جامعة نايف العربية للعلوم الأمنية
Naif Arab University for Security Sciences

Naif Arab University for Security Sciences
Journal of Information Security & Cybercrimes Research
مجلة بحوث أمن المعلومات والجرائم السيبرانية
https://journals.nauss.edu.sa/index.php/JISCR

JISCR

Mitigation of Application Layer DDoS Flood Attack Against Web Servers

CrossMark

Ahamed Aljuhani [1,3] *, **Talal Alharbi** [2,3], **Bradley Taylor** [3]
[1] *Department of Information Technology, Faculty of Computers & Information Technology, University of Tabuk, Tabuk, Saudi Arabia.*
[2] *College of Computer Science and Engineering, University of Jeddah, Jeddah, Saudi Arabia.*
[3] *Electrical Engineering & Computer Science Department, School of Engineering, The Catholic University of America, United States.*

Received 14 Dec. 2018; Accepted 22 Feb. 2019; Available Online 25 Mar. 2019

Abstract

The Application-layer Distributed Denial of Service (App-DDoS) attack is one of the most menacing types of cyber-attacks that circumvent web servers. Since the attackers have developed different techniques and methods, preventing App-DDoS attacks has become more difficult than ever before. One of the most commonly and targeted protocols in the application-layer is HTTP-GET flooding attacks. The attacker sends a large number of HTTP-GET requests from different infected devices to force the server to assign the maximum resources available in response to every single request. The objective of this attack is to exhaust the server's resources and deny service to the legitimate users. The App-DDoS attacks affect Quality of Service (QoS) and are extremely costly in terms of resource exhaustion. In this paper, we discuss development and testing of an App-DDoS attack detection and mitigation model in order to defend web servers against threats. Our design model employs three principle states: normal, screening and suspicious. The defense model transits between these modes based on the server load. We use Machine Learning (ML) techniques to provide high detection accuracy of App-DDoS attacks. Our experimental results demonstrate that this defense system is effective against App-DDoS attack.

I. Introduction

Many organizations and institutions provide online services, including online sales, education and entertainment. With the increasing number of online applications, the potential of such Distributed Denial of Service (DDoS) attacks is also increasing [1]. DDoS attacks have become more common and complicated in recent years. A single DDoS attack compromises many devices "zombies" target the victim and interrupt system services by inundating resources [4]. Using botnets, the attacker installs malware known as a master DDoS, to find vulnerabilities in other devices within the same network. Then the attacker controls these compromised devices and continues to attack the victim. Most malware programs have a high degree of automation to generate massive traffic directed toward the target [5]. When an attack occurs, it overwhelms the web servers and makes them inaccessible for valid users.

The App-DDoS attack is one of the major threats to web servers. The attacker employs botnets to send a large number of requests to the target server. One of the most commonly targeted protocols in the application-layer is

Keywords: *DDoS; Application layer DDoS; App-DDoS Attack; HTTP GET; DDoS Http; Security; App-DDoS Detection; Mitigation; NFV.*

* Corresponding Author: Ahamed Aljuhani
Email: 91aljuhani@cua.edu

doi: 10.26735/16587790.2019.002

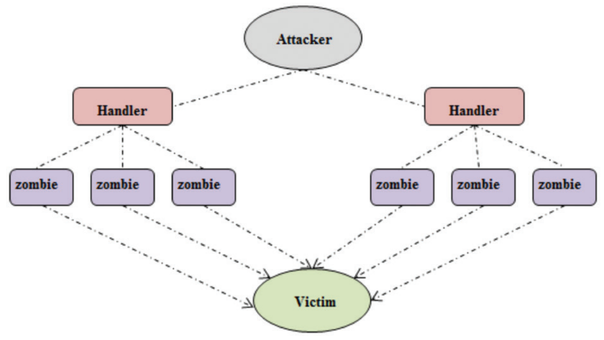

Fig. 1. The architecture of DDoS flooding attacks.

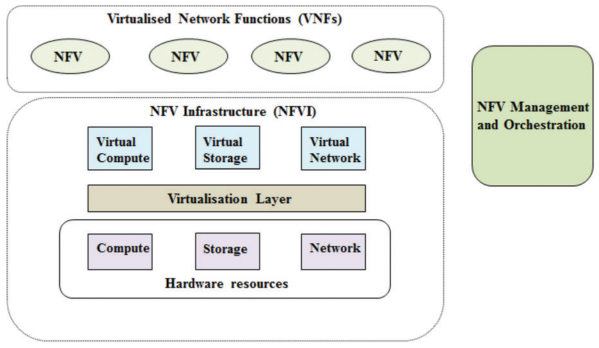

Fig. 2. Network Functions Virtualization (NFV) Architecture.

the HTTP protocol; flooding DDoS attacks on HTTP-GET are particularly of concern due to its wide integrate with online services [2]. Therefore, most web servers are vulnerable to such attacks. When an attack occurs, online services become unavailable and legitimate users cannot access a web server. As a result, businesses suffer significant financial and operating expenses for every hour a system is down. The impact of an attack depends on the size of the attack and the length of time the system remains unavailable. Businesses often lose range from $84,000 to $108,000 per hour when online services become unavailable [6].

With rapid improvement in technology and online systems, the application-layer DDoS attacks have been increasingly used by attackers. One common technique to launch such attack is to use botnets, a large group of compromised devices which attack a web server simultaneously. The volume and complexity of application-layer DDoS attacks have increased recently. Further, application-layer DDoS attacks are easy to use because necessary sources and tools are publicly available in the Internet.

Network Functions Virtualization (NFV) has been given more attention recently aiming to deploy network functions as software instances running on Virtual Machines (VMs) [7]. A main characteristic of the NFV is automation: instantiation, modification and deletion of Virtualized Network Functions (VNFs) can be automatically performed. Another essential feature is cost reduction; NFV reduces the capital expenditure (CapEx) and operating expenses (OpEx) [8].

The organization of this paper is as follows: Section 2 demonstrates the contribution of this work. Section 3 discusses the background and related work of the application-layer DDoS attack. Section 4 presents the methodology and system design of our defense model. Section 5 expresses important considerations when designing the defense system. Section 6 shows the experimental evaluation and results of this work. Finally, Section 7 summarizes the most essential points of the research and recommendations for future work.

II. Contribution

App-DDoS attacks require robust mitigation to help protect from malicious attacks. Current mechanisms use different procedures to defend against App-DDoS attacks, though they have several limitations. Limitations include slow/delayed attack detection, increased computational complexity, and reduced computational capacity of the dedicated hardware. In view of these problems, we presented a holistic DDoS mitigation framework that may be applied against all types of DDoS attacks enumerated in previous research studies [4, 13]. The focus of the present study, however, is on the App-DDoS attacks. We designed and developed an App-DDoS attack detection and mitigation model to defend web servers against threats. Three primary modes of system operation are: normal, screening and suspicious. Our system selects these modes based on the server's load. In our model, the resource monitoring protocol trigger upon changes in the server's load and generates alert messages when it is overloaded.

To ensure better detection of App-DDoS attacks, the system employs Machine Learning (ML) techniques during the screening mode to determine whether a given user is normal or an attacker. However, when the trained model cannot automatically discriminate whether the traffic belongs to a legitimate user or an attacker and the webserver continues to suffer from resource depletion (overload), the system switches to the suspicious mode. In that mode, each user must pass the CAPTCHA test in order to connect to the webserver. The defense system is designed to automatically to defeat App-DDoS attacks; every action is recorded into a reporting module for the security evaluation.

III. Background and Related Work

A. Background

Distributed Denial of Service (DDoS) attacks are a major threat to the infrastructure of a network. Network providers suffer from various types of DDoS attacks; each attack uses different, advanced techniques, such as botnets and malware, to augment and prolong an attack. Recently, sophisticated attacks using botnets involve IoT devices and utilize a new DDoS method to disrupt network services.

In current, DDoS attacks have been characterized by greater magnitude and greater complexity than attacks of the past. One of the remarkable trends in both scenarios is the large amount of traffic generated by the botnets. For example, in 2012, the botnet-based DDoS attack that targeted US bank generated traffic up to 75 GBPS [9]. In 2013, the Spamhaus website was attacked with generated traffic of 300 GBPS [9]. In 2014, an unnamed ISP was knocked down by a Network Time Protocol (NTP) DDoS attack with 400 GBPS of generated traffic [9]. In October 2016, the Mirai botnet targeted the DNS company named Dyn with a flood that reached 1.2 Tbps; the Mirai botnet attack was the largest DDoS attack to date [10]. Recent report shows that DDoS attacks cost businesses on average more than $2.5 million [10].

This operating and financial impact of DDoS attacks on businesses is tremendous. The concern of businesses about their network infrastructure grows after they become a victim or a target of DDoS attacks. The cost of an attack varies with the magnitude of the attack and the duration of the system's unavailability. Fig. 3 displays the distribution of DDoS financial impact, which 69% operational expense, 33% losses in revenue, 31% customer attrition and 14% employee turnover [12].

The application layer, seventh layer of the OSI model, interacts directly with the end user. This layer provides many protocols, including the File Transfer Protocol (FTP), Telnet, HTTP, the Domain Name System (DNS) and the Simple Network Management Protocol (SNMP).

In the application-layer, DDoS attacks focus on draining the server's resources, such as Sockets, CPU, memory, disk/database bandwidth, and I/O bandwidth. As a result, legitimate users cannot access online services, which are emptied of resources throughout the DDoS attack. App-DDoS flooding attacks endeavor to target a specific application protocol such as the HTTP- GET. The attack floods the application server with many requests and keeps the server busy handling these requests until the server runs out of resources and becomes unavailable. As a result, legitimate clients are not able to access the application. The attacker usually launches the application flooding attacks by using botnets, which make the server unresponsive to legitimate users.

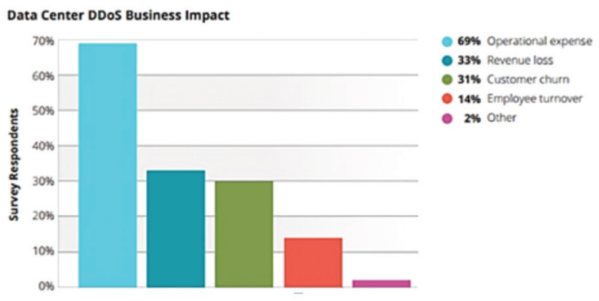

Fig. 3. DDoS financial impact.

B. Related Works

Somani et al. in [15], proposed DDoS mitigation technique "DARAC: DDoS Mitigation Using DDoS Aware Resource Allocation in Cloud" was deployed in the cloud. The detection method is based on human behavior analysis, and, blacklist of source IPs to filter out the malicious traffic. Also, it provided auto resource scaling for services operating onto a virtual machine. The resource utilization and traffic analysis were monitored within a window size of three minutes. However, the resource scaling is subject to service agreement constraints with the cloud provider.

In a recent study, Singh and De in [16] presented a defense model entitled DDoS Attack Detection and Mitigation Technique Based on Http Count and Verification Using CAPTCHA. The authors developed a method for IP blacklisting to block the whole blacklisted IP address. They used http counter to detect DDoS attacks. The suspicious IP was received CAPTCHA (puzzle test) in order to distinguish a normal user form a zombie machine. This method, however, depends only on the proposed http counter: attackers were able to deceive the proposed solution and mimic the user behavior through sending requests from a large group of devices below the http counter threshold which will be considered as normal users.

Devi and Yogesh in [17] proposed a DDoS detection method as an approach to counter application layer DDoS attacks. An Access Matrix was defined to capture the access information from legitimate clients of a web server. The access matrix contained the HTTP request rate, HTTP session rate, and duration of users' access. The method to counter DDoS counted a suspicious assignment once sessions were established. Then, based on the score, the counter method decided whether to accept or drop the packet. However, this method relied on the score calculation; if the proposed counter miscalculates, it drops legitimate traffic.

Wang et al. in [18] suggested two methods for resisting App-DDoS attacks. The App-DDoS attacks consist of two categories, flooding attacks and asymmetric attacks. In flooding attacks, the authors used entropy predictions, based on selected features, to detect attacks. In the second type, asymmetric attacks, the paper used a second-order Markov Model to determine the normal user behavior and calculate the deviation between the current access sequence and the prediction sequence of each user. However, the authors stated that the detection accuracy needs to be improved in future work.

IV. System Discription

Our model is designed to mitigate App-DDoS flood attacks: aiming to detect attacks at an early stage by receiving alert messages from the resource monitoring protocol. As a result, the server assures that App-DDoS attacks will not degrade Quality of Service (QoS) for legitimate users keeping the server from becoming unavailable. The system has three primary modes: normal, screening and suspicious. When the system runs in a normal condition even if there is high demand from certain users, the normal mode continues to operate as long as the system remains in normal circumstances. However, when a webserver resource depletion exceeds a predefined criteria, an alert message is sent and the system switches over to the screening mode. In the screening mode, the screener uses our trained ML model to check the traffic and determine whether the current user is malicious or not. If an attack is detected, the screener calls the mitigation algorithm to prevent the attacker from accessing the server. However, if the trained ML model has insufficient information to determine whether the traffic belongs to a legitimate user or an attacker and the webserver continues to suffer from resource depletion, the system shifts into the suspicious mode. In the suspicious mode, each user must pass a CAPTCHA test in order to connect to the webserver. When the user solves the graphical test, the connection is verified legitimate.

Our goal is to minimize system disruption. During an incident, the system switches to the screening mode (intermediate phase). If the load remains above the accepted limit, the system transits into the suspicious mode. However, it returns to the normal mode when server's load falls below the acceptable limit. After exiting the suspicious mode, the system returns to the normal mode. All three modes are implemented in the screener.

In our system, modes are run exclusively. The normal mode is the initial starting point for the system; expectation of the system to operate in this mode most of the time. Fig. 4 shows the state transition diagram for our model. Our App-DDoS defense system as shown in Fig. 5 consists of several modules that together achieve this robust mitigation in the face of App- DDoS attacks. The details of the App-DDoS defense system is elucidated in the rest of the paper.

A. Screening Mode

The normal mode is the initial starting point for the system; expectation of the system to operate in this mode

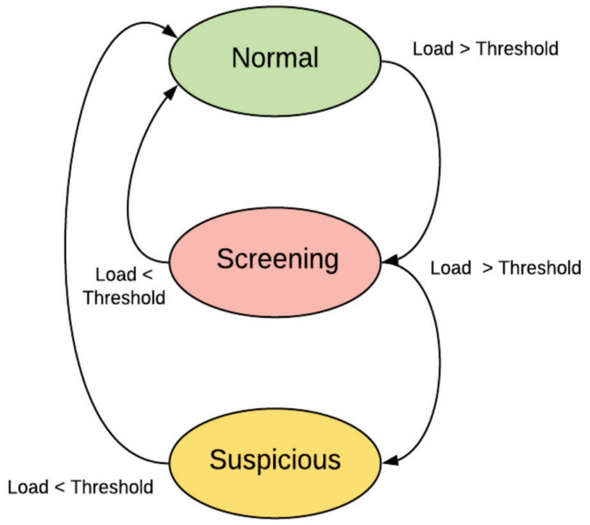

Fig. 4. Transitioning mode based on the server load.

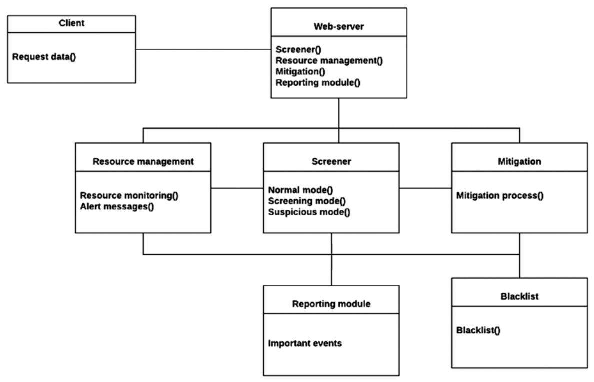

Fig. 5. App-DDoS defense class.

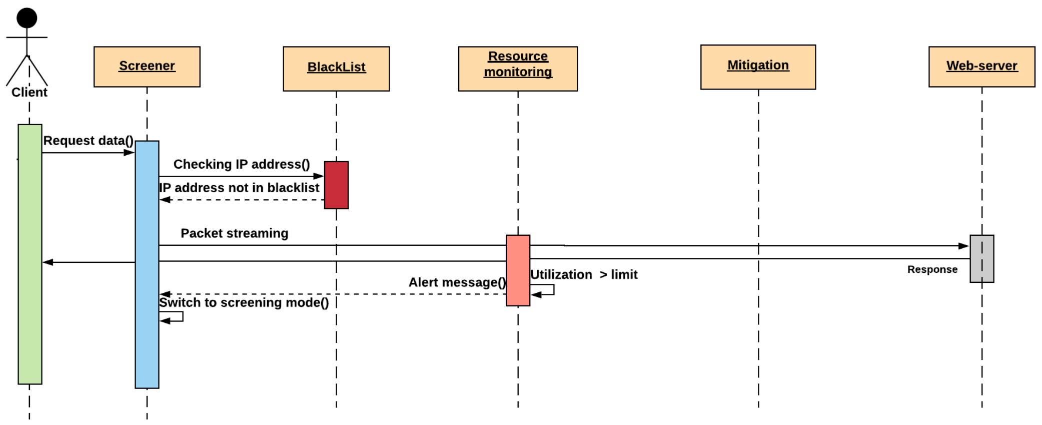

Fig. 6. System's operation during the screening mode.

most of the time. The normal mode indicates that there is no overload of the system's resources. The system runs in the normal mode as the system operates in normal conditions. Fig. 6 presents the sequence of the system operations during the normal mode; showing how the screener interacts when the system encounters resource exhaustion and then switching to the screening mode.

B. Suspicious Mode

When the resource usage of the web server exceeds a certain limit, such as if it encounters a flood of App-DDoS attacks, an alert message is generated by the resource monitoring protocol, and the system switches into the screening mode. In the screening mode, the screener uses the trained model of the logistic regression method to check the traffic and identify the user's behavior whether a normal user or an attacker.

1) Logistic Regression

Logistic regression is a common used method for binary classification due to its simplicity and efficiency. We employ it to determine whether a given sequence of requests is normal or an attacker (0 or 1). Formally, given a set of parameters w to learn for the model, a dataset X of size K and true labels y, logistic regression minimizes the following objective:

$$J(w) = \sum_{i=1}^{K} -(y_i \log(\sigma(w^T X_i)) + (1-y_i)\log(1-\sigma(w^T X_i))) \qquad (1)$$

where $\boldsymbol{X}_i$ is the i^{th} sample in $\boldsymbol{X}$ and $\sigma(t) = \frac{1}{1+e^{-t}}$

is the logistic function. To construct the detection

algorithm, we train and test the dataset during offline processing. After the algorithm is trained and tested, we model the algorithm online using its trained parameters. The following subsection illustrates this process.

2) Dataset Description

In order to train and test the algorithm, the App-DDoS dataset has been created for both normal and attack. We use the ClarkNet-HTTP to define normal user behavior [14]. This dataset includes host name, time stamp, HTTP-GET requests, and destination address. The attack data has been 'collected by simulating attacks toward the target. The traffic analyzer software was used to capture all incoming traffic. Before feeding the dataset to the algorithm, analysis and data cleaning are performed. Missing information and gaps in the dataset are filtered to not be trained in the model, to assure both the quality of training and accuracy of the classification.

- **Feature Extraction**

We used the difference in time between requests made by a user. As a normal user might show large variations in a sequence of time requests, the attacker may present a sequence of requests with small variations, aiming to flood a webserver with many requests in short time. The difference in time of those requests would explore a much more telling pattern to distinguish whether users are legitimate or not. When processing complete, the dataset sample helps identify a set of k features. We denote a given sample as a vector x of dimension k, where xi represents the times at which requests i is arrived in sample x.

- **Training and Testing**

The algorithm is trained and tested offline to construct the online prediction model for help in detecting App-DDoS attacks during the screening mode. The offline process is outlined in Algorithm 1.

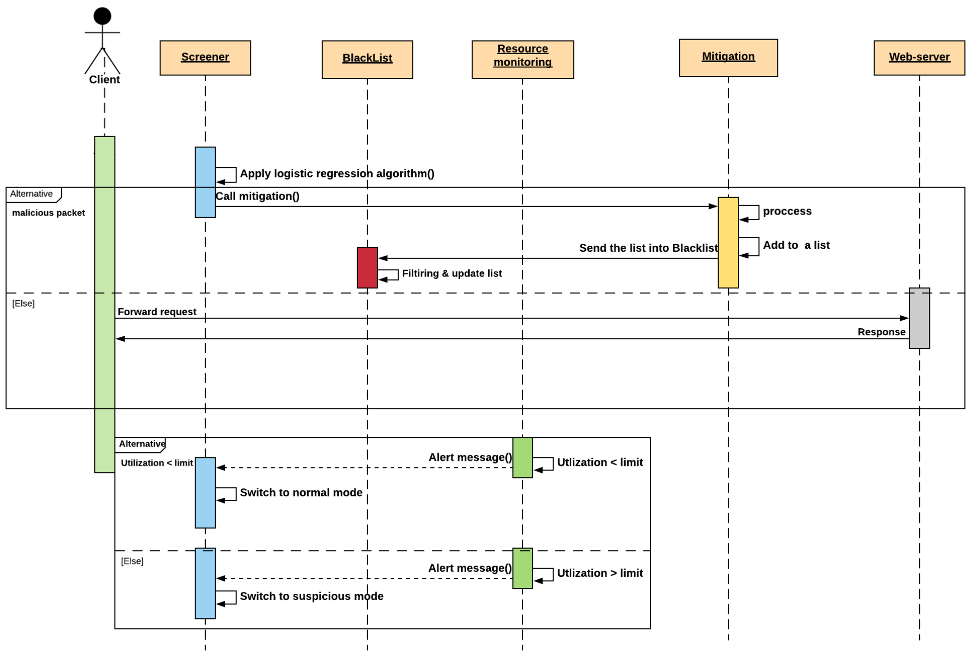

Fig. 7. System's operation during the suspicious mode.

```
Algorithm 1: Offline Process
1: Procedure: Training and testing
2: Input: Dataset
3: Output: learning parameters
4: Analyze and structure the dataset
5: Feature extraction
6: Training
7: Store the learning parameters
8: End procedure
```

After the offline processing, the prediction model based on the logistic regression method will be used during the screening mode to help identify the user as a legitimate user or an attacker. Fig. 7 presents the sequence of the system operations during the screening mode; showing how the screener interacts when the system encounters resource exhaustion and then running the detection algorithm to distinguish a normal user from an attack. When an attack is detected, the detection algorithm activates the mitigation process to stop and block the attack. As a final step, the system returns to the normal mode.

C. Mitigation Module

Often after shifting to the screening mode, attacks can be detected and the mitigation algorithm immediately applied to prevent attack and return system loading to normal. However, if the server's resources continue to be drained, the system shifts into the suspicious mode and applies the CAPTCHA mechanism to distinguish legitimate user from zombie machines. However, our goal is to identify attacks with minimal use of the CAPTHCA method to avoid annoying users.

When the system shifts into the suspicious mode, it sends a CAPTHCHA puzzle to users for authentication. The normal user can easily solve the puzzle and obtain access to a web server. However, zombie machines struggle to solve CAPTCHA puzzles; they either try to mimic human users who are not able to solve puzzles after many attempts or keep sending requests continuously without solving puzzles. Those who keep sending requests without solving the test are identified as zombie machines. However, for attackers who try to mimic human users and cannot solve puzzles after several attempts, they will be considered zombie machines, and then mitigation procedure will run. We consider situations exist where a normal user sometimes fails to solve a graphical test on the first attempt; the system will not consider that user as a zombie machine. However, if a user continuously fails to solve a graphical test and exceeds a baseline limit for attempts, it will be classified a zombie machine. The suspicious mode returns to the normal mode when loading returns below threshold. Algorithm 2 illustrates our application of the CAPTCHA method.

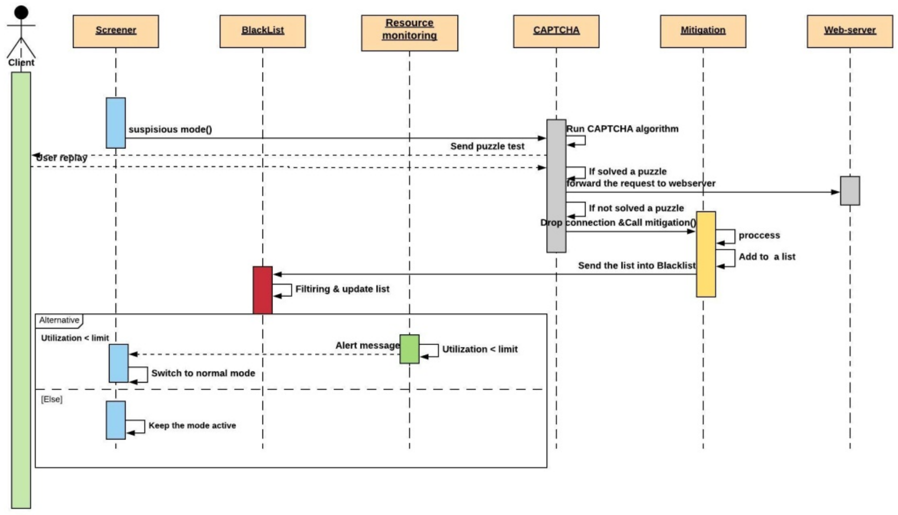

Fig. 8. System's operation during the suspicious mode.

Algorithm 2: CAPTCHA Method

```
1: Procedure: Puzzle test
2: Input: Incoming packet
3: Output: Whether a user is ligitimate or a zomby
4:    for each IP address do
5:        send CAPTCHA
6:        number of attempts ++
7:        if a user solves the puzzle then
8:            the user is ligitimate & forward the request
9:        else if number of attempts > threshold then
12:   10:         drop the connection & add the IP to a black
11:       endif
12:   end for
13: end procedure
```

Algorithm 3: Mitigation Process

```
1: Procedure: Block IPs
2: Input: attack's information
3:   For all IPs ∈ screener do
4:       If IPs are not in the black list then
5:           blacklist + +
6:       end if
7:   end for
8:   For all IPs ∈ blacklist do
9:       If IPs in the black list then
10:          Increase blocking time interval
11:      else
12:          block_IPs temporally
13:  end for
14: end procedure
```

Fig. 8 presents the sequence of the system operations during the suspicious mode. After applying the screening mode, the system checks the resource and determines if the resource is above the acceptable limit, it switches into the suspicious mode. This process ultimately will be applied the CAPTCHA mechanism to distinguish legitimate users from zombie machines.

D. Mitigation Module

When an attack is detected, the detection algorithm activates the mitigation process to stop and block the attack. The mitigation algorithm receives the attacker's information detected by the screener detection algorithm. Then the algorithm checks if the source IP already has been captured; if not already in the list, it will be added. If a given IP address comes up again, it may increase the blocking time interval. The mitigation module does not block the IP address permanently because some legitimate IPs are included in zombie groups for a short time. When legitimate users discover misuse devices and perform security updates, the source IP might once again be a legitimate user. Such zombie machines are usually created from the same network in order to launch an effective DDoS attack. Attackers seek a large group of devices, in order to do that, the attacker may infect a specific network to turn its machines into botnets for the purpose of attacking a specific target. To enhance the performance of mitigation, our mitigation algorithm includes the capability to block IPs which come from the same network or subnet instead of blocking each individual source IP. The blacklist is updated; it expires within a pre-determined time to prevent blocking legitimate users in the future. Algorithm 3 illustrates the mitigation process.

E. Resource Management

To determine the critical level and normal level, resource utilization is managed according to the predefined criteria. When any resource exceeds the acceptable limit, resource monitoring protocol sends an alert message to the screener. Then, the screener switches into the screening mode and applies the detection algorithm. By the end of the screening stage, we expect the resources to be back to the normal level. However, if the resource remains above the acceptable limit, the screener switches into the suspicious mode. The system switches between modes based on the server's load Algorithm 4.

Algorithm 4: Resource Monitoring Protocol

```
1:Input: Server (ID, M) // M is the maximum capacity
2:Procedure: Trigger resource usage and send alert messages
3:loop:
4: calculate $U = \frac{C}{M}$ U: used resource, C: current utlization
5:      if $U >$ threshold
6:          {send an alert message to the screener
7:          send alert_message to the reporting module}
8:      else if $U <$ threshold
9:          {send an alert message to the screener
10:         send alert_message to the reporting module}
11:     end if
12: end loop
13: end procedure
```

F. Screener

The screener component is deployed on the application server to protect the server from App-DDoS attacks. It continuously checks for alert messages sent by resource monitoring protocol. It is also responsible for changing detection mode based on the type of the alert message. For example, if an alert message is received indicating that the resource usage is exceeded a threshold and the system operates in the normal mode, the screener changes the system mode to screening. If the screener receives another alert message indicating that the resource usage is exceeded a threshold, the screener changes the mode to suspicious. Algorithm 5 illustrates the pseudocode of the screener process.

Algorithm 5: Screener Process

```
1: Input: Alert messages
2: Procedure: Switching between system modes
3:mode=normal
3: loop:
4: check for alert message
5: if alert message and mode == normal
6:      mode = screening
7:      for each incoming IP
8:          apply logistic regression
9:          if pass test
10:             forward request
11:         else
12:             call mitigation
13:         end if
14:     end for
15: else if alert message and mode == screening
16:     mode = suspicious
17:     for each incoming IP
18:         apply CAPTCHA method
19:         if pass test
20:             forward request
21:         else
22:             drop connection and call mitigation
23:         end if
24:     end for
25:else if alert message and (mode == screening or
        suspicious)
26:     mode=normal
27:end if
28:end loop
```

G. Reporting Module and Policy Evaluation

The administrator evaluates important events that have been recorded during an incident. Based on the report, the administrator should make some arrangements to the policy and security rules as needed. Using this report, the administrator is able to review system's capabilities and make changes necessary to improve QoS in future.

V. Defense Against App-DDoS Attacks

The following considerations are taken into account when designing the protection model that provides effective mitigation against App-DDoS attacks.

- Early indications of the possibility of App-DDoS attacks: We use resource monitoring protocol to send an alert message when any server's resources are above the accepted limit. This warning limit gives an early indication of the possibility of App-DDoS attacks and allows the detection module to investigate traffic. Our method defines early detection by the server consumed resources.

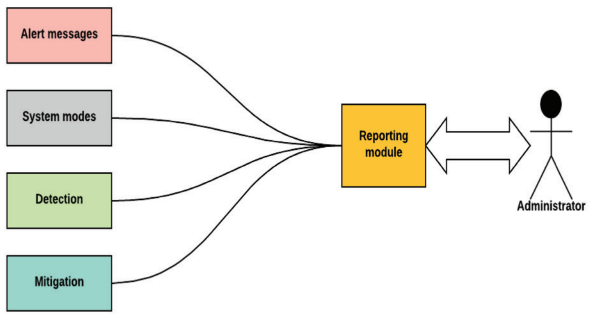

Fig. 9. Reporting analysis and policy evaluation

The resource monitoring triggers the web server resources in all modes in order to generate alert messages about resource depletion. As a result, the system transitions between modes based on the web server load.

- Automated response: DDoS attacks of this sort require a rapid response; the screener uses the detection method once it receives an alert message from the resource monitoring protocol. Because of this, the administrator does not need to be present in order to respond during an incident.
- A flash crowd event: A flash crowd forms when many legitimate users access the webserver simultaneously [19]. Our defense system is designed to distinguish a flash crowd from DDoS attacks. A flash crowd has mostly identical attributes to normal users while the attributes of App-DDoS attacks are radically different from those of legitimate users. Additionally, a flash crowd usually occurs during a special event, and a defense system should expect more users during such a time. Furthermore, during a flash crowd, users usually have small number of requests in comparison to App-DDoS attacks, which employ a large number of requests.
- Attackers mimicking normal users: In order to imitate legitimate users' behavior, attackers need an enormous group of zombie to emulate normal users [20]. Because of this, an adversary seeks for diverse of zombie machines within the same network. Our designed model is able to detect attackers who intimate normal users during the screening mode. Additionally, CAPTCHA mechanism will be performed during the suspicious mode in case if attackers have not detected through the screening mode.
- Minimal use of CAPTCHA: Our goal is to identify attacks without using the CAPTHCA method or at least to rely upon minimal use of CAPTCHA. We do not want to authenticate users using a CAPTCHA puzzle each time they need to access a web server because this is sometimes annoying for users. Our model is designed to detect attacks without using graphical tests. However, at some point, we use a graphical test when the server' resources continue to drain after the screening mode is complete. Otherwise, the graphical test does not appear during the normal and screening mode.
- Policy control: The administrator determines security rules for detection mechanisms and the learning parameters that are required during the screening stage. The administrator also defines the threshold values for server's resources.
- Effective mitigation: In our design system, we detected and mitigated attacks in an appropriate manner for the sake of performance and fast protection. Like a large group of zombies, they usually come from same LAN network, the mitigation algorithm prevents zombies by blocking subnet address instead of suspended each IP address, which takes time to process and influence the performance of the system. Also, our mitigation method temporarily adds the IP address to a black list, so malicious traffic will be blocked. However, when the IP comes up again, the mitigation may increase the suspended time.
- Reporting module: In the designed model, we record significant events in the reporting module. However, the reporting module acts like a summary report containing important events taken by the system like alert messages; whereas log file provides detailed information for each process. Therefore, the administrator reviews these actions and makes necessary changes to the system for QoS.

VI. Experiment and Evaluation

A. Testing Environment and Attack Scenarios

To evaluate the system design against App-DDoS attacks, a virtual environment was created. The test environment consisted of a web server with 2 VCPU and 2 GB memory running on Ubuntu 16.04 operating system. The resource monitoring algorithm was implemented to run continuously in the server's background to trigger the server's resources and generate alert messages. The screener was implemented with three primary modes as mentioned in the system description section. In the screening mode, we deployed the trained model based on logistic regression analysis. The offline process was performed to obtain the best learning parameters prior to deploying the algorithm for real-time detection.

For the external network, a legitimate user machine with 1 VCPU and 1 GB memory was created to send requests to the webserver. Additionally, two attacker machines with 1 VCPU and 1 GB memory were specified to generate App-DDoS attacks. The two attacker nodes are configured with Goldeneye attack tool [21]. We also installed Wireshark analyzer tool [22] at each station in order to capture and analyze the network traffic. Details of our experimental setup and description of machines are presented in the Figure 10 and Table 1, respectively .

To test and validate the defense system, we performed different attack scenarios targeting the webserver. The webserver was targeted first with a host of light attacks to determine the impacts on the target side. Next, we conducted heavy attack scenarios in which the server' resources were highly increased in the webserver and a main influence was occurred in a short time. We also created pulse attack scenarios to target the webserver with a series of short but powerful pulses of App-DDoS attacks. During light attack scenarios, no major impact on the webserver was occurred, which, in turn, showed no alert messages. Through the heavy and pulse attack scenarios, however, the defense system encountered resource depletion. The resource triggers sent alert messages to the screener trying to detect and prevent the attacks. In the following section, the results and discussion are presented.

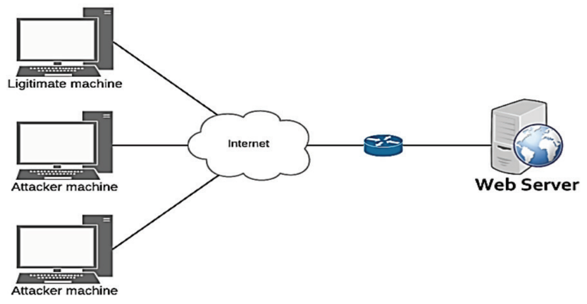

Fig. 10. Experimental setup.

TABLE I
MACHINE'S SPECIFICATION

Machine	Operating System	CPU	CPU
Server	Ubuntu 16.04	2 VCPU	2 GB
Legitimate user	Ubuntu 16.04	1 VCPU	1 GB
Two attacker machines	Ubuntu 16.04	1 VCPU	1 GB

B. Light App-DDoS Attack

We started with light App-DDoS attacks; no major effect on the webserver was noticed. The traffic rate was also low during this scenario Fig. 11 (a). With light attack scenarios, there were little errors over TCP connection see Fig. 11 (b). In addition, the CPU usage was influenced little between 30% and 35% Fig. 11 (c). The consumed memory was increased little and then remained steady Fig. 11 (d). The transmission rate was slightly increased before become fluctuated Fig. 11 (e).

C. Heavy App-DDoS Attack

During the heavy attack without mitigation scenarios, we were able to see the packet rate per second which was highly increased in moments with the high amount of App-DDoS attacks Fig. 12 (a). Meanwhile, the webserver was busy of handling all packets coming from attackers in a short time resulting in an increase in TCP errors Fig. 12 (c). With the heavy attack phase, the amount of CPU utilization was gradually occupied by attackers. As the attacks increased, the CPU utilization became overwhelmed; a response to the large number of requests Fig. 12 (e). The memory usage was also extremely influenced by the attacks, increased linearly with the raise in attacks and continued to drain Fig. 12 (g). When the webserver was flooded by heavy attacks,

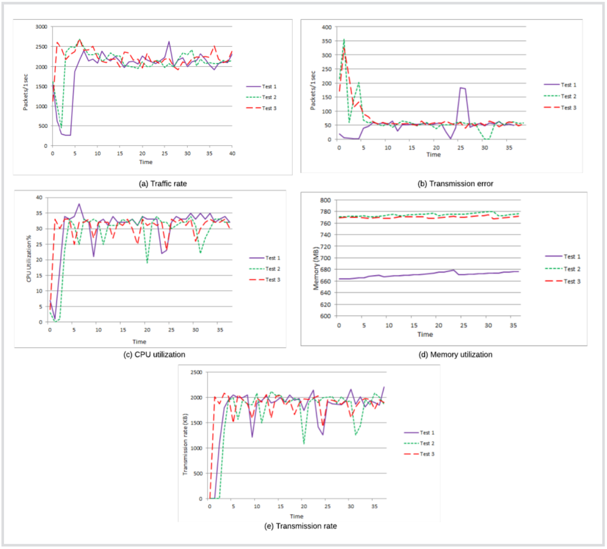

Fig. 11. Light attack.

the transmission rate was tremendously increased and continued to drain Fig. 12 (i).

When heavy attack scenarios occurred with mitigation, the attack led to a rapid increase in the traffic rate in a short time. When the attack was detected and blocked, the traffic rate decreased Fig. 12 (b). As the CPU usage also increased when the attack started, it returned

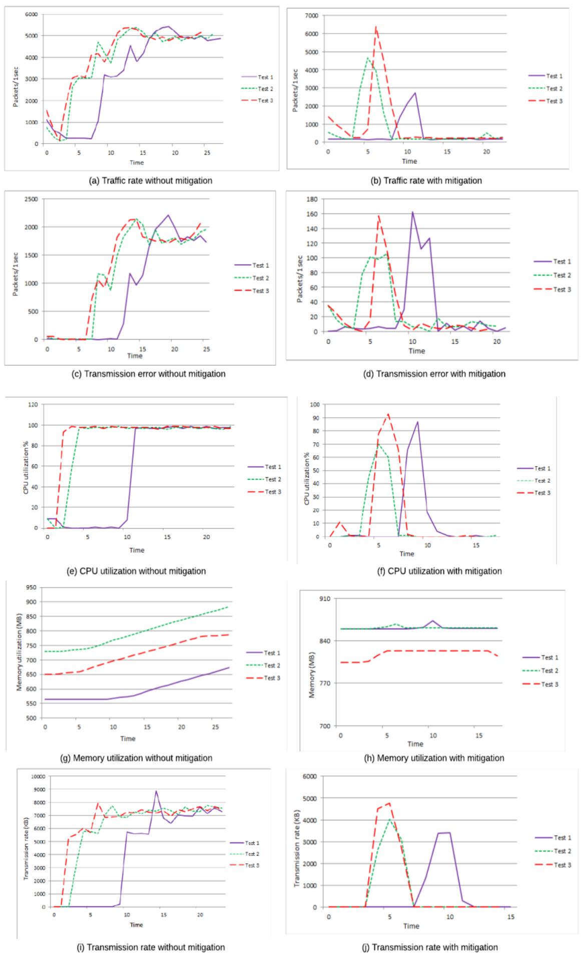

Fig. 12. Heavy attack.

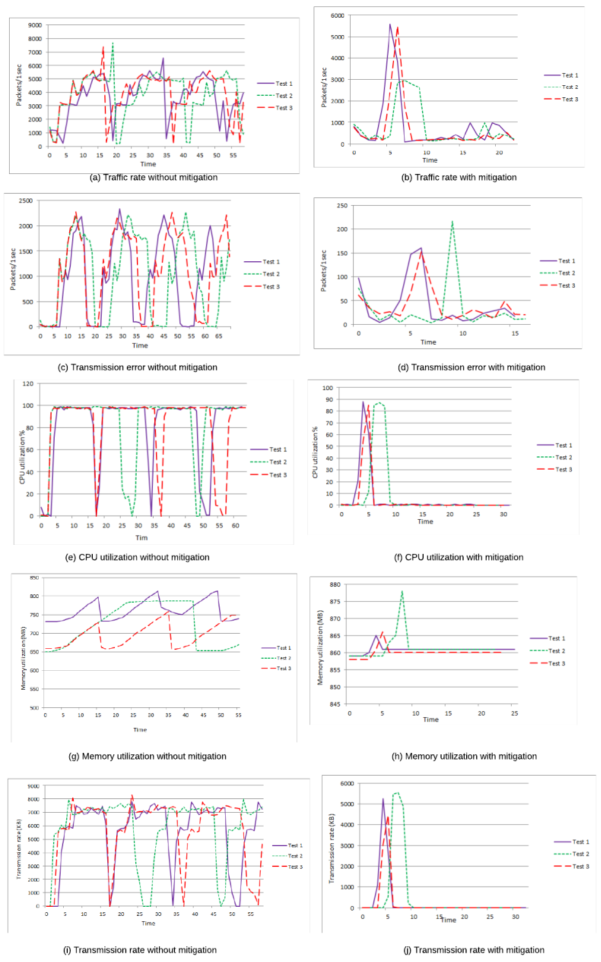

Fig. 13. Pulse attack.

to the normal condition with the detection and blocking processes Fig. 12 (e). Further, the memory usage was increased slightly at the beginning of the attacks. When the attacks were detected and blocked, the memory usage stopped to increase and remained stable Fig. 12 (h). The transmission rate was also raised during the attack. However, when the attack was successfully mitigated, the transmission rate was dropped down and remained constant Fig. 12 (j).

D. Pulse App-DDoS Attack

Another attack scenario was considered pulse attacks: the attacker floods the webserver with multiple attacks during a short time. When the attacks reached the peak, another attack started and so on. As seen in Fig. 13 (a) and 13 (c), this attack influenced traffic rate and increased the transmission error connection with high rate of error at each round. The CPU utilization reached about 100% of CPU usage for a few seconds before turning down. This attack repeated this strategy multiple times Fig. 13 (e). The memory utilization was fluctuated between high and low level of used memory. The transmission rate was high, and then sharply fell down, and started again to increase. This is associated with the series of short impulse attack multiple times made by the attackers Fig. 13 (i).

During pulse attack scenarios with mitigation, the traffic rate and the transmission error were increased and then dropped down in response to detection of hacking Fig. 13 (b) and 13 (d). On the other hand, the CPU and memory utilization were also influenced by attacks as the mitigation method discovered and blocked the malicious traffic. Based on this, the CPU and memory utilization returned to the normal conditions Fig. 13 (f) and 13 (h). The transmission rate was increased when attacks started and then returned to the normal level Fig. 13 (j).

TABLE II
COMPARISON OF OUR SOLUTION METHOD WITH OTHER AVAILABLE APPROACHES

Citation	Attack identification method	Screening for DDoS	Detection method	Flash crowd detection	Automated response	Early detection	Early detection	Detect zombie
Detect zombie	Traffic Evaluator	auto-scaling overload method	Analyzing user behavior	No	Has to check with Service Level Agreement (SLA)	Not specified	Low	No
Singh, K.J. and De, T	Http count	continuously	http counter	No	Not specified	No	Low	Yes
Devi & Yogesh	User Features	Not specified	Access matrix analysis	Yes	Not specified	No	Low	No
Wang et al	Not specified	Not specified	ML technique	No	Not specified	Not specified	High	Yes
Our solution	Alert message	Upon receiving alert message	ML technique	Yes	Yes	Yes	Low	Yes

E. Effectiveness Comparison

Table II compares our proposed mitigation methods with existing techniques. To do so, we identified the following standards:

- Attack identification method: identify signs of DDoS attacks.
- Screening for DDoS: whether method scans for DDoS attacks continuously or on demand.
- Detection method: detection algorithm being used
- Flash crowd detection: forms when many legitimate users access a webserver simultaneously.
- Automated response: DDoS attacks response occurs without any involvement from the administrator.
- Early detection: early sign of DDoS attacks before damaging the target.
- Complexity: the level of computation for method being used.
- Detect zombie: attackers mimicking normal users.

VII. CONCLUSIONS AND FUTURE WORK

In this article, we have proposed an App-DDoS defense system that effectively mitigates App-DDoS attacks. The App-DDoS attack is one of the common types of cyber-attacks that currently circumvent web server security. Organizations and businesses are suffering from App-DDoS attacks when relying on the Internet to provide online services to their customers. In this research, we have shown that our defense system effectively mitigates App-DDoS attacks in a variety of different scenarios. The system designed has three primary modes: normal, screening and suspicious. The normal mode indicates that the system is running in normal conditions. However, when our monitor notices a resource depletion exceeding predefined criteria, it generates an alert message. The system switches over to the screening mode. During screening stage, our model checks the traffic and determines whether the current user is an attacker or not. Then, the mitigation algorithm runs to prevent attackers from accessing the webserver. However, if the webserver continues to suffer from additional resource depletion, the system shifts into the suspicious mode. In the suspicious mode, each user must pass the CAPTCHA test in order to connect to the webserver. Our defense model records important events during the attack to aid in the forensic evaluation of security. The experimental results demonstrate that the defense system is effective against App-DDoS attack.

Our system was designed to mitigate HTTP-GET flooding DDoS attacks, one of the primary vectors of DDoS attacks. However, there are numerous other types of attacks in this layer that worthy of study, including Domain Name System (DNS) attacks, Simple Message Transfer Protocol (SMTP) attacks, and Session Initiation Protocol (SIP) attacks. Each of these types can be used by attackers to exploit vulnerabilities and launch effective DDoS attacks. Another consideration for future work is the development of a dynamic threshold for the resource monitoring protocol based on the requirements of the system during high flood traffic. A fixed threshold depends on the system's capabilities and resource availability. The system failure problem is another direction for future work. A method could be found that provides backup and recovery for the

system after unexpected incidents have occurred. Even during the mitigation process, a single point of failure may stop the entire system from functioning correctly.

References

[1] K. Singh, P. Singh and K. Kumar, "Application layer HTTP-GET flood DDoS attack: Research landscape and challenges," *Comput. Secur.*, vol. 65, pp. 344-372, Mar. 2017, doi: 10.1016/j.cose.2016.10.005.

[2] K. Singh, P. Singh and K. Kumar, "User behavior analytics-based classification of application layer HTTP-GET flood attacks," *J. Netw. Comput. Appl.*, vol. 112, pp. 97-114, June 15, 2018, doi: 10.1016/j.jnca.2018.03.030.

[3] S. Mansfield-Devine, "The growth and evolution of DDoS," *Netw. Secur.*, vol. 2015, no. 10, pp. 13-20, Oct. 2015, doi: 10.1016/S1353-4858(15)30092-1.

[4] T. Alharbi, A. Aljuhani, H. Liu and C. Hu, "Smart and Lightweight DDoS Detection Using NFV," in *Proc. Int. Conf. Comput. Data Anal. (ICCDA '17)*, Lakeland, FL, USA, May 2017, pp. 220-227, doi: 10.1145/3093241.3093253.

[5] T. Thapngam, S. Yu, W. Zhou and G. Beliakov, "Discriminating DDoS attack traffic from flash crowd through packet arrival patterns," *2011 IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Shanghai, 2011, pp. 952-957, doi: 10.1109/INFCOMW.2011.5928950.

[6] eCommerce web site performance today. An updated look at consumer reaction to a poor online shopping experience, 2012. [Online].Available: http://www.damcogroup.com/white papers/ecommerce_website_perf_wp.pdf

[7] ETSI Industry Specification Group, "Network Function Virtualization (NFV); Architectural Framework," ETSI, France, RGS/NFV-002 V1.2.1, 2014.

[8] A. Aljuhani and T. Alharbi, "Virtualized Network Functions security attacks and vulnerabilities," *2017 IEEE 7th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Las Vegas, NV, 2017, pp. 1-4, doi: 10.1109/CCWC.2017.7868478.

[9] D. Boro and D. K. Bhattacharyya, "DyProSD: a dynamic protocol specific defense for high-rate DDoS flooding attacks," *Microsyst. Technol.*, vol. 23, no. 3, pp 593-611, May 18, 2016, doi: 10.1007/s00542-016-2978-0.

[10] Nicky Woolf, " DDoS attack that disrupted internet was largest of its kind in history, experts say," Oct. 26, 2016. [Online] .Available: https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet (accessed Dec. 2, 2016).

[11] Neustar, "Worldwide DDoS Attacks & Cyber Insights Research Report," May 2017. [Online].Available: https://www.discover.neustar/201705-Security-Solutions-DDoS-SOC-Report-LP.html

[12] Worldwide infrastructure security report, 2017. [Online].Available: https://www.arbornetworks.com/images/documents/WISR2016_EN_Web.pdf

[13] T. Alharbi, A. Aljuhani and Hang Liu, "Holistic DDoS mitigation using NFV," *2017 IEEE 7th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Las Vegas, NV, 2017, pp. 1-4, doi: 10.1109/CCWC.2017.7868480.

[14] ClarkNet-HTTP, http://ita.ee.lbl.gov/html/contrib/Clark-Net-HTTP.html.

[15] G. Somani, A. Johri, M. Taneja, U. Pyne, M. S. Gaur and D. Sanghi, "DARAC: DDoS Mitigation Using DDoS Aware Resource Allocation in Cloud," in *Int. Conf. Inf. Syst. Secur.*, in Information System Security, in Lecture Notes in Computer Science, vol. 9478, pp. 263-282, doi: 10.1007/978-3-319-26961-0_16.

[16] K. J. Singh and T. De, "DDOS Attack Detection and Mitigation Technique Based on Http Count and Verification Using CAPTCHA," *2015 Int. Conf. Comput. Intell. Netw.*, Bhubaneshwar, 2015, pp. 196-197, doi: 10.1109/CINE.2015.47.

[17] S. R. Devi and P. Yogesh, "An effective approach to counter application layer DDoS attacks,' in *2012 Third Conf. Comput. Commun. Netw. Technol. (ICCCNT 2012)*, Coimbatore, 2012, pp. 1-4, doi: 10.1109/ICCCNT.2012.6395941.

[18] Y. Wang, L. Liu, C. Si and B. Sun, "A novel approach for countering application layer DDoS attacks," *2017 IEEE 2nd Adv. Inf. Technol. Electron. Autom. Control Conf. (IAEAC)*, Chongqing, 2017, pp. 1814-1817, doi: 10.1109/IAEAC.2017.8054326.

[19] Q. Le, M. Zhanikeev and Y. Tanaka, "Methods of Distinguishing Flash Crowds from Spoofed DoS Attacks," *2007 Next Generation Internet Netw.*, Trondheim, 2007, pp. 167-173, doi: 10.1109/NGI.2007.371212.

[20] H. Beitollahi and G. Deconinck, "ConnectionScore: a statistical technique to resist application-layer DDoS attack," *J. Ambient Intell. Humanized Comput.*, vol. 5, no. 3, pp. 425-442, Jul. 10, 2013, doi: 10.1007/s12652-013-0196-5.

[21] Github, DDoS Attack Tools, 2013. [Online].Available: https://github.com/jseidl/GoldenEye

[22] H. Kaur, S. Behal and K. Kumar, "Characterization and comparison of Distributed Denial of Service attack tools," *2015 Int. Conf. Green Comput. Internet Things (ICGCIoT)*, Noida, 2015, pp. 1139-1145, doi: 10.1109/ICGCIoT.2015.7380634.