Informatic.*, 2016.

[49] A. Bhandari, A.L. Sangal, and K. Kumar, “Performance metrics for defense framework against distributed denial of service attacks,” *Int. J. Netw. Security*, vol. 6, pp. 38-47, 2014.

[50] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, and A. Wesslén, “Experimentation in Software Engineering,” *Springer Berlin Heidelberg*, 1st ed. 2012.

[51] J. Yu, C. Fang, L. Lu, and Z. Li, “A lightweight mechanism to mitigate application layer DDoS attacks,” In: *Proc. Int. Conf. Scalable Inform. Syst.*, 2009, pp. 175-191.

[52] S. Wen, W. Jia, W. Zhou, W. Zhou, and C. Xu, “Cald: Surviving various application-layer DDoS attacks that mimic flash crowd,” In: *Proc. Int. Conf. Netw. Syst. Security*, 2010, pp. 247-254.

[53] J. Wang, M. Zhang, X. Yang, K. Long, and C. Zhou, “HTTP-sCAN: Detecting HTTP-flooding attack by modeling multi-features of web browsing behavior from noisy dataset,” In: *Proc. Asia-Pacific Conf. Commun.*, 2013, pp. 677-682.

[54] J. Wang, M. Zhang, X. Yang, K. Long, and J. Xu, “HTTP-sCAN: Detecting HTTP-flooding attack by modeling multi-features of web browsing behavior from noisy web-logs,” *China Commun.*, vol. 12, no. 2, pp. 118-128, 2015.

[55] J. Wang, X. Yang, and K. Long, “Web DDoS detection schemes based on measuring user’s access behavior with large deviation,” In: *Proc. IEEE Global Telecommun. Conf.*, 2011, pp. 1-5.

[56] M. Jamshed and J. Brustoloni, “In-network Server-directed Client Authentication and Packet Classification,” In: *Proc. IEEE Conf. Local Comput. Netw.*, 2010, pp. 328-331.

[57] W. G. Morein, A. Stavrou, D. L. Cook, A. D. Keromytis, V. Misra, and D. Rubenstein, "Using graphic turing tests to counter automated DDoS attacks against web servers," In: *Proc. ACM Conf. Comput. Commun. Security*, 2003, pp. 8-19.

[58] M. Srivatsa, A. Iyengar, J. Yin, and L. Liu, "A middleware system for protecting against application level denial of service attacks," In: *Proc. ACM/IFIP/USENIX Int. Conf. Middleware*, 2006, pp. 260-280.

[59] T. Thapngam, S. Yu, W. Zhou, and G. Beliakov, "Discriminating DDoS attack traffic from flash crowd through packet arrival patterns," In: *Proc. IEEE Conf. Comput. Commun. Workshops*, 2011, pp. 952-957.

[60] H. Beitollahi and G. Deconinck, "Tackling application-layer DDoS attacks," In: *Proc. Int. Conf. Ambient Syst. Netw. Techn.*, 2012, pp. 432-441.

[61] Y. Xie and S.-Z. Yu, "Detecting Shrew HTTP Flood Attacks for Flash Crowds," In: *Proc. Int. Conf. Comput. Sci.*, 2007, pp. 640-647.

[62] Y. Xie and S.-Z. Yu, "A novel model for detecting application layer DDoS attacks," In: *Proc. Int. Multi-Symp. Comput. Computat. Sci.*, 2006, vol. 2, pp. 56-63.

[63] Y. Xie and S. Tang, "Online anomaly detection based on web usage mining," In: *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops PhD Forum*, 2012, pp. 1177-1182.

[64] D. Stevanovic, N. Vlajic, and A. An, "Unsupervised clustering of web sessions to detect malicious and non-malicious website users," In: *Proc. Int. Conf. Ambient Syst. Netw. Techn.*, vol. 5, pp. 123-131, 2011.

[65] D. Stevanovic, A. An, and N. Vlajic, "Detecting web crawlers from web server access logs with data mining classifiers," In: *Proc. Int. Symp. Methodologies Intell. Syst.*, 2011, pp. 483-489.

[66] D. Stevanovic and N. Vlajic, "Application-layer DDoS in dynamic web-domains: Building defenses against next-generation attack behavior," In: *Proc. IEEE Conf. Commun. Netw. Security*, 2014, pp. 490-491.

PRIMARY STUDIES

[S1] J. Yu, Z. Li, H. Chen, and X. Chen, "A detection and offense mechanism to defend against application layer DDoS attacks," In: *Proc. Int. Conf. Netw. Services*, 2007, pp. 54-54.

[S2] Y. Xie and S.-Z. Yu, "A detection approach of user behaviors based on HsMM," In: *Proc. Int. Teletraffic Congr. Perform. Challenges Efficient Next Generation Netw.*, 2005, pp. 461-450.

[S3] Y. Xie and S.-Z. Yu, "A large-scale hidden semi-Markov model for anomaly detection on user browsing behaviors," *IEEE/ACM Trans. Netw.*, vol. 17, no. 1, pp. 54-65, 2009.

[S4] J. Wang, X. Yang, and K. Long, "A new relative entropy based app-DDoS detection method," In: *Proc. IEEE Symp. Comput. Commun.*, 2010, pp. 966-968.

[S5] M. A. Saleh and A. A. Manaf, "A novel protective framework for defeating HTTP-based denial of service and distributed denial of service attacks," *Scientific World J.*, vol. 2015, article ID 238230, 19 pages, 2015.

[S6] Y. G. Dantas, V. Nigam, and I. E. Fonseca, "A selective defense for application layer DDoS attacks," In: *Proc. IEEE Joint Intell. Security Informatics Conf.*, 2014, pp. 75-82.

[S7] Y. Choi, I.-K. Kim, J.-T. Oh, and J.-S. Jang, "AIGG threshold based HTTP GET flooding attack detection," In: *Proc. Int. Workshop Inform. Security Applicat.*, 2012, pp. 270-284.

[S8] S. Limkar and R. K. Jha, “An effective defence mechanism for detection of DDoS attack on application layer based on hidden markov model,” In: *Proc. Int. Conf. Inform. Syst. Design Intel. Applicat.*, 2012, pp. 943-950.

[S9] W.-Z. Lu and S. Yu, “An HTTP flooding detection method based on browser behavior,” In: *Proc. Int. Conf. Comput. Intell. Security*, 2006, vol. 2, pp. 1151-1154.

[S10] Y. Xi, W. Wushao, and Y. Yiru, “An OTP-based mechanism for defending application layer ddos attacks,” In: *Proc. Int. Conf. Appl. Informatics and Commun.*, 2011, pp. 388-396.

[S11] Q. Liao, H. Li, S. Kang, and C. Liu, “Application layer DDoS attack detection using cluster with label based on sparse vector decomposition and rhythm matching,” *Security Commun. Netw.*, 2015.

[S12] M. Zhang, W. Zhang, and K. Fan, “Application layer DDoS detection model based on data flow aggregation and evaluation,” In: *Proc. Int. Conf. Critical Infrastructure Protection*, 2012, pp. 37-45.

[S13] C. Ye, K. Zheng, and C. She, “Application layer DDoS detection using clustering analysis,” In: *Proc. Int. Conf. Comput. Sci. Netw. Techn*, 2012, pp. 1038-1041.

[S14] S. Kandula, D. Katabi, M. Jacob, and A. Berger, “Botz-4-sale: Surviving organized DDoS attacks that mimic flash crowds,” In: *Proc. Conf. Netw. Syst. Design Implemen.*, 2005, pp. 287-300.

[S15] L. V. Ahn, M. Blum, N. J. Hopper, and J. Langford, “CAPTCHA: Using hard AI problems for security,” In: *Proc. Int. Conf. Theory Applicat. Cryptographic Techn.*, 2003, pp. 294-311.

[S16] H. Beitollahi and G. Deconinck, “ConnectionScore: A statistical technique to resist application-layer DDoS attacks,” *J. Ambient Intell. Human Comput.*, vol. 5, no. 3, pp. 425-442, 2013.

[S17] Y. Tang, “Countermeasures on application level low-rate denial-of-service attack,” In: *Proc. Int. Conf. Inform. Commun.*, 2012, pp. 70-80.

[S18] M. Walfish, M. Vutukuru, H. Balakrishnan, D. Karger, and S. Shenker, “DDoS defense by offense,” *ACM Trans. Comput. Syst.*, vol. 28, no. 1, pp. 3:1-3:54, Aug. 2010.

[S19] T. Thapngam, S. Yu, and W. Zhou, “DDoS discrimination by linear discriminant analysis (LDA),” In: *Proc. Int. Conf. Comput. Netw. Commun.*, 2012, pp. 532-536.

[S20] S. Ranjan, R. Swaminathan, M. Uysal, and E. W. Knightly, “DDoS-Resilient scheduling to counter application layer attacks under imperfect detection,” In: *Proc. Int. Conf. Comput. Commun.*, 2006, pp. 1-13.

[S21] W. Yen and M.-F. Lee, “Defending application DDoS with constraint random request attacks,” In: *Proc. Asia-Pacific Conf. Commun.*, 2005, pp. 620-624.

[S22] H. Liu and K. Chang, “Defending systems against tilt DDoS attacks,” In: *Proc. Int. Conf. Telecommun. Syst. Services Applicat.*, 2011, pp. 22-27.

[S23] G. Maciá-Fernández, R. A. Rodríguez-Gómez, and J. E. Díaz-Verdejo, “Defense techniques for low-rate dos attacks against application servers,” *Comput. Netw.*, vol. 54, no. 15, pp. 2711-2727, Oct. 2010.

[S24] H. Y. Suen, W. C. Lau, and O. Yue, “Detecting anomalous web browsing via diffusion wavelets,” In: *Proc. IEEE Int. Conf. Commun.*, 2010, pp. 1-6.

[S25] L. C. Giralte, C. Conde, I. M. Diego, and E. Cabello, “Detecting denial of service by modelling web-server behaviour,” *Comput. Elect. Eng.*, vol. 39, no. 7, pp. 2252-2262, 2013.

[S26] Y. Xie, S. Tang, X. Huang, C. Tang, and X. Liu, "Detecting latent attack behavior from aggregated web traffic," *Comput. Commun.*, vol. 36, no. 8, pp. 895-907, 2013.

[S27] W. Zhou, W. Jia, S. Wen, Y. Xiang, and W. Zhou, "Detection and defense of application-layer DDoS attacks in backbone web traffic," *Future Generation Comput. Syst.*, vol. 38, pp. 36-46, 2014.

[S28] P. Chwalinski, R. Belavkin, and X. Cheng, "Detection of application layer DDoS attack with clustering and likelihood analysis," In: *Proc. IEEE GLOBECOM Workshops*, 2013, pp. 217-222.

[S29] P. Chwalinski, R. Belavkin, and X. Cheng, "Detection of application layer DDoS attacks with clustering and bayes factors," In: *Proc. IEEE Int. Conf. Syst. Man Cybernetics*, 2013, 2013, pp. 156-161.

[S30] P. Arun Raj Kumar and S. Selvakumar, "Detection of distributed denial of service attacks using an ensemble of adaptive and hybrid neuro-fuzzy systems," *Comput. Commun.*, vol. 36, no. 3, pp. 303-319, 2013.

[S31] D. Das, U. Sharma, and D. K. Bhattacharyya, "Detection of HTTP flooding attacks in multiple scenarios," In: *Proc. Int. Conf. Commun. Comput. Security*, 2011, pp. 517-522.

[S32] P. Chwalinski, R. Belavkin, and X. Cheng, "Detection of HTTP-GET attack with clustering and information theoretic measurements," In: *Proc. Int. Symp. Found. Practice Security*, 2013, pp. 45-61.

[S33] T. Yatagai, T. Isohara, and I. Sasase, "Detection of HTTP-GET flood attack based on analysis of page access behavior," In: *Proc. IEEE Pacific Rim Conf. Commun. Comput. Signal Process.*, 2007, pp. 232-235.

[S34] D. Stevanovic, N. Vlajic, and A. An, "Detection of malicious and non-malicious website visitors using unsupervised neural network learning," *Appl. Soft Comput.*, vol. 13, no. 1, pp. 698-708, 2013.

[S35] D. Gavrilis and E. Dermatas, "Detection of web denial-of-service attacks using decoy hyperlinks," In: *Proc. Int. Symp. Commun. Syst. Netw. Digital Signal Process.*, 2006.

[S36] C. Xu, G. Zhao, G. Xie, and S. Yu, "Detection on application layer DDoS using random walk model," In: *Proc. IEEE Int. Conf. Commun.*, 2014, pp. 707-712.

[S37] S. Yu, W. Zhou, W. Jia, S. Guo, Y. Xiang, and F. Tang, "Discriminating DDoS attacks from flash crowds using flow correlation coefficient," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 6, pp. 1073-1080, 2012.

[S38] T. Thapngam, S. Yu, W. Zhou, and S. K. Makki, "Distributed Denial of Service (DDoS) detection by traffic pattern analysis," *Peer-to-peer Netw. Applicat.*, vol. 7, no. 4, pp. 346-358, 2014.

[S39] D. Stevanovic, A. An, and N. Vlajic, "Feature evaluation for web crawler detection with data mining techniques," *Expert Syst. Applicat.*, vol. 39, no. 10, pp. 8707-8717, 2012.

[S40] L. Qin, L. Hong, K. Songlin, and L. Chuchu, "Feature extraction and construction of application layer DDoS attack based on user behavior," In: *Proc. Chin. Control Conf.*, 2014, pp. 5492-5497.

[S41] S. Yu, S. Guo, and I. Stojmenovic, "Fool me if you can: Mimicking attacks and anti-attacks in cyberspace," *IEEE Trans. Comput.*, vol. 64, no. 1, pp. 139-151, Jan. 2015.

[S42] J. Wang, X. Yang, M. Zhang, K. Long, and J. Xu, "HTTP-SoLDiER: An HTTP-flooding attack detection scheme with the large deviation principle," *Sci. China Inf. Sci.*, vol. 57, no. 10, pp. 1-15, Apr. 2014.

[S43] H. Kim, B. Kim, D. Kim, I.-K. Kim, and T.-M. Chung, "Implementation of GESNIC for web server protection against HTTP GET flooding attacks," In: *Proc. Int. Workshop Inform. Security Applicat.*, 2012, pp. 285-295.

[S44] S. Yu, W. Zhou, and R. Doss, “Information theory based detection against network behavior mimicking DDoS attacks,” *IEEE Commun. Lett.*, vol. 12, no. 4, pp. 318-321, 2008.

[S45] X. Q. Di, H. M. Yang, and H. Qi, “Low-rate application-layer DDoS attacks detection by principal component analysis (PCA) through user browsing behavior,” *Appl. Mech. Mater.*, vol. 397-400, pp. 1945-1948, 2013.

[S46] C. Huang, J. Wang, G. Wu, and J. Chen, “Mining web user behaviors to detect application layer ddos attacks,” *J. Softw.*, vol. 9, no. 4, pp. 985-990, 2014.

[S47] J. Yu, C. Fang, L. Lu, and Z. Li, “Mitigating application layer distributed denial of service attacks via effective trust management,” *IET commun.*, vol. 4, no. 16, pp. 1952-1962, 2010.

[S48] M. Srivatsa, A. Iyengar, J. Yin, and L. Liu, “Mitigating application-level denial of service attacks on web servers: A client-transparent approach,” *ACM Trans. Web*, vol. 2, no. 3, pp. 15:1-15:49, 2008.

[S49] G. Oikonomou and J. Mirkovic, “Modeling human behavior for defense against flash-crowd attacks,” In: *Proc. IEEE Int. Conf. Commun.*, 2009, pp. 1-6.

[S50] Y. Xie and S. Yu, “Monitoring the application-layer DDoS attacks for popular websites,” *IEEE/ACM Trans. Netw.*, vol. 17, no. 1, pp. 15-25, 2009.

[S51] M. Abliz and T. Znati, “New approach to mitigating distributed service flooding attacks,” In: *Proc. Int. Conf. Syst.*, 2012, pp. 13-19.

[S52] D. Stevanovic and N. Vlajic, “Next generation application-layer DDoS defences: Applying the concepts of outlier detection in data streams with concept drift,” In: *Proc. Int. Conf. Mach. Learn. Applicat.*, 2014, pp. 456-462.

[S53] P. Du and A. Nakao, “OverCourt: DDoS mitigation through credit-based traffic segregation and path migration,” *Comput. Commun.*, vol. 33, no. 18, pp. 2164-2175, 2010.

[S54] S. Umarani and D. Sharmila, “Predicting application layer ddos attacks using machine learning algorithms,” *Int. J. Comput. Elect. Automat. Control Inform. Eng.*, vol. 8, no. 10, 2014.

[S55] A. Ramamoorthi, T. Subbulakshmi, and S. M. Shalinie, “Real time detection and classification of DDoS attacks using enhanced SVM with string kernels,” In: *Proc. Int. Conf. Recent Trends Inform Techn.*, 2011, pp. 91-96.

[S56] T. Ni, X. Gu, H. Wang, and Y. Li, “Real-time detection of application-layer DDoS attack using time series analysis,” *J. Control Sci. Eng.*, vol. 2013, pp. 4:4-4:4, 2013.

[S57] Y. Xie, S. Tang, Y. Xiang, and J. Hu, “Resisting web proxy-based http attacks by temporal and spatial locality behavior,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 7, pp. 1401-1410, 2013.

[S58] P. Djalaliev, M. Jamshed, N. Farnan, and J. Brustoloni, “Sentinel: hardware-accelerated mitigation of bot-based DDoS attacks,” In: *Proc. Int. Conf. Comput. Commun. Netw.*, 2008, pp. 1-8.

[S59] S. Lee, G. Kim, and S. Kim, “Sequence-order-independent network profiling for detecting application layer DDoS attacks,” *EURASIP J. Wireless Commun. Netw.*, vol. 2011, no. 1, pp. 1-9, 2011.

[S60] M. Emami-Taba, M. Amoui, and L. Tahvildari, “Strategy-Aware mitigation using markov games for dynamic application-layer attacks,” In: *Proc. IEEE Int. Symp. High Assurance Syst. Eng.*, 2015, pp. 134-141.

[S61] Y. S. Choi, J. T. Oh, J. S. Jang, and I. K. Kim, “Timeslot monitoring model for application layer DDoS attack detection,” In: *Proc. Int. Conf. Comput. Sci. Convergence Inform. Techn.*, 2011, pp. 677-679.

[S62] E. Doron and A. Wool, “WDA: A web farm distributed denial of service attack attenuator,” *Comput. Netw.*, vol. 55, no. 5, pp. 1037-1051, 2011.

[S63] A. Stavrou, D. L. Cook, W. G. Morein, A. D. Keromytis, V. Misra, and D. Rubenstein, “WebSOS: An overlay-based system for protecting web servers from denial of service attacks,” *Comput. Netw.*, vol. 48, no. 5, pp. 781-807, 2005.