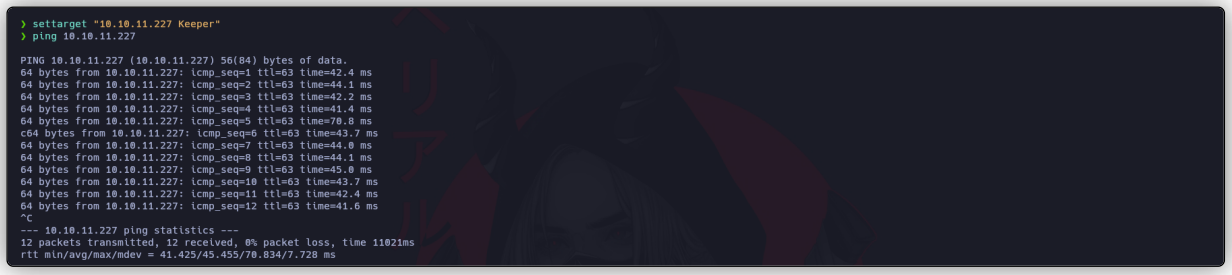
243- KEEPER

1. KEEPER

https://app.hackthebox.com/machines/Keeper

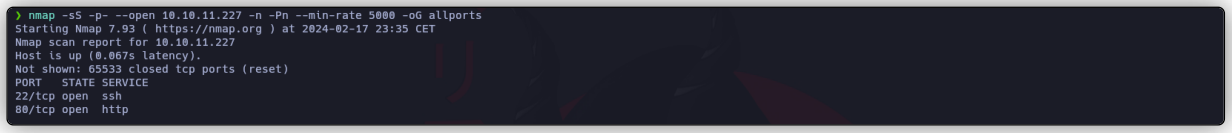
1.1. Preliminar

- Comprobamos si la máquina está encendida, averiguamos qué sistema operativo es y creamos nuestro directorio de trabajo. Parece que nos enfrentamos a una máquina *Linux*.

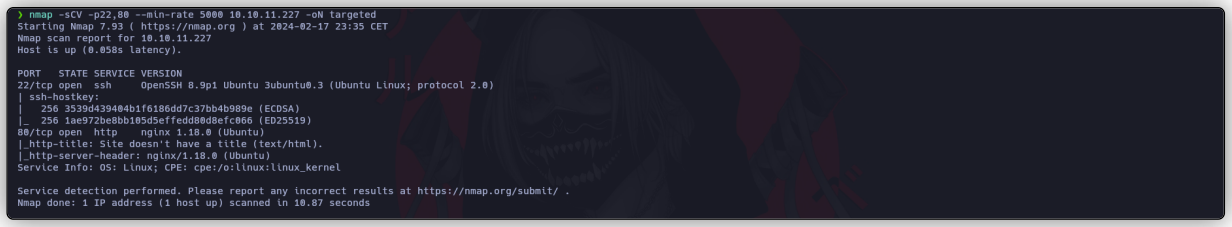
- 

```
❯ settarget "10.10.11.227 Keeper"
❯ ping 10.10.11.227

PING 10.10.11.227 (10.10.11.227) 56(84) bytes of data.
64 bytes from 10.10.11.227: icmp_seq=1 ttl=63 time=42.4 ms
64 bytes from 10.10.11.227: icmp_seq=2 ttl=63 time=44.1 ms
64 bytes from 10.10.11.227: icmp_seq=3 ttl=63 time=42.2 ms
64 bytes from 10.10.11.227: icmp_seq=4 ttl=63 time=41.4 ms
64 bytes from 10.10.11.227: icmp_seq=5 ttl=63 time=70.8 ms
c64 bytes from 10.10.11.227: icmp_seq=6 ttl=63 time=43.7 ms
64 bytes from 10.10.11.227: icmp_seq=7 ttl=63 time=44.0 ms
64 bytes from 10.10.11.227: icmp_seq=8 ttl=63 time=44.1 ms
64 bytes from 10.10.11.227: icmp_seq=9 ttl=63 time=45.0 ms
64 bytes from 10.10.11.227: icmp_seq=10 ttl=63 time=43.7 ms
64 bytes from 10.10.11.227: icmp_seq=11 ttl=63 time=42.4 ms
64 bytes from 10.10.11.227: icmp_seq=12 ttl=63 time=41.6 ms
^C
--- 10.10.11.227 ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 11021ms
rtt min/avg/max/mdev = 41.425/45.455/70.834/7.728 ms
```

1.2. Nmap

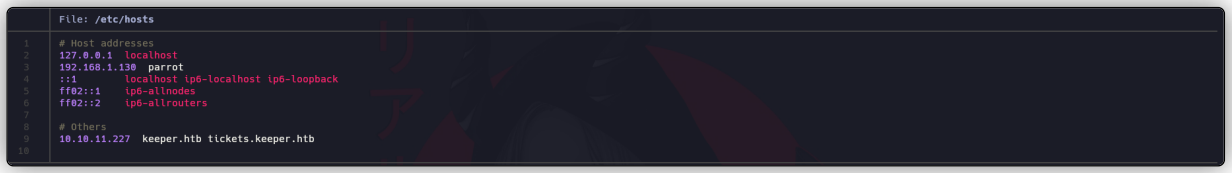
- Escaneo de puertos sigiloso. Evidencia en archivo *allports*. Solo tenemos los *puertos 22 y 80* abiertos.
  - 

```
❯ nmap -sS -p- --open 10.10.11.227 -n -Pn --min-rate 5000 -oG allports
Starting Nmap 7.93 ( https://nmap.org ) at 2024-02-17 23:35 CET
Nmap scan report for 10.10.11.227
Host is up (0.067s latency).
Not shown: 65533 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
```

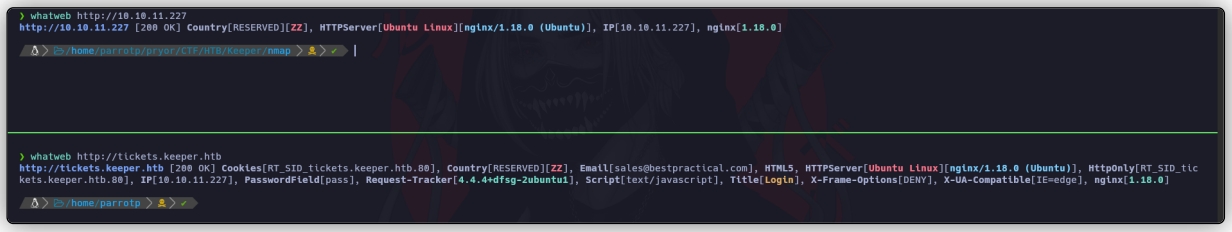
- Escaneo de scripts por defecto y versiones sobre los puertos abiertos, tomando como input los puertos de *allports* mediante `extractPorts`.
  - 

```
❯ nmap -sCV -p22,80 --min-rate 5000 10.10.11.227 -oN targeted
Starting Nmap 7.93 ( https://nmap.org ) at 2024-02-17 23:35 CET
Nmap scan report for 10.10.11.227
Host is up (0.058s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 3539d439404b1f6186dd7c37bb4b989e (ECDSA)
|_  256 1ae972be8bb105d5effedd80d8efc066 (ED25519)
80/tcp open  http    nginx 1.18.0 (Ubuntu)
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: nginx/1.18.0 (Ubuntu)
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 10.87 seconds
```

1.3. Tecnologías web

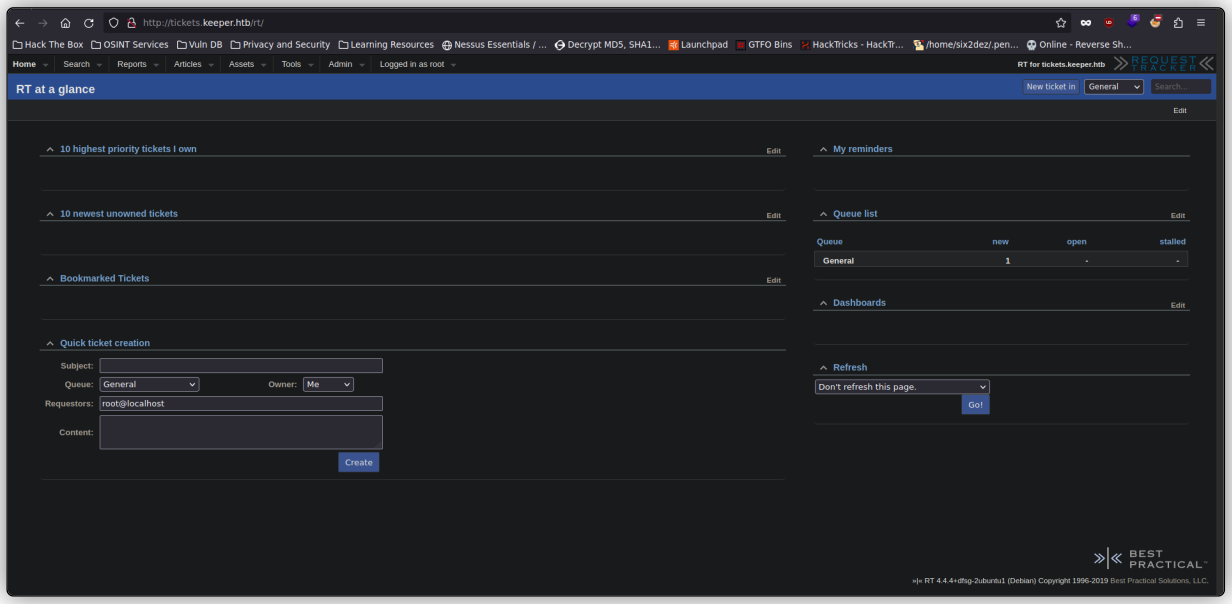
- Cuando visitamos la web, ésta nos redirige a *"tickets.keeper.htb/rt"*. Así que añadimos este dominio a nuestro /etc/hosts.
  - 

```
File: /etc/hosts

# Host addresses
127.0.0.1  localhost
192.168.1.130  parrot
::1        localhost ip6-localhost ip6-loopback
ff02::1    ip6-allnodes
ff02::2    ip6-allrouters

# Others
10.10.11.227  keeper.htb tickets.keeper.htb
```

- **Whatweb**: nos reporta lo siguiente. Entre otra información, vemos un correo electrónico. Vemos que se está usando un servicio llamado *Request Tracker*.
  - 

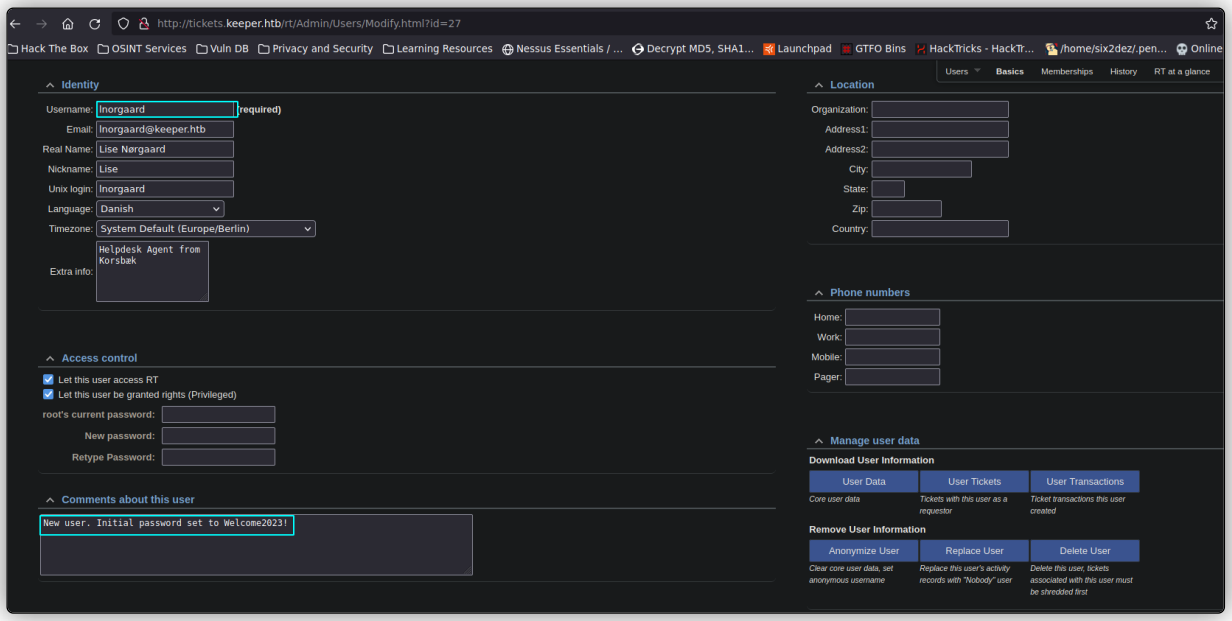
```
❯ whatweb http://10.10.11.227
http://10.10.11.227 [200 OK] Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][nginx/1.18.0 (Ubuntu)], IP[10.10.11.227], nginx[1.18.0]

❯ whatweb http://tickets.keeper.htb
http://tickets.keeper.htb [200 OK] Cookies[RT_SID_tickets.keeper.htb.80], Country[RESERVED][ZZ], Email[sales@bestpractical.com], HTML5, HTTPServer[Ubuntu Linux][nginx/1.18.0 (Ubuntu)], HttpOnly[RT_SID_tickets.keeper.htb.80], IP[10.10.11.227], PasswordField[pass], Request-Tracker[4.4.4+dfsg-2ubuntu1], Script[text/javascript], Title[Login], X-Frame-Options[DENY], X-UA-Compatible[IE=edge], nginx[1.18.0]
```

1.4. Default web service credentials

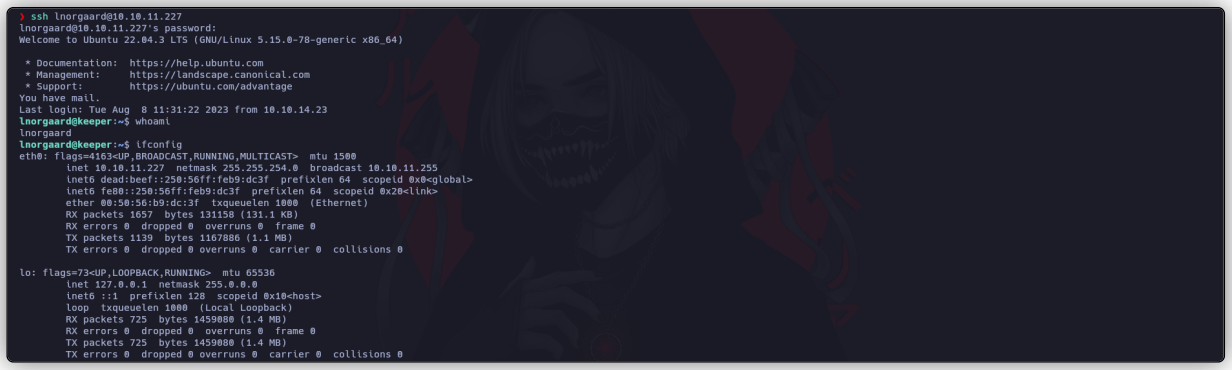
- Accedemos a *"tickets.keeper.htb/rt"* y nos encontramos con un panel de login. Buscamos **credenciales por defecto** del servicio *Request Tracker*, el cual detectamos anteriormente con **Whatweb**. Este servicio es simplemente un sistema de emisión de incidencias y seguimiento de problemas. Finalmente, encontramos unas credenciales que usamos para conseguir acceso, siendo el usuario *root* y contraseña *password*. En la siguiente imagen, podemos ver que tenemos acceso a lo que parece ser un panel administrativo.
    - 

1.5. Leaked SSH credentials

- Investigando la página web, vamos a una sección /Users, en la que, seguidamente, encontramos unas credenciales para un tal *Inorgaard*, con contraseña *Welcome2023!*.

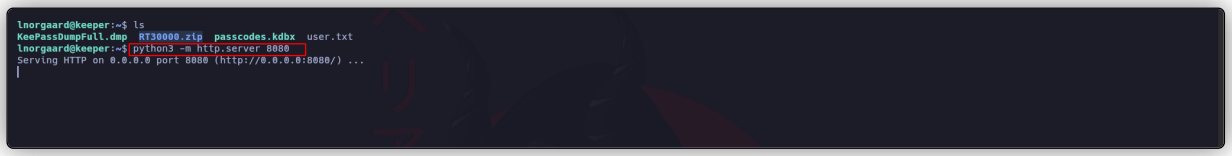


- Usamos estas credenciales para conectarnos por SSH. Conseguimos acceso.

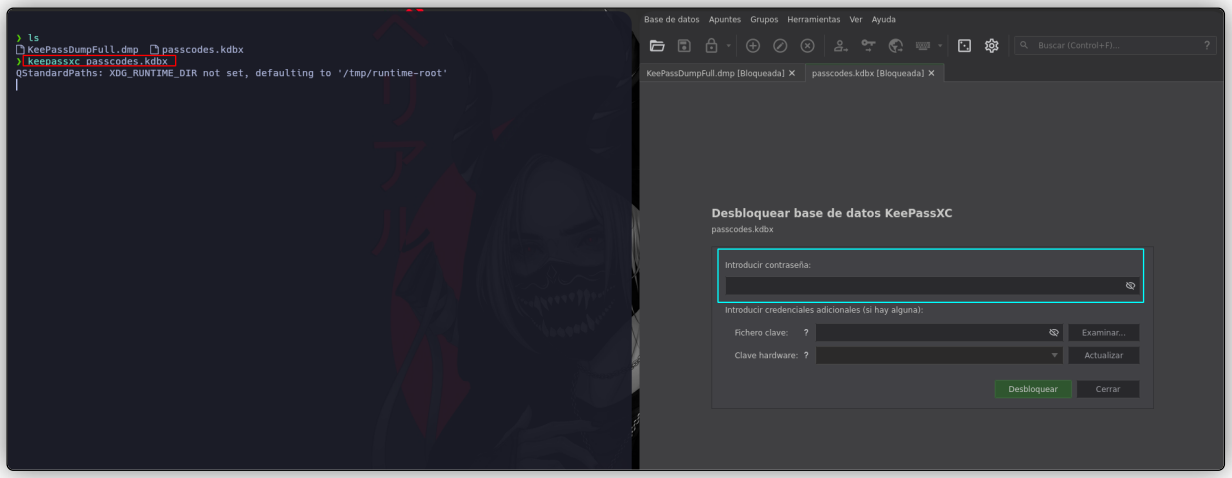


1.6. KeePass master key cracking via dump file

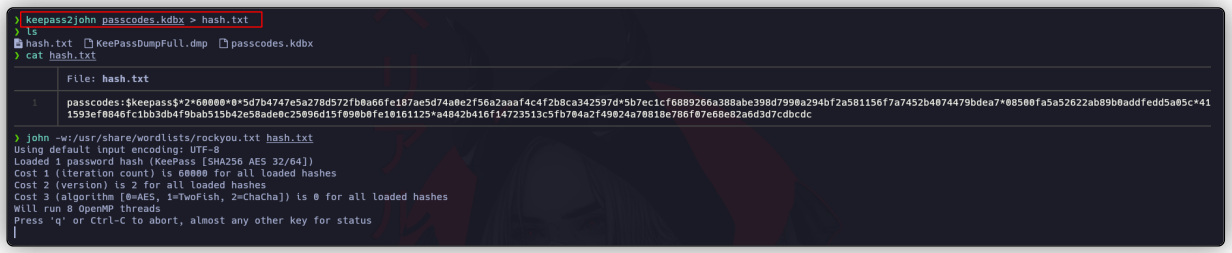
- CVE-2023-32784:
- En el directorio /home de la máquina encontramos un archivo *.zip* que descomprimimos a continuación con: `unzip RT30000.zip`. Al descomprimirlo, obtenemos dos archivos, uno *.dmp* y otro *.kdbx*. Parece que se trata de un dumpeo de una base de datos de KeePass. Nos abrimos un servidor con Python para descargarlos desde nuestra máquina de atacante.



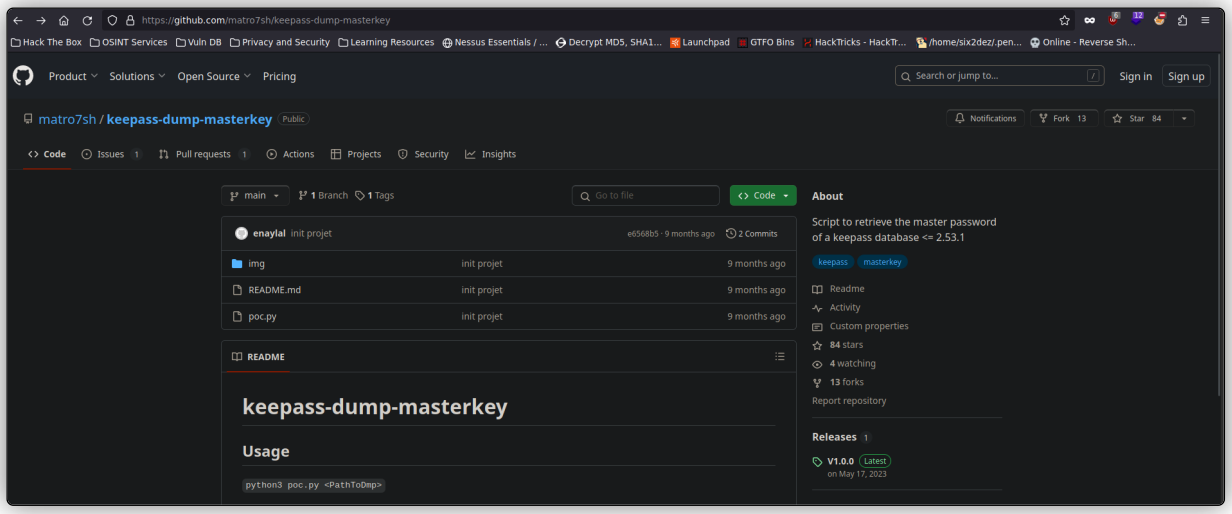
- Nos descargamos estos archivos con `wget`. Para tratar de descifrar el archivo *.kdbx*, vamos a usar la herramienta KeePassXC con `keepassxc passcodes.kdbx`. Necesitamos una contraseña maestra para este archivo, la cual de momento no tenemos. Esta contraseña maestra lo que hace es cifrar la base de datos de contraseñas de KeePass. De momento, intentaremos ver el otro archivo *.dmp*. No obstante, este archivo nos lo interpreta como binario, y es demasiado grande como para que podamos sacar algo en claro.



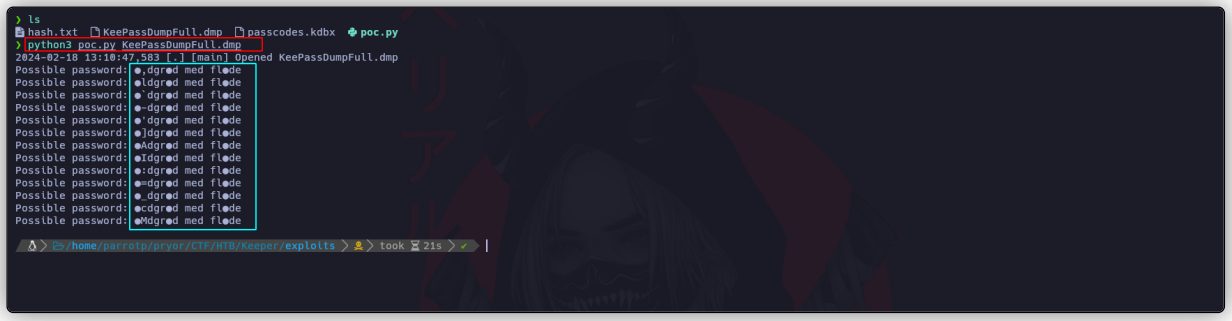
- En este punto, recurrimos a Keepass2john para extraer el hash del archivo *passcodex.kdbx*: `keepass2john passcodes.kdbx > hash.txt`. Esto realiza una conversión de *.kdbx* al formato aceptado por John the Ripper, con la finalidad de crackear este hash. Tratamos de romperlo hash a continuación, pero no tenemos éxito.



- Tratamos de buscar más información por internet sobre cómo podemos extraer en texto claro la contraseña maestro de un archivo *.kdbx*. Encontramos un exploit para KeePass, el cual extrae la contraseña maestra desde la memoria de la aplicación. Esto nos permite que, al comprometer el dispositivo, recuperemos la contraseña incluso cuando la base de datos está bloqueada. Para esta vulnerabilidad, no se requiere ejecución de código en el sistema objetivo, sino tan solo un volcado de memoria. Asimismo, se nos comparte una herramienta para realizar esta explotación, la cual clonamos en nuestro directorio de trabajo. Compartimos este exploit a continuación.
  - https://github.com/matro7sh/keepass-dump-masterkey



- Para usar esta herramienta, tenemos que pasarle como parámetro el archivo *.dmp*, es decir, el dumpeo de memoria. Por tanto, ejecutamos este exploit con `python3 poc.py KeePassDumpFull.dmp`. Obtenemos posibles contraseñas, pero éstas muestran caracteres no imprimibles.

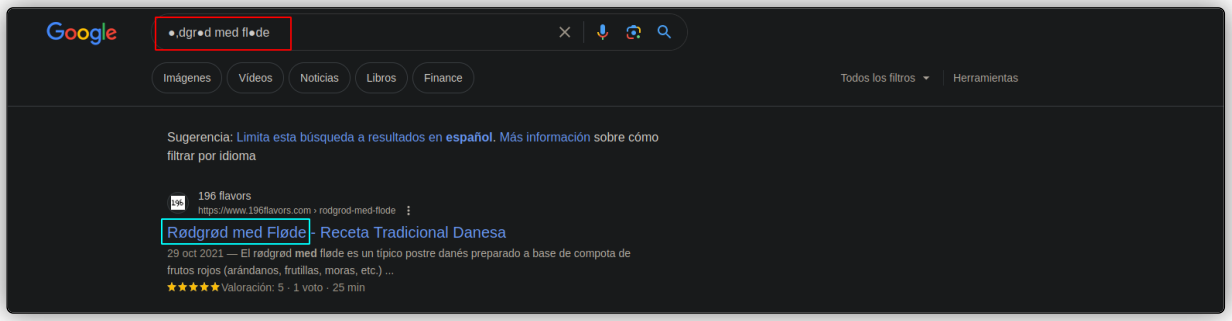


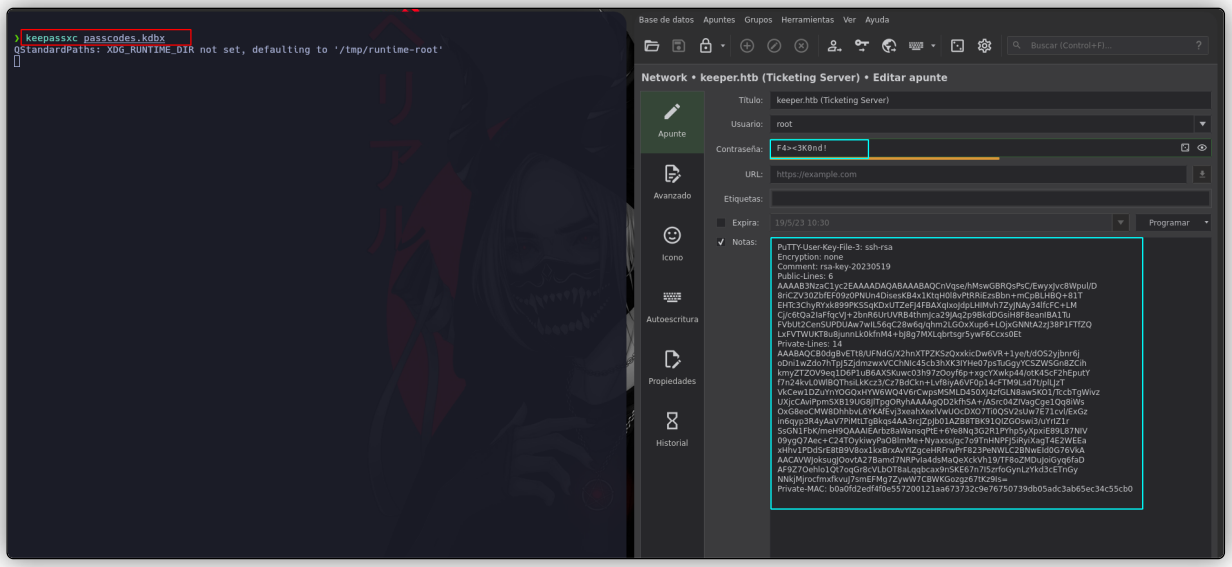
> - **Archivo .dmp**:
> - El formato de archivo **.dmp** se utiliza comúnmente para archivos de **volcado de memoria**. Estos archivos contienen un volcado de la memoria de un programa o sistema en un momento específico en el tiempo. Por lo general, se generan cuando un programa o sistema experimenta un error grave o se bloquea inesperadamente.
> - Los archivos de volcado de memoria **.dmp** pueden ser útiles para diagnosticar problemas de software. Los desarrolladores y los equipos de soporte técnico a menudo utilizan estos archivos para analizar el estado del sistema o del programa en el momento del fallo y así identificar la causa subyacente del problema.
> - **Archivo .kdbx**:
> - La extensión de archivo **.kdbx** se asocia comúnmente con una base de datos cifrada creada por el programa de gestión de contraseñas **KeePass**. KeePass es una aplicación de software de código abierto que permite almacenar de forma segura contraseñas y otra información confidencial en una base de datos cifrada. Los archivos de base de datos de **KeePass (.kdbx)** almacenan las contraseñas y otros datos de forma segura utilizando un algoritmo de cifrado, lo que garantiza que solo las personas autorizadas puedan acceder a la información contenida en la base de datos mediante una clave maestra.
> - La extensión **.kdbx** se ha vuelto muy popular debido a la creciente preocupación por la seguridad de los datos en línea y la necesidad de gestionar múltiples contraseñas de manera segura.

> - **CVE-2023-32784**:
> - *En KeePass 2.x antes de la versión 2.54*, es posible recuperar la contraseña maestra en texto claro a partir de un volcado de memoria, incluso cuando un espacio de trabajo está bloqueado o ya no está en ejecución. El volcado de memoria puede ser un volcado del proceso de KeePass, un archivo de intercambio (pagefile.sys), un archivo de hibernación (hiberfil.sys) o un volcado de RAM de todo el sistema. El primer carácter no puede ser recuperado. En la versión

> 2.54, se implementaron cambios en el uso de API y/o inserción de cadenas aleatorias para mitigar este problema.

1.7. Looking for password coincidence on internet

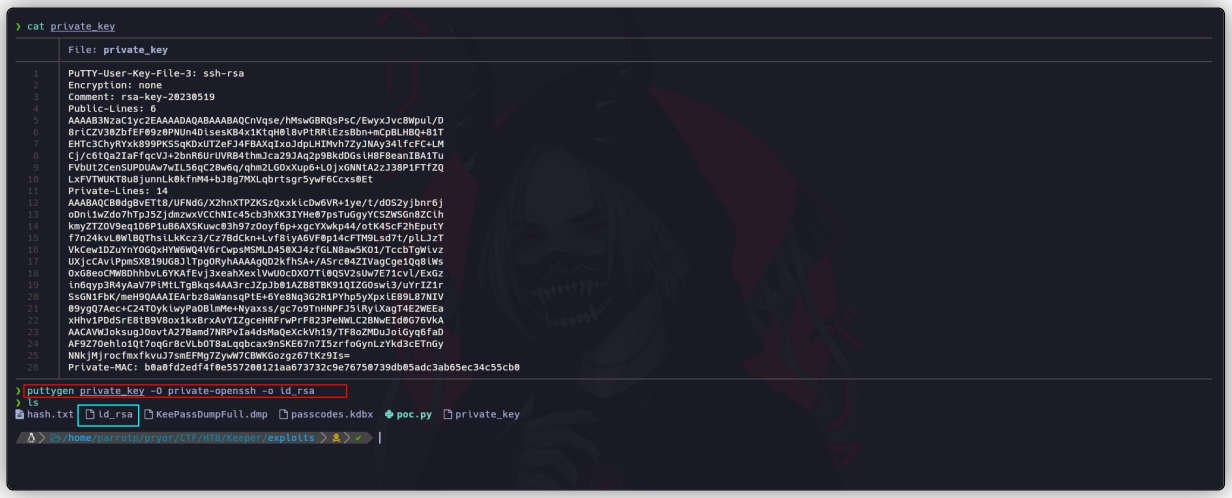
- Buscamos estas contraseñas en Google para encontrar coincidencias. Obtenemos lo siguiente. Probaremos estas alternativas como posible contraseña maestra.
  - 

- Hacemos nuevamente `keepassxc passcodes.kdbx` para abrir la base de datos e introducimos esta contraseña: *Rødgrød med Fløde*. En un principio, no obtenemos acceso, pero tras diferentes intentos, obtenemos la clave maestra: *rødgrød med fløde*. Tuvimos que pasar toda la contraseña a minúsculas. Al obtener acceso a la base de datos, vemos una posible contraseña para el usuario **root**, con la cual probamos acceder, pero tampoco tenemos éxito. Nos centraremos ahora en lo que parece ser una clave SSH que encontramos.
  - 

1.8. Changing SSH key format

- Copiamos toda esta clave y la pegamos en un archivo en nuestro directorio de trabajo. No obstante, debemos saber que esta no es la típica clave SSH **id_rsa**. Siendo ésta una clave *Putty-user-key-file-3*, la cual suele tener una extensión *.ppk*. La idea entonces es tratar de transformar este formato de clave a un formato **PEM**, que es el usado por las claves privadas SSH con la

extensión *.key*. Para ello, vamos a usar la herramienta Puttygen con `puttygen private_key -O private-openssh -o id_rsa`. Ya tenemos nuestra clave id_rsa.



- Hacemos `chmod 600 id_rsa` para dar los permisos necesarios a esta clave. Ahora, ejecutamos: `ssh -i id_rsa root@10.10.11.227` para conectarnos como root a la máquina víctima proporcionando el archivo id_rsa. Conseguimos acceso.