DOI: https://doi.org/10.31341/jios.44.2.6
Open Access
UDC 004.056:004.7
Original Scientific Paper

Controlled DDoS Attack on IPv4/IPv6 Network Using Distributed Computing Infrastructure

Michal Čerňanský *michal.cernansky@ucm.sk*
Department of Applied Informatics
University of SS. Cyril and Methodius, Trnava, Slovakia

Ladislav Huraj *ladislav.huraj@ucm.sk*
Department of Applied Informatics
University of SS. Cyril and Methodius, Trnava, Slovakia

Marek Šimon *marek.simon@ucm.sk*
Department of Applied Informatics
University of SS. Cyril and Methodius, Trnava, Slovakia

Abstract

The paper focuses on design, background and experimental results of real environment of DDoS attacks. The experimental testbed is based on employment of a tool for IT automation to perform DDoS attacks under monitoring. DDoS attacks are still serious threat in both IPv4 and IPv6 networks and creation of simple tool to test the network for DDoS attack and to allow evaluation of vulnerabilities and DDoS countermeasures of the networks is necessary. In proposed testbed, Ansible orchestration tool is employed to perform and coordinate DDoS attacks. Ansible is a powerful tool and simplifies the implementation of the test environment. Moreover, no special hardware is required for the attacks execution, the testbed uses existing infrastructure in an organization. The case study of implementation of this environment shows straightforwardness to create a testbed comparable with a botnet with ten thousand bots. Furthermore, the experimental results demonstrate the potential of the proposed environment and present the impact of the attacks on particular target servers in IPv4 and IPv6 networks.

Keywords: DDoS attacks, network security, real-time environment, IPv6

1. Introduction

Enterprises will be soon deploying Internet Protocol version 6 (IPv6) in all their networks. Deployment of IPv6 brings with it old and new threats. Therefore, network administrators need to be able to detect risks that threaten their network. In addition, comparing the state of the network and its defense mechanisms for IPv4 and IPv6 can help discover hidden vulnerabilities and propose an appropriate defense strategy [1].

One possible attack on a network or on a server on the network is distributed denial of service (DDoS) attack. DDoS attacks use a group of distributed computers

to lead the attack from different locations with different IP addresses and the attacker is able to coordinate the attack on the victim. The task of multiple misused/compromised sources (bots) distributed across the Internet is to flood victim in a way that victim server cannot handle it. The bots are used by attacker to send enormous traffic to the victim network or server to overwhelm them and so make them useless for requests from legitimate users. The botnet is created by a group of distributed misused/compromised sources and an average botnet size of attacking bots is of the order of tens of thousands of computers [2].

The article focuses on establishing and testing DDoS attacks using a testbed based on Ansible orchestration tool. Ansible is open-source tool for IT automation offering both configuration management and also high-end orchestration. Ansible is a simple, model-driven configuration management system to deploy on multiple nodes and to remotely run tasks. Any language can be used to write extension modules which are consequently transferred to remote computers automatically [3].

The study illustrates several types of DDoS attacks tested by the testbed over both IPv4 and IPv6 networks. Three basic types of attacks were considered: SYN flood attack as one of the oldest DDoS attacks; HTTP Get flood attack as attack using standard URL requests hence it is quite challenging to differentiate from valid traffic; and SMTP mail flood attack as attack on a specific type of mail server.

Even though the DDoS attacks themselves are well known and even though the software for launching DDoS attack is existing, the simulation in the laboratory on a small scale of a complex and large system as DDoS attack is fraught with difficulty [4]. A tool to assistance network administrators to test and understand the state of their real networks is needed in order to allow secure, available and legitimate communications among machines on networks.

Several aspects have been taken into account in the design, implementation, and testing of the proposed DDoS attack system. First of all, the possibility to reach the number of attacking computers of the order of thousands of computers so that the attacks are comparable in size to real botnets. Another aspect was the ability of real testing under monitoring, i.e. the controlled attack runs in real-time on a particular network, and various scenarios are tested. Besides that, from the proposed system was required to be easy to handle which was achieved by engaging the Ansible environment. Even in heterogeneous environments, Ansible automatically configures, coordinates, and manages distributed computing systems.

The Ansible client-server topology is similar to the basic structure of botnet used in DDoS attacks. Our previous ad hoc grid experiments show that ad hoc grid software can also be used to create a P2P test botnet for real networks [5], [6]. However, the management of such an approach is complicated and time-consuming. Moreover, ad hoc software must be installed on all nodes which is a problem with some devices, such as smartphones [7]. In this respect, the use of the Ansible orchestration tool is a powerful aid and simplifies the implementation of the test environment.

The paper illustrates the ability of the proposed environment to serve as a testing tool for DDoS attacks in testing of network security both for IPv4 networks and for IPv6 networks. In addition, the test environment allows network administrators as well as researchers to compare defense mechanisms for both protocols, as well as to

investigate the state of network security and proposed countermeasures against DDoS attacks under real-world conditions. The security of two real production servers (web server, mail server) was tested with proposed testbed where different metrics including the performance impact on the machines and the client throughput during testing DDoS attacks were measured.

The remainder of this paper is structured as follows. Sections 2 describes related work, Section 3 presents principles of basic types of DDoS attacks. In Section 4, design of the DDoS testing environment is illustrated where its general features are shown. Section 5 evaluates proposed testbed and shows experimental results of DDoS attacks for both IPv4 and IPv6 networks and Section 6 Section 6 discusses the impact of the results. Finally, Section 7 deals with future goals and limitations in conclusions followed by necessary references.

2. Related Work

A considerable amount of testbeds for DDoS attacks testing has already been developed. They may use virtual or physical components for testing and in terms of functionality, these can be specialized test environments or general-purpose environments. There are four plain approaches to DDoS testbeds [8], [9], [10]:

1. *Mathematical models approach*: description of attacks using mathematical concepts, language, symbolic modeling and mathematical validation.
2. *Simulation approach*: In simulation approach, a software-simulated attack and the behavior of the individual participants of the attack is simulated mostly on one computer. Various network simulators such as Omnet++, ns-2, ns-3, Opnet, GloMosim or Qualnet3 can be used. Compared to the real implementation of the problem, the simulation process is cheaper, easier to implement and often safer; therefore, the simulation allows to analyze the behavior of experiment participants at an overall lower cost. On the other hand, since simulation simplifies the simulated platform, it is not possible to set and to model all the parameters during the simulation as it is in reality due to the complexity and diversity of real-world networks. In addition, the simplified simulation model does not have unexpected shortcomings and weaknesses [4].
3. *Emulation approach*: The main advantage of emulation approach used for security attacks is higher fidelity of experiments. Some components as traffic or links are simulated and some components are applied on real hardware, e.g., real devices with limited resources, real operating systems and applications are used. Emulation, unlike simulation, can reveal unpredictable errors in implementation, protocol interaction, and resource constraints in more detail.

   Also, the experiment time in simulation and emulation is different. The time in the simulation is not real, it runs in virtual simulated time; however, the emulation flows in real-time. Examples of emulation systems are environments such as DETER, ns-3, WIL (WAN-in-Lab), Emulab [4], [8].

4. *Real systems approach*: Overly simplified models as well as unrealistic assumptions in simulation and emulation can lead to inaccurate results. Although the use of real testing systems is more expensive than simulation and emulation, by running the experiments on real platforms and applications in real terms, the outcomes of experiments have higher fidelity [8]. Most known testbeds for computer networking and distributed systems research are PlanetLab [11] or GENI [12]. Nowadays, even real systems are implemented on virtual machines.
   Testing DDoS attacks during simulation and emulation is often distorted and real systems approach can lead to more realistic results for specific networks [4], [9]. In addition, the ability to test DDoS attacks for a particular network is missing; most of the existing testbeds are of academic interest only and only relate to testing of DDoS attacks with high detection rate, low false alarm rate, or with secure and isolated environment [13].

In the proposed testbed, the aim is to perform DDoS attacks as controlled attacks on real-world networks under continual control and monitoring from a testing person rather than on only an isolated environment, and so to give detailed feedback about the network status and its security.

3. Attacks

The goal of DDoS attack is to make the victims' services unavailable until the attack is either effectively mitigated or terminated. Denial of Service attacks make the server unavailable or unable to process legitimate requests. Duration of these attacks can range from a few seconds to several days and the financial loss of service providers can be significant. The motivation for such attacks can vary; for example, it can be done from competitive, financial, or political reasons [14]. Although there are various defense techniques based on artificial intelligence, high performance computing, or intrusion prevention system to defend against DDoS attacks, DDoS attack is still serious threat and there are still mass DDoS attacks that computer systems and networks must withstand [15], [16].

3.1. SYN Flood Attack

A SYN flood attack is a form of DDoS attack focused on TCP connection queue capacity where an attacker sends a succession of SYN requests to a victim's system in an attempt to overflow the queue, thus denying legitimate requests. Usually, the attacker sends SYN requests with spoofed IP addresses to manipulate the 3-way handshake in a TCP connection. Normally the SYN requests remain in the queue until the connection is completed or the request times are out. While SYN flood attack works by not responding to the server with the expected ACK code to complete the SYN request, the TCP connection queue capacity is exhausted and other users cannot establish new connections, Figure 1. Even server with a large range of the memory space allocated for the TCP connection queue can be overloaded in a mass DDoS

attack, the attack fills the queue and the SYN flood attack disrupt a server providing services [17], [18].

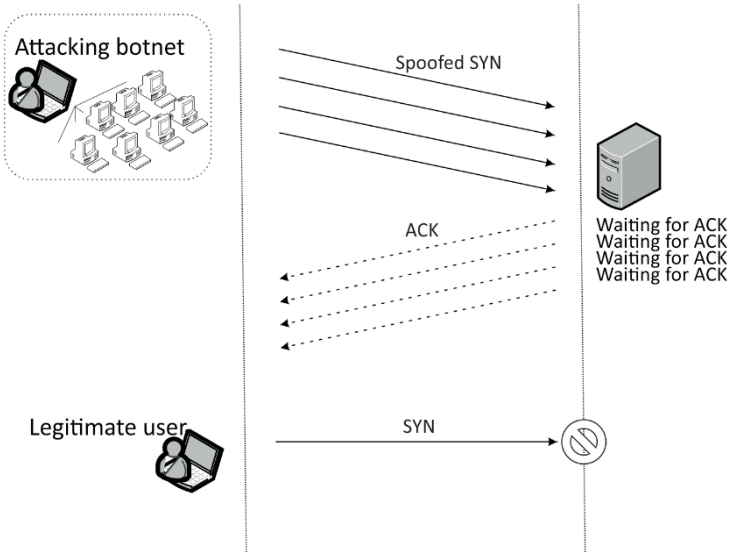


Figure 1. SYN flood DDoS attack

Currently, there are a number of defense mechanisms against the SYN flood attack. One is, for example, the mechanism *Syn cookies* [19] to improve tolerance capacity of the server during attack. Since the handshaking process is turned into a stateless process by the SYN cookies defense mechanism using an algorithm for hashing, encoding and packet processing, it can significantly enhance the capacity of the server to handle SYN packets [20].

The idea of SYN cookies is that when the TCP connection queue is full, incoming requests are not stored until their valid ACKs are received. Once a SYN request arrives, the server replies with a SYN-ACK including a SYN cookie as its Initial Sequence Number (ISN). A cryptographic hash of some header fields from the original SYN request is calculated as a SYN cookie. Subsequently, the server is able to verify the cookie through the client's next ACK with that ISN on it, and the server can create a full connection using the encoded information [17], [20], [21].

The SYN cookies are now a standard feature of FreeBSD and Linux operating systems.

3.2. HTTP Get Flood Attack

HTTP Get flood attack can overwhelm victim resources as the attacker sends plenty of HTTP GET requests to a server to generate a large amount of responses. The server replies to each of the clients' requests and waits for acknowledgement of each of it. A socket queue, which is responsible for holding all the HTTP GET requests until dedicated threads, is assigned to serve the request, Figure 2. However, the attacker

never replies the request and the queue becomes full, consequently dropping the incoming requests sent by legitimate users.

An attack-related HTTP GET request is indistinguishable from legitimate request so the web server does not perform filtering to determine whether it is fake or genuine. The server assumes the request is from legitimate source even during the HTTP GET flood attack and the server will continuously receive and process the request. Therefore, it is difficult to prevent this attack [22], [23], [24].

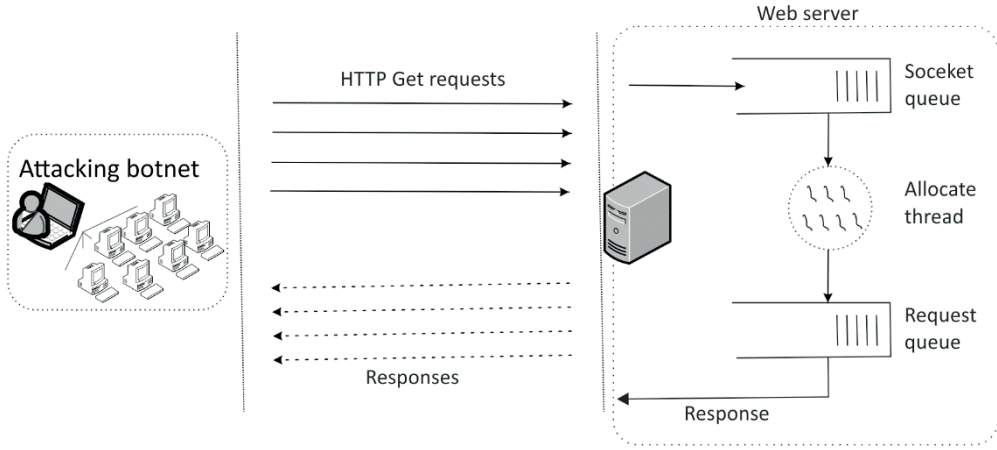


Figure 2. HTTP Get flood DDoS attack [14]

3.3. SMTP Mail Flood Attack

Mail flood attack, also called e-mail bombing, floods an inbox and mail server with enormous amount of messages in a short period of time, Figure 3. It can be SMTP (Simple Mail Transfer Protocol) also POP (Post Office Protocol) relaying. During the attacks, victim server is overwhelmed by bogus emails that the legitimate e-mails are influenced by a non-tolerable delay, even the internet connectivity may be lost [25].

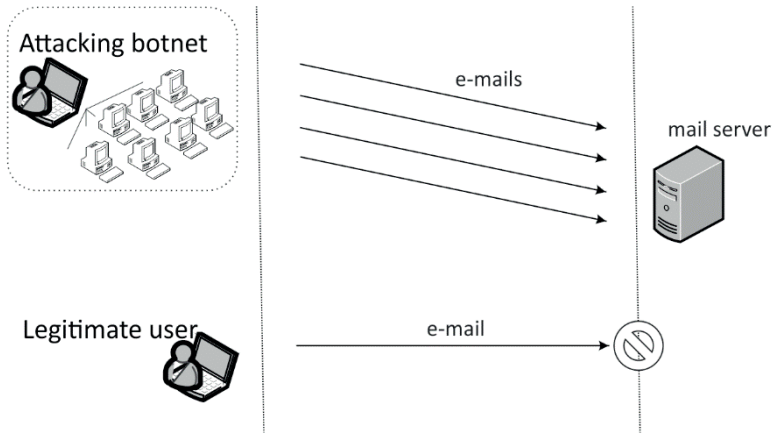


Figure 3. SMTP mail flood DDoS attack

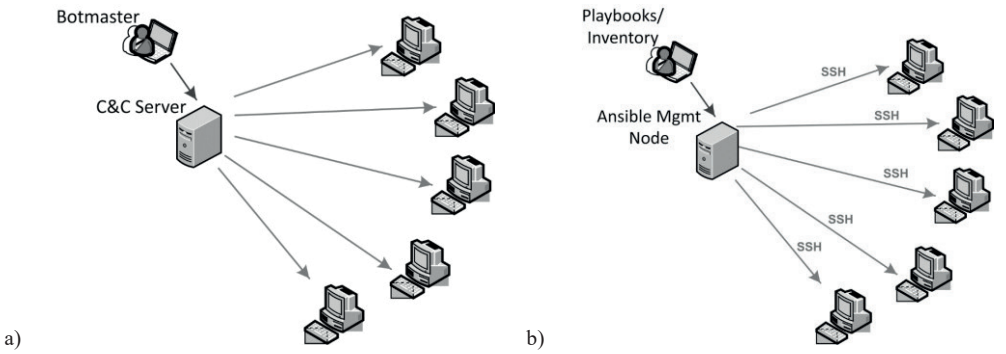
SMTP mail flood attack is a common type of application layers attack and belongs to the slow or low-rate DDoS attacks. Traffic generated by the slow DDoS attacks behaves as legitimate traffic; therefore, the slow DDoS attacks are very hard to detect. Although a slow DDoS attack consumes much less bandwidth and resources, it may cause a large destruction. No particular pattern of message or any proper format of mail need to be sent in SMTP mail flood attack, which makes it more difficult for countermeasures to deployed [25], [26].

4. Design of DDoS Testbed

Basic botnet type is centralized C&C botnet where the botnet owner can simply control the botnet using command and control (C&C) software. The structure of centralized C&C botnet consists of one centralized source, from which the entire attack and fraudulent activities are coordinated and which directly manages the compromised machines (bots) by commands via C&C (command and control) channel, Figure 4a). In this kind of botnet, the central C&C source sends commands directly to each bot [27]. The structure of the proposed testbed was inspired by the structure of the centralized C&C botnet.

Four different categories of botnet architecture are described in [28] and the centralized C&C botnet belongs to the first one. The botnets are categorized depending on how botnets control network and how they hide their detection. Four categories are distinguished as follows: the centralized C&C botnet, P2P botnet, hybrid botnet, and botnet combining the http protocol with P2P. Although botnets from all categories are used for a DDoS attack, the structure of Ansible tool similar to the first category determines the design of proposed DDoS testbed and the orchestration feature from Ansible was applied to organize the DDoS attack.

Ansible as a provisioning tool is employed for IT automation. Since the Ansible is an agentless tool, the remote host is only required to support Python and Secure Shell (SSH) connections. Moreover, Ansible supports idempotency, i.e., the result of performing an operation once is exactly the same as the result of performing it repeatedly without any intervening actions. It means for the Ansible that a playbook



a) b)

Figure 4. Similarity of architecture of a) Centralized C&C botnet and b) Ansible

can be run many times and the state of the machine would not be changed. Ansible is still under active improvement and offers many modules that can download remote content, install packages, copy files or change system configuration and provide many other features. Anyone can use the Ansible as well as develop further modules.

A central point called Controller Machine is used by Ansible to remotely manage a fleet of machines and the Ansible software is installed on this machine, Figure 4b). Since Ansible utilizes an agentless structural engineering without installing an agent managed machines, the remote machines are connected directly over the SSH protocol. Ansible retrieves the information about managed machines from the inventory file and then executes the code over SSH from playbook file consisting of tasks represented by a YAML file. No additional software should be installed or run on the remote machines but instead of it Ansible pushes, executes and removes modules on the machines using only Python and a few other packages [29], [30].

Figure 4 illustrates comparison of structure of Ansible orchestration tool with centralized C&C botnet. As revealed in the diagram above, the design of Ansible tool is comparable to the centralized C&C botnet architecture. The centralized C&C botnet contains a group of bots and C&C server and all this is controlled by botmaster, Figure 4a). A group of bots receives and responds to commands from the C&C server that acts as a rendezvous mechanism for orders from the botmaster. Likewise, the Ansible tool includes a group of remoted machines, Ansible management node and playbooks/inventory, Figure 4b). As an analogy to a botmaster, a simple human-readable format of playbook is available in Ansible environment. The playbook is used by an administrator to manage configuration which can be installed on the remote machines and the inventory covers all the machines that are managed in the group. An Ansible management node is a machine which is responsible for configuration of all remoted machines and where the Ansible software is installed [3].

Commands from botmaster are delivered to the compromised machines through C&C channel through which botmaster can control the remote machine in the centralized C&C botnet. In this way the botmaster can guide and control the whole botnet. Since C&C infrastructure interconnects the botnet components and offers data transferring among them, it is critical to have this relation stable in order to sustain the work of botnet effectively [31]. Compared with the Ansible ecosystem, a connection of the Ansible management node to each host defined in the inventory is done through SSH connection and an execution of each task defined in human-readable Ansible playbook is remotely performed [32].

However, the purpose and behavior of both environments is different. While the main aim of the Ansible tool is to assist to automate the job processing, the main aim of the botnet is attacking the victim. In addition, confidentiality protection is the reason for using encrypted SSH communication in Ansible; while in the botnet the reason for the encrypted C&C communication is to make it difficult to perform payload analysis and try to hide any clear behavior of the botnet at the network [33].

Although the process of the testbed establishment is relatively straightforward and is easy to achieve in organizations, the preparatory phase for its creation requires some effort and takes time. The establishment process of the testbed itself and whole test environment in the case study contained the following eight steps:

i. KVM hypervisor preparing for the powerful servers;
ii. creating virtual machine with Ansible environment (Python and a few other packages, CentOS7),
iii. creating virtual machines (in this case 32 VMs with CentOS7, 36 VMs with Debian10, Debian10 installation on a physical server and its cloning by Clonezilla SE and PXE on 28 additional servers),
iv. creating network infrastructure for the experiments,
v. migration of a copy of the production web server (CentOS7 LAMP) to the hypervisor into test environment; anonymization of sensitive data on the production web server,
vi. migration of a copy of the production mail server (Debian10, postfix, dovecot) to the hypervisor into test environment; anonymization of sensitive data on the production mail server,
vii. creating scripts (shell, python, Perl) generating and measuring the attacks,
viii. creating Ansible playbooks to launch attacks and to measure them. The creation process resulted in establishing of testbed infrastructure illustrated on Figure 5 that provides almost ten thousand different IP addresses usable for carrying out the attacks.

It should be noted that Ansible tool is not directly designed for DDoS attack and it was necessary to address some minor problems with this environment to achieve the desired goal during Ansible preparation. For example, one of the Ansible strategy as a way to control play execution is strategy free which allows each node to run until the end of the play simultaneously as fast as it can without waiting for all nodes. However, it is important to watch out the race condition problem, i.e., not to allow a new task to start until the previous block of tasks is completed.

Moreover, it should be underlined that the Ansible-based testbed achieves all the requirements for a DDoS testbed environment described in [34]: (i) an experiment completed in the testbed should be specified, saved and replayed; (ii) an experiment should be deployed, run and stopped; (iii) the DDoS attack should be monitored during the run as well as the logs should be archived for later repetition or examination.

5. Experimental Setup and Results

The case study of the testbed environment involved 97 physical or virtual servers, Figure 5. The number of physical servers was 29 running OS Debian10, Intel(R) Core(TM) 2 Quad CPU Q8400 @ 2.66GHz, 2 GB RAM. Virtual servers based on Kernel-based Virtual Machine (KVM) module that allows the kernel to function as a hypervisor were used, explicitly 32 servers running CentOS7, one core Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz, 1GB RAM and 36 servers running Debian10, Intel(R) Xeon(R) CPU E3-1230 v3 @ 3.30GHz, 1 GB RAM. Each server can emulate 100 virtual nodes to increase the number of unique IPs in the network and so the final number of all unique IPs reached the value of 9 700. The bandwidth of all links in the network topology was 1 Gbps.

The copies of two real production servers were used for test purposes, a web server for the first two scenarios and a mail server for the last attack. It means that there are no default settings set but the operation is adapted to real run, e.g., mail server uses real spam filters and antivirus software and controls each incoming email or defense mechanisms are installed. Moreover, effects of the attacks were discussed with servers' administrators and have been proven as useful to create more appropriate countermeasures against server attacks.

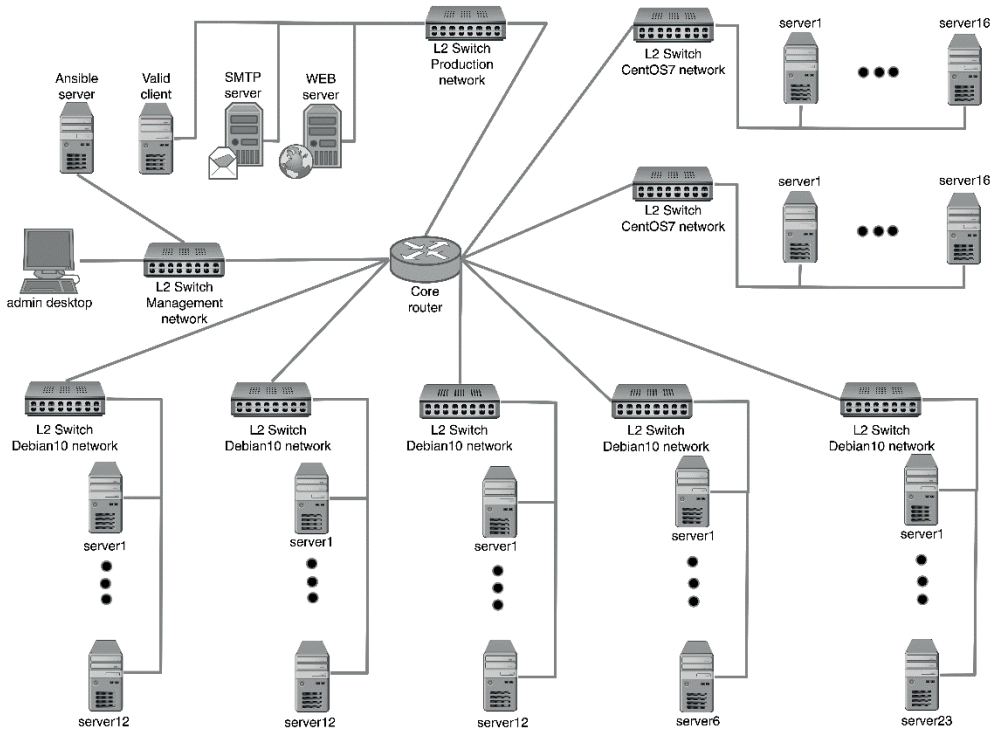


Figure 5. Structure of the testbed based on Ansible environment

It should be noted that it is particularly easy to increase the number of connected machines into the testbed environment and so to increase the number of attacking bots; it depends entirely on the number of available computers on the tester's side. However, with an increase in devices involved in the testbed, the attacking force of the attacks does not rise in a direct mathematical proportion. In the case study, there were used real production servers with defense mechanisms and the results showed that, e.g., for SYN flood attack, the web server is resistant even against attacking enormous force of 10,000 bots. Existing production servers were intentionally used to demonstrate the effectiveness of the Ansible-based attack tool for the real-world conditions used in practice.

On the other hand, although orchestration increases the ease of maneuvering, it also makes it easier for potential attacker to control the entire system from one central

node. If an administrator's Ansible account is compromised, the attacker receives direct access to all computers managed from that account. For this reason, it is necessary to require intensified protection to the Ansible management node and to prevent its compromising.

As main metrics in the attack analysis, the CPU load and memory usage of the victim server, and average serve rate for all performed experiments were taken into account. CPU load is a measure of the amount of computational work, i.e., of tasks running, of tasks waiting for CPU and of tasks blocked. Memory usage of the victim server is the amount of main memory utilized during the DDoS attack. Average serve rate is an average amount of responses generated by server as responses to requests generated by a legitimate client during the DDoS attack; during attacks, the legitimate requests are often dropped. During the DDoS attack, one legitimate client for each attack scenario was used as a test client to demonstrate the availability of server service during the attack.

The length of the performed experiments lasted mostly 60 seconds. The chosen time of attack was based on the authors' previous experience with DDoS attacks. The experiments in the article employ a medium-sized botnet, which can completely flood the victim in 60 seconds, and then the results of the experiment can be analyzed. During the attack on the target, flood flows from the attacking machines controlled by the Ansible tool and one valid client tries to connect to the target server affected by DDoS attack.

5.1. SYN Flood Attack Scenario

The first scenario was a SYN flood attack targeting the server TCP/IP stack. The attack is focused towards a listening port of the server where an enormous amount of TCP-SYN requests with spoofed IP addresses of sources is sent by the attacker. 9700 unique IP nodes sent SYN requests to the given port from the attacking botnet. Consequently, the computational resources of the victim server are exhausted because of the number of open connections and server services become unavailable. A python script using an open source packet manipulation tool Scapy was used for the attack. Scapy is able to build or decode packets, send and capture them, as well as to match requests and replies. It combines the functions of scanning, probing, tracerouting, unit testing, attacking and network discovering [35]. OS CentOS7, one core Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz, 2GB RAM was running on the victim web server. The duration of SYN flood attack was 60 seconds for both IPv4 and IPv6 networks.

As can be seen in Figure 6, The CPU load increased only slightly at the beginning of the attack and then decreased. Memory reduction was greater for IPv6 network; for IPv4 network, it was 5500 kilobytes of memory, for IPv6 25500 kilobytes of memory.

In SYN flood attack, average serve rate is represented by the round trip time (RTT) response (SYN+ACK) to client's legitimate SYN request, i.e. a legitimate client tries to send its request to the server every second during a DDoS attack and it is determined whether it has received a response from the server. As can be seen from

Figure 6, the legitimate user managed to use the service even during the SYN flood attack. The DDoS attack did not affect the availability of the service. The response time fluctuated only slightly for both IPv4 and IPv6 networks.

The SYN flood attack belongs to one of the most common types of DDoS well-known for a decade. From this reason, the attack was tested as first. Moreover, the employed production server contains countermeasures called SYN cookies against the SYN flood attack. Currently, the SYN cookies defense mechanism is a standard feature of Linux operating system. The SYN cookies mechanism closes each SYN queue entry until it receives a subsequent ACK response from the client. In this way, the SYN queue is not filled up with attacking connections and the SYN queue entry is reconstructed only when the server receives legitimate client ACK response [17]. A more detailed description of the effectiveness of the SYN cookies defense mechanism against SYN flood attacks can be found in [36].

Thanks to this mechanism, the attack had no significant impact on the server and values of all metrics correspond to normal operation conditions for both IPv4 and IPv6 networks, Figure 6. On the other hand, the experiment proved the functionality of the defense mechanism under such a heavy load not only for IPv4 network but also for an IPv6 network.

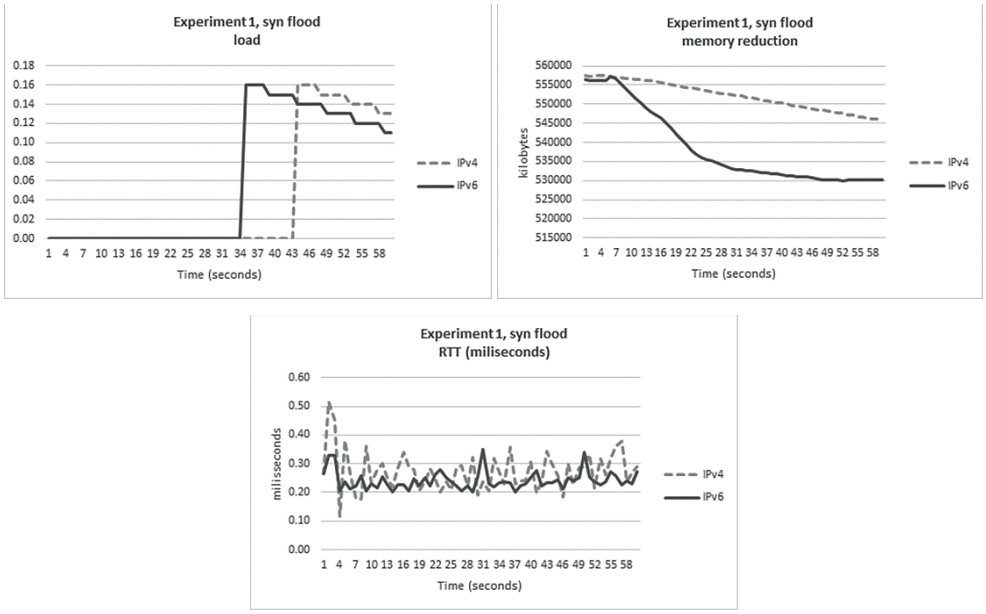


Figure 6. Metrics values for SYN flood attack

5.2. HTTP Get Flood Attack Scenario

The HTTP Get flood was led from proposed environment to target server as the second experiment again for 60 seconds. Overwhelming of the server's computing resources is done through basic GET request conducted in application layer from attacker to victim server. A load testing and benchmarking tool for HTTP server ApacheBench [37] was used for the attack in the performed experiment. The same

victim web server as in previous experiment was employed. Figure 7 displays the results of the HTTP Get flood attack for both IPv4 and IPv6 networks.

The average serve rate is represented by web page view requested from legitimate client; i.e. the legitimate client tried every second during the DDoS attack to reach the website to the web servers.

The second performed DDoS attack had an overwhelming impact on the victim server. As Figure 7 illustrates, the load of the server during 60 seconds of the HTTP Get flood attack reached the value of 16 for IPv4 network and the value of 14 for IPv6 network, the amount of the usable main memory decreased to 23 percent of the initial amount for IPv4 network and to 27 percent for IPv6 network. Additionally, serving of the client requests was unavailable after 19 seconds in IPv4 network and after 29 seconds in IPv6 network, Figure 7. It should be noted that although http services were unavailable, the server was not shut down because of the security limitation set on the Apache web server.

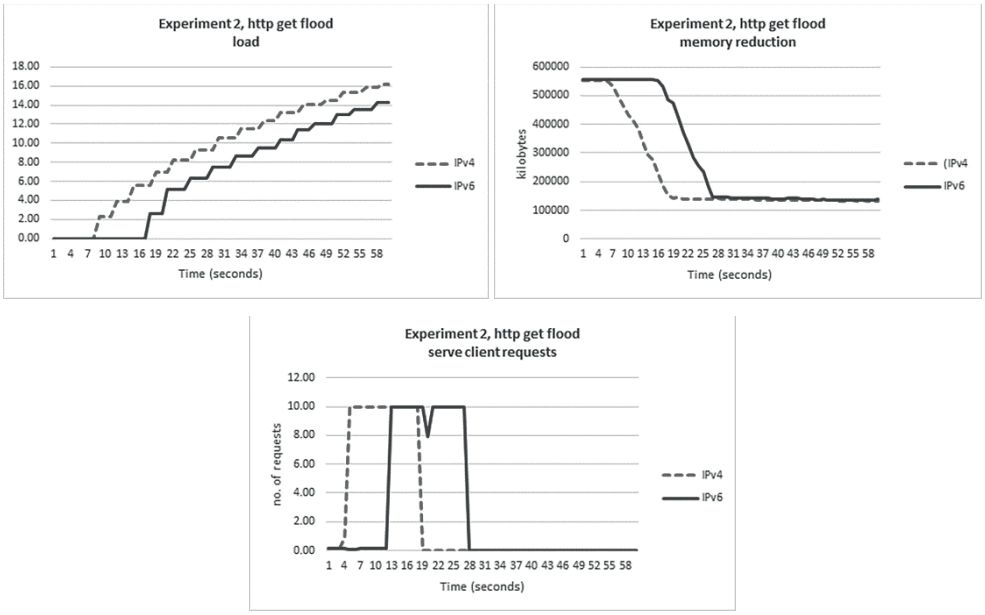


Figure 7. Metrics values for Http Get flood attack

5.3. SMTP Mail Flood Attack Scenario

The last type of performed attacks was SMTP mail flooding which tried to overwhelm the mail server, i.e., to send an enormous number of successive emails to mail server in an attempt to collapse the target mailbox or affect a non-tolerable delay of the mail server. The size of the emails sent was fixed; each email sent had the same size, 4 kilobytes. Emails were sent from the attacking machines every 3 seconds; the interval was chosen with regard to the configuration of the mail server, when it was possible to send a maximum of 20 emails from one IP address in 60 seconds. A stricter

setting than the default for similar types of mail servers (e.g., Postfix under Linux) was intentionally chosen, so that the effectiveness of the attack is obvious even with the established defense policies on the mail server side.

The SMTP mail flood attack can also result in loss of connectivity of whole network [38]. For the SMTP mail flood, the Scapy tool was used for 90 seconds; since SMTP mail is a connection-less service, the attack was designed for a slightly longer time as in previous attacks. The victim mail server configuration was the same as for web server for both IPv4 and IPv6 networks, i.e., OS CentOS7, one core Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz, 2GB RAM. Figure 8 shows the result of the SMTP mail flood attack.

The average serve rate is represented by client's email resending, i.e. a legitimate client sends a mail to the mail server every 5 seconds during a DDoS attack and the mail server tries to forward it. From the point of view of the mail server, this means that the mail server puts the received email in the mail queue and then gradually processes the mail queue, while trying to progressively deliver all the emails in the queue. From emails sent in this way from a legitimate client, only one email was successfully delivered during the test period for IPv4 network and two emails for IPv6 network, Table 1.

IPv4	IPv6
3s	2s
5h 29 m 16s	1m 5s
8h 18 m 47s	6h 5m 43s
10h 2m 53s	7h 18m 7s
11h 1m 2s	9h 20m 16s

Table 1. The time of delivery of the first client e-mails forwarded by the web server

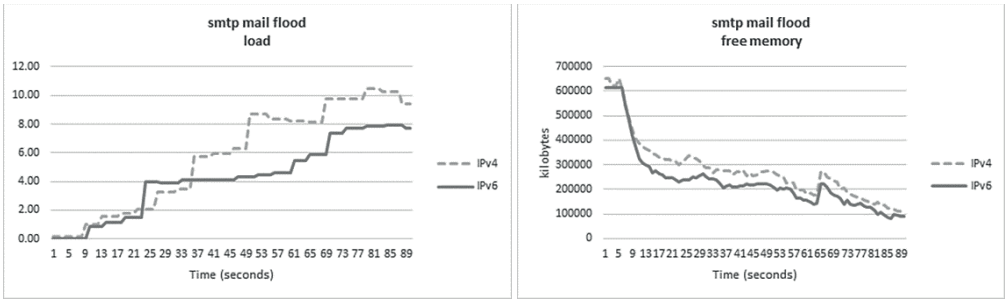


Figure 8. Metrics values for SMTP mail flood attack

Since the attack ended after 90 seconds, the attacked mail server could more gradually deliver undelivered emails from the mail queue in which it had in both cases more than 100,000 pending messages even 10 hours after the attack stopped. Long time delays of legitimate client's emails resending ranged in the order of hours, Table

1. Since most mail programs repeatedly send the undelivered email, the attack is more likely to slow down, not to completely disable the server. On the other hand, the CPU load of the server during 90 seconds of the attack exceeded the value of 10 for IPv4 network and the value of 8 for IPv6 network; the amount of the usable main memory decreased to 17 per cent of the initial amount for IPv4 network and to 14 percent for IPv6 network, which is a high server burden, Figure 8. Moreover, at the end of the attacks, the SMTP server was often disconnected from the client and the exception SMTPServerDisconnected was appearing more and more often.

6. Discussion

As can be seen from Figure 6, 7 and 8, all performed experiments led to real DDoS attacks for both IPv4 and IPv6 networks. Although DDoS SYN flood attacks did not pose a threat to the targeted server due to the implemented SYN cookie mechanism, HTTP Get flood and SMTP mail flood attacks were a serious threat to the targeted server and caused system overload and loss of communication with the server.

Ansible is a convenient software tool for easily performing test DDoS attacks. Attacks are still under the control of the administrator during their execution. Ansible allows to orchestrate a large number of devices controlled from a single point to create a botnet that matches the size and power of real botnets used by attackers in a cyber environment. In addition, its deployment is possible not only for IPv4 networks, but this tool can also be applied to IPv6 networks.

DDoS attacks are gaining strength in the cyber environment, and with the advent of real IPv6 networks, they have spread to these networks using the features of the new IP protocol. In February 2018, the first documented native IPv6 DDoS attack occurred, flooding from approximately 1,900 native IPv6 hosts on more than 650 different networks [39].

IPv6 protocol introduces many features to improve IPv4 performance. New features of the IPv6 protocol include, for example, larger address space, easier TCP/IP administration, IPsec, QoS, auto-configuration, mobility. Moreover, IPv6 protocol eliminates NATing and makes possible for each TCP/IP device to obtain a public IP address. Furthermore, IPv6 supports end-to-end communication. Two types of IP address configuration are provided in the IPv6 protocol, which are stateful and stateless automatic configurations and use NDP instead of ARP, etc. Adding these features will affect the behavior of IPv6 networks. It brings new security issues, but many attacks remain inherited from IPv4 [40].

In addition to showing how an orchestration tool can be used to create a test botnet, the article seeks to show whether adding new features to the IPv6 protocol will affect the overall impact of DDoS attacks on IPv6 networks, either positively or negatively.

The experiments performed showed that the impact on IPv6 and IPv4 networks during DDoS attacks is comparable. Although the IPv6 network showed slightly better DDoS resistance to the IPv4 network, the results were in the same order of magnitude. DDoS attacks on servers have shown that by deploying only the IPv6

protocol, there is no better resistance to DDoS attacks led from a large botnet. Testing of the environment for both versions of the IP protocol showed similar results which led to degradation of victim's services under the attack.

From this point of view, DDoS attack in IPv6 networks is not more dangerous than in IPv4 networks. On the other hand, IPv6 offers several times more IP addresses available whereas the length of the IP address is 32 bits for IPv4 and the length of the IP address is up to 128 bits for IPv6, i.e. up to 2^{128} or 3.4 x 10^{38} different IPv6 addresses can be defined. This means the possibility of connecting an enormous number of other devices to IPv6 networks such as Internet of Things devices [41]. Greater involvement of attacking devices allows for a much larger volume of attacks and therefore a greater impact can be expected for many DDoS attacks in IPv6 networks, as DDoS attacks after the deployment of IPv6 networks will be made from a large number of heterogeneous devices.

7. Conclusion

Distributed Denial of Service attacks is still one of the major threats for both IPv4 as well as IPv6 networks and the malicious enormous traffic from many attacking computers can overflow victim server or network. It is difficult to evaluate DDoS attack of the real network. The study proposed a design and implementation of controlled DDoS attack environment based on Ansible orchestration tool. The environment features are distributed environment, management scalability, open source, easy installation and system implementation. Moreover, no special hardware is required, the testbed uses existing infrastructure in organization; the size of the testbed in the study is comparable with a 10,000-bots botnet. The paper showed the capability of the proposed testbed to conduct real-conditions DDoS attack.

The experiments of SYN flood, HTTP Get flood and SMTP mail flood attack demonstrated that employment of Ansible tool to create a DDoS testbed for real systems is an adequate way how to construct a controlled DDoS attack environment. The effectiveness and the functionality of this tool in both IPv4 and IPv6 networks was proved. The proposed testbed can help experimenters to test realistic DDoS attack under monitoring on their networks for scientific purposes as well as to evaluate vulnerabilities and DDoS countermeasures of their networks. The given DDoS testbed can serve as a basic tool for behavioral testing of computing system under DDoS attack.

In the near future, the same testbed will be used to analyze possibility to involve IoT devices into automated job processing to perform IoT DDoS attacks.

Acknowledgements

This work was supported by the grant VEGA 1/0145/18 *Optimization of network security by computational intelligence*.

References

[1] M. Šimon, L. Huraj, M. Čerňanský, "Performance Evaluations of IPTables Firewall Solutions under DDoS attacks," *Journal of Applied Mathematics, Statistics and Informatics*, Volume 11, Issue 2, 2015, pp. 35-45.

[2] L. D. Paulson, "Hackers strengthen malicious botnets by shrinking them," *Computer*; *News Briefs*. IEEE Computer Society. 39 (4), pp. 17–19, 2006.

[3] P. Masek, et al. "Unleashing Full Potential of Ansible Framework: University Labs Administration," in *The 22nd Conf. of Open Innovations Association (FRUCT)*, IEEE, Finland, pp. 144–150, 2018.

[4] D. Schmidt, and S. M. Shalinie, "*DDoS Testbed. An Investigation into the Detection and Mitigation of Denial of Service (DoS) Attacks*," Springer Science & Business Media, India, pp. 115-129, 2011.

[5] M. Šimon, L. Huraj, "DDoS testbed based on peer-to-peer grid," In *IEEE Int. conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*. IEEE, pp. 1181-1186, 2016.

[6] M. Šimon, L. Huraj, V. Siládi, "Analysis of Performance Bottleneck of P2P Grid Applications," *J Appl Math Stat Inf.*, 9(2), pp. 5-11, 2013.

[7] M. Šimon, L. Huraj, and M. Hosťovecký. "A Mobile Botnet Model Based on P2P Grid," In: *Conf. on Creativity in Intelligent Technologies and Data Science*. Springer, Cham, pp. 604-615, 2017.

[8] S. Behal, and K. Kumar, "Trends in Validation of DDoS Research," *Procedia Computer Science*, 85, pp. 7-15, 2016

[9] J. Dykstra, "*Essential cybersecurity science: build, test, and evaluate secure systems*," O'Reilly Media, Inc., 2015.

[10] I. Halenar, M. Juhas, B. Juhasova, and D. Borkin. "Virtualization of Production Using Digital Twin Technology," In: *20th International Carpathian Control Conference (ICCC)*. IEEE, Poland, pp. 1–5, 2019.

[11] PlanetLab. An open platform for developing, deploying, and accessing planetary-scale services. [Online]. Available: https://www.planet-lab.org/

[12] Geni, Exploring Networks of the Future, 2020. [Online]. Available: https://www.geni.net/

[13] S. Behal, and K. Kumar, "Detection of DDoS attacks and flash events using novel information theory metrics," *Computer Networks*, 116, pp. 96-110, 2017.

[14] K. Singh, P. Singh, K. Kumar, "Application layer HTTP-GET flood DDoS attacks: Research landscape and challenges," *Computers & Security*, 65, pp. 344-372, 2017.

[15] R. Vishwakarma, A. K. Jain, "A survey of DDoS attacking techniques and defence mechanisms in the IoT network." *Telecommunication Systems*, pp. 1-23, 2019.

[16] A. Dudáš, P. Voštinár, J. Škrinárová, J. Siláči, "Improved Process of Running Tasks in the High Performance Computing System," In: *16th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, Slovakia, pp. 133-140, 2018.

[17] J. J. Echevarria, P. Garaizar, J. Legarda, "An experimental study on the applicability of SYN cookies to networked constrained devices," *Software: Practice and Experience*, *48*(3), 740-749, 2018.

[18] R. K. Deka, D. K. Bhattacharyya, J. K. Kalita, "Granger Causality in TCP Flooding Attack," *IJ Network Security*, 21.1, pp. 30-39, 2019.

[19] D. J. Bernstein, "SYN cookies," 1996. Available: http://cr.yp.to/syncookies.html [Accessed: Jan. 15, 2020].

[20] V. T. Dang, T. T. Huong, N. H. Thanh, P. N. Nam, N. N. Thanh, A. Marshall, "SDN-Based SYN Proxy—A Solution to Enhance Performance of Attack Mitigation Under TCP SYN Flood," *The Computer Journal*, *62*(4), pp. 518-534, 2018.

[21] J.J. Echevarria, P. Garaizar, J. Legarda, "An experimental study on the applicability of SYN cookies to networked constrained devices," *Software: Practice and Experience*, 48.3, pp. 740-749, 2018.

[22] G. A. Jaafar, S. M. Abdullah, S. Ismail, "Review of Recent Detection Methods for HTTP DDoS Attack," *Journal of Computer Networks and Communications*, Volume 2019, Article ID 1283472, 10 pages, 2019.

[23] K. Aburada, Y. Arikawa, S. Usuzaki, H. Yamaba, T. Katayama, M. Park, N. Okazaki, "Use of access characteristics to distinguish legitimate user traffic from DDoS attack traffic," *Artificial Life and Robotics*, Springer, 24.3, pp. 318-323, 2019.

[24] A. Kumari, N. Agrawal, U. Lilhore, "Attack over Email System," *International Journal of Scientific Research & Engineering Trends,* Volume 3, Issue 5, pp. 200-206, 2017.

[25] R. Swami, M. Dave, V. Ranga, "Software-defined Networking-based DDoS Defense Mechanisms," *ACM Computing Surveys (CSUR)*, *52*(2), pp.1-5, 2019.

[26] G. Cartier, J. F., Cartier, J. M. Fernandez, "Next-generation DoS at the higher layers: A study of SMTP flooding," In: *International Conference on Network and System Security*, Springer, Berlin, pp. 149-163, 2013.

[27] R.A. Rodríguez-Gómez, G. Maciá-Fernández, and P. García-Teodoro, "Survey and taxonomy of botnet research through life-cycle," *ACM Computing Surveys (CSUR)*, 45(4), pp. 1-33. 2013.

[28] I. Ullah, N. Khan, and H. A. Aboalsamh, "Survey on botnet: Its architecture, detection, prevention and mitigation," In: *10th IEEE Int. Conf. on Networking, Sensing and Control*, France, pp. 660-665, 2013.

[29] S. Thakur, S.C. Gupta, N. Singh, and S. Geddam, "Mitigating and patching system vulnerabilities using ansible: A comparative study of various configuration management tools for IAAS cloud," In *Information Systems Design and Intelligent Applications*. Springer, New Delhi, pp. 21-29, 2016.

[30] J. O. Benson, J.J. Prevost, and P. Rad, "Survey of automated software deployment for computational and engineering research," In: *Annual IEEE Systems Conference (SysCon)*. IEEE, USA, pp. 1-6, 2016.

[31] H. Beiknejad, H. Vahdat-Nejad, and H. Moodi, "P2P Botnet Detection Based on Traffic Behavior Analysis and Classification," *Int. J. of Comp. & Info. Tech.*, 6(1), pp. 1-12, 2018.

[32] P. Rémi, "Ground Control Segment automated deployment and configuration with ANSIBLE and GIT," In: *SpaceOps Conference*. France, p. 2337, 2018.

[33] L. Lu, Y. Feng, and K. Sakurai, "C&C session detection using random forest," In: *Proceedings of the 11th Int. Conf. on Ubiquitous Information Management and Communication*. ACM, 34, pp. 1–6, 2017.

[34] J. Berleur et al., "A Distributed Denial of Service Testbed," In: *HCC9/CIP 2010*, IFIP AICT 328, pp. 338–349, 2010.

[35] Scapy 2.4.3.dev33 documentation https://scapy.readthedocs.io/en/latest/

[36] Scholz, D., Gallenmüller, S., Stubbe, H., Jaber, B., Rouhi, M., and Carle, G. "Me Love (SYN-) Cookies: SYN Flood Mitigation in Programmable Data Planes, " *arXiv preprint arXiv:2003.03221*, 2020.

[37] ab - Apache HTTP server benchmarking tool, Available: http://httpd.apache.org/docs/2.0/programs/ab.html

[38] M. Haddadi, and R. Beghdad, "DoS-DDoS: Taxonomies of attacks, countermeasures, and wellknown defense mechanisms in cloud environment," *EDPACS*, 57:5, pp. 1-26, 2018.

[39] A. Chadd, "DDoS attacks: past, present and future." *Network Security* 2018.7 (2018): 13-15.

[40] A. Shiranzaei, and Rafiqul Z. Khan, "IPv6 security issues—A systematic review." *Next-Generation Networks*. Springer, Singapore, 2018. 41-49.

[41] L. Huraj, M. Šimon, T. Horák, "Resistance of IoT Sensors against DDoS Attack in Smart Home Environment," *Sensors* 2020, *20*, 5298.