Networked

En este post se explicarán los pasos que se han seguido para conseguir vulnerar la seguridad de la máquina Networked en Hack The Box, tal y como se refleja, es un sistema Linux con un nivel de dificultad fácil (4.4).

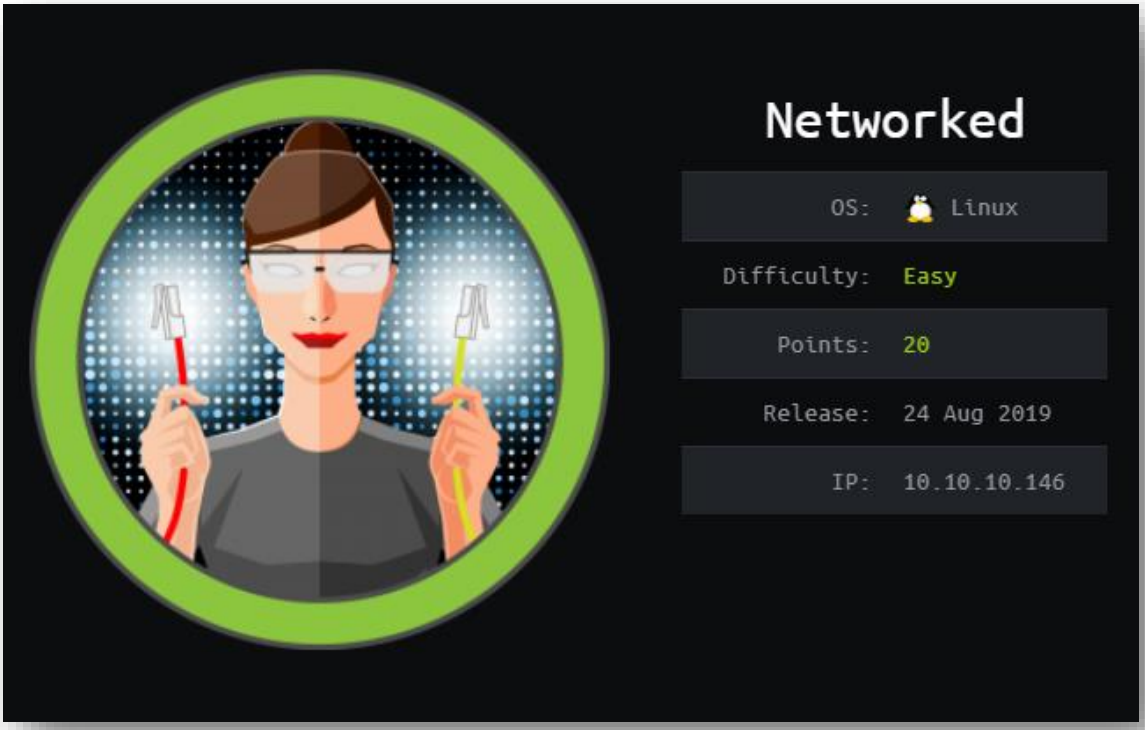

Ilustración 1: Networked.

Se procedió a realizar un escaneo de servicios y puertos haciendo uso de NMAP:

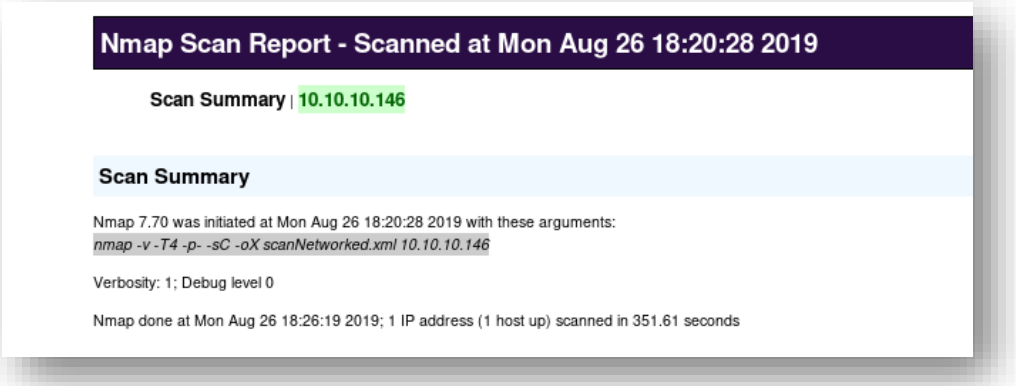

Ilustración 2: Comando de NMAP usado.

Port		State (toggle closed [1] | filtered [0])	Service	Reason	Product	Version	Extra info
22	tcp	open	ssh	syn-ack			
	ssh-hostkey	2048 22:75:d7:a7:4f:81:a7:af:52:66:e5:27:44:b1:01:5b (RSA) 256 2d:63:28:fc:a2:99:c7:d4:35:b9:45:9a:4b:38:f9:c8 (ECDSA) 256 73:cd:a0:5b:84:10:7d:a7:1c:7c:61:1d:f5:54:cf:c4 (ED25519)					
80	tcp	open	http	syn-ack			
	http-methods	Supported Methods: GET HEAD POST OPTIONS					
	http-title	Site doesn't have a title (text/html; charset=UTF-8).					

Ilustración 3: Resultados de la ejecución del comando NMAP.

Como se puede observar existe un puerto en el que se ejecuta un servicio Apache (80) y el puerto 22 está habilitado para SSH.

La web que se muestra en el puerto 80 es la siguiente:

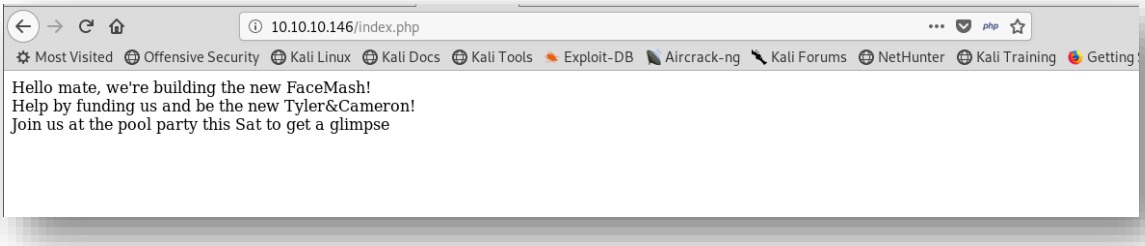

Ilustración 4: Web en http://10.10.10.146/index.php.

A primera vista la web no reflejaba mucha más información que la mostrada en el mensaje de la página de inicio. Así que se usó la herramienta DIRB para encontrar rutas más interesantes o ficheros vulnerables:

- DIRB:

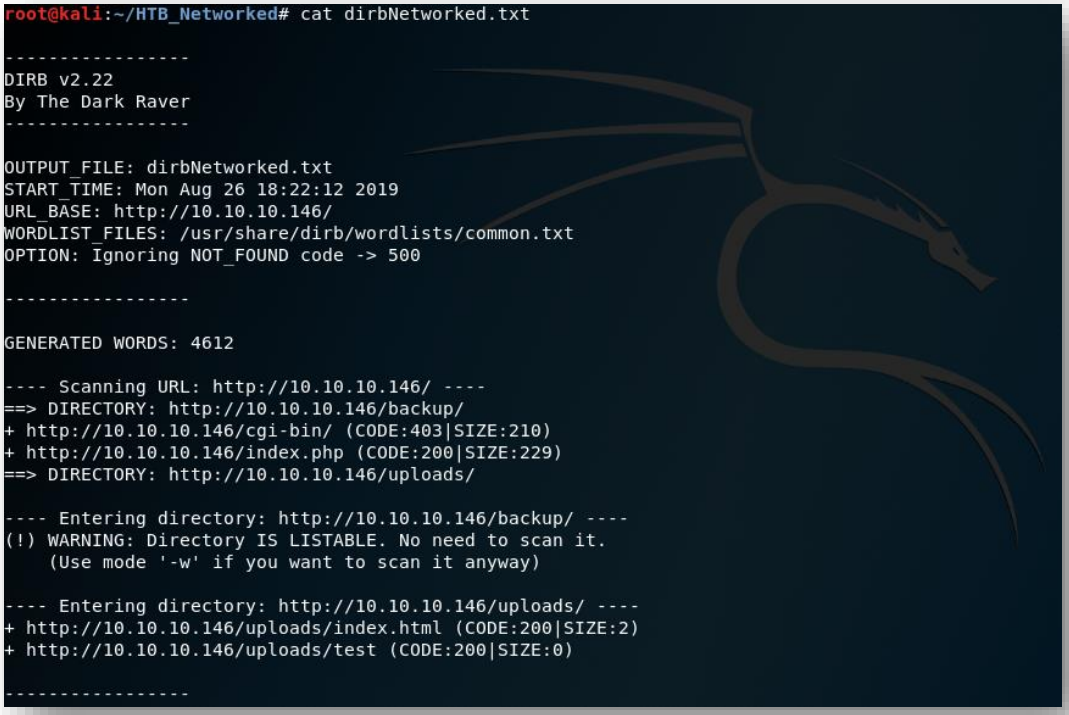

Ilustración 5: DIRB en http://10.10.10.146.

Analizando los resultados se encontraron algunos ficheros y directorios a los cuales se podían acceder:

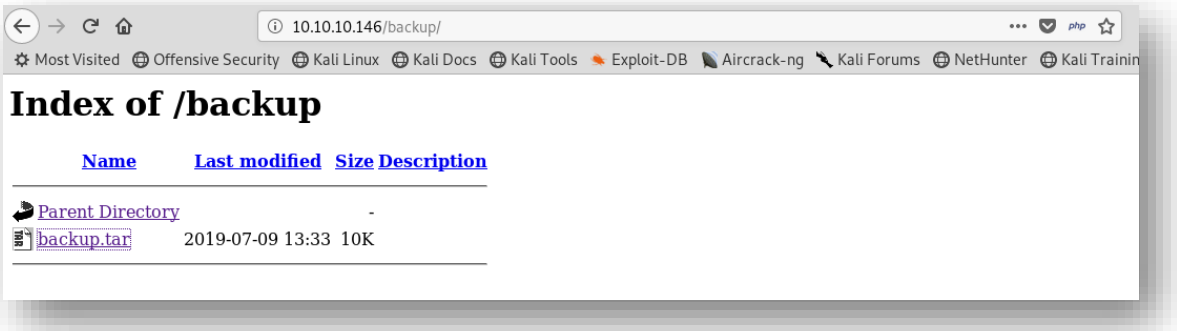

Ilustración 6: Directorio Backup.

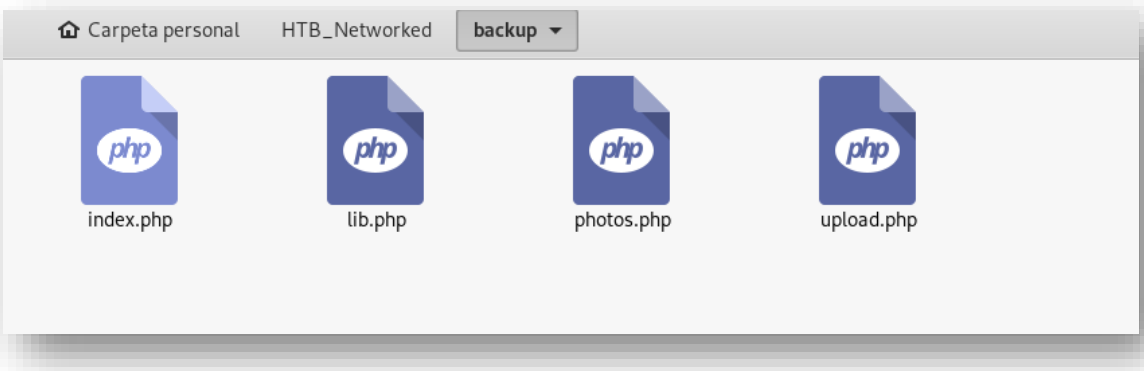

Ilustración 7: Contenido del directorio Backup en http://10.10.10.146/backup.

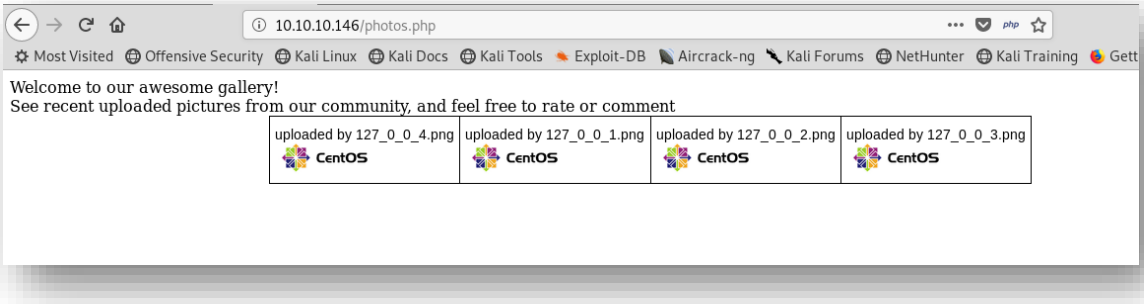

Ilustración 8: Acceso al fichero photos.php.

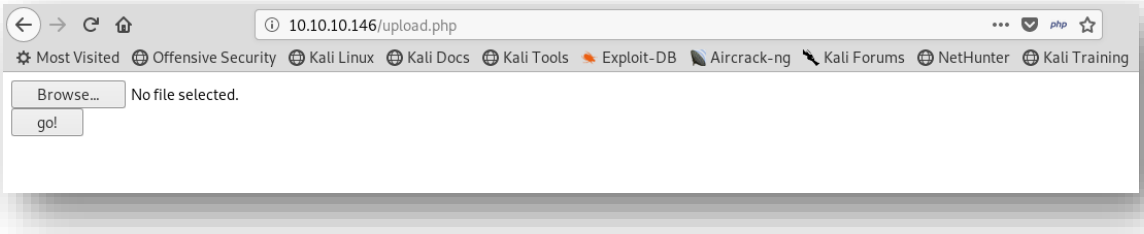

Ilustración 9: Acceso al fichero upload.php.

Con toda la información recabada, parecía bastante claro el siguiente paso que se debía seguir para poder entrar en el sistema. Crear un fichero con código PHP que ejecutara una *shell* en el equipo víctima y subirla como un fichero de tipo imagen (extensiones JPEG o PNG), haciendo uso del fichero *upload.php. (Fuente:* https://www.youtube.com/watch?v=nNB9XlRfvzw*).*

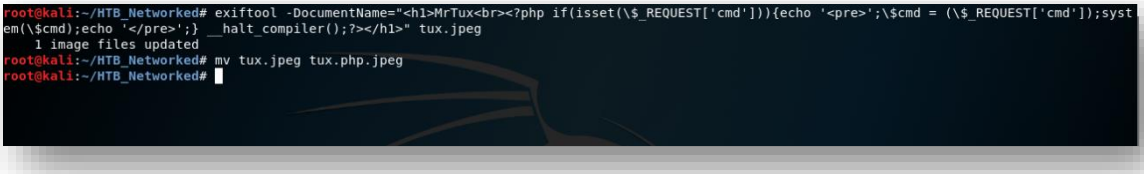

Ilustración 10: Usando exiftool para introducir código PHP en una imagen con extensión JPEG.

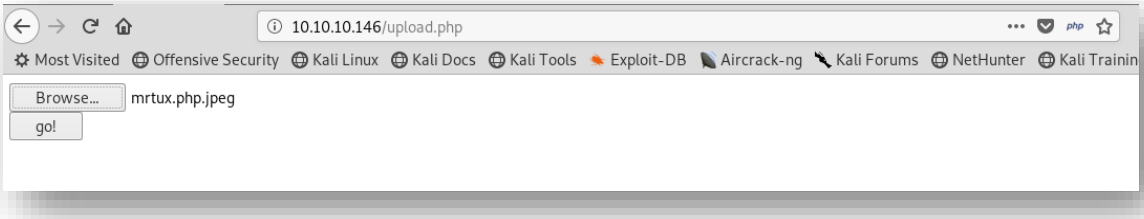

Ilustración 11: Subiendo fichero con web shell.

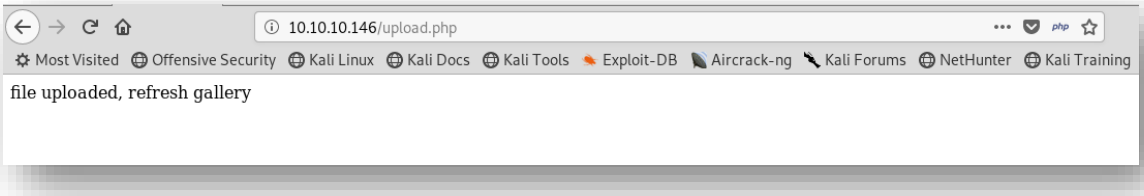

Ilustración 12: Fichero subido con éxito.

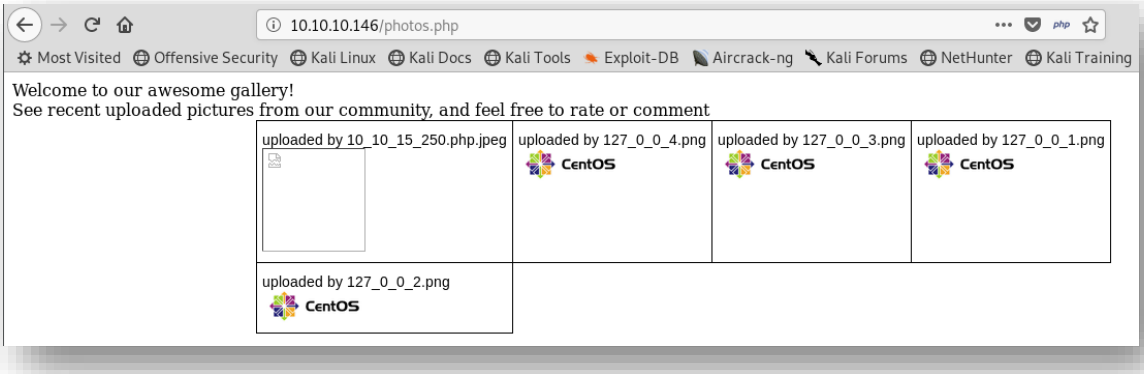

Ilustración 13: El fichero photos.php lista las imágenes subidas en el sistema, incluyendo la que se había subido anteriormente.

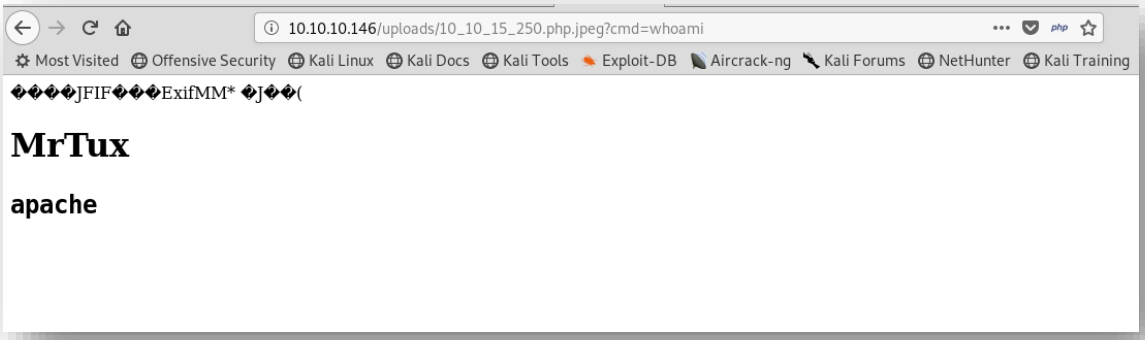

Ilustración 14: Ejecutando comando whoami en la web shell.

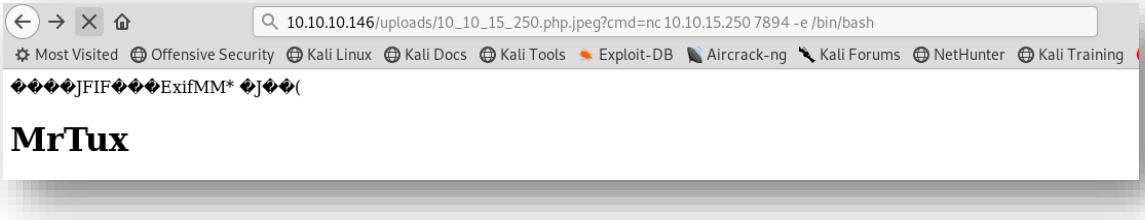

Ilustración 15: Abriendo una reverse shell mediante la web shell con el comando nc.

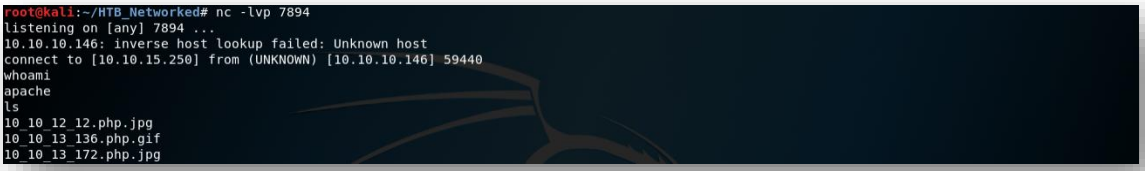

Ilustración 16: reverse shell obtenida con el usuario apache.

Ya una vez dentro del sistema con un usuario sin privilegios (*apache*), se optó por realizar un pequeño reconocimiento, en el cual, se pudo hallar el directorio de otro usuario en el sistema (*guly*), con los siguientes ficheros:

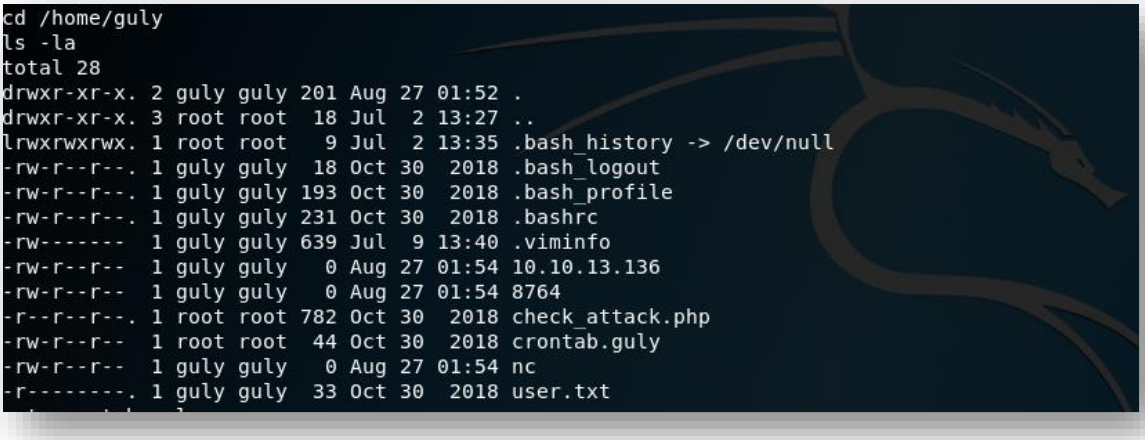

Ilustración 17: Contenido del directorio /home/guly.

Existía un fichero llamado "*crontab.guly*" que ejecutaba el fichero "*check_attack.php*" cada tres minutos. Por tanto, se analizó el código del programa para buscar posibles vulnerabilidades y realizar una escalada de privilegios al usuario *guly*.

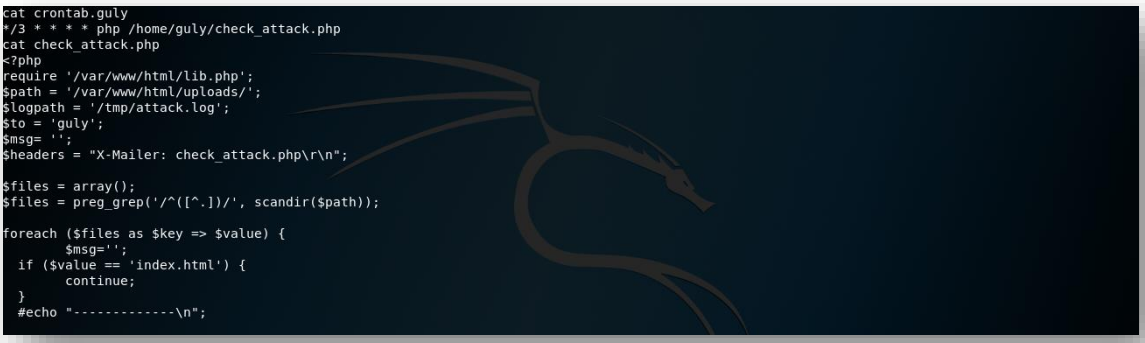

Ilustración 18: Fichero check_attack.php parte 1.

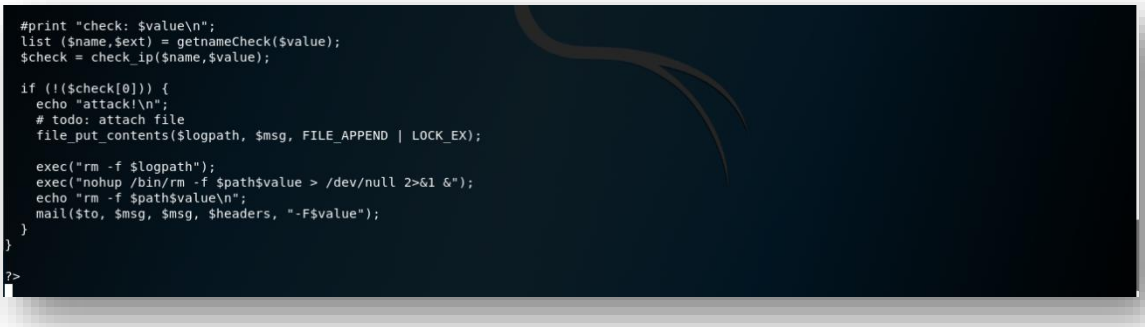

Ilustración 19: Fichero check_attack.php parte 2.

Cuando se ejecuta "*check_attack.php*" recorre el directorio */var/www/html/uploads/* en busca de ficheros maliciosos, para posteriormente eliminarlos haciendo uso del programa

/bin/rm. Si se consigue crear un fichero en el directorio *uploads* que evite el filtrado de la expresión regular del programa, donde no se permiten '/' ni '\', se puede ejecutar una *reverse shell* cuando se lance */bin/rm* y realizar una escalada de privilegios.

Se decidió codificar el nombre del fichero en base64 para así escapar del filtro de la expresión regular:

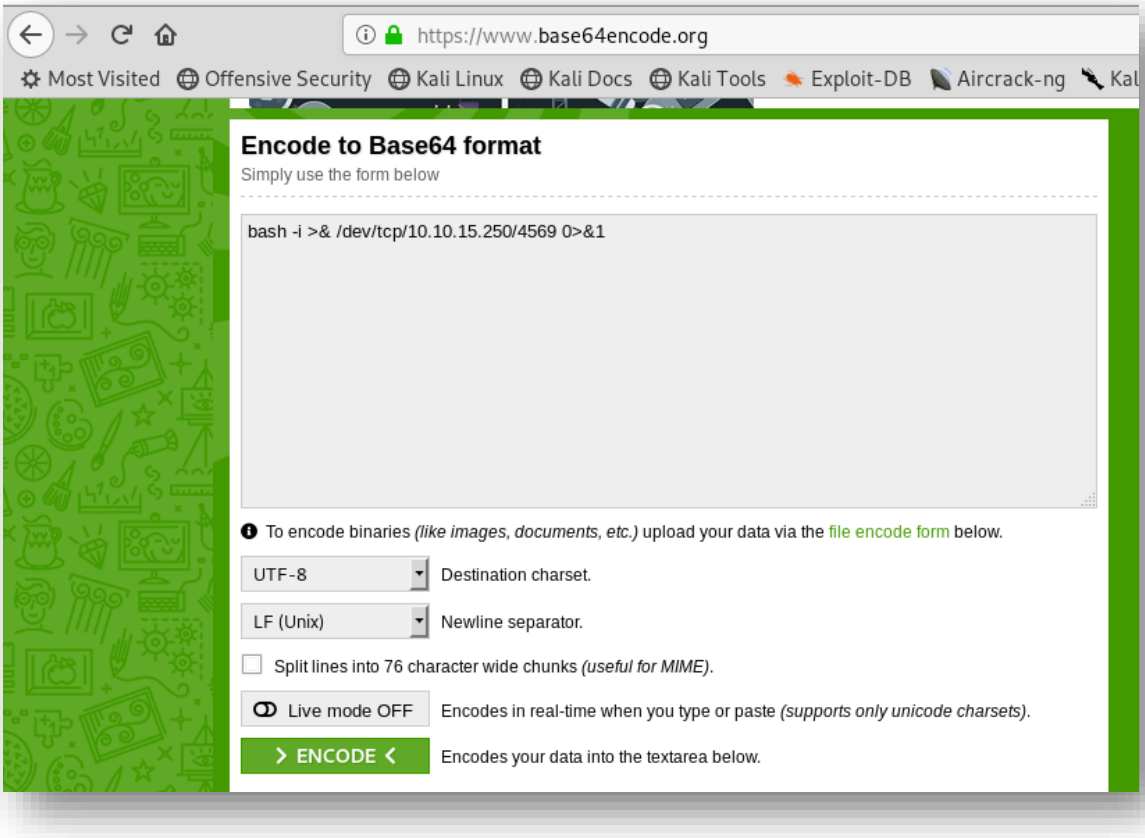

Ilustración 20: Comando que abrirá una reverse shell con el usuario guly.

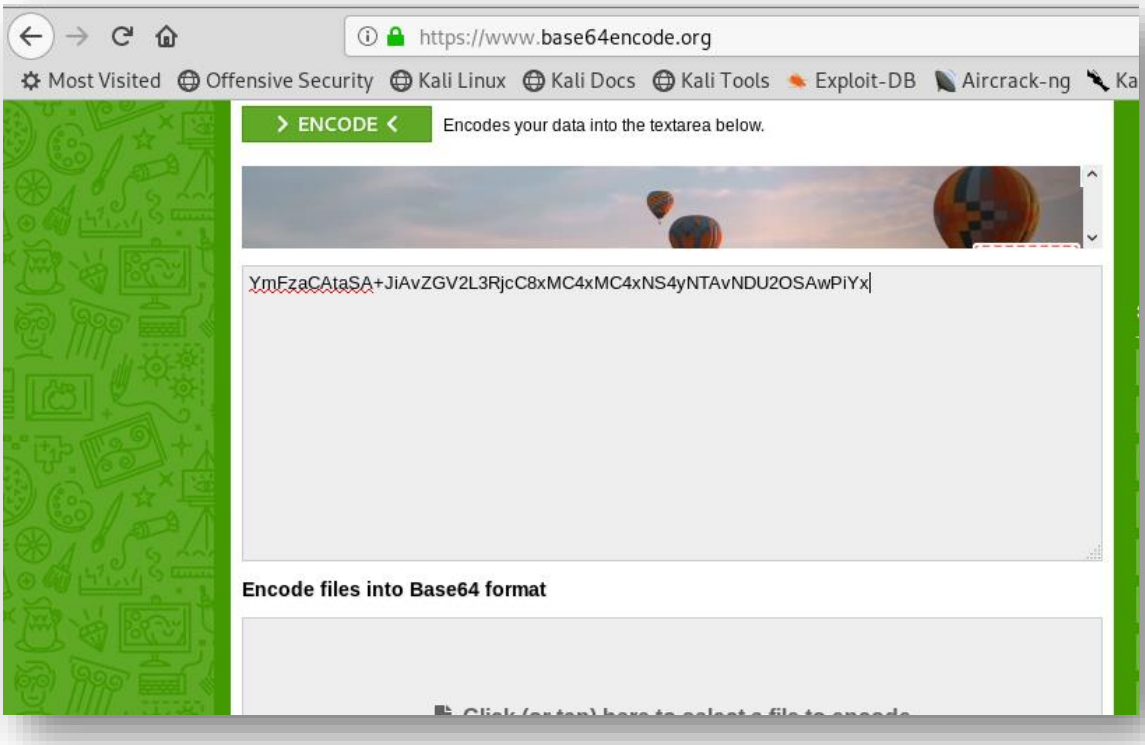

Ilustración 21: Codificando el comando en base64.

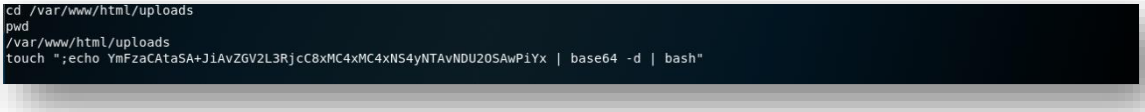

Ilustración 22: Creando el fichero que tiene por nombre el comando que se ejecutará en el sistema.

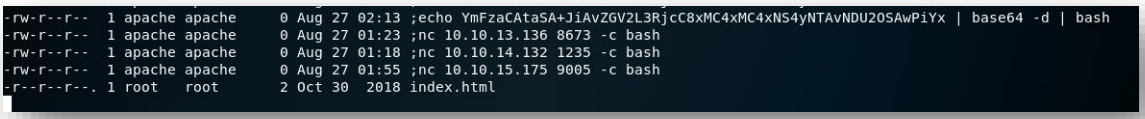

Ilustración 23: Fichero creado con éxito.

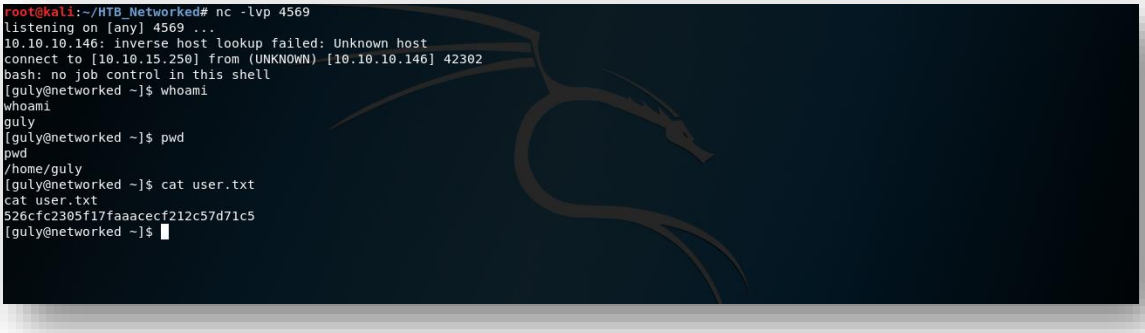

Ilustración 24: Después de tres minutos reverse shell conseguida y flag del usuario guly.

En este punto solo falta conseguir tener acceso al sistema como usuario administrador (*root*). Se usó el comando "*sudo -l*" para comprobar si el usuario *guly* podía ejecutar algún programa con privilegios de administrador.

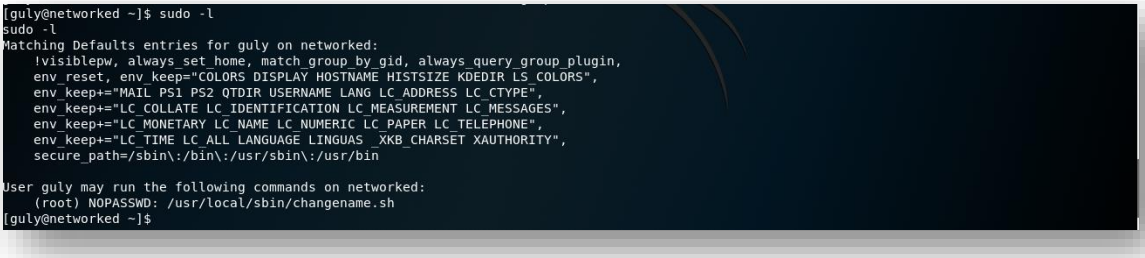

Ilustración 25: Script /usr/local/sbin/changename.sh ejecutable como root desde la sesión de guly.

Se siguió el mismo procedimiento que se usó para realizar la escalada de privilegios anterior, se analizó el código, en este caso del fichero "*/usr/local/sbin/changename.sh"*:

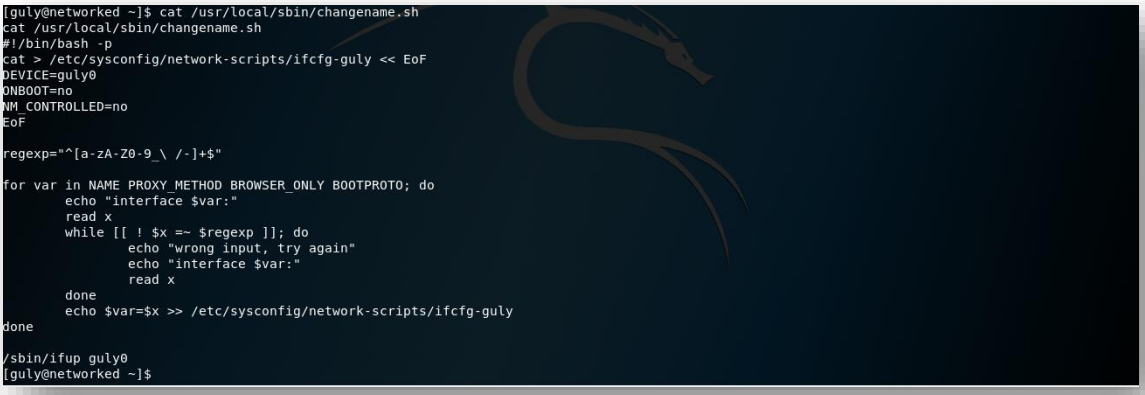

Ilustración 26: Código del script /usr/local/sbin/changename.sh.

El script permite introducir parámetros al usuario que lo ejecuta, simplemente se debe rebasar el filtro de la expresión regular que controla que no se introduzca ningún

comando. Para ello, se creó un fichero en el directorio */tmp*, con el comando que abriría una *reverse shell* como *root* y se introdujo la ruda de dicho fichero como parámetro en el script.

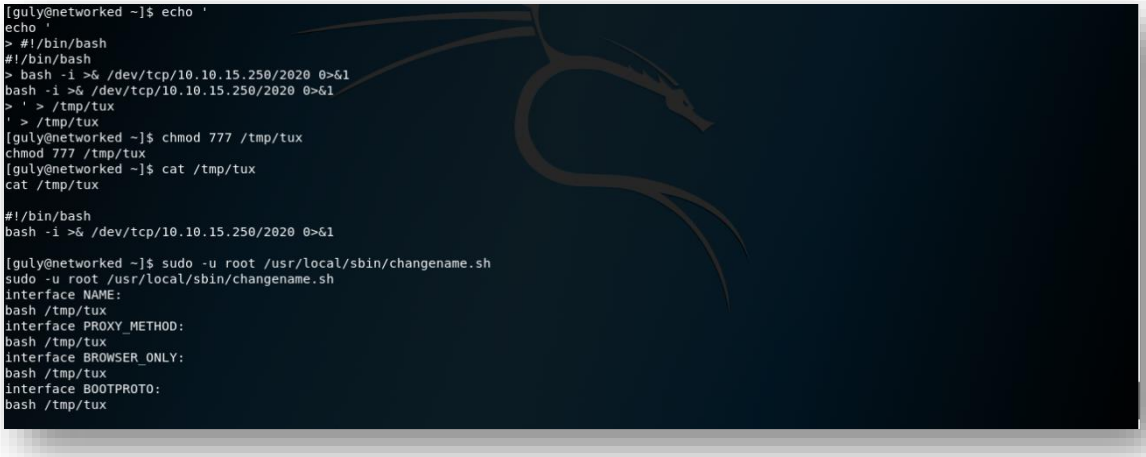

Ilustración 27: Creando script malicioso e introduciendo su ruta como parámetro.

Una vez ejecutado, se obtuvo una *shell* como administrador:

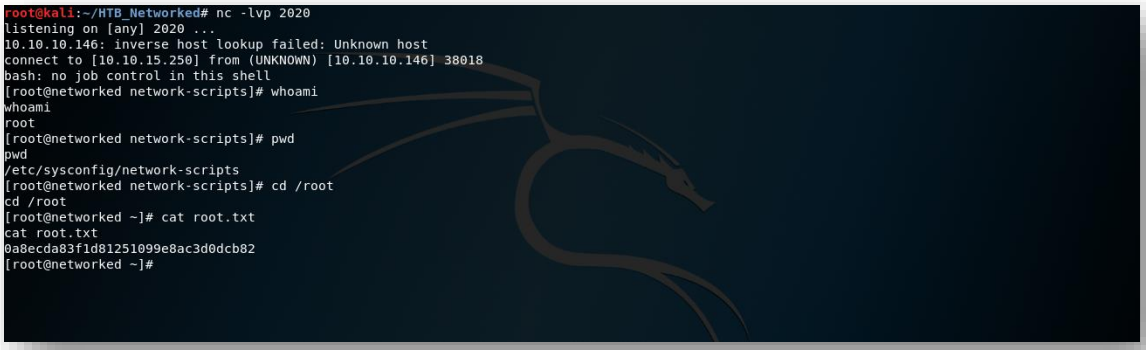

Ilustración 28: Obtención de una shell como administrador del sistema y la flag root.txt.

Como conclusión se podría decir que es una máquina sencilla a la hora de identificar las vulnerabilidades a explotar. Pero bastante divertida de realizar.