

\* NEBULA  
LEVEL 14

# NEBULA CTF '14

---

Prof. ssa Barbara Masucci



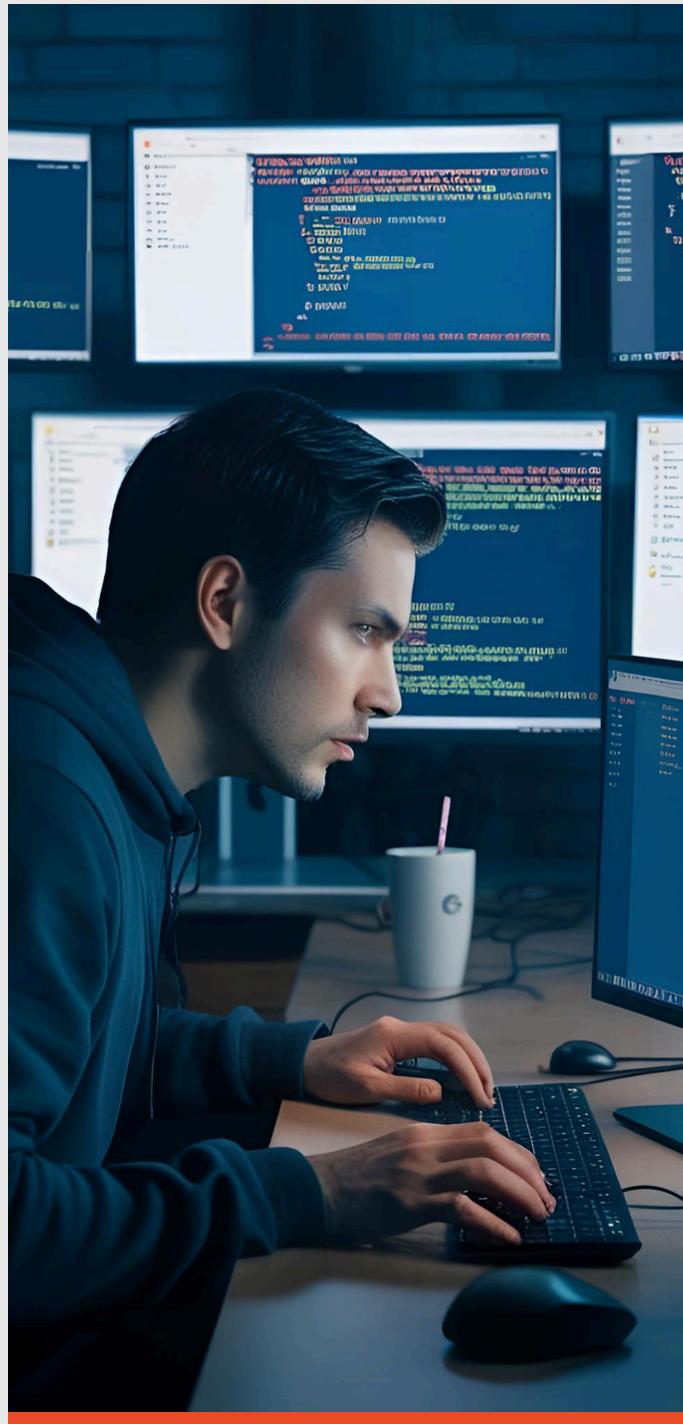
[exploit-exercises.com/nebula](http://exploit-exercises.com/nebula)

## Studenti

Carmine Calabrese 0522501853  
Marco Ciano 0522501674  
Giuseppe D'Avino 0522501747

---

# INTRODUZIONE ALLA SFIDA



“This program resides in `/home/flag14/flag14`. It encrypts input and writes it to standard output. An encrypted token file is also in that home directory, decrypt it :)"

There is no source code available



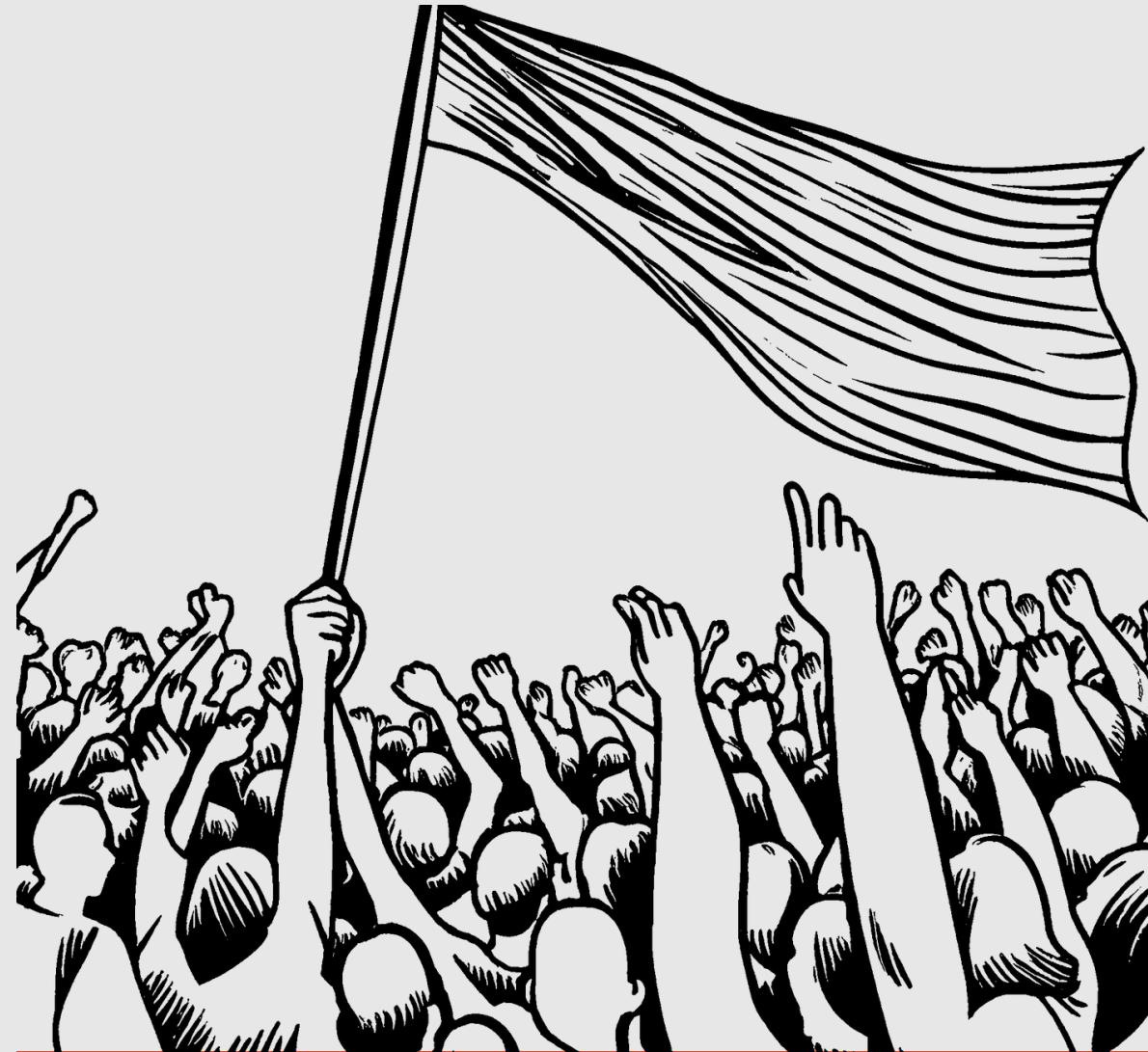
Per affrontare la sfida CTF

Occorre effettuare il login con le seguenti credenziali:

- username: **level14**
- password: **level14**

# OBIETTIVO DELLA SFIDA

---

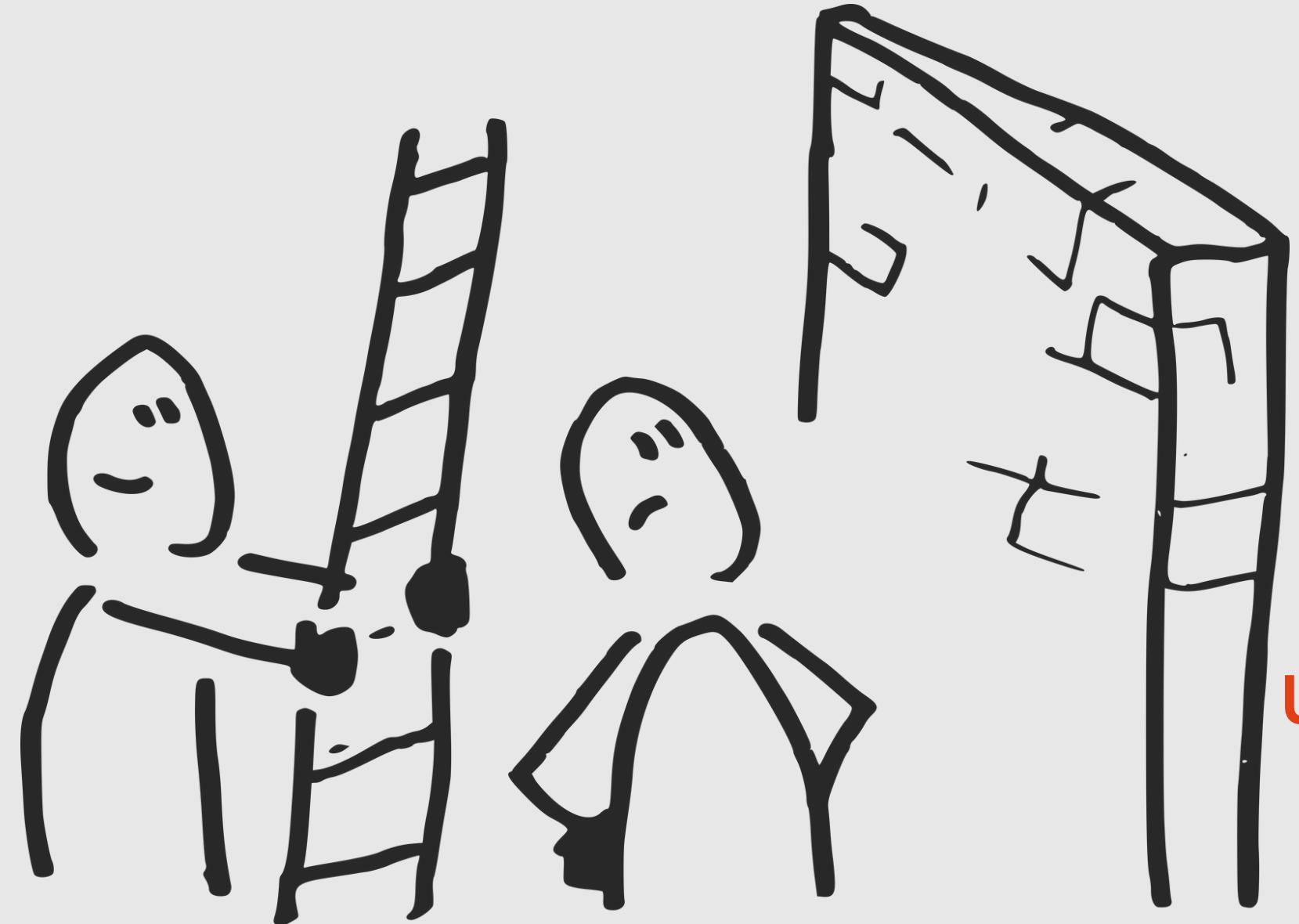


Esecuzione di **/bin/getflag** con i  
privilegi dell'utente **flag14**



You have successfully executed  
**getflag** on a target account

# POSSIBILI SOLUZIONI

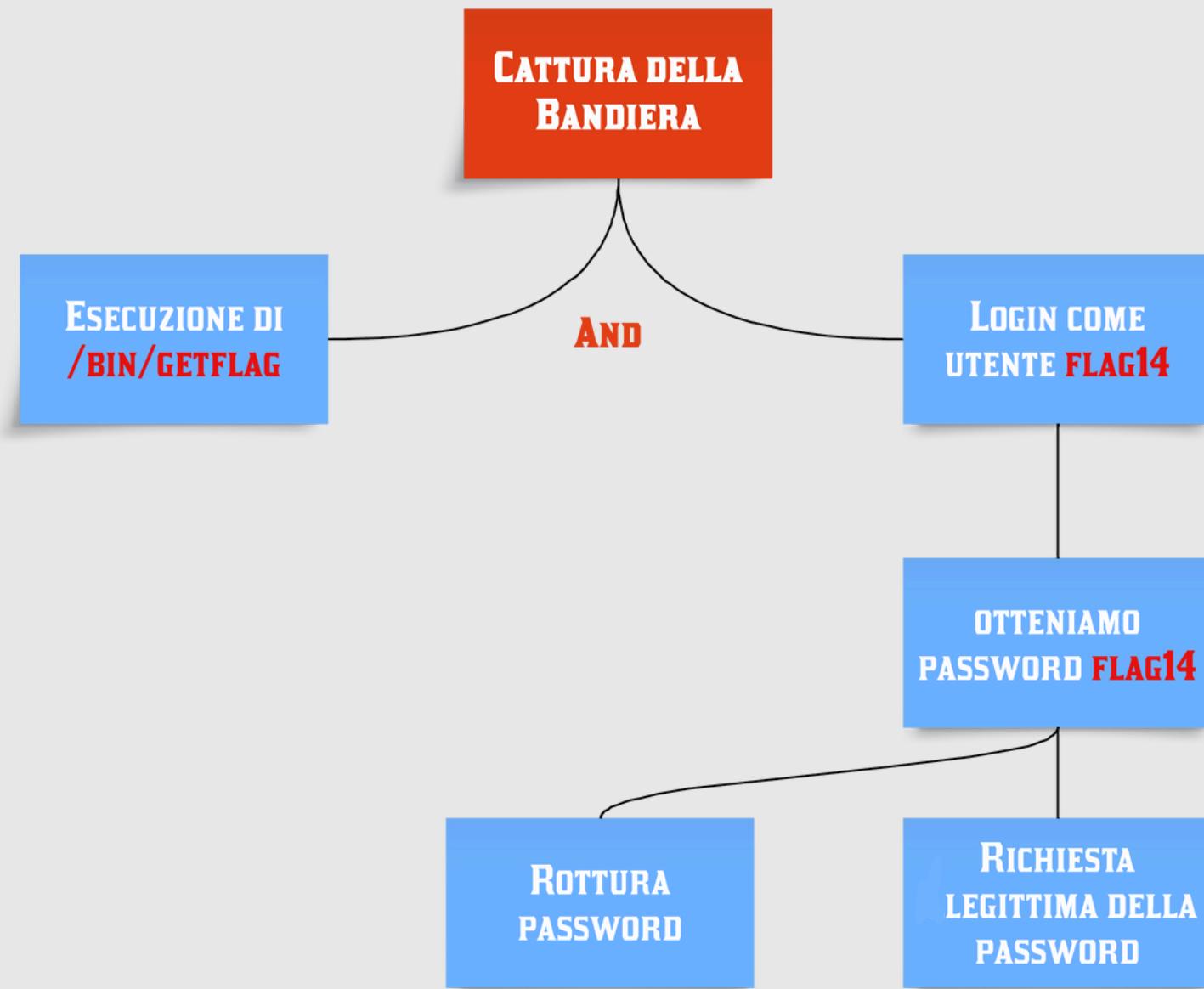


**Rottura** della password

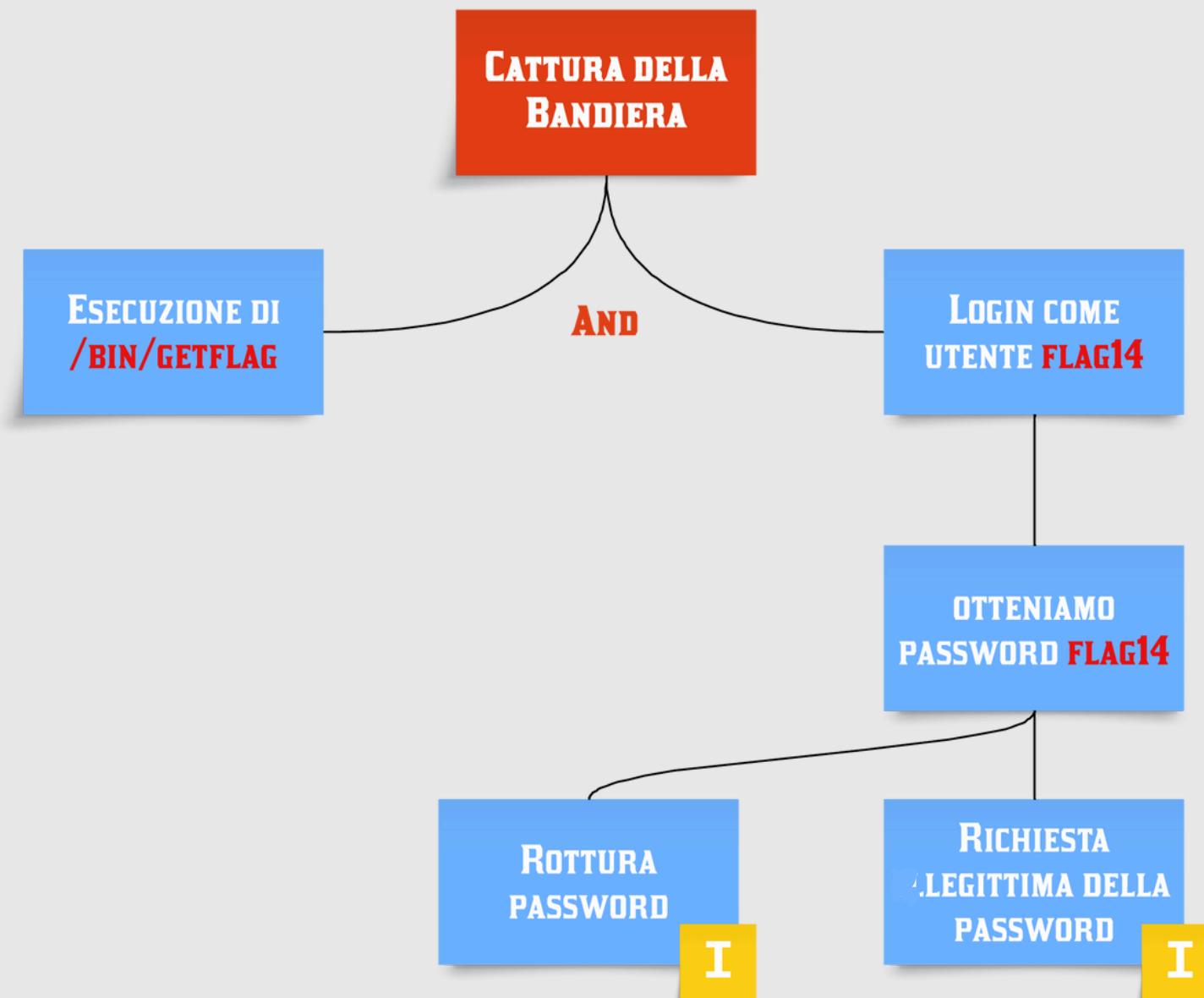
**Richiesta legittima** della password

**Utilizzo dei file** a nostra disposizione

# ALBERO DI ATTACCO



Il primo albero d'attacco che andiamo a realizzare, rappresenta lo standard per le sfide CFT di Nebula. In questo modo possiamo verificare se le conoscenze delle sfide pregresse, possano apportare una base di partenza significativa per la risoluzione della CFT level14.



# ALBERO DI ATTACCO

Tentare di ottenere o forzare la password è una strada possibile, ma estremamente complessa e poco efficiente. Inoltre, andrebbe contro lo **spirito** e gli **obiettivi** della challenge



I nodi riguardanti **rottura** e **richiesta legittima** sono stati marcati come *impossibili*, quindi bisognerà esplorare nuove alternative

# ANALISI DELLE DIRECTORY

---

La directory **/home/level14** non contiene file interessanti

```
level14@nebula:~$ ls -la /home/level14
total 7
drwxr-x--- 1 level14 level14 60 2025-04-14 08:01 .
drwxr-xr-x 1 root     root    80 2012-08-27 07:18 ..
-rw-r--r-- 1 level14 level14 220 2011-05-18 02:54 .bash_logout
-rw-r--r-- 1 level14 level14 3353 2011-05-18 02:54 .bashrc
drwx----- 2 level14 level14 60 2025-04-14 08:01 .cache
-rw-r--r-- 1 level14 level14 675 2011-05-18 02:54 .profile
-rw----- 1 level14 level14 1392 2012-08-23 03:41 .viminfo
level14@nebula:~$ _
```

# ANALISI DELLE DIRECTORY

---

**ls- la /home/flag14/flag14**

```
level14@nebula:~$ ls -la /home/flag14/flag14
-rwsr-x--- 1 flag14 level14 7272 2011-12-05 18:59 /home/flag14/flag14
level14@nebula:~$ _
```

- Il file è di proprietà **flag14**
- Il **bit setUID** è acceso
- Il file è eseguibile dal gruppo **level14**

# ANALISI DELLE DIRECTORY

---

**ls- la /home/flag14/token**

```
level14@nebula:~$ ls -la /home/flag14/token
-rw----- 1 level14 level14 37 2011-12-05 18:59 /home/flag14/token
level14@nebula:~$
```

L'utente **level14** ha i permessi di lettura e scrittura

# ANALISI DELLE DIRECTORY

---

## cat token

Tramite il comando **cat** visualizziamo il contenuto del file

```
level14@nebula:/home/flag14$ cat token
857:g67?5ABBo:BtDA?tIvLDKL{MQPSRQWW.
level14@nebula:/home/flag14$
```

Il file contiene una stringa alfanumerica all'apparenza randomica

# ANALISI DEL FUNZIONAMENTO

---

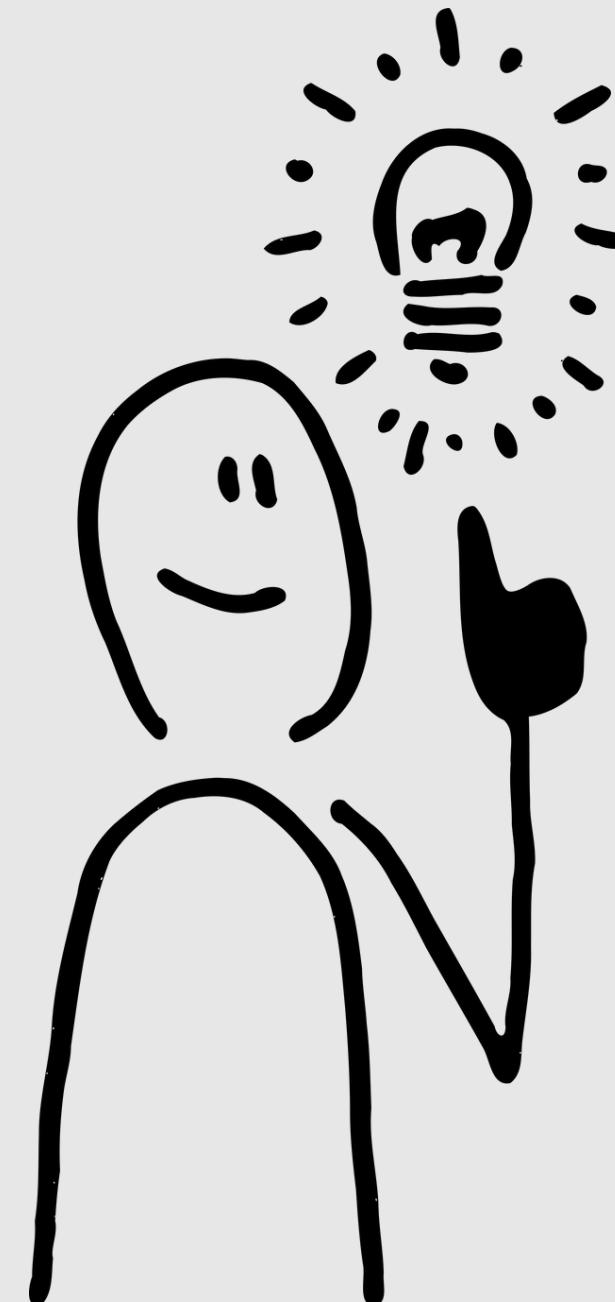
In mancanza di un file sorgente, vengono effettuate delle **prove** per comprendere il funzionamento del programma

```
level14@nebula:~$ ./home/flag14/flag14
./home/flag14/flag14
      -e      Encrypt input
level14@nebula:~$
```

Il file richiede un parametro **-e** per effettuare la cifratura di una stringa in input

# STRATEGIA INIZIALE

---

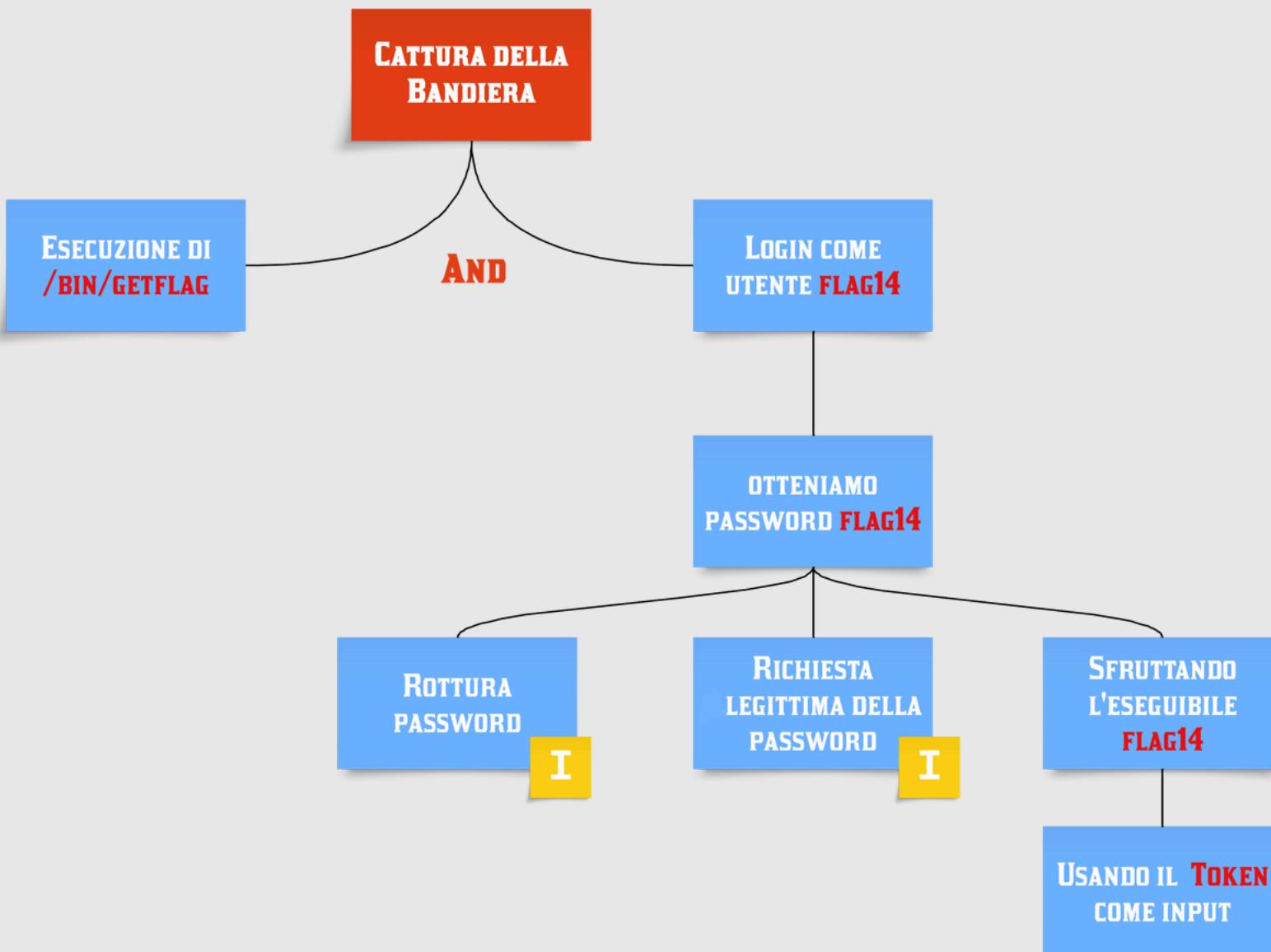


Essendo **flag14** un programma per cifrare stringhe, forse la stringa contenuta in **token** non è del tutto casuale



**token** potrebbe essere il risultato di una stringa cifrata mediante **flag14**

# ALBERO DI ATTACCO



Aggiorniamo il nostro albero di attacco, aggiungendo l'utilizzo del file **token** come input

# STRATEGIA PRELIMINARE

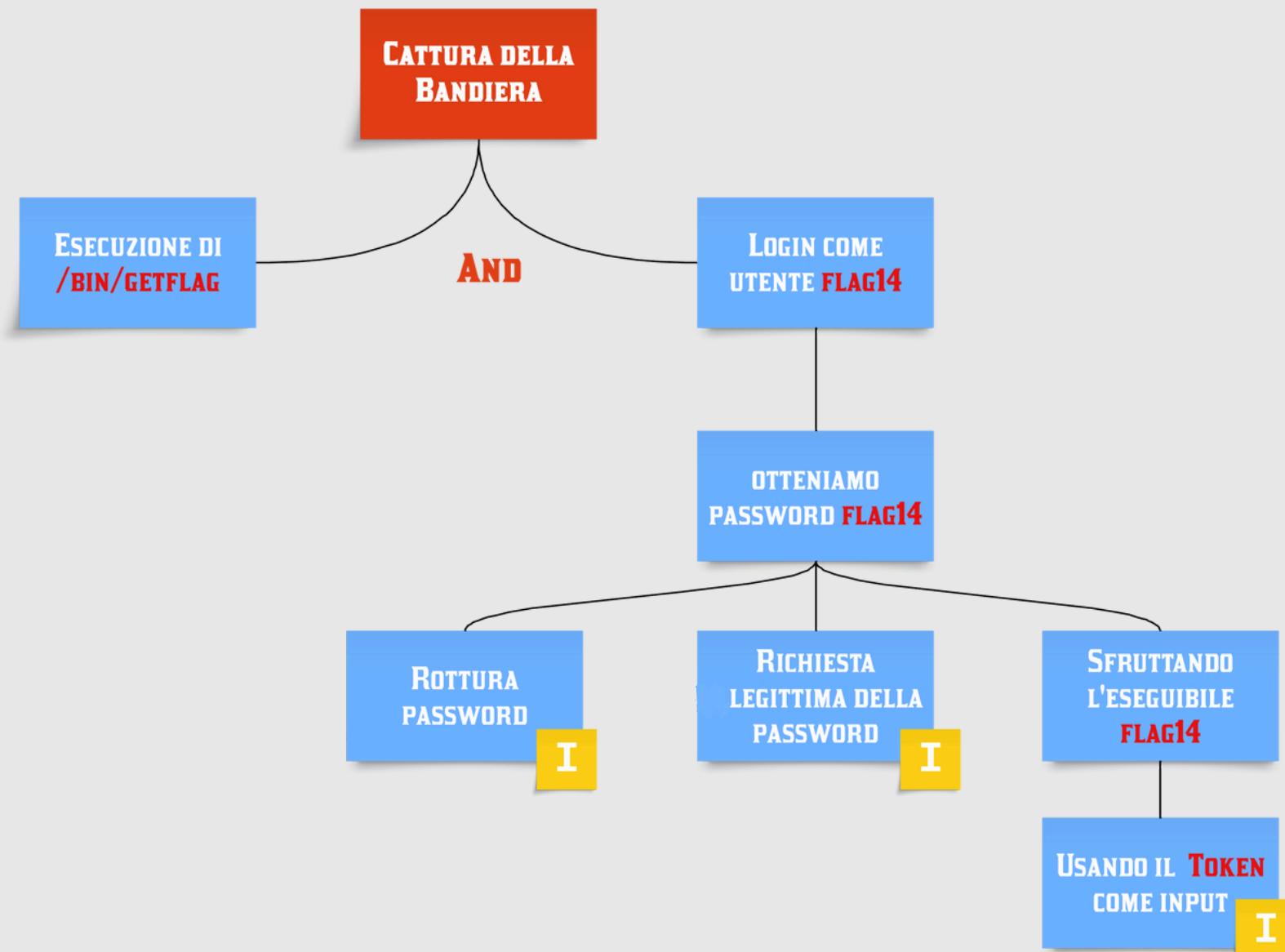


Eseguiamo il programma dando in input la stringa contenuta nel file **token** e verifichiamo se l'output ottenuto risulta esserci di aiuto

```
level14@nebula:~$ ./home/flag14/flag14 -e  
857:g67?5ABBo:BtDA?t1vLDKL{MQPSRQWW.  
869=k ;=F=JLM{GP♦TRQ♦♦♦b [ce♦hmmqqqxyQ._
```

L' **output** non può essere utilizzato in alcune modo, data la presenza di caratteri non riconosciuti dall' **OS**

# ALBERO DI ATTACCO



Aggiorniamo il nostro albero, marcando il nodo riguardante l'uso del **token** come input ad *impossibile*

# STRATEGIA ALTERNATIVA

---



Una strategia provata, e che ha portato alla risoluzione della sfida, è stata la strategia del **black-box analysis**, una tecnica di **Reverse Engineering** che prevede lo studio di output per comprendere il comportamento del programma



Quindi, dare in input stringhe mirate così da poter studiare l'algoritmo di cifratura

# STRATEGIA ALTERNATIVA

Proviamo quindi a studiare il comportamento del programma **immettendo come input** la stringa “salveatutti”



```
level14@nebula:/home/flag14$ ./flag14 -e  
salveatutti  
sbnyifz||}s_
```

Tale output non sembra dirci nulla, se non che lettere uguali **non producono** lo stesso risultato

# STRATEGIA ALTERNATIVA

---

Eseguiamo un'altra prova immettendo come input la stringa “salvesalve” ed effettuiamo un confronto con quello che era l'output precedente

Output precedente

```
salveatutti  
sbnyifz||}s
```

Output corrente

```
salvesalve  
sbnyixgs~n
```

Possiamo notare che:

- I due output hanno una **parte in comune** corrispondente alla parte di input uguale.
- L'output sembra tener conto della **posizione** delle lettere nell'input.

# STRATEGIA ALTERNATIVA



Per avere conferma di quanto appena detto, eseguiamo il programma inserendo come input una **stringa numerica** “**123456**”

Posizione: 0 1 2 3 4 5

|   |   |   |   |   |   |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 3 | 5 | 7 | 9 | ; |

La domanda che ci siamo posti è

Perchè il carattere che segue l'elemento “9” all'interno dell'output non è un numero?

# STRATEGIA ALTERNATIVA

## man ascii

Per rispondere a tale domanda, facciamo riferimento alla tabella ASCII per studiare come i caratteri sono ordinati.



| Oct | Dec | Hex | Char |
|-----|-----|-----|------|
| 060 | 48  | 30  | 0    |
| 061 | 49  | 31  | 1    |
| 062 | 50  | 32  | 2    |
| 063 | 51  | 33  | 3    |
| 064 | 52  | 34  | 4    |
| 065 | 53  | 35  | 5    |
| 066 | 54  | 36  | 6    |
| 067 | 55  | 37  | 7    |
| 070 | 56  | 38  | 8    |
| 071 | 57  | 39  | 9    |
| 072 | 58  | 3A  | :    |
| 073 | 59  | 3B  | ;    |
| 074 | 60  | 3C  | <    |
| 075 | 61  | 3D  | =    |
| 076 | 62  | 3E  | >    |
| 077 | 63  | 3F  | ?    |

carattere '6' con **ASCII code 54**



carattere ';' con **ASCII code 59**

# STRATEGIA ALTERNATIVA

Posizione: 0 1 2 3 4 5

123456  
13579;\_

| INPUT | Codice ASCII<br>INPUT | OUTPUT | Codice ASCII<br>OUTPUT | Differenza<br>tra i Codici<br>ASCII |
|-------|-----------------------|--------|------------------------|-------------------------------------|
| 1     | 49                    | 1      | 49                     | 0                                   |
| 2     | 50                    | 3      | 51                     | 1                                   |
| 3     | 51                    | 5      | 53                     | 2                                   |
| 4     | 52                    | 7      | 55                     | 3                                   |
| 5     | 53                    | 9      | 57                     | 4                                   |
| 6     | 54                    | ;      | 59                     | 5                                   |

# ALGORITMO DI CIFRATURA

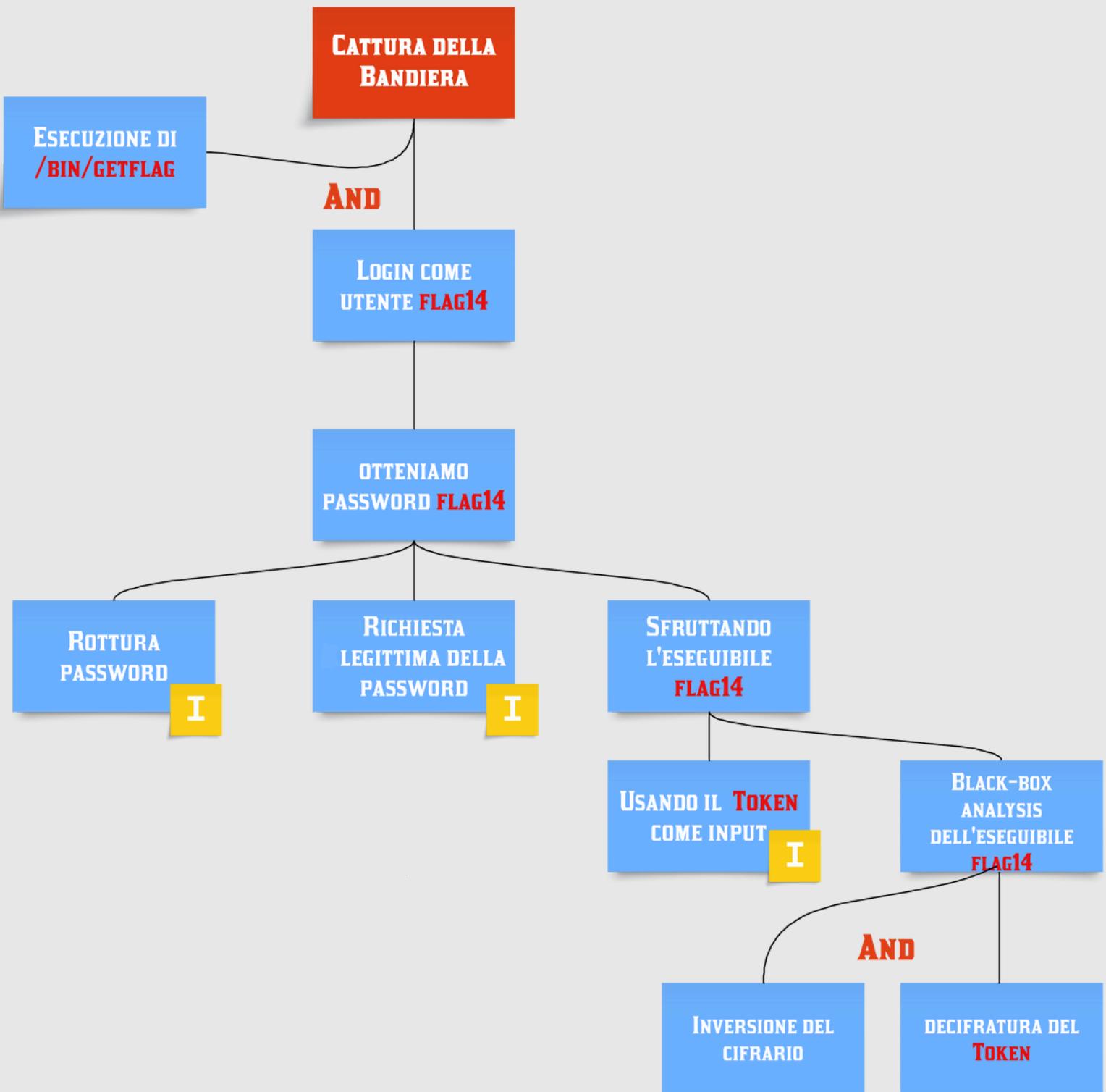
```
def encrypt(string):
    encrypted_str = ""
    for index, char in enumerate(string):
        encrypted_str += chr(ord(char) + index)
    return encrypted_str
```

Per un'ulteriore conferma è stato emulato l'algoritmo ed è stata effettuata una comparazione tra i risultati



Gli output prodotti risultano essere uguali

# ALBERO DI ATTACCO



Aggiorniamo l'albero di attacco con i nuovi nodi, aggiungendo la nuova clausola

# IMPLEMENTAZIONE DELLO SCRIPT

---

Con le conoscenze acquisite possiamo **costruire uno script** basato sulla lettura del file “token” per effettuare la decifratura del suo contenuto.



# IMPLEMENTAZIONE DELLO SCRIPT

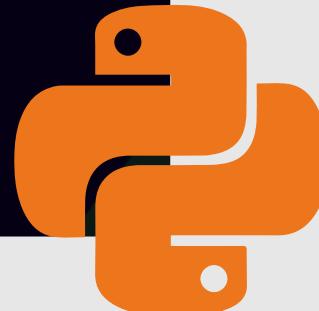
Vengono presentate due soluzioni di script rispettivamente utilizzando i linguaggi

**Python e C**

```
def decrypt(string):
    decripted_string = ''
    for index, char in enumerate(string):
        decripted_string += chr(ord(char) - index)
    return decripted_string

file_path = raw_input("Inserisci path del file:")
try:
    with open(file_path, "r") as file:
        token = file.read().strip()
        print(decrypt(token))
except IOError:
    print("Errore apertura file")
```

decrypt.py



```
#include <stdio.h>

int main(int argc, char *argv[]){
    FILE *file = fopen(argv[1], "r");
    int ch;
    int new_ch;
    int i = 0;

    while((ch = fgetc(file)) != EOF){
        new_ch = ch - i;
        printf("%c", new_ch);
        i +=1;
    }

    fclose(file);
    return 0;
}
```

decrypt.c



# IMPLEMENTAZIONE DELLO SCRIPT

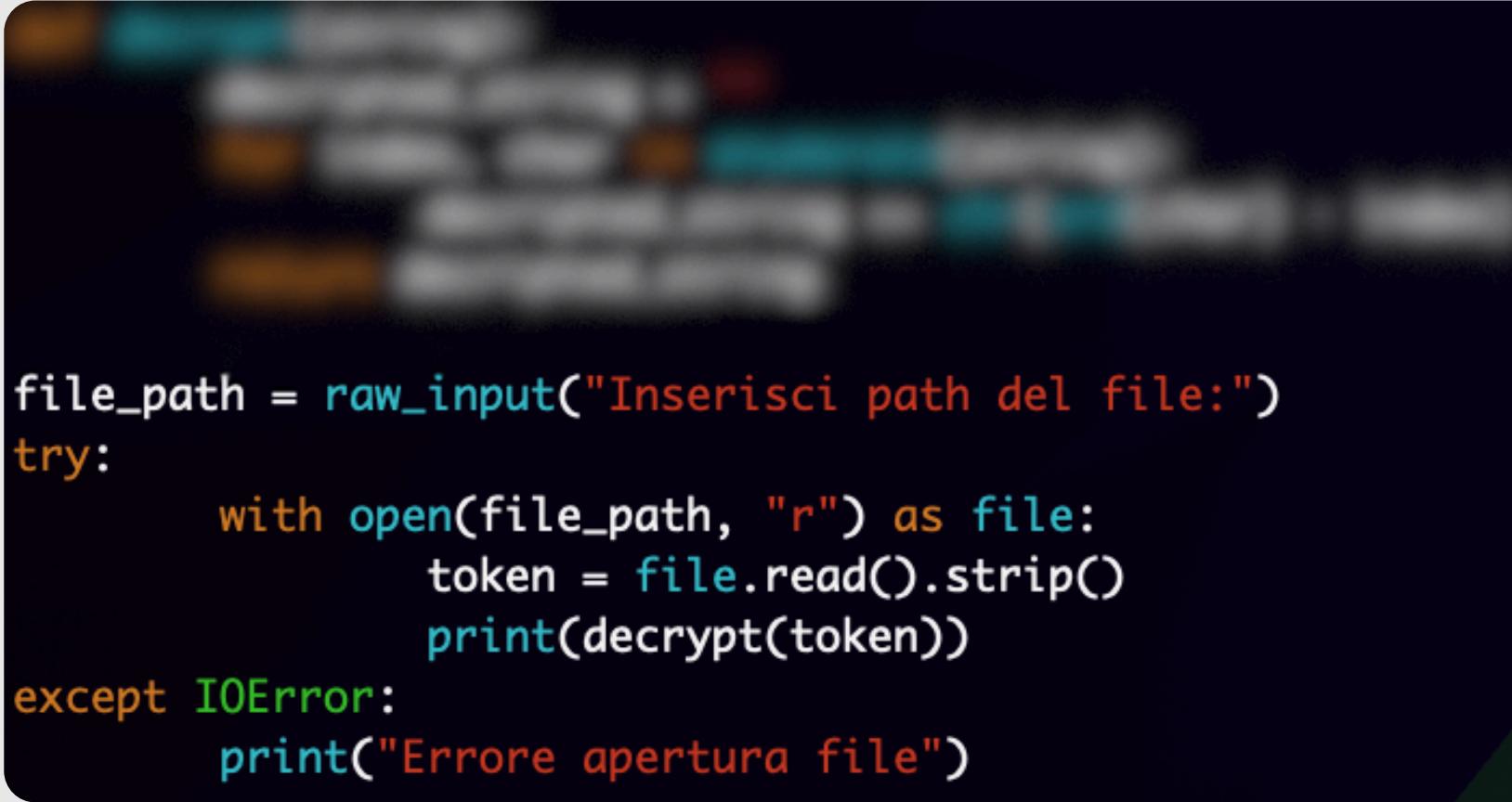
---

```
def decrypt(string):
    decripted_string = ''
    for index, char in enumerate(string):
        decripted_string += chr(ord(char) - index)
    return decripted_string
```

Questa funzione prende in input una stringa cifrata e la "decifra" con una semplice logica basata su ASCII.

- **enumerate(string)** dà in output sia l'indice (index) sia il carattere (char) per ogni carattere della stringa.
- **ord(char)** converte il carattere nel suo codice ASCII.
- **chr(...)** fa l'operazione inversa: da codice ASCII → carattere.
- Viene sottratto l'indice al codice ASCII, rendendo la codifica dipendente dalla posizione del carattere.

# IMPLEMENTAZIONE DELLO SCRIPT



```
file_path = raw_input("Inserisci path del file:")
try:
    with open(file_path, "r") as file:
        token = file.read().strip()
        print(decrypt(token))
except IOError:
    print("Errore apertura file")
```

- `raw_input()` chiede all'utente il percorso del file contenente la stringa da decifrare.
- Tenta di aprire il file specificato.
- `file.read().strip()` legge tutto il contenuto del file e rimuove eventuali spazi bianchi iniziali/finali.
- Il contenuto viene passato alla funzione `decrypt()` e stampato.

# INVERSIONE DEL TOKEN

---

**python decrypt.py**

Mandando in esecuzione lo script creato ed inserendo il path del file “token”, riceviamo come output il suo contenuto decifrato

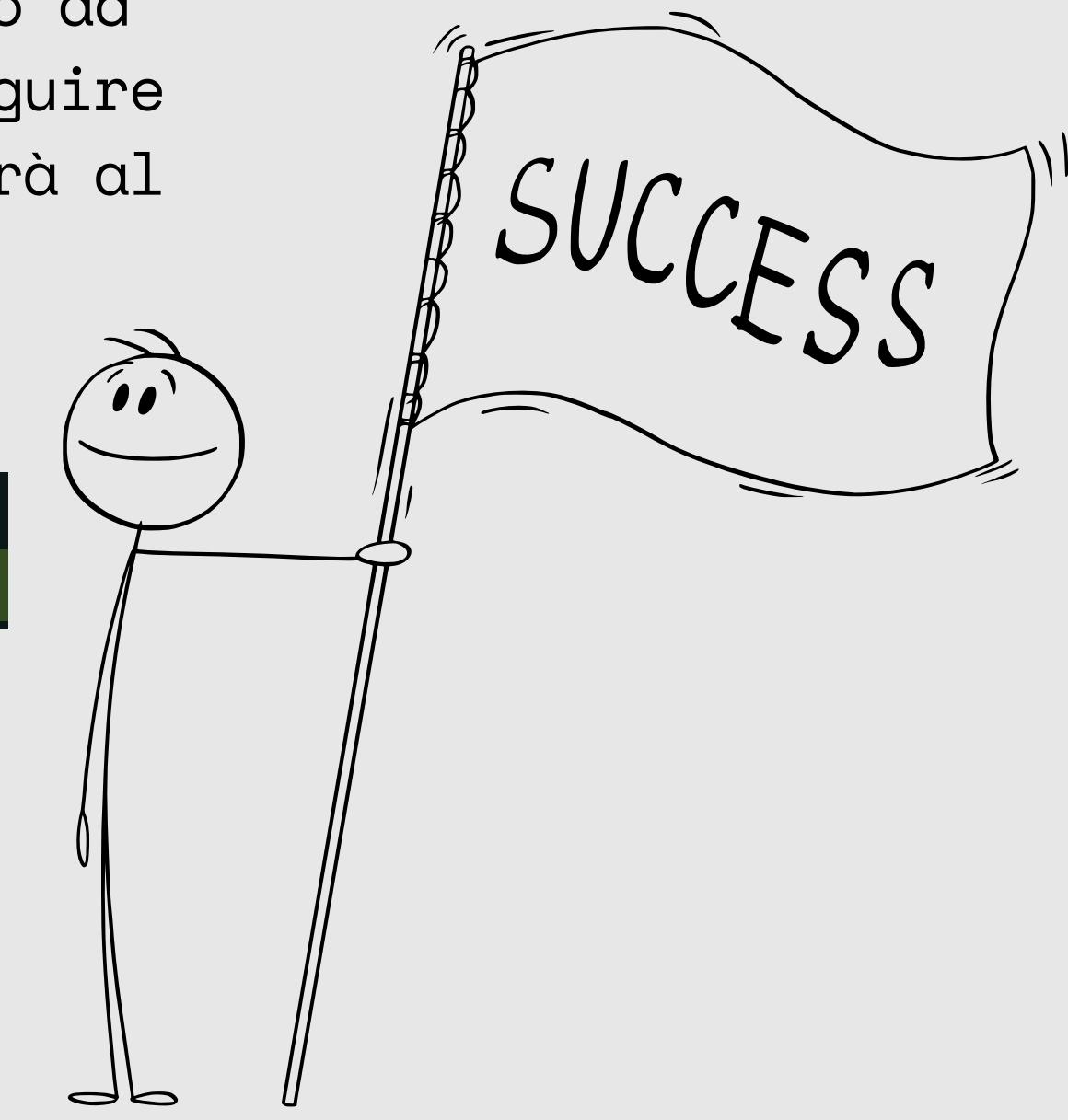
```
[level14@nebula:~$ python script_py.py
Inserisci path del file:/home/flag14/token
8457c118-887c-4e40-a5a6-33a25353165]
```

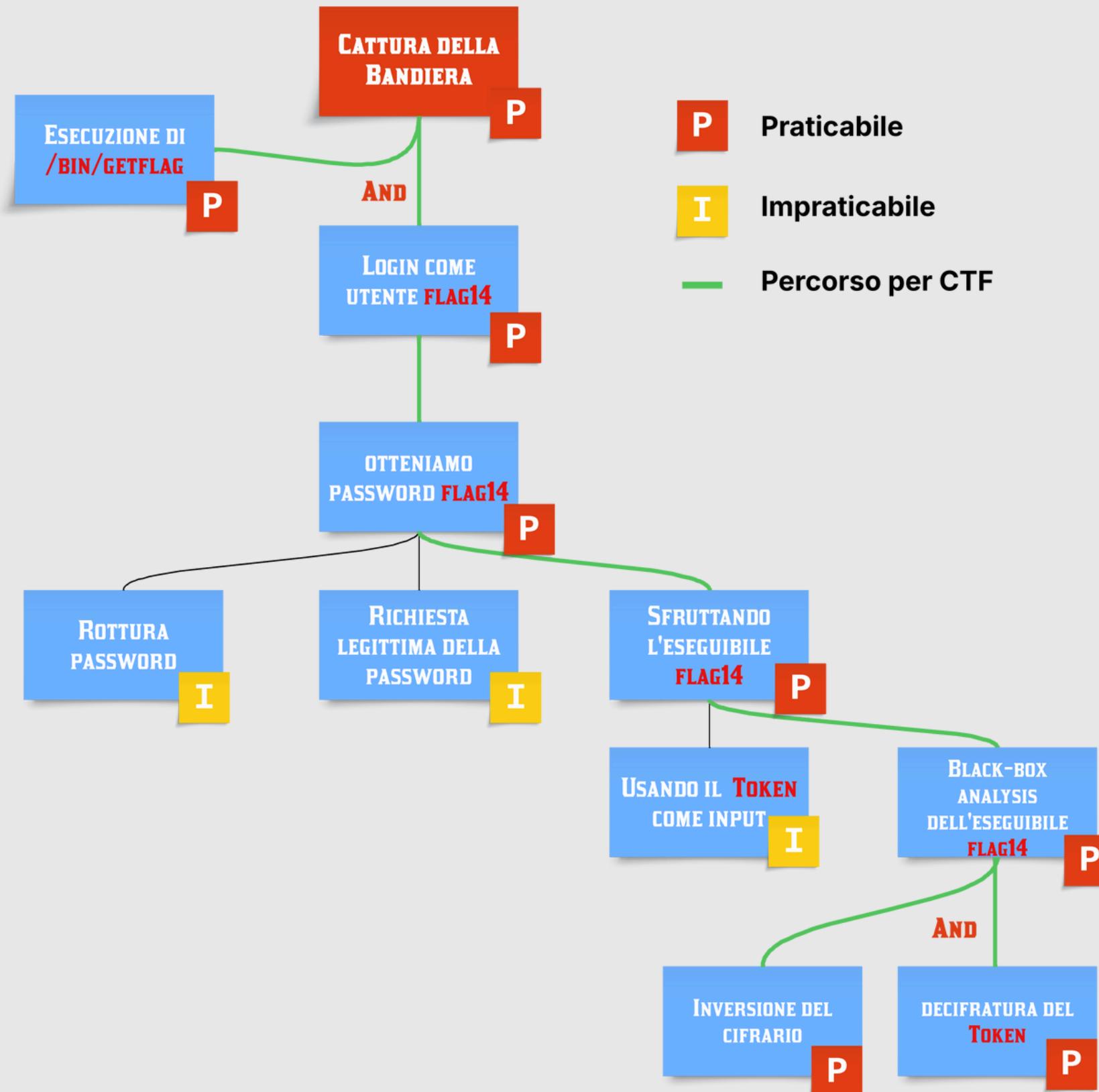
# SUCCESSO DELLA SFIDA

---

Tale output rappresenta la password che andremo ad inserire per autenticarci come **flag14** e per eseguire successivamente il comando **getflag** che ci porterà al compimento della sfida.

```
[flag14@nebula:~$ getflag
You have successfully executed getflag on a target account]
```





# ALBERO DI ATTACCO FINALE

In questa sezione possiamo notare la strada praticabile segnalata come da legenda.

# QUALI SONO LE DEBOLEZZE?

---

- [CWE-272] Least Privilege Violation
- [CWE-276] Incorrect Default Permissions
- [CWE-326] Inadequate Encryption Strength
- [CWE-327] Use of a Broken or Risky Cryptographic Algorithm
- [CWE-328] Use of Weak Hash

CWE-272  
LEAST PRIVILEGE VIOLATION

La versione di Bash usata dalla VM Nebula non presenta l'abbassamento automatico dei privilegi, infatti il nostro file eseguibile **/home/flag14/flag14** risulta avere il bit **setUID** acceso, quindi qualsiasi utente che mandi in esecuzione il file binario **flag14** lo fa con i permessi del proprietario (flag14).

CWE-272  
MITIGAZIONE DEBOLEZZA

Per mitigare il problema, bisognerebbe **aggiornare la versione di bash** usata dalla VM.

Nel nostro livello non è stato fatto per evitare di manomettere la VM ed avere problemi nell'eseguire altre sfide

Per risolvere la **vulnerabilità**, si può direttamente risolvere la debolezza successiva, così da evitare l'aggiornamento della versione, seppur fondamentale in ambienti reali

CWE-276  
INCORRECT DEFAULT PERMISSION

Entrambi i file a nostra disposizione presentano problemi riguardanti i permessi dei vari utenti:

- **/home/flag14/flag14** - l'algoritmo usato risulta essere semplice, quindi l'esecuzione del programma da parte di utenti esterni permette una facile rottura dell'algoritmo
- **/home/flag14/token** - il file può essere letto e scritto dall'utente level14, quindi il token non presenta alcuna protezione

CWE-276  
MITIGAZIONE DEBOLEZZA #1

Limitare l'**esecuzione** del file **/home/flag14/flag14**

**chmod g-x /home/flag14/flag14**

```
root@nebula:~# ls -la /home/flag14/flag14
-rwsr-x--- 1 flag14 level14 7272 2011-12-05 18:59 /home/flag14/flag14
root@nebula:~# chmod g-x /home/flag14/flag14
root@nebula:~# ls -la /home/flag14/flag14
-rwsr----- 1 flag14 level14 7272 2011-12-05 18:59 /home/flag14/flag14
root@nebula:~# su - level14
level14@nebula:~$ ./home/flag14/flag14
-su: ./home/flag14/flag14: Permission denied
level14@nebula:~$ _
```

CWE-276  
MITIGAZIONE DEBOLEZZA #2

Limitare la **lettura e scrittura** del file **token** all'utente flag14

**chown flag14 /home/flag14/token**

```
root@nebula:~# ls -la /home/flag14/token
-rw----- 1 level14 level14 37 2011-12-05 18:59 /home/flag14/token
root@nebula:~# chown flag14 /home/flag14/token
root@nebula:~# ls -la /home/flag14/token
-rw----- 1 flag14 level14 37 2011-12-05 18:59 /home/flag14/token
root@nebula:~# su - level14
level14@nebula:~$ cat /home/flag14/token
cat: /home/flag14/token: Permission denied
level14@nebula:~$ _
```

CWE-326  
INADEQUATE ENCRYPTION STRENGTH

L'attuale algoritmo di cifratura si è dimostrato **vulnerabile** ad attacchi di **forza bruta**, analogamente al metodo di risoluzione da noi impiegato.

<https://cwe.mitre.org/data/definitions/326.html>

CWE-327

## USE OF A BROKEN OR RISKY CRYPTOGRAPHIC ALGORITHM

---

L'algoritmo di cifratura attualmente in uso dall'utente è **obsoleto**. La sua semplicità strutturale lo rende suscettibile a una **rapida compromissione**, consentendo a potenziali aggressori di decifrare le informazioni protette con uno sforzo computazionale minimo.

CWE-328  
USE OF WEAK HASH

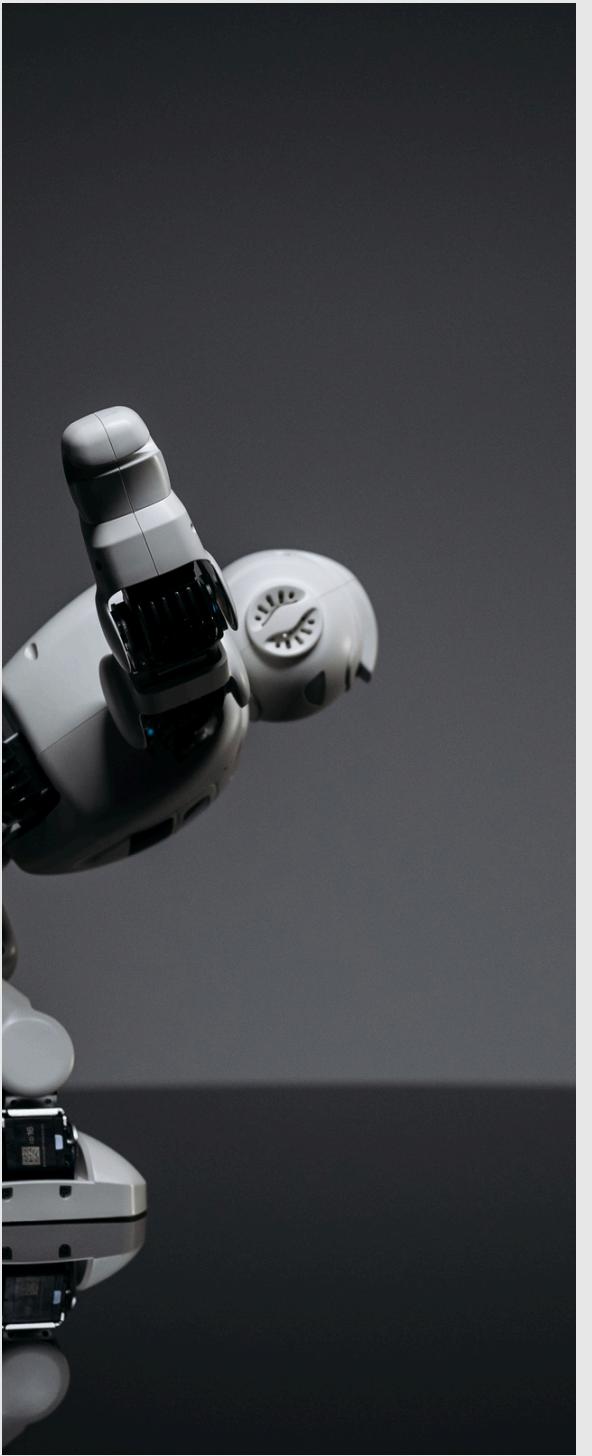
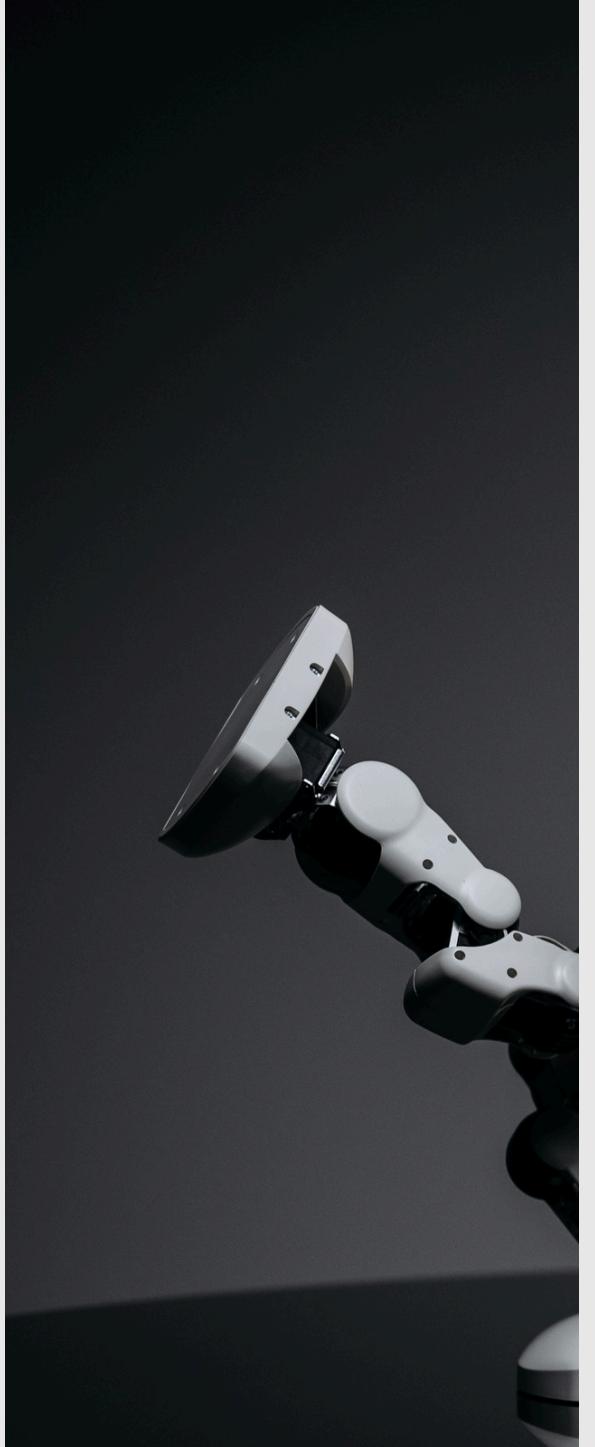
L'algoritmo di cifratura risulta essere un algoritmo **deterministico**, ovvero, dato un input da parte dell'utente, l'output risulta essere sempre lo stesso, rendendo molto debole la sicurezza della cifratura adottata

## CWE-326 CWE-327 CWE-328 MITIGAZIONE DEBOLEZZA

La **mitigazione** delle debolezze risulta essere la medesima. L'utente flag14 per un'ulteriore protezione della propria password può migrare a sistemi **crittografici moderni** e **resilienti** quali:

- **AES** - Advanced Encryption Standard
- **ChaCha20-Poly1305**
- **RSA** - Rivest-Shamir-Adleman
- **ECC** - Elliptic Curve Cryptography

\* NEBULA  
LEVEL 14



---

THANK  
YOU