Analysis of the Keeper Machine

Ali BA FAQAS

March 10, 2024

1 Preliminary Note

The analysis of this machine was conducted prior to its retirement, hence no external write-ups were consulted.

2 about keeper

Keeper, a Linux machine of easy difficulty, incorporates a support ticketing system with default credentials. By enumerating this service, we can view plaintext credentials that provide SSH access. This SSH access allows us to obtain a KeePass database dump file and extract the master password. Once we have access to the KeePass database, we can retrieve the root SSH keys, enabling us to establish a privileged shell on the host.

3 Vulnerability Discovery

3.1 Network Mapping (Nmap)

Our initial step, as always, is to scan for open ports and services. We utilized the -sV option to identify the software versions associated with the open ports. The results are as follows:

```
┌──(aloosh㉿kali)-[~/Desktop/S2/SE/machines]
└─$ sudo nmap -sV 10.10.11.227
[sudo] password for aloosh:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-02-05 21:00 CET
Nmap scan report for 10.10.11.227
Host is up (0.12s latency).
Not shown: 997 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.3 (Ubuntu Linux; protocol 2.0)
80/tcp   open  http    nginx 1.18.0 (Ubuntu)
8080/tcp open  http    SimpleHTTPServer 0.6 (Python 3.10.12)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.40 seconds
```

Figure 1: Nmap Scan Results.

The scan revealed three open ports:

1. SSH (Port 22).
2. HTTP (Port 80).
3. HTTP (Port 8080).

Given that we lack the necessary credentials for SSH (Port 22), we decided to focus on HTTP (Port 80).

Upon accessing the IP address via a browser on Port 80, we discovered a single link redirecting to tickets.keeper.htb/rt.

Figure 2: Main Page.

Subsequently, we added the IP to our /etc/hosts file, associating it with both keeper.htb and tickets.keeper.htb. Clicking on the aforementioned link led us to a login page.

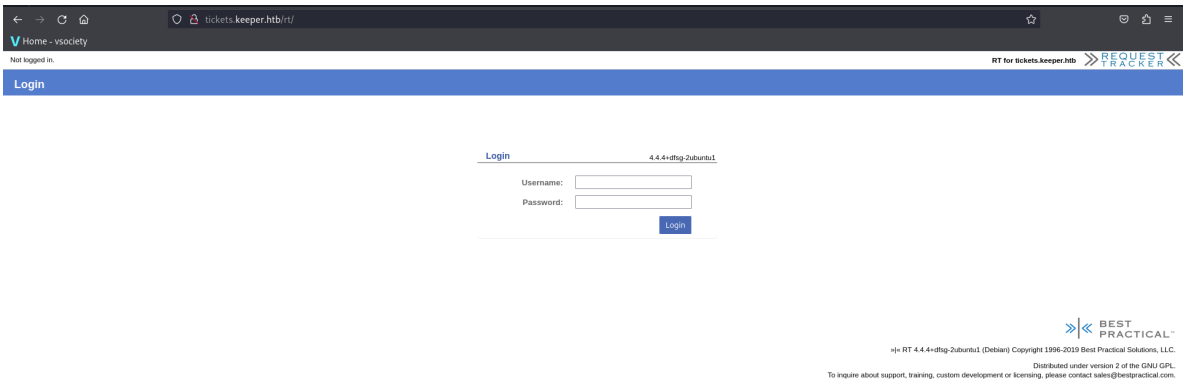

Figure 3: Login Page.

3.2 Gaining Access

We identified a ticketing system called RT, developed by Best Practical Solutions. This system is commonly used by businesses to manage tasks, issues, jobs, or other types of requests reported by their customers or users.

Before delving into more complex strategies, we attempted to use the default username and password for the RT system.

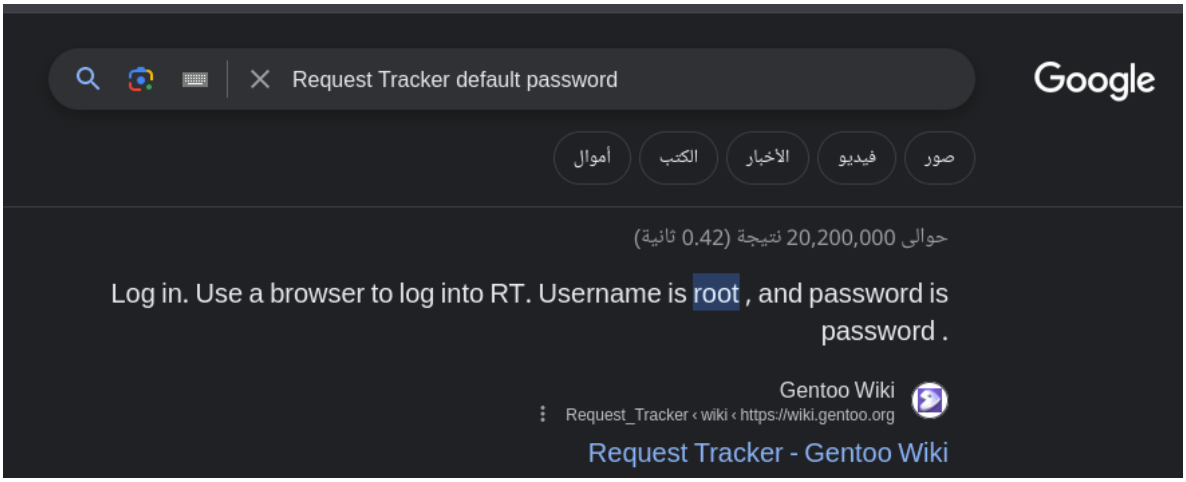

Figure 4: Default RT System Credentials.

To our surprise, these credentials were successful, and we gained root access to the website.

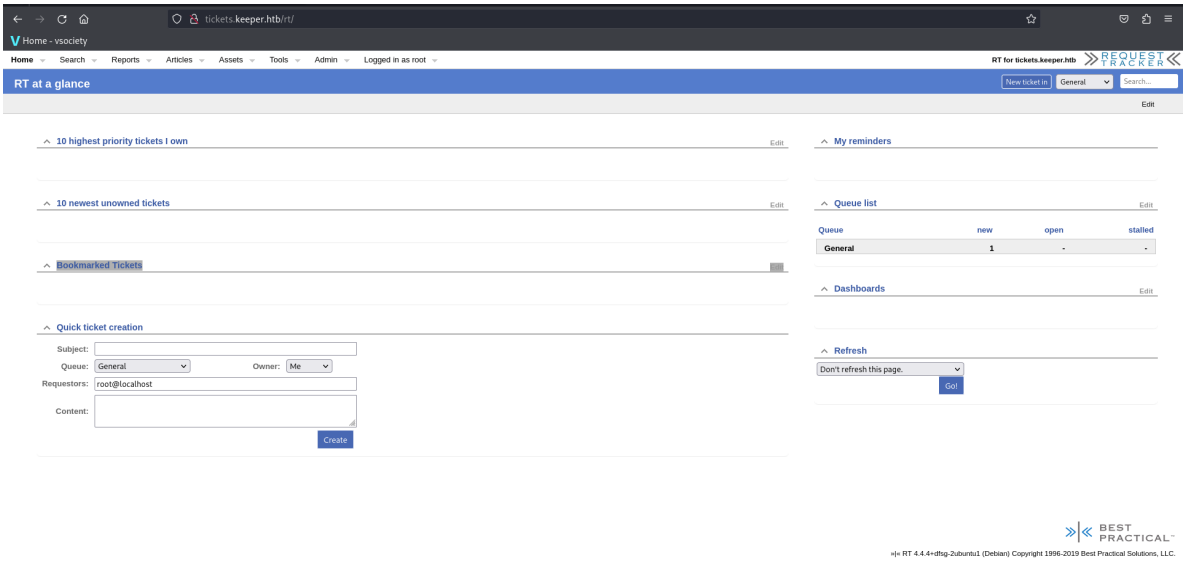

Figure 5: Successful Login.

Upon further exploration of the website, we discovered a page where we could modify admin permissions. Notably, some permissions pertained to scripts, which piqued our interest.

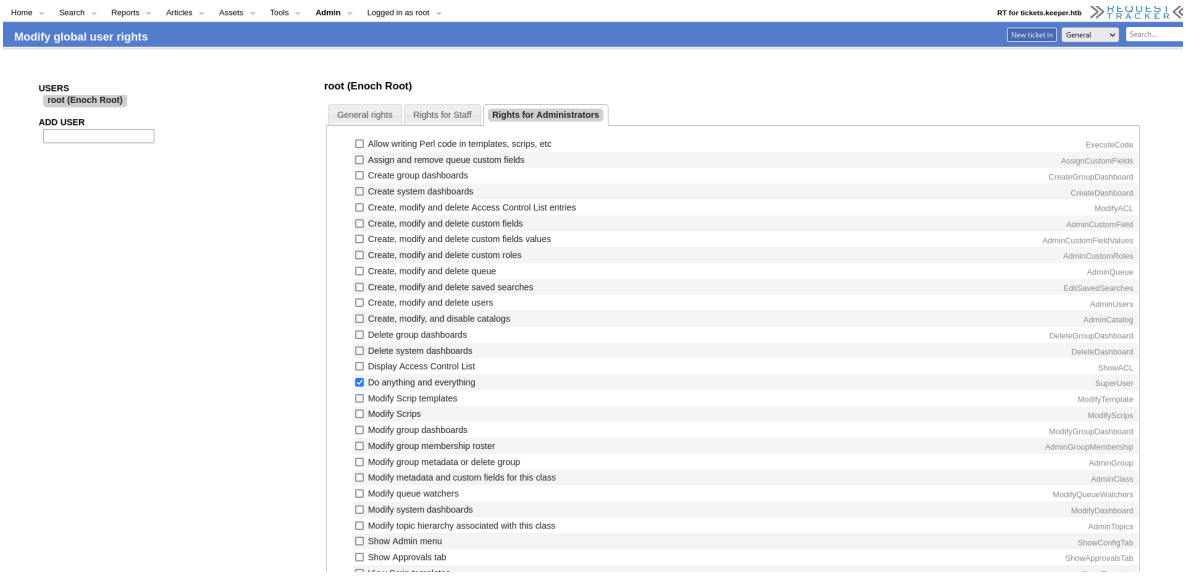

Figure 6: Attempt to Modify Admin Permissions.

Unfortunately, our attempt to modify the permissions was denied by the system, which suspected malicious activity. Thus, this approach was unfeasible.

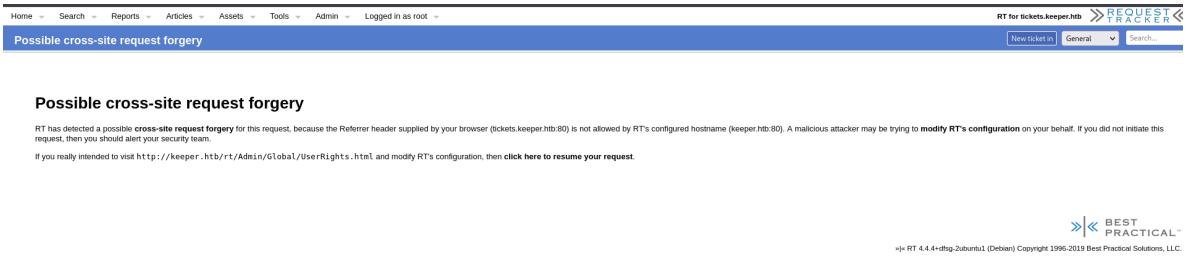

Figure 7: Failed Attempt to Modify Admin Permissions.

Continuing our search for exploitable elements, we found a user named lnorgaard on the users page. Upon opening their page, we discovered the password "welcome2023!" written in the comments section.

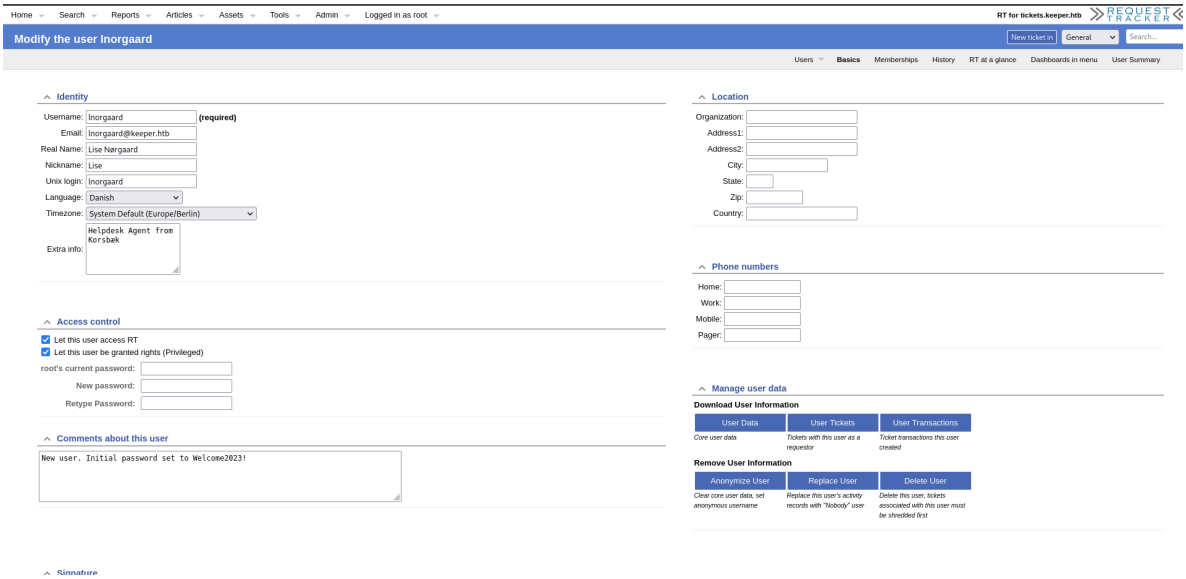

Figure 8: User Page with Discovered Password.

3.3 User Flag

Given that the SSH port was identified as open from the Nmap scan, we proceeded to log in using the credentials of the user lnorgaard.

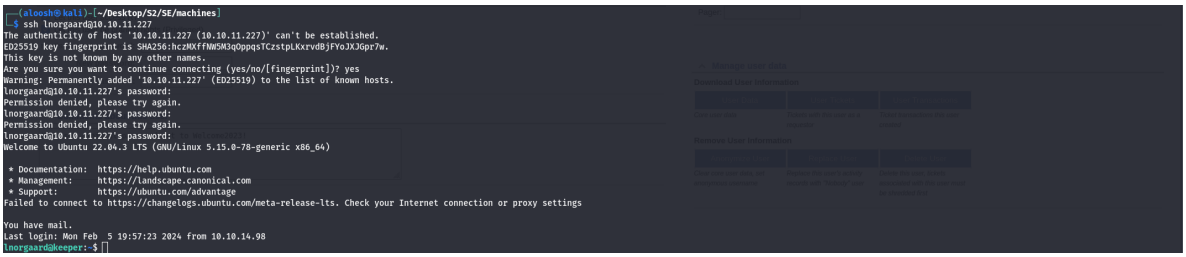

Figure 9: Establishing an SSH Connection.

We found the user flag conveniently located in the home directory of the user.

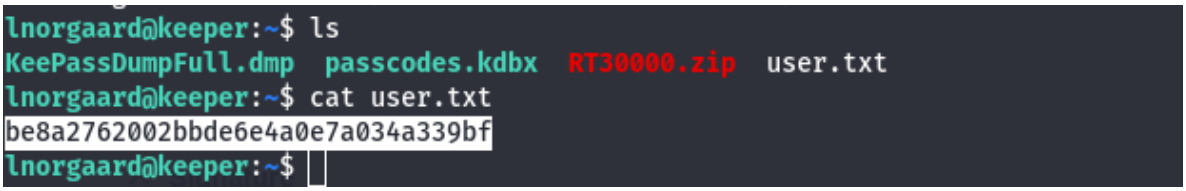

Figure 10: Retrieved User Flag.

3.4 Root Flag

In our attempt to escalate privileges, we first checked lnorgaard's sudo privileges. However, lnorgaard was not a member of the sudo group.

```
lnorgaard@keeper:~$ id
uid=1000(lnorgaard) gid=1000(lnorgaard) groups=1000(lnorgaard)
lnorgaard@keeper:~$ sudo -l
[sudo] password for lnorgaard:
Sorry, user lnorgaard may not run sudo on keeper.
lnorgaard@keeper:~$
```

Figure 11: Checking lnorgaard's Sudo Privileges.

Despite exploring other potential avenues for privilege escalation, such as listing all processes with setuid, we were unsuccessful.

Our attention then turned to a zip file in lnorgaard's home directory.

```
lnorgaard@keeper:~$ mkdir zipped
lnorgaard@keeper:~$ ls
KeePassDumpFull.dmp  passcodes.kdbx  RT30000.zip  user.txt  zipped
lnorgaard@keeper:~$ cd zipped/
lnorgaard@keeper:~/zipped$ unzip ../RT30000.zip
Archive:  ../RT30000.zip
  inflating: KeePassDumpFull.dmp
 extracting: passcodes.kdbx
lnorgaard@keeper:~/zipped$ ls
KeePassDumpFull.dmp  passcodes.kdbx
lnorgaard@keeper:~/zipped$
```

Figure 12: Unzipping the File.

Interestingly, the unzipped contents included a .kdbx file, a password database created by KeePass Password Safe. This free password manager stores an encrypted database of passwords, which can only be accessed using a user-set master password.

To access this encrypted database, we needed the master key. This is where the .dmp file came into play. A .dmp file, or dump file, is a memory image created by Windows when a device crashes. In the context of a .dmp file, it's theoretically possible that sensitive information such as passwords could be present if they were in memory at the time of the crash.

In our research, we also discovered a relevant CVE about KeePass.

The vulnerability CVE-2023-32784, affecting KeePass 2.x versions prior to 2.54, allows an attacker to recover the cleartext master password from various types of memory dumps.

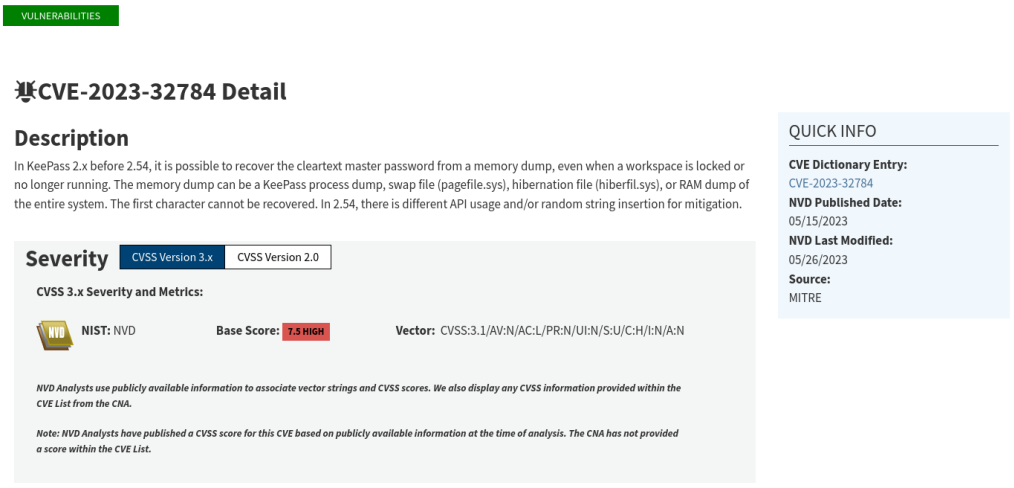

Figure 13: KeePass CVE.

With the goal of finding the master key in the .dmp file, we transferred the zip file to our local machine and used a POC code found on GitHub. This program attempts to find all possible master keys within the file.

Figure 14: Searching for the Master Key.

The result was incomplete, "**dgr*d med fl*de", missing some letters. However, by taking the last possible password found by the program and searching it on Google, we discovered a Danish meal called Rødgrød med fløde. Given that lnorgaard's user profile indicated he is Danish, we hypothesized that the program couldn't find the letter ø because it's a special character.

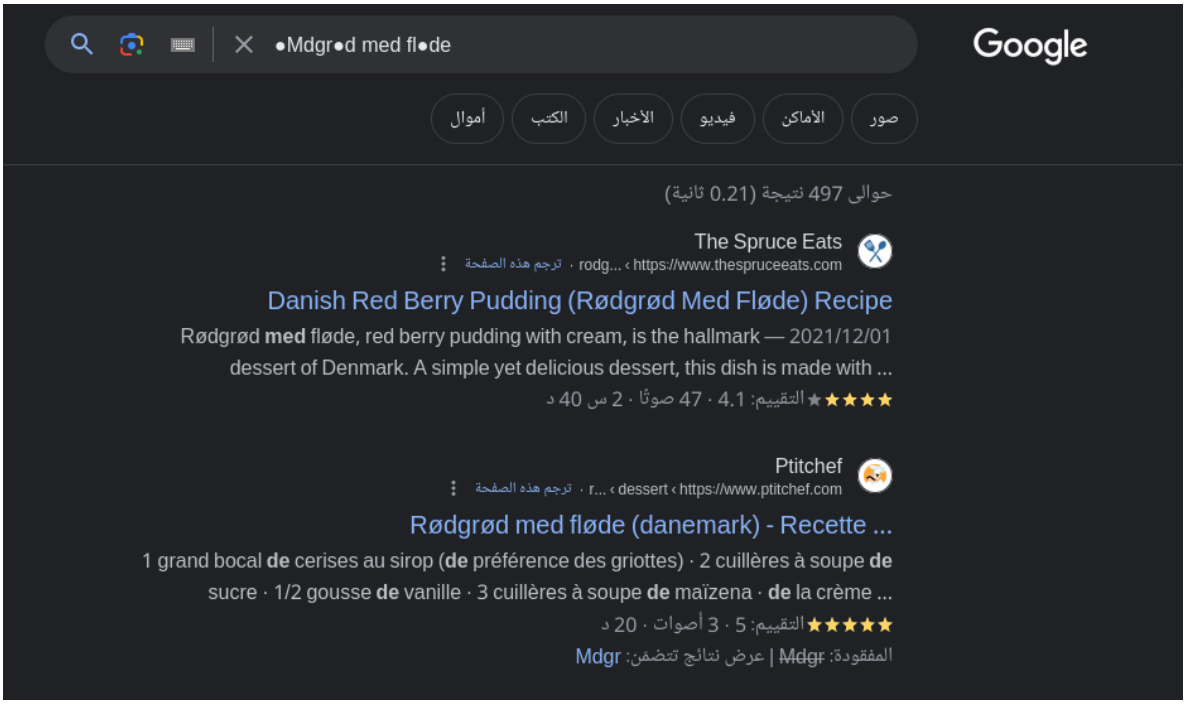

Figure 15: Identifying the Master Key.

We used the web-based KeePass KeeWeb platform to open the password manager, using Rødgrød med fløde as the master key.

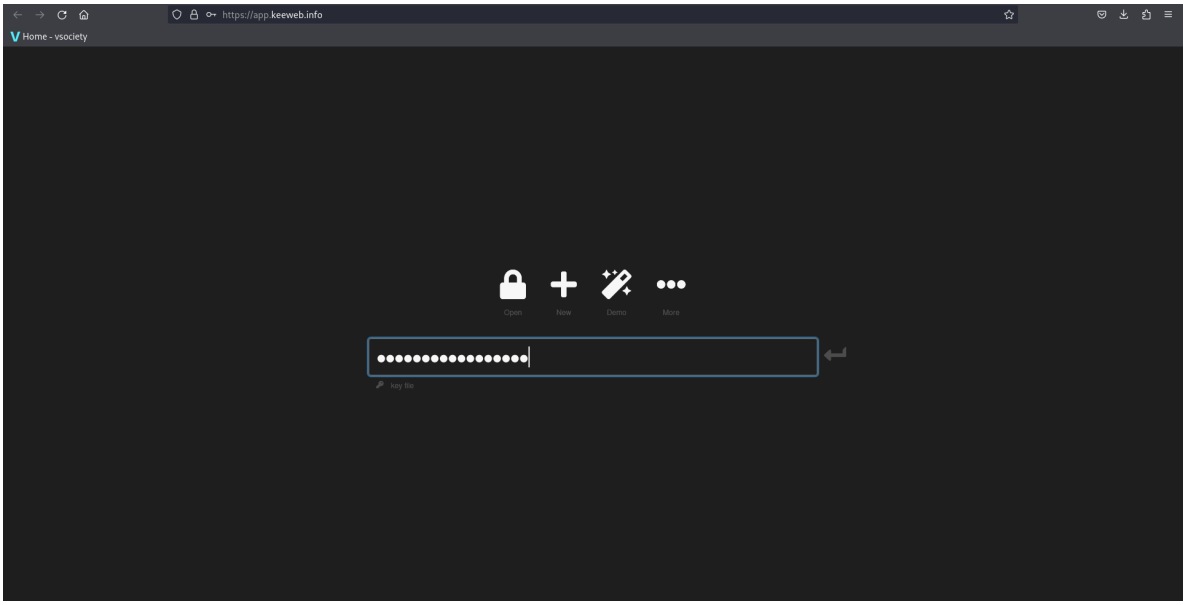

Figure 16: Accessing KeeWeb Platform.

Fortunately, the master key was correct, and we were able to access the contents of a PuTTY PPK file for the root user.

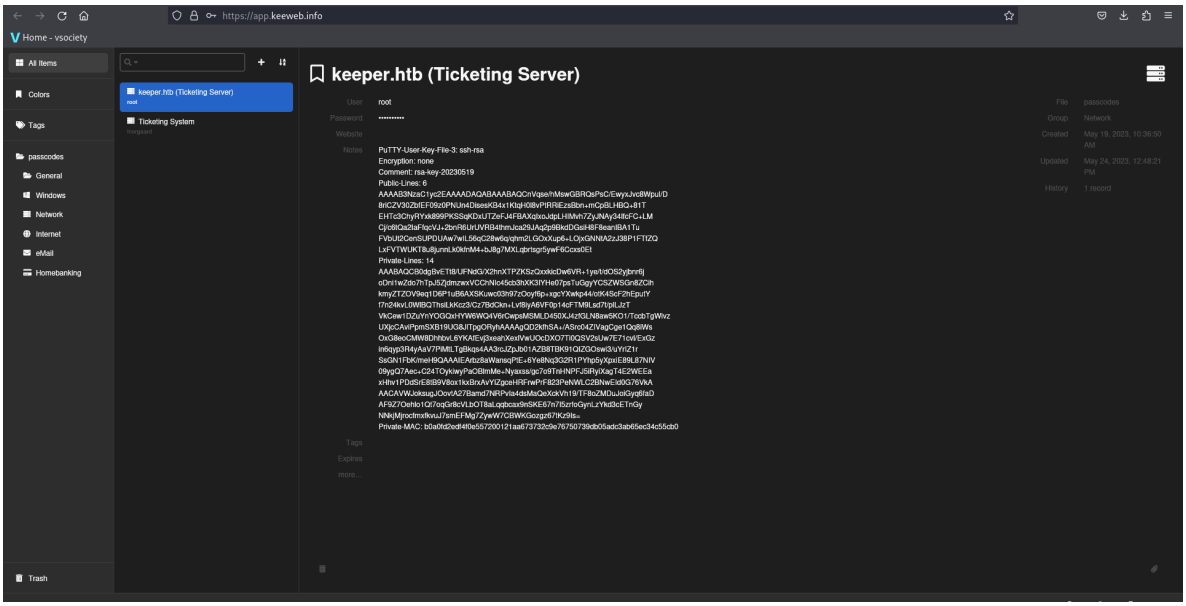

Figure 17: Accessing PuTTY PPK File.

With the help of puttygen, we were able to convert the PPK to a .pem SSH private key. This allowed us to SSH into the machine as root, marking the end of our journey with the capture of the root flag:

```
$ puttygen key.ppk -O private-openssh -o root.pem
```

Figure 18: Generation of root.pem.

With the .pem file ready, we were prepared to establish an SSH connection as follows:

```
$ ssh  root@keeper.htb -i root.pem
```

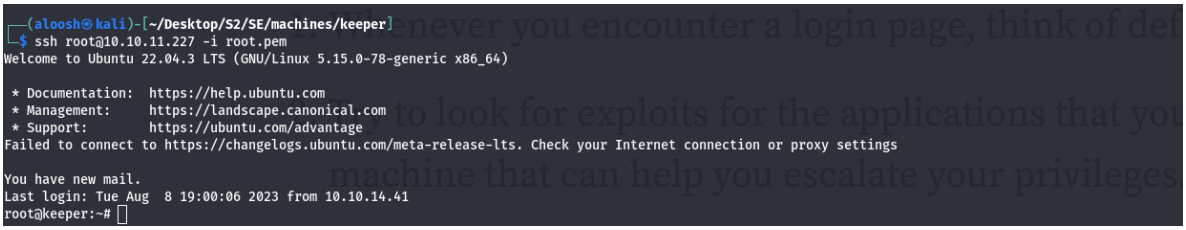

Figure 19: Establishing SSH Connection as Root.

As shown in the figure, we successfully gained root access and were able to open the root.txt file located in the root home directory.

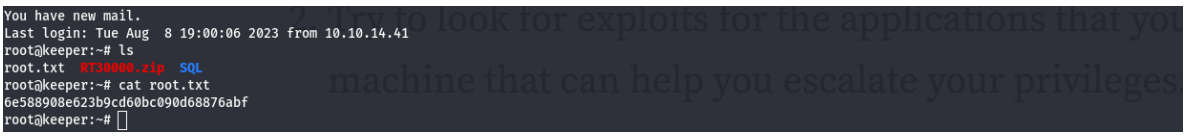

Figure 20: Retrieved Root Flag.