Log4Shell

20195178 서영재

요 약

Log4Shell 취약점(CVE-2021-44228)은 Apache Log4j2 2.0-beta9 ~ 2.12.1, 2.13.0 ~ 2.15.0 에서 제공하는 JNDI 기능에서 LDAP을 포함한 기타 JNDI 관련 엔드포인트가 악용되어 RCE 등 악성 행위가 가능한 취약점이다. Log4Shell에 대한 심각성을 알고, 점검 및 대응의 필요성을 상기시켜 위협을 조기에 예방하고 대응할 수 있도록 노력을 기울여야 한다.

Ⅰ. 서 론

본 취약점(CVE-2021-44228)은 2021년 11월 24일 Alibaba Cloud's Security에 의해 발견되었다. 이는 자바 로깅 프레임워크 Log4j의 제로데이 임의 코드 실행 취약점이다. LDAP와 JNDI(Java Naming and Directory Interface)요청을 검사하지 않는 Log4j의 특징을 이용하여 공격자가 원경으로 코드를 실행(RCE)할 수 있게 한다. 본 취약점은 CVSS(Common Vulnerability Scoring System)[1] 점수 최고점인 10점을 부여받았다. 해커가 해당 취약점을 악용할 경우 시스템 관리자 권한을 획득할 수 있는 문제가 있다. 최초 공격은 MS社 마인크래프트 게임에서 이루어진 것으로 알려진다. 이 후 디도스 악성코드 및 랜섬웨어 악성코드 등을 유포하기 위한 공격이 지속되었다.

1) 컴퓨터 시스템 보안의 심각도 및 위험을 평가하는 데 사용되는 지표

Ⅱ. 본 론

2.1. Log4j

소프트웨어 개발자가 자신의 애플리케이션 안에 다양한 데이터의 로그 기록을 남길 수 있게 하는 오픈소스 로깅 프레임워크이다. Ceki G□lc□ 가 처음 개발한 Log4j는 현재(2022년 기준) 아파치 로깅 서비스의 일부로, 콘솔 및 파일 출력 형태의 로깅을 지원한다. 다양한 자바 기반의 서비스, 기업용 소프트웨어 전반에서 로그 기록을 위해 사용된다. Log4j와 같이 소프트웨어 동작 시 시스템의 상태와 이벤트를 시간 경과에 따라 기록하는 로깅 관련 프레임워크는 LOGBACK, Log4j2 등이 있다.

2.2. JNDI(Java Naming and Directory Interface)

자바에서 디렉토리를 이용하여 데이터를 호출할 수 있게 해주는 디렉토리 서비스로, 다양한 디렉토리 서비스를 이용할 수 있게 하는 CORBA COS, Java RMI Registry, LDAP

등의 SPI를 제공한다. Java 애플리케이션이 조회를 수행하고 LDAP, RMI, DNS 등과 같은 프로토콜을 사용하여 Java 객체를 검색할 수 있도록 하는 Java API이다. 애플리케이션이 RMI 레지스트리에 등록된 원격 객체 또는 LDAP와 같은 디렉토리 서비스와 상호 작용할 수 있는 API를 제공한다. 공격자는 LDAP URL을 컨트롤하여 자신의 통제 하에 Java 프로그램을 실행시켜 오브젝트를 로드할 수 있게 되는데 본 CVE-2021-44228 취약점은 이점을 공격에 활용한 것이다.

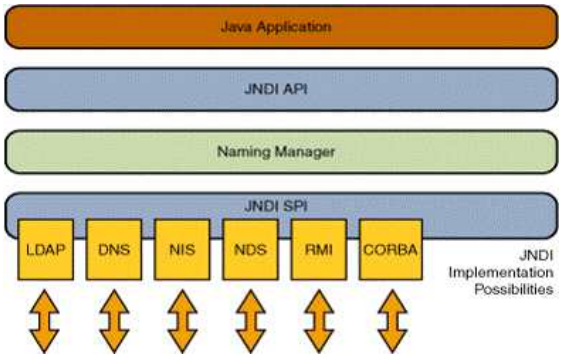

그림 2 JNDI 개념 및 동작구조

2.3 영향도 및 타임라인

날짜	내용
21/11/24	알리바바社 취약점 최초 발견
21/12/06	CVE-2021-44228 보안업데이트 배포
21/12/09	알리바바社 공격PoC 공개
21/12/10	마인크래프트 공격 탐지
21/12/11	주요 기관/기업 긴급 상황 1차 전파
21/12/13	CVE-2021-44228 보안업데이트 배포
21/12/15	주요 기관/기업 긴급 상황 2차 전파
21/12/16	CVE-2021-44228 보안업데이트 배포
21/12/18	벨기에 국방부 공격 탐지
	CISO 간담회
21/12/19	주요 기관/기업 긴급 상황 3차 전파
21/12/28	CVE-2021-44228 보안업데이트 배포
21/12/29	주요 기관/기업 긴급 상황 4차 전파

표 2 Log4j 취약점 타임라인

2.4 분석

Log4j에서는 로깅을 위해 아래와 같은 메소드를 사용하며 설정 파일을 참고한다.

```
org.apache.logging.log4j.core.Logger.log
privateConfig.loggerGonfig.getReliability
strategy()
```

이후 noLookups 옵션을 체크한 후 '${' 문자열을 찾아 해당 문자열이 존재할 경우 config.getStrSubstitutor().replace() 메소드를 호출한다. 본 메소드는 value 값을 인자값으로 하여 StrSubstitutor.substitue() 메소드를 호출한다.
StrSubstitutor.substitue() 메소드는 전잘된 물자열에서 '$' , '{' , '}' 을 제외한 값을 StrSubstitutor.resolveVariable() 메소드의 인자로 전달한다. 이 값은 다시 resolve.lookup() 메소드로 실행된다. resolve.lookup() 메소드는 PREFIX_SEPARATOR을 기준으로 구분하여 변수를 설정하고 prefix로 실행할 lookup() 메소드를 설정한다. 취약점 공격을 위해서는 JNDI를 사용하므로, JNDILookup.lookup() 메소드를 실행한다. 본 메소드는 다시 jndiManager.lookup() 메소드를 실행한다. 본 취약점이 바로 jndiManager.lookup() 메소드로 인해 발생한다. 인자로 전달된 값에 eogs 검증 절차 없이 바로 lookup() 기능을 통해 접근을 시도하여 context 검색함으로써 공격자 LDAP 서버에 Exploit 엔티티를 요청하여 악성 행위가 가능하게 된다.

2.3 취약점 조치방안

여러 보안업체에서 배포중인 Log4j 취약점 스캐너, 운영체제 기본 검색 등을 이용하여 Log4j 취약점 또는 Log4j 버전을 확인할 수 있다. 이를 활용하여 취약한 버전이 확인된 경우, 아래와 같이 JAVA 버전에 따라 Log4j를 최신 버전으로 업데이트해야 한다. 업데이트가 어려운 경우, 임시 조치 방안으로 JndiLookup 클래스만 제거한다.

2022년 1월 기준, Log4j 최신 버전	
JAVA 8 이상	2.17.1 이상
JAVA 7	2.12.4 이상
JAVA 6	2.3.2 이상

표 4 JAVA 버전에 따라 업데이트해야하는 Log4j 버전

2.4 보안담당자에게 필요한 노력

1) 보안담당자들은 취약점 공격을 탐지 및 차단할 수 있도록 주기적으로 보안장비 탐지룰을 업데이트해야 한다. Log4j 취약점 공격을 시도하는 것으로 알려진 IP를 방화벽에서 차단해야 한다. 2) 공격자는 중앙관리시스템과 같은 내부 호스트들과 접점이 많은 지점을 거점으로 악용하는 경우가 많다. 따라서 공격자가 내부 피해 확산을 위해 거점으로 장악할 수 있으므로 주요 자원에 대한 선별적인 점검과 집중 모니터링을 해야한다. 3) 주요 시스템에 대한 모니터링을 강화하기 위해 각 시스템별 보안 관점에서 활용 가능한 로그를 철저히 기록하고 공격자에 의해 삭제되지 않도록 관리해야 한다. 4) 소프트웨어 관리 체계에 대해 개선해야 한다.

Ⅲ. 결 론

Log4j 관련 보안 취약점이 계속 발견되면서 장기화될 전망이 나오고 있다. 이에 Log4Shell 취약점에 대한 중대한 위협을 조기에 예방하고 대응할 수 있도록 노력을 기울여야 한다. 방어자 관점에서 면밀히 분석하고자 구체적인 테스트를 통해 취약점 발현원리와 공격자들의 해당 취약점 악용 기업을 파악한다. 향후 지속되는 공격을 효과적으로 대응하여 더 큰 문제가 발생되지 않도록 해야 한다.

참 고 문 헌

[1] Douglas Everson, Long Cheng, Zhenkai Zhang, “Log4shell: Redefining the Web Attack Surface”

[2] Raphael Hiesgen, Marcin Nawrocki, Thomas C. Schmidt, Matthias Wahlisch, “The Race to the Vulnerable : Measuring the Log4j Shell Incident”

[3] 과학기술정보통신부, 한국인터넷 진흥원 “Log4j 위협 대응 보고서 v1.0”

[3] Alvaro Mu□oz (@pwntester) Oleksandr Mirosh, “A JOURNEY FROM JNDI/LDAP MANIPULATION TO REMOTE CODE EXECUTION DREAM LAND” , Blackhat USA 2016

[4] as3617, “Log4j2 vulnerability analysis” , https://blog.ssrf.kr/m/70