View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by DigitalCommons@Kennesaw State University

Kennesaw State University

DigitalCommons@Kennesaw State University

Master of Science in Information Technology Theses

Department of Information Technology

Spring 4-2-2020

Distributed Denial of Service Attack Detection

Travis Blue

Follow this and additional works at: https://digitalcommons.kennesaw.edu/msit_etd

Part of the Information Security Commons

Recommended Citation

Blue, Travis, "Distributed Denial of Service Attack Detection" (2020). *Master of Science in Information Technology Theses*. 6.
https://digitalcommons.kennesaw.edu/msit_etd/6

This Thesis is brought to you for free and open access by the Department of Information Technology at DigitalCommons@Kennesaw State University. It has been accepted for inclusion in Master of Science in Information Technology Theses by an authorized administrator of DigitalCommons@Kennesaw State University. For more information, please contact digitalcommons@kennesaw.edu.

Distributed Denial of Service Attack Detection

Travis Blue

A thesis in the field of Information Technology for the Degree of Master of Science in Information Technology

Kennesaw State University

2020

ABSTRACT

Distributed Denial of Service (DDoS) attacks on web applications has been a persistent threat. Successful attacks can lead to inaccessible service to legitimate users in time and loss of business reputation. Most research effort on DDoS focused on network layer attacks. Existing approaches on application layer DDoS attack mitigation have limitations such as the lack of detection ability for low rate DDoS and not being able to detect attacks targeting resource files. In this work, we propose DDoS attack detection using concepts from information retrieval and machine learning. We include two popular concepts from information retrieval: Term Frequency (TF)-Inverse Document Frequency (IDF) and Latent Semantic Indexing (LSI). We analyzed web server log data generated in a distributed environment. Our evaluation results indicate that while all the approaches can detect various ranges of attacks, information retrieval approaches can identify attacks ongoing in a given session. All the approaches can detect three well known application level DDoS attacks (trivial, intermediate, advanced). Further, these approaches can enable an administrator identifying new pattern of DDoS attacks.

ACKNOWLEGEMENT

I like to express thanks to Dr. Hossain Shahriar for his supervision and guidance of this thesis including providing and directing me to the resources. I also like to thank the committee members Drs. Chi Zhang and Seyedamin Pouriyeh for their thoughtful comments to improve the work. Finally, I am grateful for my family to receive the support during the degree program from start until completion.

Table of Contents

Chapter I. Introduction

Denial of Service (DoS) attacks generate a large number of requests to a target application occupying resources for processing the requests (e.g., memory, CPU time) [1]. As a result, legitimate users do not get an access to the required services in time. Distributed DoS (DDoS) attacks involve issuing huge number of requests from a set of computers (bots) controlled by an attacker. Currently, DDoS attack is one of the largest threats in the Internet [2]. A recent survey from Symantec [4] indicates that DDoS have been prevalent for network layer (e.g., ICMP flooding, TCP SYN flooding) where the goal of an attacker is to exhaust network bandwidth and buffer memory.

DDoS attacks issue large number of GET or POST requests to random web pages, which seem normal traffic. Existing network layer approaches are not applicable for detecting DDoS attacks. Unlike network layer attacks, application layer attacks are hard to detect and mitigate. A number of bots are available in the market that can automate application layer DDoS attacks. Worst, DDoS as a service is now currently available to mount attacks on legitimate entities [2]. DDoS attacks have been mounted against various websites such as game (Sony Play Station) [5] and bitcoin [6]. DDoS attacks can lead to loss in revenue ($5600 per minute), productivity, and reputation [7]. Given that application layer DDoS attack mitigation is important.

There are a number of available mitigation approaches in the literature [24-31]. These approaches have limitations such as high computational complexity (random graph model [31], Hidden Markov Model [29], clustering [27]). These approaches can discover slow rate (GET or POST) and intermediate attacks. However, they do not address advanced attacks where attackers issue GET/POST requests in correct order from many computers simultaneously. Attackers may avoid these existing techniques by launching attacks to web applications not only requesting to the main page of webserver, but also pages having links to resources (e.g., large sized image files). Further, current approaches do not let an administrator discover customized DDoS attack types to raise early warning. This thesis addresses these issues.

In this thesis, we propose a framework to detect DDoS attacks. We first identify page ranking using a popular Term Frequency (TF)-Inverse Document Frequency (IDF) from web server logs [41]. Then, we identify ranking of resources that are accessed most to build normal profile. For a given web session, we form a query of accessed resources and find how close it is with respect to the normal profile using Latent Semantic Indexing (LSI). If a large deviation is identified, the session is identified as part of DDoS attack.

Our proposed approach can address some limitations of the existing DDoS approaches. First, it can detect three well known attacks types such as low rate attacks, flash mob attacks and stochastic rate attacks. Moreover, it can identify attacks where attacker targets an inner page of website, or arbitrary resources of large size (e.g., large image or video files).

This thesis is organized as follows: Chapter 2 presents an example of DDoS attack; Chapter 3 highlights related work; Chapter 4 presents the proposed approach; Chapter 5 discusses evaluation results; and Chapter 6 concludes the paper.

Chapter 2

Distributed Denial of Service Attack Example

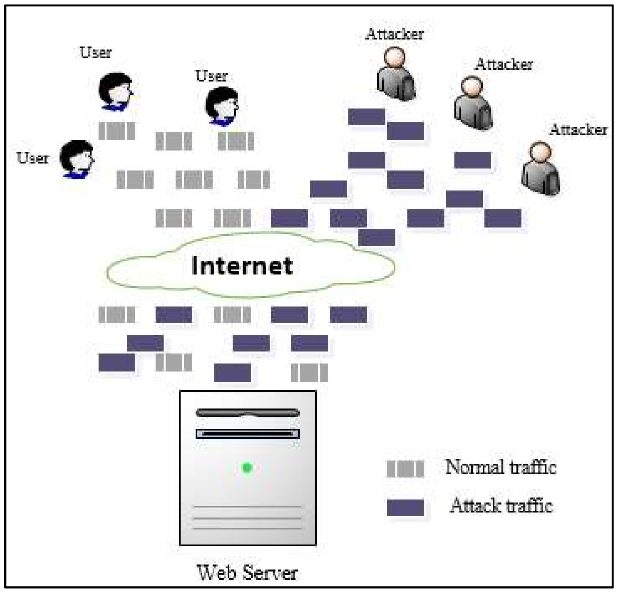


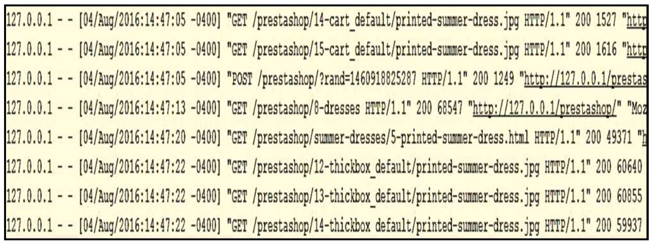
Figure 1. General DDoS Scenario

Figure 1 shows a general DDoS scenario where a web server can be targeted by multiple attackers disrupting user access to the resources and services provided by the server. Within the context of the current work, Slowloris (slow read) was executed as an example AL-DDoS attack, and the corresponding Apache web server logs were used to build the initial profile following the proposed approach. Slowloris is a type of attack that allows a single computer or machine to halt the processes and functions of another machine's web server by slowly occupying all connections on the server. The attack uses minimal bandwidth and has few, if any, side effects on unrelated services. Most AL-DDoS attacks, Slowloris included, are executed through the use of some type of DDoS tool. For Slowloris, the tool is entitled Switchblade and it is openly available from the Open Web Application Security Project (OWASP). When the attack was launched in a controlled VMware environment, the Apache webserver was crashed in less than five seconds. The attack successfully crashed the Apache server every time it was executed. Each time, the server was inoperable until Apache was restarted, which would lead to drastic reductions in productivity in any organization or company setting.

In addition to Slowloris, Switchblade allows the user to exploit SSL half connections and HTTP Post attacks. A proxy server was put in place alongside Slowloris to enable the interception of http level traffic and experimentation. Due to the high volume of available literature on network-layer attacks and data, this work was specifically geared towards application-layer attacks alone. Denial of Service attacks occurring at the application-layer have not been studied in much detail to date. Therefore, the current study intends to address this gap in the field, despite the implementation of other approaches by researchers within the literature. Previous mitigation techniques were too focused on the network-layer, or were not applicable to DDoS attacks.

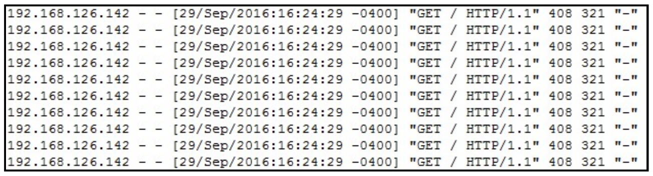
Through the use of this tool, an attacker can launch HTTP GET slowheaders, HTTP POST slowheaders or SSL renegotiation. In the HTTP GET slowheaders attack, the attacker sends multiple (up to 40,000) incomplete HTTP requests to the web server until all available connections to the web server are taken by the incomplete GET requests. Similarly, in the HTTP POST slowheaders attack, several incomplete requests to upload a file or submit a filled-in web form, etc. are sent to the web server until all connections are occupied by the attacker. Finally, in a SSL renegotiation attack, an attacker can inject code or commands into a HTTPS session, downgrade the connection from HTTPS to HTTP, craft custom responses and so on. The Figures below illustrate samples of the logs obtained from the web server. Figure 2 shows an example of good traffic logs without malicious input. The next Figure demonstrates malicious GET requests in the logs that were part of a DDoS attack simulation. Figure 4 illustrates an excerpt from the POST

DDoS attack execution. Finally, in Figure 5 the SSL renegotiation from the Apache error log is provided.



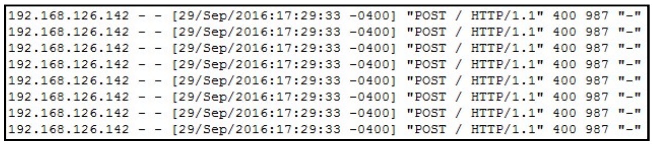
```
127.0.0.1 - - [04/Aug/2016:14:47:05 -0400] "GET /prestashop/14-cart_default/printed-summer-dress.jpg HTTP/1.1" 200 1527 "http
127.0.0.1 - - [04/Aug/2016:14:47:05 -0400] "GET /prestashop/15-cart_default/printed-summer-dress.jpg HTTP/1.1" 200 1616 "http
127.0.0.1 - - [04/Aug/2016:14:47:05 -0400] "POST /prestashop/?rand=1460918825287 HTTP/1.1" 200 1249 "http://127.0.0.1/prestas
127.0.0.1 - - [04/Aug/2016:14:47:13 -0400] "GET /prestashop/8-dresses HTTP/1.1" 200 68547 "http://127.0.0.1/prestashop/" "Moz
127.0.0.1 - - [04/Aug/2016:14:47:20 -0400] "GET /prestashop/summer-dresses/5-printed-summer-dress.html HTTP/1.1" 200 49371 "h
127.0.0.1 - - [04/Aug/2016:14:47:22 -0400] "GET /prestashop/12-thickbox_default/printed-summer-dress.jpg HTTP/1.1" 200 60640
127.0.0.1 - - [04/Aug/2016:14:47:22 -0400] "GET /prestashop/13-thickbox_default/printed-summer-dress.jpg HTTP/1.1" 200 60855
127.0.0.1 - - [04/Aug/2016:14:47:22 -0400] "GET /prestashop/14-thickbox default/printed-summer-dress.jpg HTTP/1.1" 200 59937
```

Figure 2. Good Log Sample



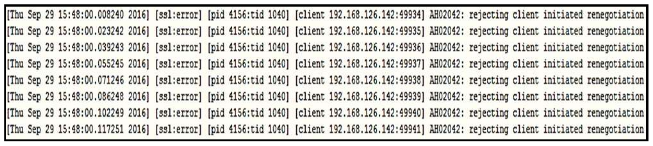
```
192.168.126.142 - - [29/Sep/2016:16:24:29 -0400] "GET / HTTP/1.1" 408 321 "-"
192.168.126.142 - - [29/Sep/2016:16:24:29 -0400] "GET / HTTP/1.1" 408 321 "-"
192.168.126.142 - - [29/Sep/2016:16:24:29 -0400] "GET / HTTP/1.1" 408 321 "-"
192.168.126.142 - - [29/Sep/2016:16:24:29 -0400] "GET / HTTP/1.1" 408 321 "-"
192.168.126.142 - - [29/Sep/2016:16:24:29 -0400] "GET / HTTP/1.1" 408 321 "-"
192.168.126.142 - - [29/Sep/2016:16:24:29 -0400] "GET / HTTP/1.1" 408 321 "-"
192.168.126.142 - - [29/Sep/2016:16:24:29 -0400] "GET / HTTP/1.1" 408 321 "-"
192.168.126.142 - - [29/Sep/2016:16:24:29 -0400] "GET / HTTP/1.1" 408 321 "-"
192.168.126.142 - - [29/Sep/2016:16:24:29 -0400] "GET / HTTP/1.1" 408 321 "-"
192.168.126.142 - - [29/Sep/2016:16:24:29 -0400] "GET / HTTP/1.1" 408 321 "-"
```

Figure 3. Malicious GET Request Log Sample



```
192.168.126.142 - - [29/Sep/2016:17:29:33 -0400] "POST / HTTP/1.1" 400 987 "-"
192.168.126.142 - - [29/Sep/2016:17:29:33 -0400] "POST / HTTP/1.1" 400 987 "-"
192.168.126.142 - - [29/Sep/2016:17:29:33 -0400] "POST / HTTP/1.1" 400 987 "-"
192.168.126.142 - - [29/Sep/2016:17:29:33 -0400] "POST / HTTP/1.1" 400 987 "-"
192.168.126.142 - - [29/Sep/2016:17:29:33 -0400] "POST / HTTP/1.1" 400 987 "-"
192.168.126.142 - - [29/Sep/2016:17:29:33 -0400] "POST / HTTP/1.1" 400 987 "-"
192.168.126.142 - - [29/Sep/2016:17:29:33 -0400] "POST / HTTP/1.1" 400 987 "-"
192.168.126.142 - - [29/Sep/2016:17:29:33 -0400] "POST / HTTP/1.1" 400 987 "-"
```

Figure 4. Malicious POST Request Log Sample



```
[Thu Sep 29 15:48:00.008240 2016] [ssl:error] [pid 4156:tid 1040] [client 192.168.126.142:49934] AH02042: rejecting client initiated renegotiation
[Thu Sep 29 15:48:00.023242 2016] [ssl:error] [pid 4156:tid 1040] [client 192.168.126.142:49935] AH02042: rejecting client initiated renegotiation
[Thu Sep 29 15:48:00.039243 2016] [ssl:error] [pid 4156:tid 1040] [client 192.168.126.142:49936] AH02042: rejecting client initiated renegotiation
[Thu Sep 29 15:48:00.055245 2016] [ssl:error] [pid 4156:tid 1040] [client 192.168.126.142:49937] AH02042: rejecting client initiated renegotiation
[Thu Sep 29 15:48:00.071246 2016] [ssl:error] [pid 4156:tid 1040] [client 192.168.126.142:49938] AH02042: rejecting client initiated renegotiation
[Thu Sep 29 15:48:00.086248 2016] [ssl:error] [pid 4156:tid 1040] [client 192.168.126.142:49939] AH02042: rejecting client initiated renegotiation
[Thu Sep 29 15:48:00.102249 2016] [ssl:error] [pid 4156:tid 1040] [client 192.168.126.142:49940] AH02042: rejecting client initiated renegotiation
[Thu Sep 29 15:48:00.117251 2016] [ssl:error] [pid 4156:tid 1040] [client 192.168.126.142:49941] AH02042: rejecting client initiated renegotiation
```

Figure 5. SSL Renegotiation Error Log Sample

Chapter 3.

Related Work

One study examined the potential for novel methods to make DoS attacks against web application servers. The authors illustrate that DoS attacks can be deployed through low-rate traffic. A low-rate approach has two main advantages for the attacker. The first advantage is the ease of execution because the amount of resources needed for the attack is greatly reduced. Secondly, it is easier to hide this type of attack from security systems that rely on detection of attacks during high-rate traffic flows. The researchers also offer a way to bypass an intrusion detection system, mainly by decreasing the payload, which is the case with low-rate traffic [7]. Another way to launch a low-rate DoS attack is for the attacker to use a malicious TCP receiver. The idea is that the attacker can remotely control the transmission rate and the pattern of the TCP sender to exploit it as a flood source. This is called an Induced-shrew attack. The attack does not arise from an implementation flaw, but rather from a TCP specification. By sending less traffic over a longer period of time, the attack is suddenly not a part of the suspicious traffic being monitored [8].

The authors of a study recommend a detection mechanism for DDoS attacks only. This mechanism uses packet sampling and the flow feature to ensure normal traffic is transmitted and abnormal traffic is prevented from flooding the system. Any traffic on the network was classified based on the flow feature. Each flow falls into one of four possible classifications: sampled packet amount, destination IP address, packet sampling and/or the source port entropy flow. The researchers set a threshold for the amount of network traffic, and any traffic that exceeded that threshold was considered to be dangerous or suspicious [9].

Despite the implementation of Intrusion Detection Systems (IDS), network layer attacks can bypass these systems by using anomalies. The authors discovered their own ways detect anomalies running over TCP. In the study, three compression algorithms were used to perform data compression: Huffman coding, Dynamic Markov coding and Lempel-Ziv-Welch algorithm. The algorithms utilized, perform data compression by using fewer bits to encode data, without data being lost. The results from their approach yielded an approximate DDoS detection rate of 99 percent and a false alarm rate ranged from 0.05 percent to 0.20 percent. [10]

Authors of another work applied multivariate correlation analysis model to network layer traffic. This model was used to detect SYN flood attacks, but also able to DDoS attacks. Six common TCP header flags to include URG, ACK, PSH, RST, SYN and FIN were examined for the covariance. This approach led to the detection of both low volumes and high volumes of attack traffic, but with the main drawback that there is no guarantee that the six flags are sufficient or valid in the model for DDoS detection [11].

Using neural network, another study was able to detect anomalies, implement their systems to distributed routers, identify the attack packets and filter the packets. It found that the marks in IP headers generated by IP traceback schemes, may enhance the sensitivity and accuracy of attack detection. The neural network features of network traffic were extracted to use as input. The performance metrics for this study were the average rate at which legitimate traffic and attack traffic passed through the filtering systems [12].

To detect DDoS attacks, the authors of the study use feature space modeling as a general detection method based on physical network features. The source of the data in the study was a subset of the KDD Cup 1999 dataset, which contains multiple network-layer attacks [13]. Only 32 out of 41 physical features were selected from the data set for construction of the covariance feature space. Six types of DDoS attacks were used for the training profiles and four additional DDoS attacks from the dataset for testing purposes. Results showed a detection rate above 99 percent in both cases based on their pre-set threshold discovered during the training phase.

In the context of another work, the researchers suggest an auto-adapted algorithm that is applied to the average value and the threshold value of the network traffic. If the network environment changes, the algorithm adjusts the parameters to account for the modification(s). This algorithm permits for more accurate detection of anomalies and reduces the operating costs. The method was created for detecting DDoS, but the authors note that it can be used to detect other network layer attacks, such as SYN flooding. Anomalies are found by changes in the number of data packets and by the CUSUM algorithm in the shortest possible time period [14].

Clustering is a common method for DDoS mitigation. By combining a new clustering method with feature ranking, the authors of a unique study intend to detect these attacks. The Modified Global K-means algorithm (MGKM) is applied and permits identification of the cluster structure. Linear correlation coefficient is ranked and detection of separate phases of the DDoS attack. The data source is the 2000 DARPA Intrusion Detection Scenario Specific Dataset. It was discovered that the phases could be distinguished with as few as six features from the data set. These authors reference another work written by Lee et al. in 2009 to compare the two approaches, declaring that their study produced more sound results than did the other related research [15].

Flooding attacks, such as a DDoS attack, can be detected using a covariance-matrix modeling technique, as proposed by Yeung *et al.*[16]. These researchers make use of statistical covariance matrices in order to build the normal profile of activities on a system or set of systems. Any changes of the covariance matrices are directly used for detecting flooding attacks. A threshold matrix restricts the classification boundary. To determine the threshold matrix, the authors of the study use Chebyshev's inequality theory. The resulting matrix is utilized to characterize the second-order features of the flooding attack. Specific DDoS attacks under investigation were the Neptune and the Smurf attacks. The authors used the KDD Cup 1999 dataset for attack traffic. Attack detection was successful in their study 100 percent of the time [16].

Several researchers and organizations alike have offered ways to use Software Defined Networking (SDN) for network measurement. One big problem facing SDN is the DDoS attack. The authors of a recent DDoS study propose a mitigation technique for attacks that are leveraging the flow monitoring capability of the network itself. Two specific methods are suggested including the Sequential Method and the Concurrent Method to modify the flow monitoring across all the network switches and catch the dubious attacker as soon as possible. The purpose of this approach is to discover the victim IP range and the source, or attacker's, IP address. There are two main algorithms deployed to discover the victim IP range and another algorithm for revealing the attacker's IP address (or range of potential IP addresses) within the context of the study. Feature selection is used to classify the packet volume-based DDoS attacks [17].

In a study from 2008, authors emphasize the importance of DoS and DDoS attacks against the next generation mobile network (NGMN) in the near future. Their research offers a detection algorithm that points out and characterizes the network traffic by paying close attention to the frequency spectrum distribution. To determine the power spectrum of the traffic, the Lomb periodogram is used. Anomalies in the frequency spectrum based on the two derivative parameters. Both of the known DoS and DDoS attacks were simulated to create signatures and tested for accuracy. The authors note that their signatures are not widely applicable due to the closed environment used for attack simulation [18].

Hybrid systems bring solutions from multiple existing approaches and form them into one detection technique. The authors of one work combined Genetic Algorithm (GA) for feature selection with Artificial Neural Network (ANN), used for attack detection. In the given work, data comes from the CAIDA UCSD 2007 Dataset. Feature selection takes place based on the GA fitness function to select only the most fit chromosomes to reintroduce to the initial population. This layered model is a class of ANN technique called a Multi-Layer Perceptron. Authors of the work determined that the five features selected for by GA provided the most accurate results for the classification process [19].

Network layer DDoS attack cannot be successful if only one machine or attacker is sending requests, the attacks can often evade detection. Authors of a study on large-scale networks provide a detection scheme for Web Service DDoS attackers that takes into consideration whether the client was active during the detection time. The approach allows for researchers to discover and potentially block bad traffic while allowing legitimate users to access services regularly. The authors aim to detect attacks that disable an entire web application by overloading the service with a large number of requests [20]. The scheme resulted in low false positive probability and nearly zero false negative probability

Entropy is applied to detecting DDoS attacks in work by Kumar *et al.* [21]. This study emphasizes the overhead challenges in regard to memory and computational power. These authors recommend distributing such overheads across all of the points of presence (POPs) of an Internet Service Provider (ISP). Connecting Pops and multiple ISPs via high bandwidth links is called the ISP backbone. Attacks were detected based on the window size and the tolerance factor of the attack for varying entropy [21].

Researchers often apply statistics, such as Chi square, to real world situations to reach their conclusions. In a study on DoS/DDoS attack detection, the researchers propose an agent-based intrusion detection system. The system compared and examined the IP address and the associated network traffic, if a dramatic deviation from the baseline statistics is found it's often an attack. Their system (Agent-based Intrusion Detection System) compared to popular detection systems such as McAfee and Snort, AIDS detected attacks with nearly ten percent better accuracy [22].

Nashat *et al.* point out that the current detection mechanisms for network layer DDoS attacks rely on a specific number of attackers, depending on the size of the target network [14]. Application layer attacks, however, do not suffer from this problem. In fact, an application layer DDoS attack can be carried out by one individual using DDoS tools like Stacheldraht and TFN2K [28].

Deep Learning architecture is technique that is composed of at least three layers in a deep neural network that can be used to detect application layer DDoS attacks. An auto encoder learns features within the attack dataset. The authors use a stacked auto-encoder deep learning architecture, capable of receiving high level features to identify the type of attack. Finally, the authors apply logistic regression to decide if the incoming traffic is normal or attack traffic. The average accurate detection rate from experimentation was 98.99 percent with an average false positive rate of 1.27 percent [23].

Chan *et al.* [24] recommend an intrusion detection router (IDR) coupled with a Bloom filter to identify DDoS attacks and heavy traffic. The IDR is composed of three important features. First, it should be able to detect any traffic on a given network that is consuming a high amount of bandwidth. It must also analyze and reject any suspicious packets resulting from that traffic. Finally, the IDR has to react to the attack when it is detected. This happens when the IDR traces to the source of the attack and can control how the attack(s) flow through the router. No IDR is available to the general public, but the idea is to develop the functionality as multiple add-on modules for open source software routers in Linux.

Yu *et al.* [25] suggest mitigating application layer DDoS attacks, specifically session flooding, by using trust management. The authors measured four specific parameters of trust for each user after every established connection. Measurements included short-term trust, long-term trust, negative trust and misuse trust. All measures were combined to generate an overarching trust value that is used to determine if the user's next request should be accepted or not. After evaluation, they concluded that their lightweight mechanism produced a negligible amount of computational cost and an acceptable overhead bandwidth based on the typical number of user sessions [25].

Tempesta is a framework developed by Krizhanovsky [26]. This framework is made up of a combined caching HTTP server and firewall. Among the objectives in this study, the author specifically outlines five goals. The framework should provide full access to all layers of the OSI model to allow for traffic classification, modification and systems for filtering. Another goal is to integrate the framework as a component of a Linux TCP/IP stack in hopes of handling short-term connections often used in DDoS attacks. The framework should be closely intertwined with Linux security and netfilter subsystems. This is for classifying and blocking botnets and managing dynamic rules. Tempesta also mitigates web service overloads (common DDoS attacks) by using a reverse-proxy functionality. Lastly the author emphasizes the need to train classification algorithms and back-end servers on what normal HTTP messages or requests must contain so they are interpreted correctly by the device or algorithm.

In another study, Zolotukhin *et al.* present a method to detect several types of DoS attacks in a timely fashion [27]. In general, the focus is aimed at detecting application layer DoS attacks that use encryption protocols. The researchers utilize statistics to apply an anomaly-based detection approach to extracted network packets. None of the packets were decrypted, which is the main point. The authors want to detect and analyze attacks without decrypting the network-level traffic specifically between a client and a web server. Conversations between clients and web servers are divided into clusters to create a model of a normal user's behavior. Each conversation is characterized by four main parameters including the source IP address, the source port, the destination IP address and the destination port. The distribution of the conversations into clusters is examined using the stacked auto-encoder (one of many deep learning algorithms) and deviations are marked as anomalies. Their method was able to lower false positive rates to less than two percent [27].

Based on previous research the authors developed, a lightweight DDoS detection mechanism for web servers [28]. They used Transductive Confidence Machines for *K*-Nearest Neighbors (TCM-KNN) and selection methods based on genetic algorithm. The goal of the present research is to propose a more efficient instance selection method that works better for a real network situation and they call the Extend Fuzzy C-Means (E-FCM) algorithm. By utilizing the algorithm, the researchers reduced the time their mechanism takes by a factor of 4.2. However, the reduction in time comes at the cost of increasing the false positive rate.

In order to monitor application layer DDoS attacks against well-known websites, the authors of one study introduced a scheme based on document popularity. The authors suggest using an access matrix to discover any existing spatial-temporal pattern of a typical surge in the number of webpage visitors. To abstract the matrix the researchers analyzed both principal and independent components. A model is developed to detect DDoS attacks based on the entropy of the document's popularity. Their approach includes multidimensional data processing, the advantages of the Hidden Semi-Markov Model, computational complexity and the self-adapting scheme, among several parameters as well [29].

Flash events and denial-of-service (DoS) attacks both result in degraded web services. A flash event is a sudden peak in the number of visitors to a certain webpage and is considered to be normal, but flash events can keep webpages from functioning completely. The authors of the study suggest the use of enhanced content distribution networks (CDNs) to protect web sites. Through the use of traffic patterns, file reference characteristics and client characteristics, flash events are separated from DoS attacks. Authors of the study state that only about 80 percent of webpages are accessed during any one flash event and the enhanced CDN can help alleviate that percentage. Implementing an enhanced CDN will distinguish flash events from DoS attacks on a web server [30].

Mirkovic *et al.* [33] describe the taxonomy of DDoS attacks and mitigation techniques that currently exist. Based on how the attack is prepared, carried out, what characteristics it has, and the effect it has on the victim, Mirkovic *et al.* present multiple DDoS classes.

LSI and TF-IDF are popular information retrieval techniques [34]. Due to space limitation, we discuss some literature work related to the application of LSI below. LSI has been applied in several domains such as document classification [35, 36], recovering connection between documents and program source code [37, 38], identifying similarities among program source code [40], and automating the task of assigning reviewers to manuscripts [39]. However, we are not aware of any approach that explores the application of LSI for discovering DDoS attacks from web server logs. Below, we briefly describe some earlier approaches that motivate our work.

Price *et al.* [35] reported that LSI is a robust approach to classify data in presence of noise. They studied known benchmark text data corpus (Reuters) by randomly inserting, updating, and deleting texts, and similarly conducted study for OCR documents having poor quality to see performance during retrieval of information.

Wang *et al.* [37] applied advanced LSI model to improve software engineering tasks. They define terms by capturing domain specific vocabulary from source code; construct a similarity dictionary with synonymity and abbreviations, clustered source code to improve precision of retrieval process.

Marcus *et al.* [38, 39] applied LSI to recover the link between documents and source code in legacy software. They consider the written comments in the source code as part of the documentation and match them with the appropriate variable identifier names. Traceability links between requirement documentation and source code can be recovered by performing textual and structural analysis (*i.e.*, methods and variables are modeled). In comparison to all these effort, we focus on detecting DDoS attacks by applying LSI.

Chapter 4: TF-IDF and LSI Approach

4.1 TF-IDF Based Ranking

TF-IDF is a computation approach to find the importance of word in a set of documents. It is composed by computing two terms: TF and IDF as discussed below. Term Frequency (TF) measures how frequently a term occurs in a document.

TF(t) = (# of times t appears in a document) / (Total # of terms in the document) … … … (i)

Inverse Document Frequency (IDF) measures how important a term is in all document. TF assumes that each is equally important. However, certain term may be common (e.g., article in sentence). Thus, IDF consider the frequent term as rare and less occurring term as important.

IDF(t) = $\log_e$(Total # of documents /#of documents having t) … … (ii)

For a given web server log file, we identify words representing resources such as php, javascript, and image files. For example, in Figure 2, out of four requests, three are GET and one is POST type.

```
"GET /prestashop/img/a.gif HTTP/1.1" 200 5867
"POST /prestashop/administration/ajax.php?rand=1460921178266 HTTP/1.1" 200 120
"GET /prestashop/js/tiny_mce/plugins/anchor/plugin.min.js HTTP/1.1" 200 508
"GET /prestashop/administration/themes/default/js/admin-theme.js HTTP/1.1" 200 14464
```

Figure 1: Example of good logs for Prestashop application

The first line accesses to an image resource prestashop, whereas, the second line accesses to ajax.php. The last two requests access to two javascript files. If we assume total 6 terms (including GET, POST, and 200 for status code) for all four lines. TF for each of the three resources is 1/6. Let us assume that we have four days of log files. Accessing to the three resources happened to only the first day. Table 1 computer TF, IDF, and TF-IDF.

Table 1: TF-IDS example of words from web logs

Term	TF	# of document	IDF	TF-IDF
a.gif	1/6	1	$\log_e(4/1)$	0.23
ajax.php	1/6	1	$\log_e(4/1)$	0.23
plugin.min.js	1/6	1	$\log_e(4/1)$	0.23
admin-theme.js	1/6	1	$\log_e(4/1)$	0.23
GET	3/6	4	$\log_e(4/4)$	0
POST	1/6	4	$\log_e(4/4)$	0
200	1/6	4	$\log_e(4/4)$	0

As it can be seen, from the table that the most common occurring terms (GET, 200 status code) found in web logs have zero TF-IDF. However, requests accessing to specific resources are having higher TF-IDF. Thus, ranking based on TF-IDF provides us an initial profile of important and meaningful resources that legitimate users would access.

The TF-IDF is applied for several days of log files. For each day, we identify the TF-IDF value of resources (words). Then, we combine the TF-IDF value for all words and perform an average for each of the word's TF-IDF value. We select top n words having non-zero TF-IDF. For example, in Table 2, we show four days of TF-IDF value for four words (term). The last column shows the average TF-IDF. In this case, we choose top three term based on non-zero TF-IDF value (login.php, a.jpeg, registration.php).

Table 2: Example of TF-IDF value from web server log

Term	Day1	Day2	Day3	Day4	Avg.
login.php	0.092	0	0	0.092	0.46
a.jpeg	0.092	0.092	0	0.092	0.069
registration.php	0.092	0	0.092	0.092	0.069
Prestashop	0	0	0	0	0

Next, we consider placing the set of selected words having non-zero TF-IDF to a matrix to apply LSI discussed in the next subsection.

4.2 LSI Approach

Latent Semantic Indexing is used to rank a set of documents based on how closely related to a set of query terms. We apply Latent Semantic Indexing (LSI) technique to perform query and obtain the close similarity of documents for decision making. We represent access pattern from sample data in a matrix form to apply LSI, where each row represents specific resource access for a certain day, and column represents specific words found in log line with their TF-IDF values obtained from previous step. The columns would not only contain web page name, but also specific resources such as images, amount of bytes, status code, browser name.

For DDoS attack detection, we consider building a query obtained from an ongoing session data. We extract the resources (term) of interests from the log. Log files (documents) obtained for legitimate traffic for various days are used form Term-Document matrix. Terms are defined based on words representing resources (php files, image files) having non-zero TF-IDF value (discussed in previous section). Queries obtained for given sessions are used to find how close a given day's log represent for an ongoing session. Similarity measures are based on cosine metrics. The closer a query vector and a document is, the higher the distance. Hence, if distance is above certain threshold (d) level, it is not considered an attack. If the similarity is less than a threshold value, then the ongoing session is considered as an attack. Below, we discuss six general steps of applying LSI approach.

Step1: Build Term-Document matrix $A = [m*n]$, where m = number of terms, n = number of documents; $q = [m*1]$ as query vector. We denote terms as $t_1 \dots t_m$, documents as $d_1 \dots d_n$. Each entry of the term document matrix indicates the occurrence of the term in each document.

Step2: Perform Singular Value Decomposition (SVD) [14]. This means expressing A in terms of three matrices U, S, and V^T such that $A = U * S * V^T$. Here, * is matrix multiplication operator, U

is the Eigen vector of the matrix A, S is the diagonal matrix having singular value, V is Inverse of U matrix, V^T is the transpose of V matrix.

Step3: Choose k out of n ($k < n$) to reduce the dimensionality of the A matrix.

Step 4: Define A_k by reducing the dimensions of A from $m*n$ to $m*k$. We define S_K from S by reducing dimension from $m*n$ to $m*k$ ($k < n$). Finally, define V_K from V by reducing the dimension from $n*n$ to $k*k$. Now, each document d_i is approximated by the corresponding row vector of v_i from V_K matrix.

Step 5: Obtain the approximate query vector in reduced space from q to q_k, by multiplying the three matrices $q* \ U_K * \mathrm{S_K}^{-1}$. This means the original query vector is reduced from $m*1$ to $k*1$. Here, S_K^{-1} is the inverse matrix of S_K.

Step 6: Measure similarity using cosine distance (d_i) formula as follows:

$$Sim\ (q_k,\ d_k) = q_k.d_k/\ (|q_k||d_k|)$$

Here, |…| represents the norm operation of a vector, $\mathrm{q_k.d_k}$ represents the dot product for the two column vectors. We compute the average (d_{avg}) similarity of all documents (d_i). An attack is detected if the similarity level is below certain threshold value ($d_{avg} < d$).

Chapter 5: Evaluation

A. Dataset generation

An environment that is used to generate data has to be very controlled to ensure that no attacks can be introduced into the setting. For this chapter, the benchmark datasets were generated through the use of a virtual machine cluster using VMware Workstation on the host machine. The host machine is a 64-bit standalone server running an AMD FX 8350 eight core processor at 4Ghz, contains 32 GB of physical memory and 64 GB of virtual memory. The operating system on the host machine is Windows 10.

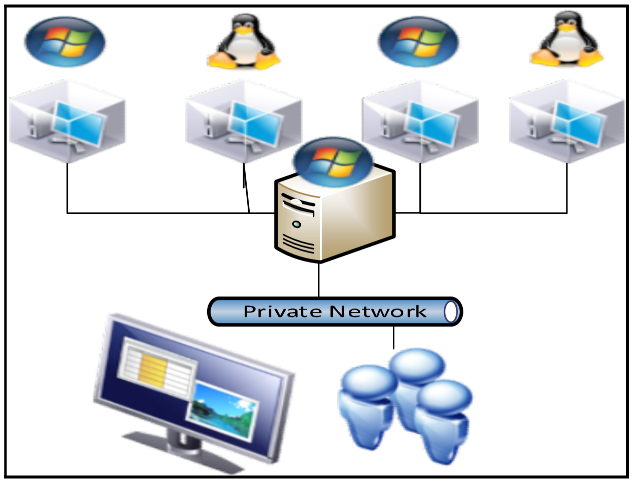


Figure 6: The Environment for Data Generation

Figure 6 is a diagram showing the environment that was used for this data generation process. Having a virtual setting for the benchmark generation provides an additional layer of defense against any out-of-network traffic. Thus, the resulting web application traffic was all generated by the user actions and the benchmark was known to be free of application layer attack attempts. Some of the virtual environments had a Windows 10 operating system while others ran on a Linux operating system. This variation in the operating system was utilized to make sure the benchmark was applicable to machines with Windows and Linux environments. All security features, such as antivirus and firewalls, were deactivated to allow for the generation of attack data. Each virtual environment had the same baseline software installed including Microsoft Office and Notepad++ and Google Chrome served as the default web browser.

In addition to the baseline software, a popular open source software named XAMPP was added to each virtual environment. This web application works across operating systems and incorporates Apache, MySQL, PHP and PERL. To generate datasets in the controlled environment, Apache was used as the web server and MySQL was used as the database management system. Implementation of the PHP and PERL features of the application were beyond the scope of this thesis work. Both the Apache web server and the MySQL database management system kept logs of information about what was occurring on the system while XAMPP was running. A total of five web applications were installed on the virtual machine cluster, and the user was only accessing one web application at a time. The web applications that were installed on the virtual cluster were all open source applications and already integrated with the XAMPP software. These applications had various functions, which led to the creation of different types of data over the course of four days for the final benchmarking dataset.

We evaluate the proposed approach using web serve log data generated using an Apache server where we deploy three large scale PHP web applications. First, we visit these applications while performing various functionalities (e.g., registration, login, posting messages in PhpBB). We employ different users performing functionalities for four different days and then combine log files of all applications. To generate realistic traffic, we use different IP addresses by deploying virtual machines having Linux in a local network.

Table 3 shows the characteristics of the log data. These include total number of lines, unique number of GET and POST requests, total # of php pages requested, and total # of images (gif) downloaded.

Table 3: Characteristics of log data

Application	# of lines	# of GET	# of POST	# of page	# of images
PhpBB	2,087	1,489	598	3,973	105
Prestashopt	5,719	5,273	445	890	28
Wordpress	2,036	1460	575	1,498	29

The applications interacted automatically using scripts, and were provided with malicious inputs. We collected inputs from the sources such as OWASP. These attack inputs are applied randomly within web requests from a browser. Table 4 displays the number of each type of attack that was distributed into the attack dataset. The Apache web server logs were referenced to manually detect successful attacks. Figure 7 shows an example of log data for a SQL injection attack where an input field (id) has a tautology attack encoded in hexa-decimal format. Similarly, Figure 8 shows an example of log data for an XSS attack where an image source has been supplied with malicious code. Finally, Figure 9 shows an example of log data for a RFI attack, where the *FORMAT* field is included with an include statement pointing to a file source from an attacker-controlled website, followed by an *exit()* command.

Table 4: Distribution of Attack Input Data

Attack type	# of samples
SQLI	1000
RFI	5
XSS	60
Total	1073

```
"GET
/sqlinj/?id=1%27+or+%271%27+%3D+%271%27%29%29%2F*&Submit=Submit&user_token
=c14e5f424d9f279c19ba507492745d50…
```

Figure 7: Example Log Data for SQL Injection Attack

```
"GET
/xss_r/?name=%3CIMG+SRC%3DJaVaScRiPt%3Aalert%28%26quot%3BXSS%26quot%3B%
29%3E&user_token=f37e5a82a994725092fd3155bb8cffba…
```

Figure 8: Example Log Data for XSS Attack

```
"GET /?FORMAT={${include("http://www.verybadwebsite.com/hacker.txt")}}{${exit()}}…
```

Figure 9: Example Log Data for RFI Attack

B. Evaluation

Table 5 shows the minimum and maximum value of TF-IDF we obtain for each day for the three applications. For each of the days, the value varied widely. This is due to users navigating pages both randomly (visiting pages that do not require doing specific functionality) and directed way while performing various functionalities.

Table 5: Minimum and maximum TF-IDF

Application	Day1		Day2		Day3		Day4	
	min TF-IDF	max TF-IDF	min TF-IDF	max TF-IDF	min TF-IDF	max TF-IDF	min TF-IDF	max TF-IDF
PhpBB	0.96	1.40	0.15	1.71	0.05	1.97	0.01	1.13
Prestashopt	0.61	1.04	0.26	1.40	0.23	1.87	0.15	1.50
Wordpress	0.35	0.40	0.33	0.63	0.14	0.65	0.04	0.77

We then perform test run of the application to emulate DDoS attacks of three types: slow rate, intermediate, and advanced. We deploy three virtual machines, each one running an automated script (implemented in python) to generate log traffic for one hour of time period. For slot rate attacks, we choose to send 60 requests randomly to web pages that do not require logging or registering to the applications. For intermediate attack, we increase the rate at 600 requests per hour and request to php pages (login, register, and search) and pages having image resources to download. For advanced attacks, we perform login to the website and perform advanced level activities (*e.g.*, add/edit posted messages, registering, updating of user information) rapidly within short time duration (1,000 requests per hour).

We evaluate performance of the proposed approach using Receiver Operating Characteristics (ROC) curve. It has two measures: True Positive Rate (TPR) on the y axis, and False Positive Rate (FPR) on the x axis. TPR is defined as the ration of TP and (TP+FN), whereas, FPR is defined as the ratio of FP and (FP+FN).

TPR = TP/(TP+FN) … … … (iii)

FPR = FP/(FP+TN) … … … (iv)

If our approach generates a warning we call it positive, if it is real, we all it True Positive (TP). If no warning is generated we denoted it negative, if it is actually not part of DDoS attack request, we call it True Negative (TN). If the approach misses the actual attack detection, we call it False Negative (FN). Ideally, we expect our approach to demonstrate TPR as 100%, while FPR would be 0%. The normal (attack free) data is used to produce FPR, whereas the dataset containing DDoS attack requests would be used to obtain TPR rate.

Table 6: FPR and TPR for low rate DDoS attack (n=3)

Similarity threshold	**2 minutes**		**5 minutes**		**10 minutes**	
	FPR(%)	TPR(%)	FPR(%)	TPR(%)	FPR(%)	TPR(%)
d <0.4	0.64	83.66	0.69	90.22	1.2	94.21
d <0.5	1.1	92.45	1.45	94.53	1.45	95.67
d <0.6	1.26	98.67	1.67	99.33	2.56	98.67
d <0.7	2.56	100	1.92	100	2.92	100

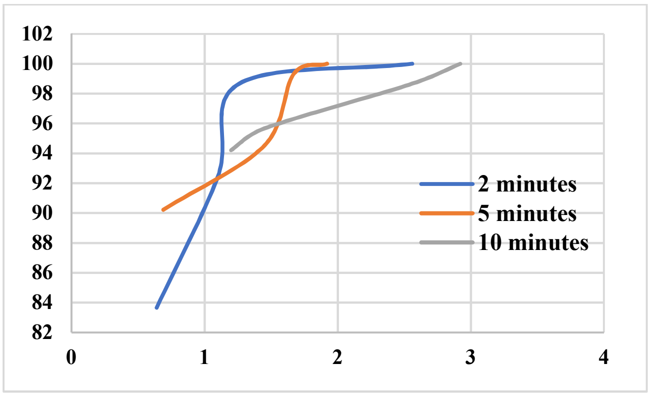


Figure 10: Comparison of performance for low rate DDoS attacks (n=3)

Table 6 shows FPR and TPR for low rate attack detection for various time intervals and having various similarity threshold level. The lowest FPR is observed for 10 minutes of session (0.64%) while the highest FPR is for 30 minutes of session (2.92%). Here, the query has three words. We find that logs of larger time duration can detect attack with increased accuracy while threshold of similarity level should be set higher. Figure 10 shows ROC curve of performance for various time duration of query logs.

Similar observation can be found for intermediate DDoS attack (Table 7, Figure 11). Here, as the threshold of similarity increases, we can detect attacks with increased accuracy and vice versa. Further, longer duration of log samples to build query results in accurate attack detection with lower similarity level. Hence, within shorter duration of time, it is possible to detect advanced DDoS attacks (Table 8, Figure 12).

Table 7: FPR and TPR for intermediate DDoS attack (n=3)

Similarity threshold	**2 minutes**		**5 minutes**		**10 minutes**	
	FPR(%)	TPR(%)	FPR(%)	TPR(%)	FPR(%)	TPR(%)
d >0.4	0.64	85.66	0.78	92.22	1.01	98.21
d >0.5	0.8	95.45	1.22	95.53	1.12	97.67
d>0.6	0.92	98.67	1.32	99.65	1.89	98.67

d>0.7	1.55	100	1.52	100	1.92	100

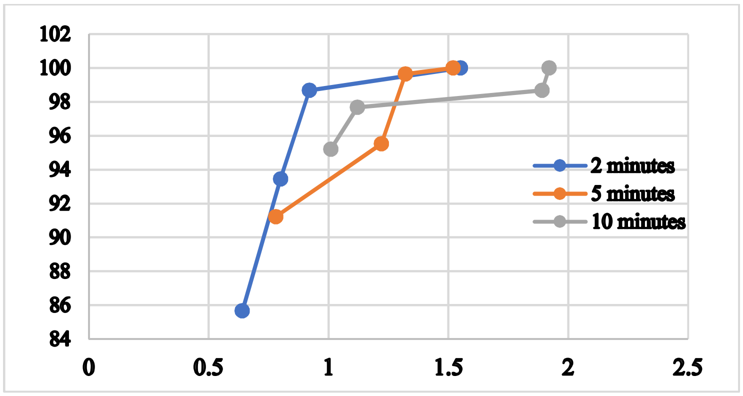


Figure 11: Comparison of performance for intermediate rate DDoS attacks (n=3)

For advanced DDoS attack, we find at a lower level of threshold, the approach is able to discover advanced DDoS attacks. The underlying reason is the query matrix has larger instances of requested resources

Table 8: FPR and TPR for advanced DDoS attack (n=3)

Similarity threshold	2 minutes		5 minutes		10 minutes	
	FPR(%)	TPR(%)	FPR(%)	TPR(%)	FPR(%)	TPR(%)
d >0.4	0.72	94.57	0.78	95.36	1.01	98.32
d >0.5	0.81	98.26	1.22	98.57	1.12	99.56
d>0.6	0.94	99.31	1.32	99.65	1.89	100
d>0.7	1.62	100	1.52	100	1.92	100

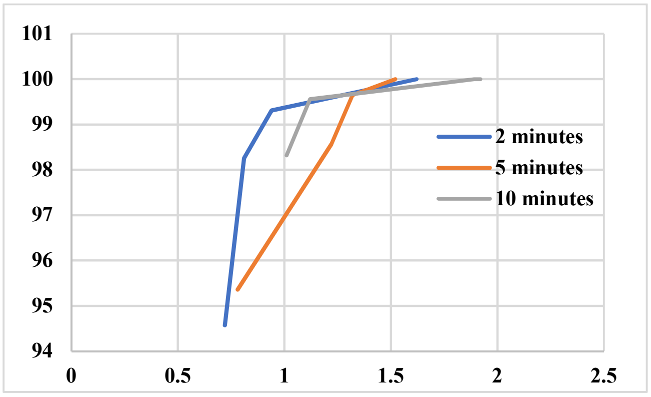


Figure 12: Comparison of performance for advanced DDoS attacks (n=3)

Chapter 6: Conclusion

DDoS attack is a major threat in today’s world. To mitigate this attack, this paper proposed an application level detection framework by employing two computing concepts from information retrieval (TF-IDF and latent semantic indexing). The approach ranks widely accessed resources (web pages, images) based on normal web page visiting pattern. The information is used to apply term document matrix. For given session, a query is formed to discover how close it is to the matrix. The approach relies on multiple log files obtained in different days. The initial results show that the approach is capable of discovering low rate, intermediate and advanced attacks. Currently, our approach focuses on discovering DDoS against commonly accessed pages and image files. In future, we plan to evaluate our approach for other types of resources (e.g., databases, tables). We plan to evaluate our approach with large scale dataset.

References

[1] Denial of Service (DoS), OWASP, Accessed from https://www.owasp.org/index.php/Denial_of_Service

[2] J. Cheng, J. Yin, Y. Liu, Z. Cai, and C. Wu, “DDoS Attack Detection Using IP Address Feature Interaction,” *Proc. of 2009 International Conference on Intelligent Networking and Collaborative Systems*, pp. 113-118.

[3] C. Wueest, The continued rise of DDoS attacks, Symantec Technical Report, October 2014, Accessed from https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/the-continued-rise-of-ddos-attacks.pdf

[4] Sony Playstation Hack, Jan 2016, Accessed from http://www.scmagazine.com/sony-psn-downed-hacking-group-claims-ddos-attack/article/463065/

[5] T. Bienkowski, Your Network or Your Bitcoins: Three Rules for Dealing with DDoS Extortion Threats, February 2016, https://www.arbornetworks.com/blog/insight/your-network-or-your-bitcoins-three-rules-for-dealing-with-ddos-extortion-threats/

[6] What is a DDoS Attack? 2016, Accessed from https://www.verisign.com/en_US/security-services/ddos-protection/what-is-a-ddos-attack/index.xhtml

[7] G. Macia, J. Diaz, and P. Garcia, “Evaluation of a low-rate DoS Attack against Application Servers,” *Computers & Security*, 2008, 27(7-8), pp. 335-354.

[8] V. Kumar, P. Jayalekshmy, G. Patra, “On Remote Exploitation of TCP Sender for Low-Rate Flooding Denial-of-Service Attack,” IEEE Communications Letters, 2009, 13(1): 46-48.

[9] J. Jun, C. Ahn, S. Kim, "DDoS attack detection by using packet sampling and flow features," *Proc. of ACM Symposium of Applied Computing (SAC)*, March 24-28, 2014, Gyeongju, Korea, pp. 711-712.

[10] Christian Callegari, Stefano Giordano, and Michele Pagano, On the Use of Compression Algorithms for Network Anomaly Detection, *Proceeding of the 2009 IEEE international conference on Communications (ICC)*, pp. 636-640.

[11] S. Jin and D.Yeung, "A covariance analysis model for DDoS attack detection," *Proc. of IEEE International Conference on Communications*, 2004.

[12] Yang Xiang and Wanlei Zhou, "Mark-aided distributed filtering by using neural network for DDoS defense," *Proc. of Global Telecommunications Conference*, IEEE, vol. 3, pp. 5 2005.

[13] Shu-Yuan Jin and D.S. Yeung, "DDoS detection based on feature space modeling", Machine Learning and Cybernetics 2004. Proceedings of 2004 International Conference on, vol. 7, pp. 4210-4215 vol.7, 2004.

[14] Shanyue Bu, Ruchuan Wang and Hong Zhou, "Anomaly Network Traffic Detection Based on Auto-Adapted Parameters Method", Proc. of 4th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM '08), pp. 1-4, 2008.

[15] Lifang Zi, John Yearwood and Xin-Wen Wu, "Adaptive Clustering with Feature Ranking for DDoS Attacks Detection", Network and System Security (NSS) 2010 4th International Conference on, pp. 281-286, 2010.

[16] Daniel S. Yeung, Shuyuan Jin and Xizhao Wang, "Covariance-Matrix Modeling and Detecting Various Flooding Attacks", Systems Man and Cybernetics Part A: Systems and Humans IEEE Transactions on, vol. 37, pp. 157-169, 2007, ISSN 1083-4427.

[17] Yang Xu and Yong Liu, "DDoS attack detection under SDN context", Computer Communications IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on, pp. 1-9, 2016.

[18] Fazirulhisyam Hashim; M. Rubaiyat Kibria; Abbas Jamalipour, "Detection of DoS and DDoS attacks in NGMN using frequency domain analysis," Proc. of 14th Asia-Pacific Conference on Communications, pp. 1-5, 2008.

[19] Mehdi Barati; Azizol Abdullah; Nur Izura Udzir; Ramlan Mahmod; Norwati Mustapha, Distributed Denial of Service detection using hybrid machine learning technique, *Proc. of International Symposium on Biometrics and Security Technologies (ISBAST)*, 2014, pp. 268 - 273.

[20] Dalia Nashat; Xiaohong Jiang; Susumu Horiguchi, “On the Detection of DDoS Attackers for Large-Scale Networks,” Proc. of International Conference on e-Business Engineering, 2009, pp. 206-212.

[21] Krishan Kumar; R. C. Joshi; Kuldip Singh, "A Distributed Approach using Entropy to Detect DDoS Attacks in ISP Domain," Proc. of 2007 International Conference on Signal Processing, Communications and Networking, 2007, pp. 331-337.

[22] Fang-Yie Leu; Chia-Chi Pai, Detecting DoS and DDoS Attacks Using Chi-Square, Proc. of Fifth International Conference on Information Assurance and Security, 2009, Volume: 2, pp. 255 - 258.

[23] Satyajit Yadav; Selvakumar Subramanian, "Detection of Application Layer DDoS attack by feature learning using Stacked AutoEncoder," Proc. of 2016 International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT), pp. 361-366.

[24] Chan, E. et al. 2004. IDR: An Intrusion Detection Router for Defending Against Distributed Denial-of-Service (DDoS) Attacks. *In Proceedings of the 7th International Symposium on Parallel Architectures, Algorithms and Networks*. pp. 581-586.

[25] Yu, J. et al. 2010. Mitigating application layer distributed denial of service attacks via effective trust management. *IET Communications*, 4, 16, pp. 1952-1962. doi: 10.1049/iet-com.2009.0809

[26] A. Krizhanovsky, "Tempesta: a framework for HTTP DDoS attacks mitigation," *Proceedings of 24th Annual International Conference on Computer Science and Software Engineering* (CASCON '14). IBM Corp., Riverton, NJ, USA, pp. 148-162.

[27] Mikhail Zolotukhin, Timo Hamalainen, Tero Kokkonen, Jarmo Siltanen, "Increasing Web Service Availability by Detecting Application-Layer DDoS Attacks in Encrypted Traffic" 2016 23rd International Conference on Telecommunications (ICT).

[28] Yang Li, Tian-Bo Lu, Li Guo, Zhi-Hong Tian, Qin-Wu Nie, "Towards Lightweight and Efficient DDoS Attacks Detection for Web Server," Proc. of WWW, 2009, pp. 1139-1140.

[29] Yi Xie and Shun-Zheng Yu, "Monitoring the Application-Layer DDoS Attacks for Popular Websites," IEEE/ACM Transactions on Networking, Vol. 17, No. 1, Feb 2009, pp. 15-25.

[30] Jaeyeon Jung, Balachander Krishnamurthy, Michael Rabinovich, "Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites," Proc. of 2002 WWW, May 2002, Honolulu, Hawaii, USA, pp. 293-304.

[31] C. Xu, G. Zhao, G. Xie, and S. Yu, "Detection on Application Layer DDoS using Random Walk Model," *Proc. of IEEE Communication and Information Systems Security Symposium*, Sydney, Australia, June 2014, pp. 707-712.

[32] DDoS Intelligence Report, Q1, 2015, Accessed from https://securelist.com/analysis/quarterly-malware-reports/74550/kaspersky-ddos-intelligence-report-for-q1-2016/

[33] Mirkovic, J. and Reiher, P. 2004. A taxonomy of DDoS attack and DDoS defense mechanisms. *SIGCOMM Computer Communication Review 34,* 2, pp. 39-53. Doi: 10.1145/997150.997156

[34] C. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*, 2008, Cambridge University Press.

[35] R. Price and A. Zukas, "Application of Latent Semantic Indexing to Processing of Noisy Text," *Proc. of IEEE International Conference on Intelligence and Security Informatics*, Atlanta, GA, USA, May 19-20, 2005, pp. 602-603.

[36] A. Zukas and R. Price, "Document Categorization Using Latent Semantic Indexing," *Proceedings of Symposium on Document Image Understanding Technology*, Greenbelt, MD, April 2003, pp. 87–91.

[37] X. Wang, G. Lai, and C. Liu, "Recovering Relationships between Documentation and Source Code based on the Characteristics of Software Engineering," *Electronic Notes in Theoretical Computer Science*, 243 (2009), pp. 121–137.

[38] A. Marcus and J. Maletic, "Recovering Documentation-to-Source-Code Traceability Links using Latent Semantic Indexing," *Proceedings of the 25th International Conference on Software Engineering (ICSE)*, Portland, OR, USA, 2003, pp. 125-135.

[39] S. Dumais, and J. Nielsen, "Automating the Assignment of Submitted Manuscripts to Reviewers," *Proc. of the 15th Annual International Conference on Research and Development in Information Retrieval, pp.* 233–244.

[40] A. Marcus and J. Maletic, "Using Latent Semantic Analysis to Identify Similarities in Source Code to Support Program Understanding," *Proc. of 12th IEEE International Conference on Tools with Artificial Intelligence*, November 2000, pp. 46–53.

[41] T. Blue and H. Shahriar, "Distributed Denial of Service Attack Detection," *Proc. of National Cyber Summit*, Huntsville, AL, USA, June 2020, 2 pp. (to appear).