New Types of Dos/Ddos Attacks Require Alternative Approaches to Security

Rashad Aliyev[1], Tural Mammadov[2]
[1]KhazarUniversity Baku, Azerbaijan
[2]Special Communication and Information Security State Service adyunkt Military Academy Baku, Azerbaijan
[1]rashad@aliev.info, [2]mammadov.t@cert.gov.az

ARTICLE INFO

Article History
Received 10 Feb 2020
Received in revised form 07 Dec 2020
Accepted 08 Mar 2021

Keywords:
DDoS attack simulation, DoS attack simulation, minimized attack vector, simulation, Hulk DoS tool, threat, vulnerability, code execution performance

ABSTRACT

This study examined cases where a system completely crashes when a user sends 2–3 DoS requests per second. The aim of these attacks is to find weak code or incorrect settings in the system. The research highlights the inadequacy of providing protection only at code and server levels or with security equipment, especially in critical infrastructures and dedicated networks. The performance and resource requirements of applications running on servers should also be considered and evaluated as security elements. The Hulk DoS tool was used to detect and simulate the problem mentioned in the research by carrying out attacks with different frequencies. Furthermore, various solutions are given to find and prevent these attacks.laboratory accreditation, the methodology, and the lesson learnt. This paper is hoped to provide guidance to those who would like to pursue accreditation for their Digital Forensics Laboratories (DFL).

I. INTRODUCTION

Denial-of-service (DoS) and distributed DoS (DDoS) attacks have increased since 2000, are estimated to cost $1.7 billion [1] and have now been ranked among the top 10 most serious threats to information security by international rating agencies such as OWASP [2]. When DDoS attacks originated in 2017, 270.6 Gbps attacks were reported, in 2018 it reached 600 Gbps attacks, and currently DDoS attacks are reaching 1Tbps. This is a possible immediate threat to the accessibility of any information resource. Further, the attempted cyber attack last year, which was recorded at 1.7 Tbps, once again confirms the above-mentioned threat [3].

More than 20 years have passed since the detection of this cyber-attack vector, and various solutions have been offered to prevent DoS or DDoS attacks, which are still used today for the most active threat source.

Thus far, the analysis of a number of projects have shown that to carry out DoS and DDoS-type cyber attacks, hackers first need to identify the weakest part of the target resource in terms of traffic bandwidth and then carry out a cyber attack according to the detected vulnerability. Such attacks are carried out using a network of zombie computers creating botnets, network, server, or application-layer vulnerabilities. The article "Optimal specifications for a protective framework against HTTP-based DoS and DDoS attacks" [4] provides a categorized table of types of DoS and DDoS attacks. The authors provide the information on the stages of DoS and DDoS attacks categorized as follows:

- Attacks focused on the network;
- Attacks focused on server power;
- Attacks focused on software vulnerabilities used.

This article discusses DoS and DDoS attacks that focus on memory overload or denial of service by manipulating the load created on the information system by the software used on that resource to disable the information resource. To carry out such attacks, malicious hackers often create a bottleneck effect for traffic flow, making it difficult for the server to process newly received requests. Further, information resources with more secure and larger network and server powers need to have a large botnet network to respond to such attacks.

II. THE PROBLEM

One study [5] showed how DDoS attacks on cloud services are implemented using XML. It was possible to observe that the attacks were carried out using the vulnerabilities in SOAP services. The research focused on SOAP services, which are no longer used to perform queries in XML; it can be observed that instead of preventing it, new technology—the use of services such as REST—was introduced. Even at a time when addressing the vulnerability of SOAP services is questionable and pending, [6] suggests that it provides a better and faster solution, and other studies [7], [8] report that it is difficult to understand but convenient. This is because SOAP technology, similar to WSDL, has a number of internal standards, which has led to the rejection of SOAP by software engineers.

Further, in a previous study [9], a DDoS attack focused on routers and traffic was simulated. Because such attacks are related to the volume of traffic, many vendors recommend the use of equipment with high traffic processing and cleaning power to prevent such attacks. By preventing such attacks at the hardware level, it is possible to ensure that end users are not impacted by these attacks. The main problem in using this type of equipment solutions is the throughput of the traffic channel. Thus, if the channel has a bandwidth of 1 Gbps, and the volume of DDoS attacks on that channel exceeds 1 Gbps, there will be a bottleneck effect on the network, which will create queues and accessibility issues to the information resource for new users.

Special attention should be paid to the research done by [10] on the simulation of DDoS attacks using SYN-flooding, NTP-flooding methods, and solutions to these problems. This research provides information on tools such as DDOSIM, TrinOO, BoNeSi, Tribe Flood Network, their user platform, as well as attack methods and rules.

Unlike the methods listed above, another DDoS attack method disables a programming module using a vulnerability in an executable program. Thus, high traffic capacity, correct installation of web server settings, and the installation of special anti-DDoS equipment to prevent DDoS attacks, as practice shows, does not offer protection from non-traditional DDoS attacks. These DDoS attacks are carried out by a programmer attempting to detect incorrect installations, errors in the program code, or any process that takes a long time to execute the code. These attacks, which are performed with minimal resource consumption and using 2-3 requests per second from one source, have the ability to bypass a security system or completely stop the operation of large information systems.

The method that allowed us to jeopardize the accessibility of information resources without the need for a large zombie computer network in the implementation of the above-mentioned DoS and DDoS attacks is considered by us as a serious security vulnerability.

III. GOAL

The main goal of this research is to explain a non-traditional DDoS attack method, which uses minimal resource consumption by using a special tool, to attract the attention of researchers in the field and to offer solutions in this direction.

Detailed information on the installation of bots for DDoS attacks and the process of DDoS attacks using zombie computers are provided in [11]. The article also provides detailed information on the different ways bots can be deployed (Scanning, Exploitation, Deployment, Propagation), as well as the categories of DDoS attacks and client/server programs used during attacks. In addition to investigating the organization of the attacks, this article discusses, in detail, the detection of attacks and the steps that can be taken to prevent it. In contrast to this research, which uses bot simulations for DDoS attacks by occupying different resources, our study makes it is possible to occupy the resources of the target server by sending a small number of requests using only one resource instead of a large bot network.

According to statistics [12], DDoS attacks can be divided into three categories: attacks that occupy resources with high volumes of traffic – "volumetric," attacks aimed at occupying connections – "TCP state-exhaustion," and "application-layer" attacks – targeting the application level. This study did not mention any threat to the accessibility of any service by filling up the memory. Based on this research, it can be concluded that application-oriented attacks can also be divided into two categories. These are:

Server-focused DDoS attack: Attacks on software running on a server (e.g., web server, SMTP server, and other services). In this case, queues for new incoming requests arise because the web server or process does not have the capacity to process the specified queries. This increases the queue because of a lack of resources, and the server is unable to process new incoming requests. As a result, the server either needs to be restarted or will freeze and not execute any requests.

Service-oriented DDoS attack: Attacks aimed at denying sub-services (modules) to be performed using software running on the server. In this case, although the resource allocated for the server and web service is sufficient, less resources are allocated for execution and time processing for the module to use (PHP module in our example). In this case, when successive requests are sent, the module postpones the execution of those requests and delays occur. As a result, owing to the large number of requests from the web server, the source code is sent to the user instead of the already executed code.

In the second example, the attack is not aimed directly at the software running on the target server but at the vulnerability of its service (module). Examples include an incorrectly coded application or an outdated application version. At this time, only an anomaly is perceived and this is observed in the log records. Because there are many intervals between automated queries, the firewall does not accept this attack as an anomalous botnet attack but as a standard user query and does not block it.

This study will focus on DDoS or DoS attacks, which are not considered abnormal by security systems and have a minimal number of connections, unlike traditional botnet attack logic.

Before investigating the problem, we first provide the principle of the operation of web servers (Figure 1):

- User sends a request over the Internet to the address of any site (example: https://www.example.com/contact.php).
- In the first stage, the DNS query is executed and the IP address where the information resource related to the domain name "www.example.com" is located is searched.
- The next step is to contact the service used on the appropriate port at the IP address specified. In our example, this will be an Apache server running on port 80.
- The Apache server analyses the incoming request, if it is a static file request (.html, .css, .js, .jpeg, etc.) then it returns the file to the user in the appropriate folder according to the incoming request. If the query is any executable script, it sends it to the appropriate module for execution. In our example, this would be a .php extension file.

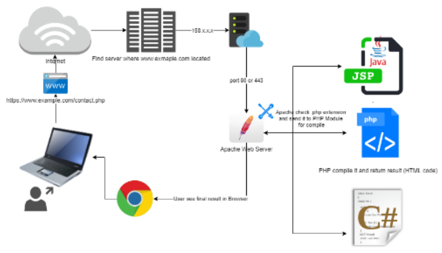

158.x.x.x
Internet
Find server where www.exmaple.com located
https://www.example.com/contact.php
port 80 or 443
Apache check .php extension and send it to PHP Module for compile
Apache Web Server
PHP compile it and return result (HTML code)
User see final result in Browser
Java
JSP
php
C#

Fig 1

The configured PHP module for the .php extension receives the corresponding file based on the request sent, processes the code, and returns the result to the Apache server.

- The Apache server returns the processed result to the user.

This article will explain the reason for the denial of service caused by DDoS or DoS attacks that is carried out by influencing the processing of simple queries in accordance with the processing of a request as described above. To do this, a process simulation was performed on a sample system using a CentOS 7 operating system, Apache web server, Nginx, and PHP scripts. The main purpose of this simulation is to present the reason for the refusal of the information resource when performing the mentioned attack, its detection, and prevention methods.

An example of a script that would slow down services for such simulations is the following:
For example, when sending a mail through SMTP in the contact

ISSN 2636-9680
eISSN 2682-9266

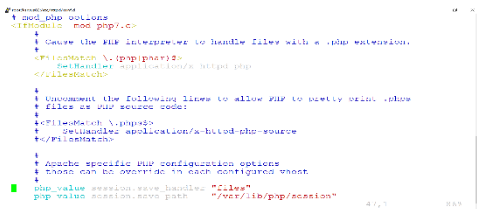
section of a website, the web server refers to the PHP module because the request is over the web. To do this, the PHP module can use the PHPMailer class [13], which is different from the standard mail() [14] function but more reliable. Here, the PHPMailer class of the PHP script connects to the SMTP host to send mail. Sometimes it is necessary to use Gmail, Mailgun, or other mail services for SMTP. In this case, a certain amount of time and resources are used to connect to the service specified by SMTP, login, and send the mail through PHP script in the PHPMailer class. As this process takes some time, it is already apparent that the web server is being loaded. By repeating the queries to this type of script 2–3 times per second, it is possible to increase the load on the processor in the information system and fully load the server.

This study used the Hulk DoS [15] tool to test service denial by sending various queries to any URL of the target by changing the user agent and referrer site. Iptables were used to block anomalous queries as a way to prevent attacks during the simulation.

IV. PROCESS FOR DENIAL OF SERVICE

Many articles [16], [9] describe DDoS attacks as fake traffic and ways of filtering them out. We suggest focusing on DDoS attacks that appear to be real and can disrupt the service.

If any information resource uses a script that is interpreted on the server side, its module is recorded in the web server's settings and the reference methods are written accordingly. For example, to install PHP in Apache, information about its extensions is saved in the config file (Figure 2).



```
# mod_php options
<IfModule  mod_php7.c>
    #
    # Cause the PHP interpreter to handle files with a .php extension.
    #
    <FilesMatch \.(php|phar)$>
        SetHandler application/x-httpd-php
    </FilesMatch>

    #
    # Uncomment the following lines to allow PHP to pretty print .phps
    # files as PHP source code:
    #
    #<FilesMatch \.phps$>
    #    SetHandler application/x-httpd-php-source
    #</FilesMatch>

    #
    # Apache specific PHP configuration options
    # those can be override in each configured vhost
    #
    php_value session.save_handler "files"
    php_value session.save_path    "/var/lib/php/session"
                                                        47,1          86%
```

Fig. 2: Setting the .php extension in the Apache configuration file

It should be noted that when the script refers to .php files, the PHP module is activated, and after the script is interpreted, the executed result is sent to the final resource. Next, describe the procedure of requests being executed as for a real user when sending an attack to deny the service to the PHP module:

- First, there should be a link in the information resource that has some vulnerabilities or when implemented will lead to higher resource use. For example (Figure 3) - www.example.com/contact.php

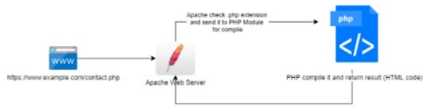

Apache check .php extension and send it to PHP Module for compile
php
www
https://www.example.com/contact.php
Apache Web Server
PHP compile it and return result (HTML code)

Fig 3

- The specified URL sends various parameter queries from the cache to disable the service, which presents itself as a new type of query. For example - contact.php?url=134, contact.php?name=Name, contact.php?surname=Surname, etc.
- Because there are small intervals between requests sent during the execution of the request, the firewall redirects it directly to the information resource without filtering. Apache executes the script on the server because it is a new query. If the execution time of the specified script is extended, the processor allocates the necessary load in memory for it. After several such difficult requests are sent to the information resource, the memory begins to fill up and the execution time increases proportionally (Figure 4).

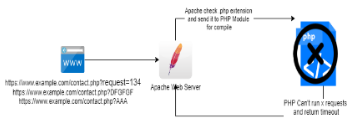

Apache check .php extension and send it to PHP Module for compile
WWW
https://www.example.com/contact.php?request=134
https://www.example.com/contact.php?DFGFGF
https://www.example.com/contact.php?AAA
Apache Web Server
php
PHP Can't run x requests and return timeout

Fig 4

- Thus, the server cannot accept new requests as a result of the download. A server with full memory can return to normal operation either after the requesting IP address is blocked or the web server is restarted.

V. Implementing a DoS Attack

Next, focus on the PHP module, specifically the first process. The Hulk DoS [15] tool was used to test the attack. Reference [17] provides information on data collection and email methods for four models with Intrusion detection systems (IDS), uses various tools for DoS attacks, and presents the results obtained. Although the article manages to simulate a DoS attack that processes more than three million queries using a variety of tools, it does not specify the time duration of this experiment. We performed tests using two servers to simulate more than three million queries using a single tool over a 24 h period:

Attacker server properties:

- 2 x Intel(R) Xeon(R) CPU E5-2650v2@ 2.60GHz
- 2 x 8 cores (total 16 cores)
- 96 GB DDR3 RAM
- CentOS Linux 7 Operating System

Victim server properties:

- Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz
- 12 cores
- 256 GB DDR4 RAM
- CentOS Linux 7 Operating System
- Web Server Nginx, Apache

ISSN 2636-9680
eISSN 2682-9266

A. Attacking Continuously for 24 h

In the first stage, we sent a request to one static address in the information resource. PHP execution time for requests at this address was milliseconds. For this request, the command on the server where the attack was organized was set as follows:

```
python /root/hulk/hulk.py
https://[ test-domain].com/test-
simple.php
```

In the test-simple.php file, we wrote a simple code that could be executed quickly as follows:

```
<?php
    echo "Hello world";
?>
```

Then, 3,056,860 such requests were sent to the target test server within 24 h. If we denote the number of queries sent in a time period as A, total queries as T, and number of seconds during the test period as S, the following formula will find A:

$$A = T / S \qquad (1)$$

Thus, A = 3,056,860/86,400 = 35.38032407 queries / second. This means an average of 35–36 queries were sent per second. It should be noted that the traffic was volatile because such attacks were carried out on real information resources (our internal projects). Thus, in addition to the 35–36 requests we send per second, up to 105 requests per second from real users were registered. Thus, despite the fact that the maximum number of parallel connections on the server at any given time could be up to 140, Figure 5 shows that the power expended by the server to perform these requests showed no significant change in 24 h.

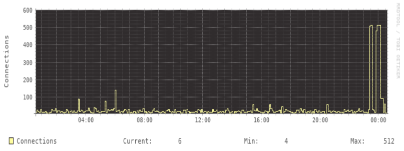


Fig. 5. Server resource usage schedule (HTTPD usage) during a 24-hour attack.

B. 20 min attack

In the next stage, we tested the results of a simulation attack on a PHP script that was running poorly on the same server using the same Hulk DoS tool.

The script sent to the query is as follows: (test-speed.php)

```
<?php
    echo "Step #01<br/>";
    sleep(1);
    echo "Step #02<br/>";
    sleep(10);
    echo "Step #03<br/>";
    sleep(100);
    echo "Final<br/>";
?>
```

Upon examining the code, we noted that the execution of this script took at least 111 s. The command on the server where the attack was organized to send requests to this script was set as follows:

python /root/hulk/hulk.py https://[test-domain].com/test-speed.php

As a result, we recorded 3022 requests sent in 20 min. If we perform the calculation as given by (1), we will get the result $A = 3022/1200 = 2.518333333$ queries/s. This means an average of 2–3 requests per second. Figure 6 shows the power usage of the server executing these requests. By sending a request to a script that works poorly for 20 min, the information resource does not stop completely; however, those requests are queued up to 500 requests.

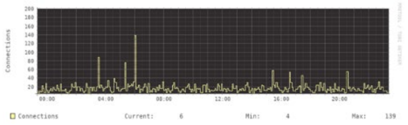


Fig. 6. Scheduled resource usage of the server during a 20 min attack.

If the sites on the server are visited in parallel, it can be observed that they stop working and the user is shown an error code 500 (Figure 7) representing an internal server problem. This was caused by the creation of a bottleneck effect on the server when a request was sent to a poorly executed script.

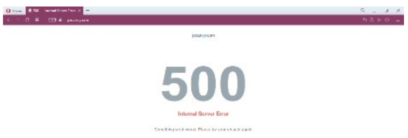


Fig. 7. Information resource frozen as a result of a 20 min attack

VI. Detection of Attacks

Because the queries simulated by a hacker during such attacks are similar to a normal user's request, it is difficult to detect it with a firewall or IDS. During such an attack, the server freezes and no operation is possible. Experts often try to solve this problem by increasing the server resources. However, observations show that a firewall, Web Application Firewall (WAF), or even a Cloudflare service installed for security sometimes fail in the face of anomalous queries that resemble real queries. With the analysis of log files, the target site received requests from aol.com, google.com, usatoday.com, etc. (referral) addresses and queries such as "?BXYE= EMNDZFUSG," "?KEDVLOX=CZKJQ." Further analysis of the log file also shows that all these requests were focused on the failure of the site's services, despite the fact that they were received from a single IP address. One of the purposes of sending these enquiries in this format (using google.com, usatoday.com, etc. as reliable, referral sites) is to allow the server to accept these enquiries sent to the cache as various enquiries by the server. Another purpose is for the firewall to accept these enquiries as a valid request. Such attacks are carried out as DoS over a single IP. However, manual blocking is not a viable solution since a new IP address can be used for attacks instead of the blocked IP address.

ISSN 2636-9680
eISSN 2682-9266

A. Requests in Log File

In such attacks, the requests in the log are usually as follows:
xx.xx.xx.xx - - [date time] "GET /test-speed.php?SIUAOJCZ=EHVOQUBC HTTP/1.0" 200 50 "http://www.usatoday.com/search/results?q=JBBFAJVR" "Opera/9.80 (Windows NT 5.2; U; ru) Presto/2.5.22 Version/10.51"

If the log is focused on the incoming request is sent to the HTTP / 1.0 and the parameter GET in all enquiries is used. All enquiries shown were sent to the script (query string) as follows.

```
/test-simple.php?DXYZTHXEH=ZEGKFOB
/test-simple.php?ABLCEUOX=CPMWNH
/test-simple.php?MAOKSZUH=XUPZBLON
```

This bypassed the cache and resulted in the processing of a new request. The purpose of sending the requests in this format was to bypass the caching function that prevents loading on the server. Thus, if the caching function was activated in the web server or content delivery network (CDN) settings, the first processed response on any URL would be kept in the cache to prevent re-processing on the server during re-enquiries. As a result, the next enquiries could be answered with a ready-made static page during the period specified for this URL's cache. It should be noted that the code is the article "Hello world" in the screen as a result of the processing process. If the "test-simple.php" request is sent to code, the processing of the request will take place and the result will be recorded on the server in accordance with the cache period. The cache period can be adjusted in accordance with the requirements for caching. For the next request, the server will return the cached version of test-simple.php. To make such attacks effective, all enquiries sent by the hacker needs to bypass the caching function for processing and resource allocation. To do this, the server can be reached as a new (first) request, by adding various pieces to the survey format. Thus, the hacker not only affects the server's processing power but also, possibly, a server unit separated for cache.

B. Processor Indicators

For Linux OS, using the "Top" Command, we noticed an increase in processes allocated to the web server. Although, we noticed no serious change in network traffic.

VII. Prevention of Attacks

In this section, we present several methods to prevent the problem:

Method 1 - Preventing a query to reach the web server: One of the ways to prevent such attacks is to simply apply filters in the .htaccess file to process enquiries from the web server. Thus, as all URLs on the attacked host use a "Rewrite Rule," changing the format of incoming queries can be prevented by filtering them at the .htaccess file level.

Method 2 – Another solution is to apply different limits to the enquiries included in the "iptables" for attacks on Linux.

Method 3 - Filtering requests using HTTP 1.0 when real enquiries are sent with HTTP 1.1 or HTTP 2 is another possible solution.

Method 4 - Considering the fact that the application code of the attack is used on the execution code, one of the most effective approaches for prevention would be to rewrite and optimize the code for the execution period and resource use. To do this, you can use methods for Static application security testing (SAST) such as sast.online [18].

Method 5 - CDN networks and caching systems also play a major role in solving this problem. Cloudflare [19] is an example of one of these resources. Thus, using the Cloudflare service, you can always use a number of preventive measures to regulate your caching periods in the properties of your site, setting it as an Always Online module.

The Special Communication and Information Security Service [20] is very useful in terms of preventing attacks or reducing the effects of attacks by optimizing in the establishment of state resources:

- Development and compilation of the Internet Information Reserve Code, measurement of Time to First BYTE (TTFB), which should not exceed 0.5 seconds;
- Sites should be focused on the loading time and should not exceed the limit of 3 seconds
- Preparation of internal static error pages (403, 404, 429, 500, 502, 504, etc.) in the information reserve;
- The implementation of the AJAX method for the establishment of information in the information resource to be downloaded on the same page or using the principle of "lazy load;"
- Execution of the "server-side" principle by the AJAX method if the information reserves are provided;
- Placement of JavaScript and CSS files used in the information resource, as well as minimizing HTML content (minify);

It should be noted that the information reserves can be inspected through a number of services such as Accessify [21].

ISSN 2636-9680
eISSN 2682-9266

VIII. CONCLUSION

In 2011, we came across this type of DDoS attack indicating that it has been used for a long time. However, we did not find any research on such an attack, and in 2020, we experienced the same attack during the implementation of several projects. Therefore, we had information to conduct detailed research on this attack. It should also be noted that during the simulation of the attack, the recurrent neural network-based intrusion detection system [17] was used with more powerful Xeon processor servers instead of the recommended i5 processor servers.

Our investigation showed that DoS or DDoS attacks could be effective even when it was not used to occupy the network channel. This, in turn, is also not optimized programs, which are equipped with high-performance servers, code and system security, as well as special security equipment in the information systems but also the optimized programs and more resources Script or Commands can cause the total system to be down and put the service in reachable. By finding a poorly written script and ensuring that the attack focuses on that script is all that is necessary to fully weaken the information reserves and affect work activities. In future, attacks will not only generate traffic but also target weakly executed code. Increased traffic leads us to think about the defence against that type of attack. However, inevitably writing better code and conducting code inspections will become a more suitable attack prevention method. This was also one of the five proposed methods to solve the problem, along with the use of cache and CDN networks to avoid such problems.

Future research will focus on the establishment of an online tool to simulate DoS attacks. Using this resource will enable users to determine whether the weaknesses presented in the article are present in their resources. Using the results generated by this tool, we plan to publish the frequency and percentage of detection and interest in various resources in future work.

IX. REFERENCES

[1] R. K. Sanodiya, "DoS attacks: A simulation study," in *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, Chennai, India, 2017.

[2] "Top 10 Web Application Security Risks," [Online]. Available: https://owasp.org/www-project-top-ten/. [Accessed 08 April 2021].

[3] "Network Security Infrastructure Report," [Online]. Available: https://www.netscout.com/report/. [Accessed 07 April 2021].

[4] M. A. S. a. A. A. Manaf, "Optimal specifications for a protective framework against HTTP-based DoS and DDoS attacks,," in *International Symposium on*

Biometrics and Security Technologies (ISBAST), Kuala Lumpur, Malaysia, 2014.

[5] T. S. a. G. A. T. Karnwal, "A comber approach to protect cloud computing against XML DDoS and HTTP DDoS attack," in *IEEE Students' Conference on Electrical, Electronics and Computer Science*, Bhopal, India, 2012.

[6] T. H. B. a. F. T. Johnsen, "Exploring SOAP and REST communication on the Android platform," in MILCOM 2015 - 2015 IEEE Military Communications Conference, Tampa, FL, USA, 2015.

[7] S. M. a. D. Kim, "A comparison of RESTful vs. SOAP web services in actuator networks," in *Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, Milan, Italy, 2017.

[8] A. W. M. a. A. M. Zeki, "Web services SOAP optimization techniques," in *4th IEEE International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, Salmabad, Bahrain, 2017.

[9] S. K. a. K. M. Carley, "Simulating DDoS attacks on the US fiber-optics internet infrastructure," in *Proceedings of the 2017 Winter Simulation Conference (WSC '17)*, Las Vegas, Nevada, 2017.

[10] N. S. K. B. a. A. S. Y. Bekeneva, "Simulation of DDoS-attacks and protection mechanisms against them," in *IEEE NW Russia Young Researchers in Electrical and Electronic Engineering Conference (EIConRusNW)*, St. Petersburg, Russia, 2015.

[11] M. P. a. M. Sathyakala, "Simulation and analysis of DDoS attacks," in *International Conference on Emerging Trends in Science, Engineering and Technology (INCOSET)*, Tiruchirappalli, India, 2012 .

[12] T. A. T. K. a. N. B. S. Daneshgadeh, "Detection of DDoS Attacks and Flash Events Using Shannon Entropy, KOAD and Mahalanobis Distance," in 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), Paris, France, 2019.

[13] "PHPMailer," [Online]. Available: https://github.com/PHPMailer/PHPMailer. [Accessed 07 April 2021].

[14] "PHP mail() function," [Online]. Available: https://www.php.net/manual/en/function.mail.php. [Accessed 08 April 2021].

[15] "Hulk DoS tool," [Online]. Available: https://github.com/grafov/hulk. [Accessed 07 April 2021].

[16] R. K. Sanodiya, "DoS attacks: A simulation study," in *International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, Chennai, India, 2017.

[17] S. A. a. M. S. S. Nayyar, "Recurrent Neural Network Based Intrusion Detection System," in *International Conference on Communication and Signal Processing (ICCSP)*, Chennai, India, 2020.

[18] "Online Static application security testing (SAST) tool," [Online]. Available: https://sast.online. [Accessed 08 April 2021].

[19] "Cloudflare," [Online]. Available: https://www.cloudflare.com. [Accessed 07 April 2021].

[20] "cert.gov.az," [Online]. Available: https://cert.gov.az. [Accessed 07 April 2021].

ISSN 2636-9680
eISSN 2682-9266

[21] "Accessify," [Online]. Available: https://www.accessify.com. [Accessed 07 April 2021].