U2W2L5 – Progetto Settimanale

Authentication cracking con Hydra

Autore: Pagnacco Elena

Sintesi:

È stato simulato un attacco wordlist sui servizi SSH e FTP tramite Hydra. In questo scenario didattico, l'attacco è stato eseguito verso l'interfaccia di rete locale della macchina attaccante stessa.

Il test ha dimostrato che una password debole compromette un sistema in poco tempo, rendendo inefficace la crittografia del *tunnel SSH*.

Richieste:

- Abilitazione e configurazione dei servizi
- Cracking dell'autenticazione con Hydra per SSH
- Cracking dell'autenticazione con Hydra per FTP

Introduzione:

Il presente documento riporta l'analisi tecnica di un'attività di **Vulnerability Assessment e Penetration Testing** mirata ai servizi di autenticazione. L'obiettivo della simulazione è verificare la resilienza dei protocolli **SSH** (Secure Shell) e **FTP** (File Transfer Protocol) di fronte ad attacchi di tipo "Wordlist Attack" (brute-force basato su dizionario).

L'attività è stata condotta in un ambiente di laboratorio controllato (Sandbox su Kali Linux), simulando uno scenario in cui l'auditor conosce l'indirizzo IP del target ed ha già effettuato la fase di *enumerazione*, ma non le credenziali di accesso. Verranno dimostrate le criticità derivanti dall'utilizzo di password deboli e protocolli non cifrati, proponendo infine le adeguate contromisure difensive.

Punti chiave:

- Creazione nuovo utente su kali
- Configurazione e attivazione servizio da testare
- Test della connessione
- Cracking con Hydra tramite liste di utenti e password

Strumenti:

- **THC-Hydra:** è un *Network Logon Cracker* modulare e parallelizzato, è progettato per eseguire attacchi di *brute-force* e *wordlist* (a dizionario) in modalità online.
  A differenza dei tool che elaborano hash offline, Hydra interagisce direttamente con i servizi di rete attivi, aprendo molteplici *socket* simultanei per simulare tentativi di autenticazione *client-server*.
  THC-Hydra fa uso intensivo di *multi-threading*, che permette la parallelizzazione delle connessioni per massimizzare il *throughput* dei tentativi su oltre 50 protocolli supportati. *L'uso intensivo di*

thread paralleli può causare Denial of Service (DoS) involontario su server datati o far scattare meccanismi di blocco (IPS/WAF) in scenari reali. THC-Hydra è uno degli standard industriali per verificare la presenza di credenziali deboli o di default su infrastrutture esposte e in tempo reale.

- **Seclists:** è il *repository open-source* di riferimento per l'aggregazione di liste utilizzate nei *security assessment*. Non è un software, ma una raccolta strutturata di dizionari per username, password, URL discovery e payload per il *fuzzing*. Centralizza i dati di test necessari per le varie fasi di un attacco, eliminando la necessità di generare manualmente input per ogni verifica.
    - **Xato (xato-net-10-million):** è una *wordlist (dizionario)* statistica inclusa in SecLists, basata sull'analisi della frequenza d'uso reale delle password nei data breach storici. A differenza delle liste generate algoritmicamente, le password sono ordinate per **popolarità** (dalla più comune alla meno comune). È fondamentale negli attacchi online per massimizzare la probabilità di accesso con i primi *N* tentativi (es. Top 1000), riducendo il rischio di *account lockout*.

Svolgimento:

Creazione utente:

In prima battuta si è proceduto a creare un nuovo utente con credenziali deboli sulla macchina Kali.

Tramite il comando "$ *adduser nomeutente"* è stato creato un utente con nome test_user e password passtest.

Le altre informazioni sono state lasciate di default in quanto non rilevanti ai fini di questo test.

Con l'utente ora configurato, è stato poi avviato il servizio SSH tramite il comando *"$ sudo service ssh start".*

```
┌──(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
```

Figura 1 Creazione utente debole e avvio del servizio ssh

Si è poi proceduto al test della connessione SSH e al login dell'utente creato tramite il comando *"$ ssh test_user@IPKALI"*.

Il login è stato effettuato con successo, come dimostrato dalla risposta al comando e dalla successiva presenza della dicitura *test_user@kali* sul terminale.

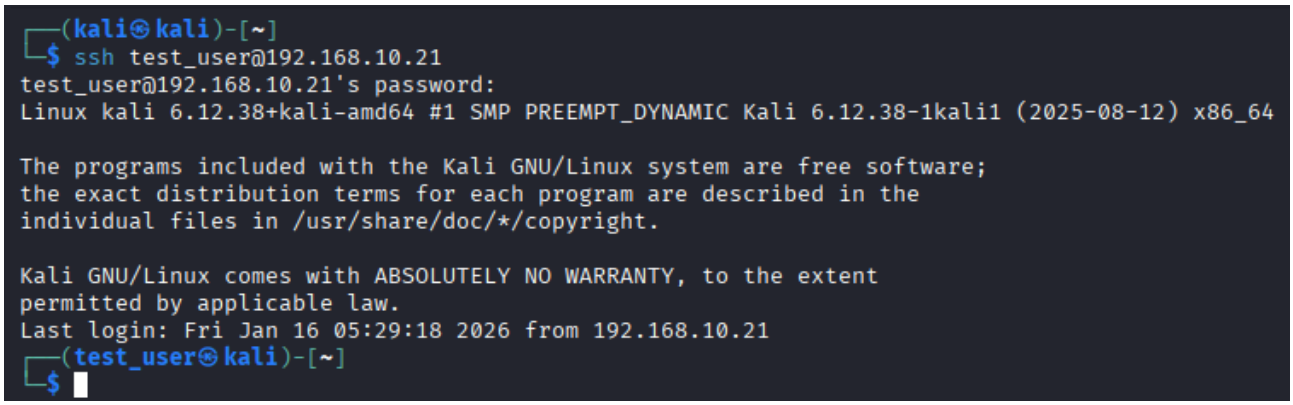

Figura 2 test di connessione e login

Con questo, la configurazione iniziale dell'utente è stata conclusa.

Scaricamento e riduzione seclists:

A scopi illustrativi, si è deciso di installare la collezione di username e password di seclists tramite il comando *"$ sudo apt install seclists*" lanciato da terminale kali@kali. Questo comando scarica diverse collezioni di username e password recuperate da violazioni precedenti, tra cui quelle xato che sono state usate per questo test.

Data la vastità dei file scaricati, è stata effettuata una riduzione di questi ultimi.

È stato lanciato il comando "$ *cat /usr/share/seclists/Usernames/xato-net-10-million-usernames.txt | grep test > xato-usernames.txt"* che legge, tramite *cat*, l'intero contenuto della *wordlist* specificata. Lo stream dei dati viene passato in *pipe* **(|)** al tool *grep*, il quale filtra le righe in base al pattern **'test'**. Infine, l'operatore di *ridirezione* **(>)** salva l'output filtrato all'interno del file di destinazione *xato-usernames.txt*.

Cosa equivalente è stata fatta per le password, il cui file di destinazione è stato nominato *xato-passwords.txt*.

Al fine di ottimizzare i tempi di esecuzione in ambiente di laboratorio, le wordlist sono state ulteriormente filtrate manualmente estraendo solo le prime dieci occorrenze contenenti la stringa 'test', assicurando comunque la presenza delle credenziali dell'utente da noi creato. Questi file sono entrambi situati nel path "/home/kali".

La riduzione tramite grep è stata effettuata esclusivamente per scopi di *Proof of Concept (PoC)* didattico, al fine di validare la metodologia riducendo i tempi di computazione. In uno scenario reale (*Black Box o PenTesting*), l'attaccante utilizzerebbe la *wordlist* completa.

Cracking delle password (servizio SSH):

Ottenute le due liste di nostro interesse e avendo già avviato il servizio, è stato lanciato il comando "$ hydra -L /home/kali/xato-usernames.txt -P /home/kali/xato-passwords.txt 192.168.10.21 -t2 ssh -f" che tenta, per il servizio SSH aperto sull'host con IP 192.168.10.21, di crackare all'effettivo le credenziali scorrendo una lista di usernames e una di passwords, il flag -f (exit on first found) interrompe l'esecuzione al momento del primo match permettendoci il risparmio di risorse.

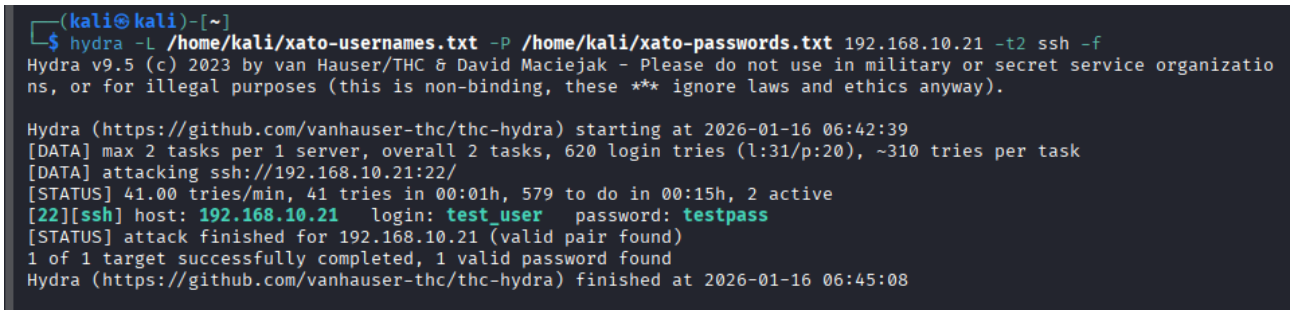

Figura 3 Cracking credenziali SSH

Come da immagine, si può vedere che le credenziali sono state correttamente trovate.

Cracking delle password (servizio FTP):

La dimostrazione è proseguita poi con la configurazione e il *cracking* del servizio *FTP*.

Configurazione:

È stato scaricato e installato il servizio FTP tramite il comando "$ *sudo apt install vsftpd*", esso è stato subito avviato ("$ *sudo service vsftpd start*").

Per controllare l'avvenuta connessione e confermare il login, è stato eseguito dal terminale del test_user il comando "$ ftp 127.0.0.1" che ha permesso di effettuare con certezza il login manuale al servizio FTP.

Questo comando è la versione estesa del comando usato per fare il login al servizio SSH, in quanto qui siamo andati a specificare in secondo momento il nome utente.

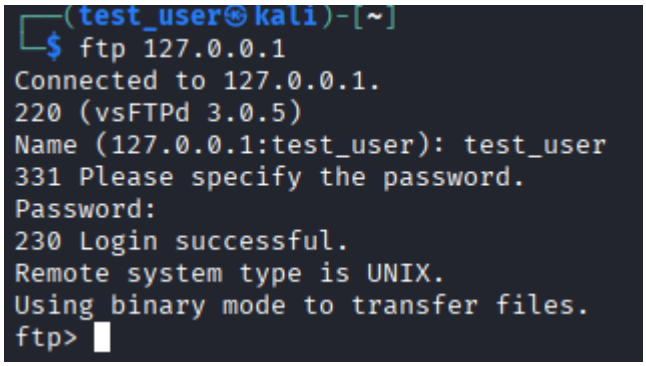

Figura 4 Login FTP

Cracking:

Il procedimento per il cracking è stato il medesimo eseguito per SSH, cambiando solamente il nome del servizio nel comando lanciato, ovvero "$ hydra -L /home/kali/xato-usernames.txt -P /home/kali/xato-passwords.txt 192.168.10.21 -t2 ftp -f".

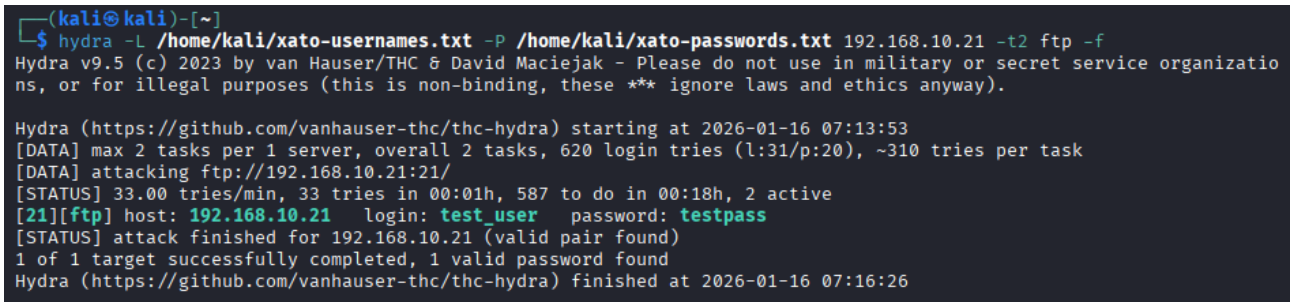

Figura 5 Cracking FTP

Utilizzando le medesime liste e i medesimi flag, anche il risultato è stato il medesimo, anche se si può notare come la porta e il servizio differiscano tra i due risultati.

Analisi Comparativa del Rischio (SSH vs FTP):

Sebbene l'output di Hydra appaia identico per entrambi i servizi (credenziali individuate), il livello di rischio intrinseco differisce sostanzialmente.

- **SSH:** Garantisce la **confidenzialità** del canale tramite *crittografia*. La vulnerabilità risiede esclusivamente nella debolezza della password.
- **FTP:** È un protocollo obsoleto e insicuro *by design*. Le credenziali e i dati viaggiano in chiaro (*Cleartext*). Il servizio FTP rappresenta una vulnerabilità critica indipendentemente dalla complessità della password, in quanto in una rete commutata (switchata), un attacco di *ARP Spoofing* permetterebbe di leggere la password in chiaro, rendendo inutile la complessità della password stessa su FTP.

Analisi dei Tempi e Proiezione Scenari Reali:

Durante il test, con il flag di limitazione -t 2, è stato rilevato un throughput medio di 41 tentativi al minuto su SSH e 33 su FTP. Proiettando questi dati su uno scenario reale (ipotizzando un flag -t 4 per massimizzare l'efficacia senza causare DoS), emergono due evidenze critiche:

- **Inefficacia del Brute-Force Totale:** L'esecuzione dell'intera wordlist *xato-net-10-million* richiederebbe circa **85 giorni** ininterrotti per SSH. Questo conferma che, contro policy di password robuste (che costringono l'uso di stringhe non presenti nei dizionari), gli attacchi online sono poco efficienti.
- **Criticità delle Password Comuni:** Tuttavia, considerando che la *wordlist* è ordinata per frequenza d'uso, un attaccante ha un'alta probabilità di successo testando solo le prime 10.000 password più comuni. In questo scenario, il tempo di compromissione scende drasticamente a circa **2 ore**.

Conclusioni:

L'attività di testing e l'analisi dei tempi hanno confermato che la sicurezza non deriva dall'impedire l'attacco brute-force (tecnicamente sempre possibile), ma nel rendere il tempo necessario per il successo (*Time-to-Crack*) talmente elevato da scoraggiare l'attaccante.

Al fine di mettere in sicurezza i sistemi analizzati, si raccomandano le seguenti azioni correttive:

1. **Password Policy:** Imporre policy stringenti che richiedano una lunghezza minima di 12 caratteri, l'uso di caratteri speciali e l'assenza di parole di senso compiuto.

2. **Dismissione FTP:** Il servizio FTP deve essere **disabilitato** e sostituito integralmente con **SFTP** (SSH File Transfer Protocol) o **FTPS**, per garantire la cifratura del traffico in transito.
3. **SSH Hardening:** Si consiglia di disabilitare l'autenticazione via password per il servizio SSH, favorendo l'utilizzo esclusivo dell'autenticazione a chiave asimmetrica (**SSH Public Key Authentication**).
4. **Mitigazione Attiva:** Implementare soluzioni di *Intrusion Prevention* come **Fail2Ban**. Tali strumenti monitorano i log di sistema e bloccano temporaneamente gli indirizzi IP che generano un numero elevato di tentativi di accesso falliti, rendendo inefficaci gli attacchi di forza bruta massivi
5. **Implementazione MFA:** L'adozione di un secondo fattore di autenticazione (es. Google Authenticator) mitiga quasi totalmente il rischio di *brute-force*, poiché conoscere la password non è più sufficiente per l'accesso.