Luke

En este post se explicarán los pasos que se han seguido para conseguir vulnerar la seguridad de la máquina Luke en Hack The Box, tal y como se refleja, es un sistema FreeBSD con un nivel de dificultad medio.

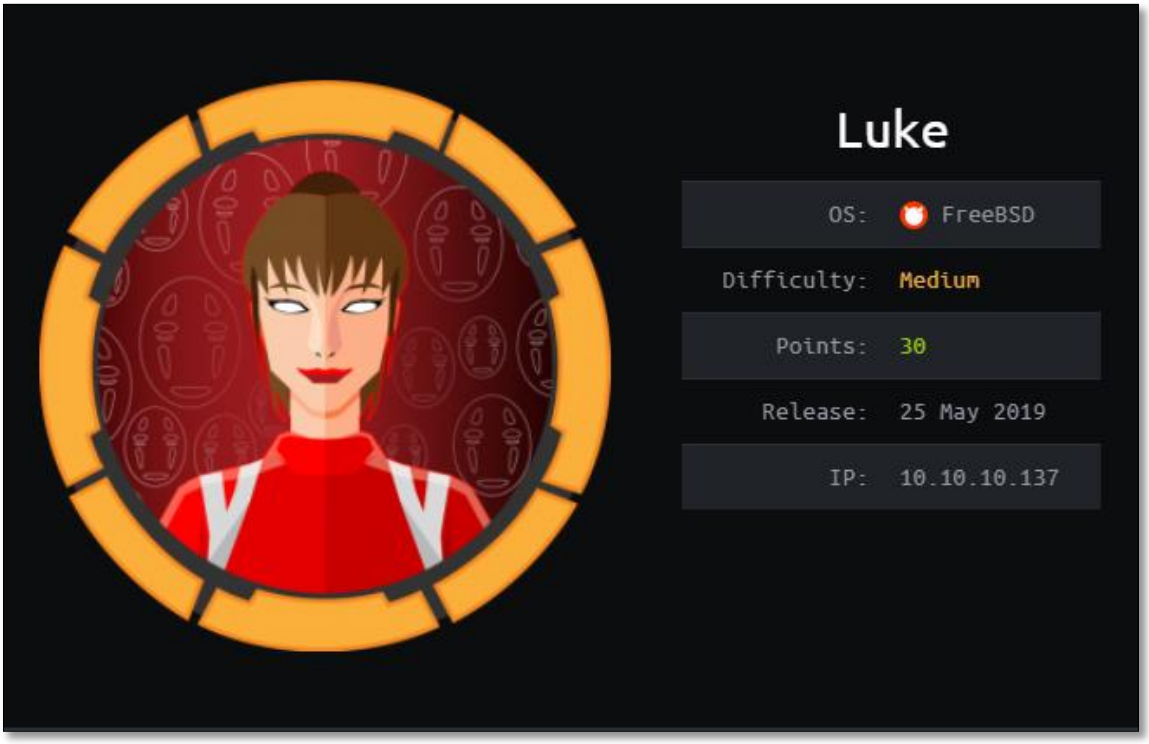

Ilustración 1: Luke.

Se comenzó realizando un escaneo del tipo SYN-SCAN, de todos los puertos del sistema:

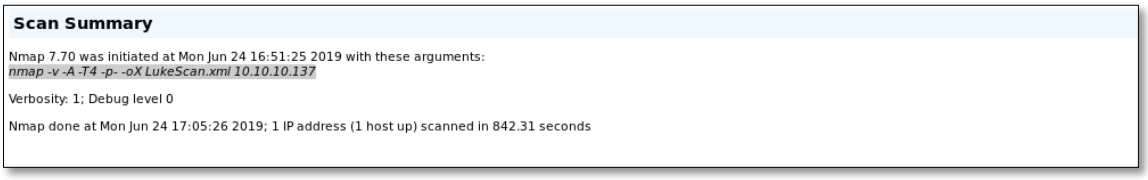

Ilustración 2: Nmap ejecutado.

Obteniendo así todos los servicios y puertos que la máquina tenía habilitado:

Port		State (toggle closed [0] filtered [21])	Service	Reason	Product	Version	Extra info
21	tcp	open	ftp	syn-ack	vsftpd	3.0.3+ (ext.1)	
	ftp-anon	Anonymous FTP login allowed (FTP code 230) drwxr-xr-x 2 0 0 512 Apr 14 12:35 webapp					
	ftp-syst	STAT: FTP server status: Connected to 10.10.15.41 Logged in as ftp TYPE: ASCII No session upload bandwidth limit No session download bandwidth limit Session timeout in seconds is 300 Control connection is plain text Data connections will be plain text At session startup, client count was 3 vsFTPd 3.0.3+ (ext.1) - secure, fast, stable End of status					
22	tcp	open	ssh	syn-ack			
3000	tcp	open	http	syn-ack	Node.js Express framework		
	http-methods	Supported Methods: GET HEAD POST OPTIONS					
	http-title	Site doesn't have a title (application/json; charset=utf-8).					
5047	tcp	filtered	iscape	host-unreach from 10.10.12.1			
7323	tcp	filtered	swx	host-unreach from 10.10.12.1			

Ilustración 3: Resultados nmap parte 1.

8000	tcp	open	http	syn-ack	Ajenti http control panel		
	http-methods	Supported Methods: GET HEAD POST OPTIONS					
	http-title	Ajenti					
9199	tcp	filtered		host-unreach from 10.10.12.1			
11738	tcp	filtered		host-unreach from 10.10.12.1			
14370	tcp	filtered		host-unreach from 10.10.12.1			
15150	tcp	filtered		host-unreach from 10.10.12.1			
15634	tcp	filtered		no-response			
20985	tcp	filtered		no-response			
23995	tcp	filtered		no-response			
25968	tcp	filtered		no-response			
33094	tcp	filtered		host-unreach from 10.10.12.1			
35235	tcp	filtered		no-response			
39987	tcp	filtered		no-response			
42175	tcp	filtered		no-response			
46806	tcp	filtered		host-unreach from 10.10.12.1			
47718	tcp	filtered		host-unreach from 10.10.12.1			
49404	tcp	filtered		no-response			
55369	tcp	filtered		no-response			
56676	tcp	filtered		host-unreach from 10.10.12.1			
64781	tcp	filtered		no-response			
65099	tcp	filtered		no-response			

Ilustración 4: Resultados nmap parte 2.

Como se puede observar la ejecución de nmap ha proporcionado unos resultados muy interesantes. Existe un servicio Express de NodeJS en el puerto 3000 así como una web en el puerto 8000. Además, el servidor FTP que se muestra en los resultados tiene el usuario anónimo habilitado, por tanto, el siguiente paso es claro, conectarse al servidor FTP:

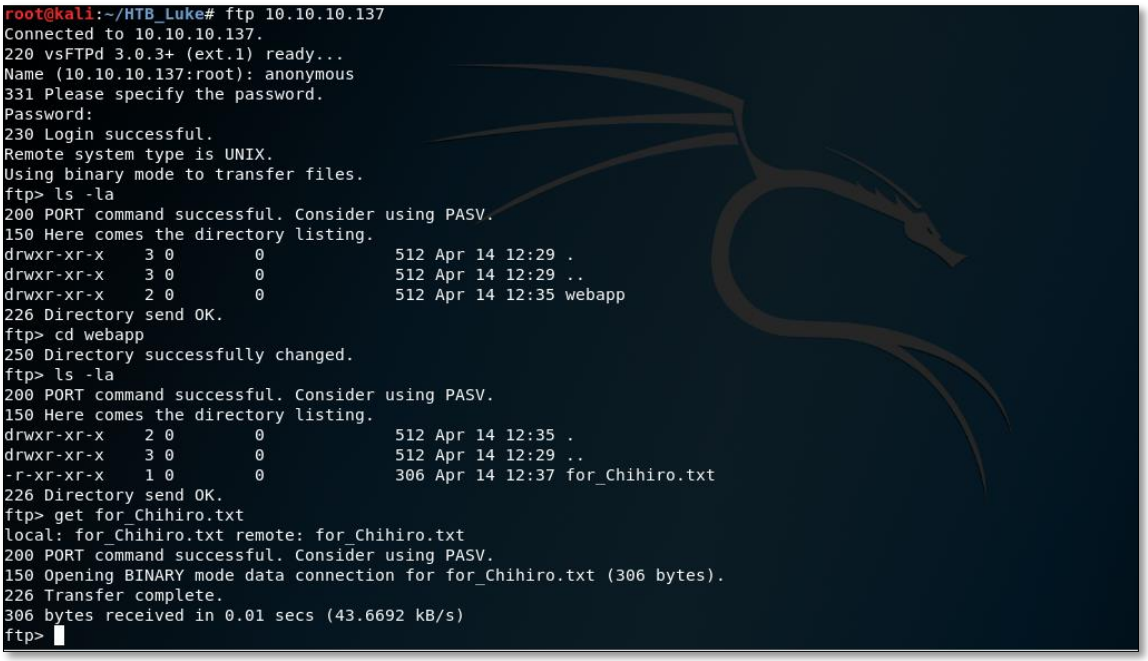

Ilustración 5: Directorios y ficheros accesibles por el usuario anónimo de FTP.

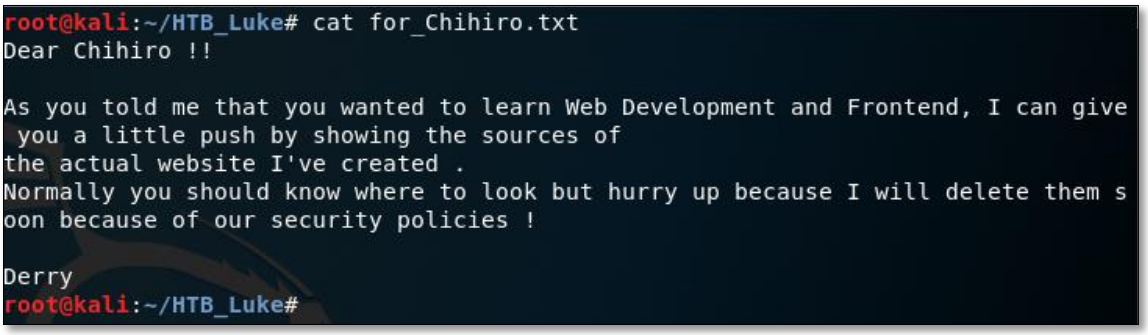

Ilustración 6: Contenido del fichero con extensión txt.

Dentro del servidor FTP no se hallaban ficheros muy interesantes excepto una nota para el usuario *Chihiro* por parte de *Derry*. En la cual se habla sobre el desarrollo de una web que tiene fallos de seguridad. Lo mas importante de este hallazgo es el hecho de la obtención de dos posibles usuarios en las webs, las cuales deben ser investigadas.

En un primer análisis se habían detectado las siguientes webs:

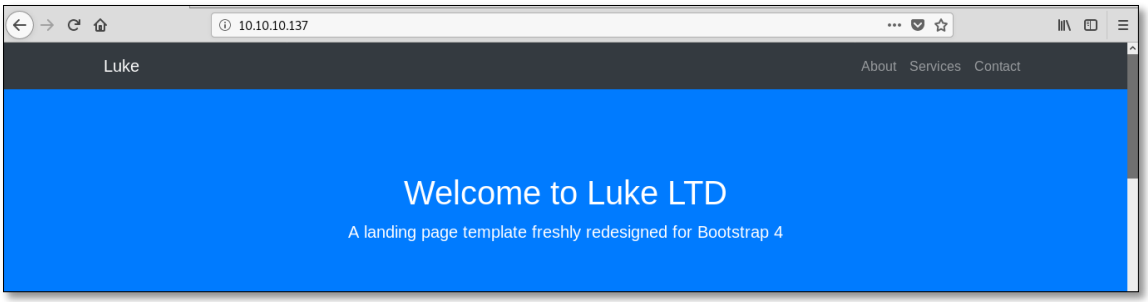

Ilustración 7: Web en el puerto 80.

A pesar de que el tipo de escaneo de nmap no reportó el puerto 80 abierto, había un servicio web en él. La enumeración es un factor importante para resolver esta máquina, así que posiblemente se debió hacer escaneos más intensos.

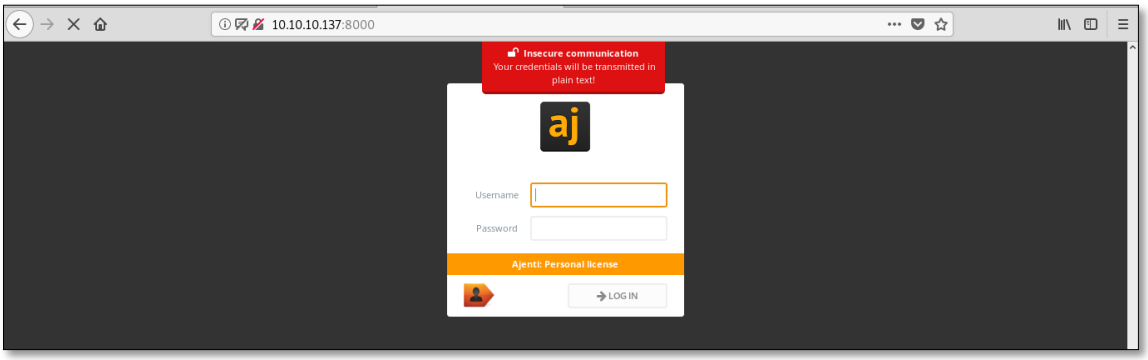

Ilustración 8: Web en el puerto 8000.

En el puerto 8000 existía una aplicación web con un panel de inicio de sesión, el cuál avisaba de la inseguridad con la que se tratan las claves.

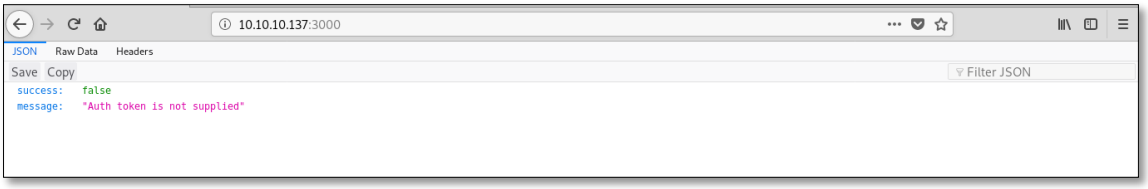

Ilustración 9: Conexión a la API en el puerto 3000.

Ya en el puerto 3000 se podían realizar las peticiones a la API, aunque no se conocían los posibles *endpoints* que podía tener. Pero haciendo uso de convencionalismos se obtuvo lo siguiente:

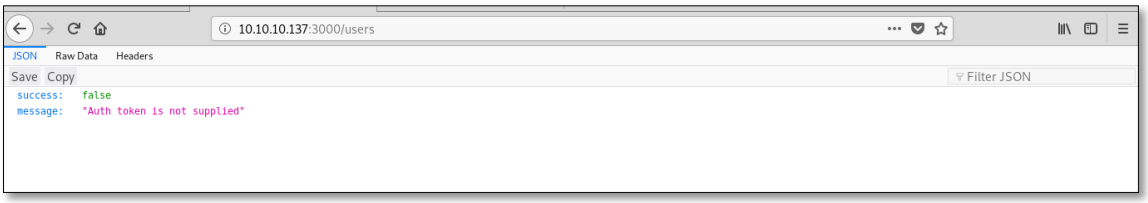

Ilustración 10: Endpoint users.

Sabiendo que existe un *endpoint users*, se pasó a comprobar que tipos de usuarios existen:

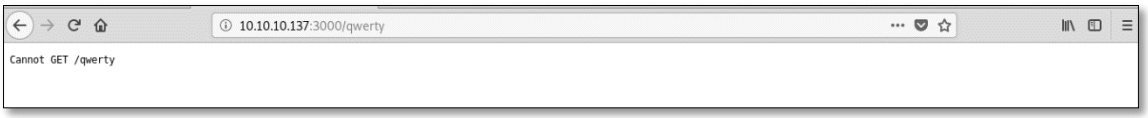

Ilustración 11: Usuario qwerty no existe.

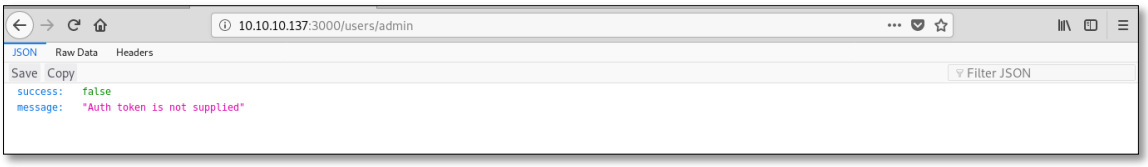

Ilustración 12: Usuario admin existe.

Ilustración 13: Usuario root no existe.

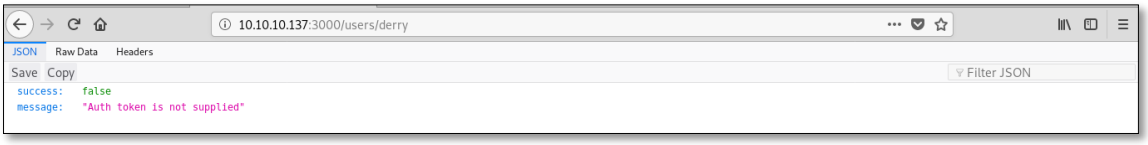

Ilustración 14: Usuario derry existe.

Ilustración 15: Usuario Chihiro no existe.

Cuando se obtuvieron los usuarios existentes, los cuales algunos coincidían con la nota encontrada en el servidor FTP, se procedió a verificar si existía un *endpoint* de login:

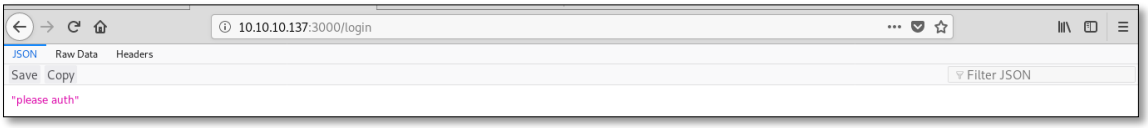

Ilustración 16: Endpoint de login existente.

Con toda esta información se podía intuir que para conseguir acceso al sistema primero se debían realizar peticiones a la API, para obtener tokens de acceso a las aplicaciones webs.

Para descubrir más paneles de inicio de sesión, así como ficheros de configuración que revelen vulnerabilidades, se hizo uso de DIRB y Nikto, en las webs que se tenían hasta el momento, así como en la API.

- Ejecuciones de DIRB:

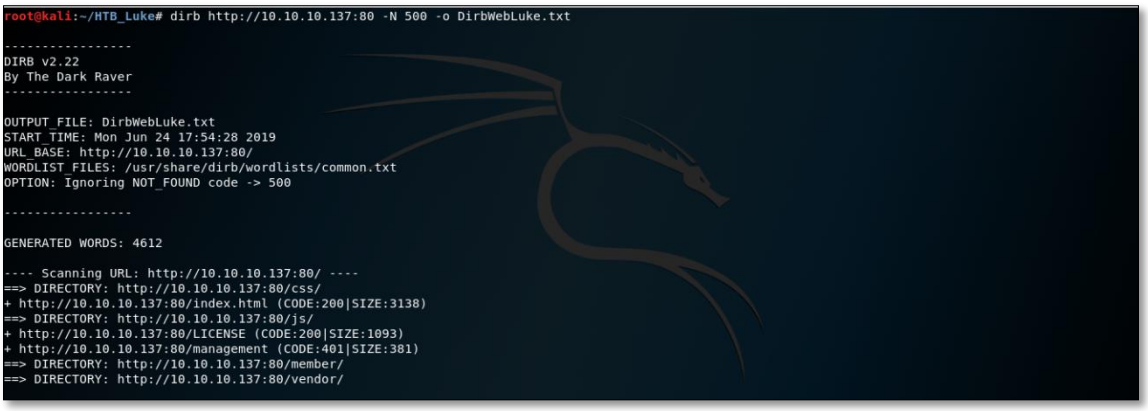

Ilustración 17: Ejecución de DIRB en el puerto 80.

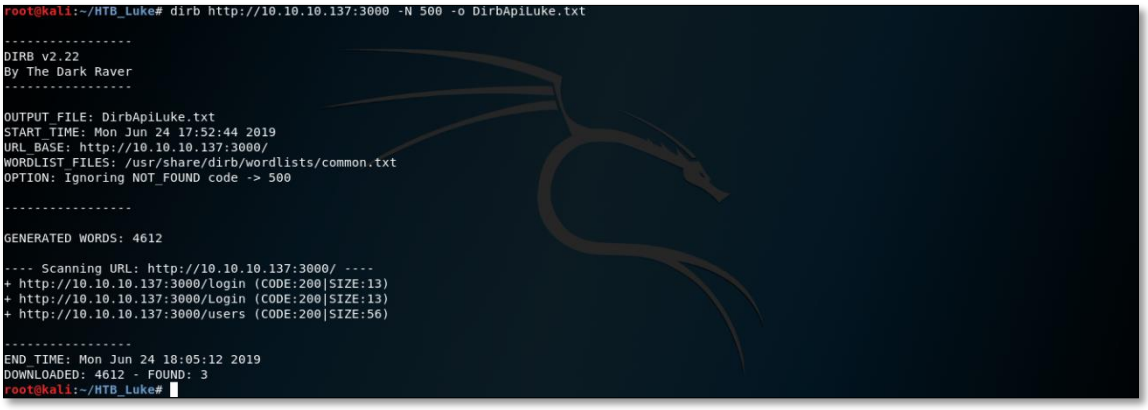

Ilustración 18: Ejecución de DIRB en el puerto 3000 (API).

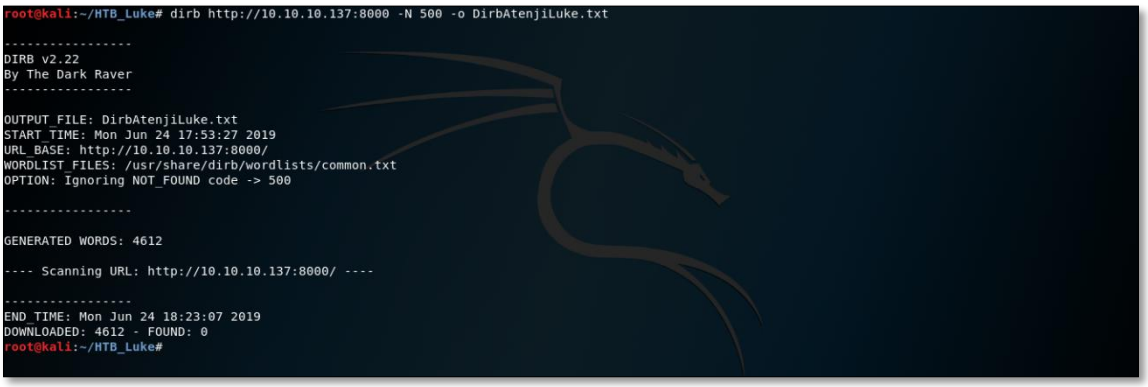

Ilustración 19: Ejecución de DIRB en el puerto 8000.

- Ejecuciones de Nikto:

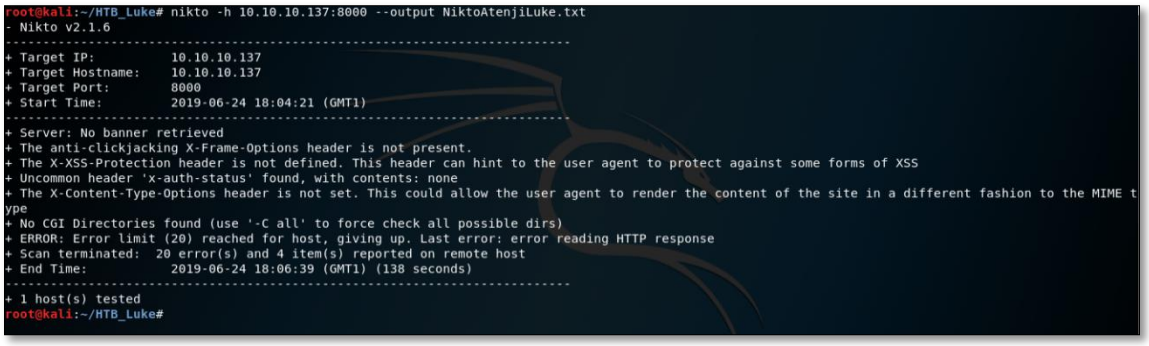

Ilustración 20: Ejecución de Nikto en el puerto 8000.

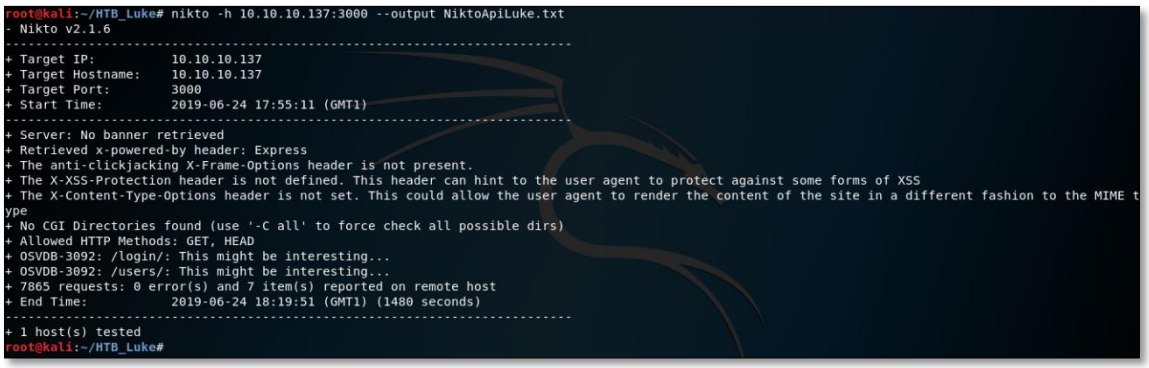

Ilustración 21: Ejecución de Nikto en el puerto 3000.

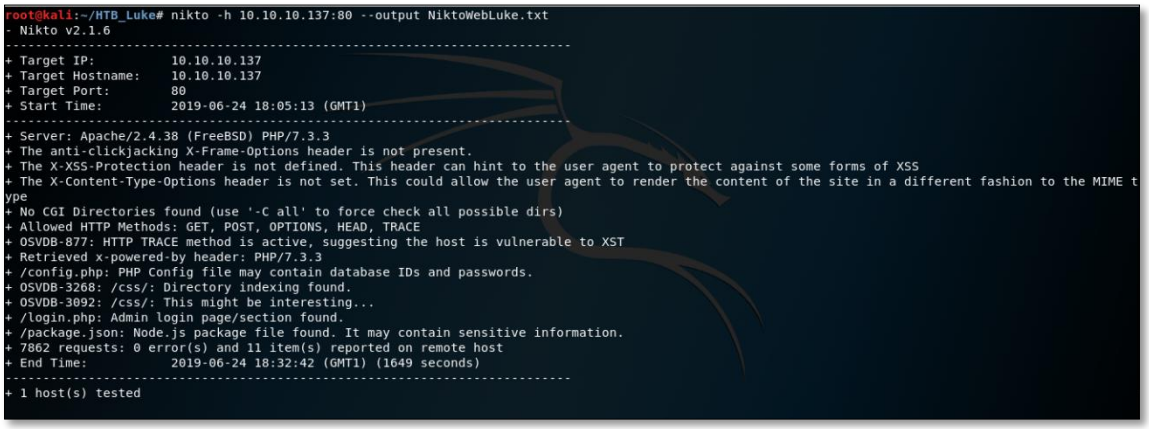

Ilustración 22: Ejecución de Nikto en el puerto 80.

Ambas herramientas no proporcionaron ninguna información relevante para la aplicación web que se accedía por el puerto 8000. En cuanto a la API se obtuvieron los mismos *endpoints* que se habían descubierto anteriormente. La información más importante que se obtuvo era referente a la web del puerto 80:

Ilustración 23: Acceso al directorio vendor.

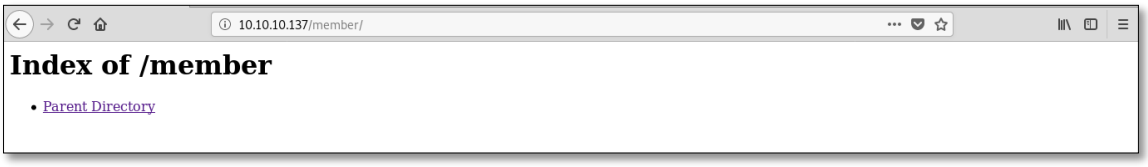

Ilustración 24: Acceso al directorio member.

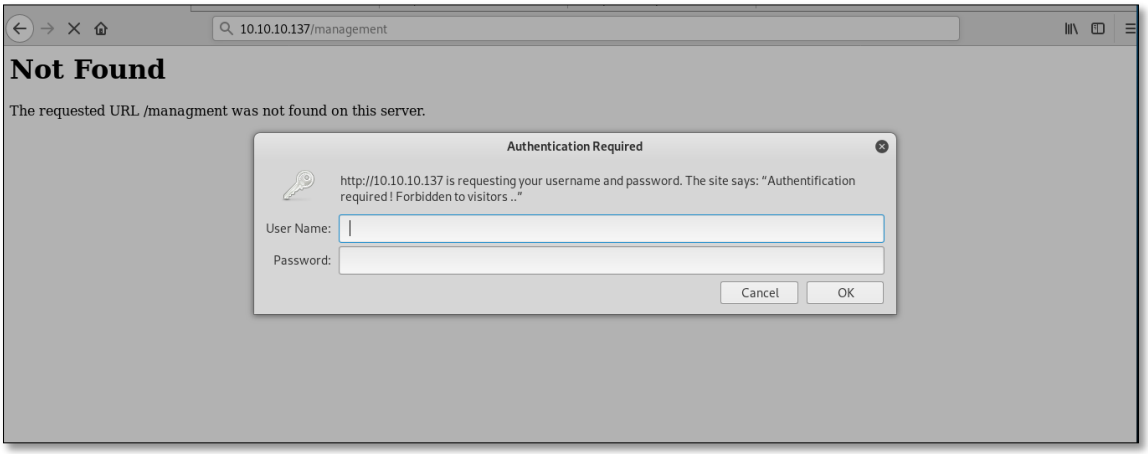

Ilustración 25: Panel de inicio de sesión en la ruta management.

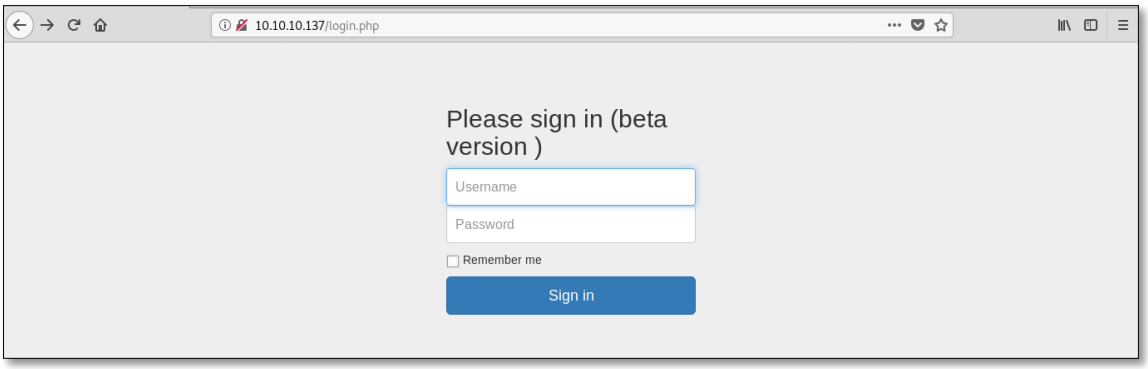

Ilustración 26: Panel de inicio de sesión a la aplicación web del puerto 80.

Se podía acceder a dos directorios cuyo contenido parecía irrelevante, además de dos paneles de login, uno que apuntaba tener relación con la gestión de la web (management) y otro a la propia aplicación.

Pero quizás lo más útil fue lo descubierto por la herramienta Nikto, el acceso al fichero con extensión PHP denominado *config* y al *package.json.*

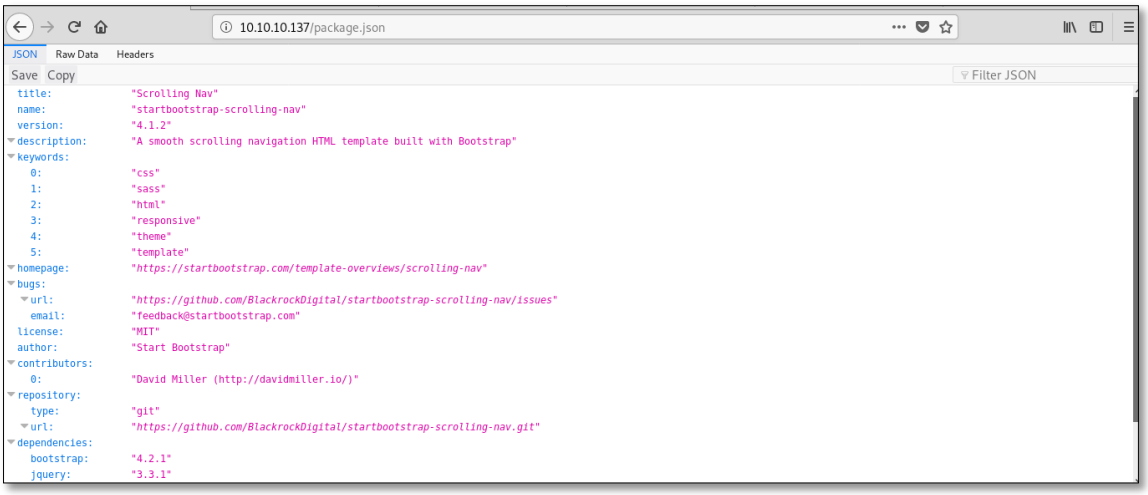

Ilustración 27: Fichero package.json.

Ilustración 28: Fichero config.php.

Como se muestra en la imagen, se tiene lo que parece ser una contraseña del usuario *root*, por tanto, con tal información ya se podían hacer peticiones a la API.

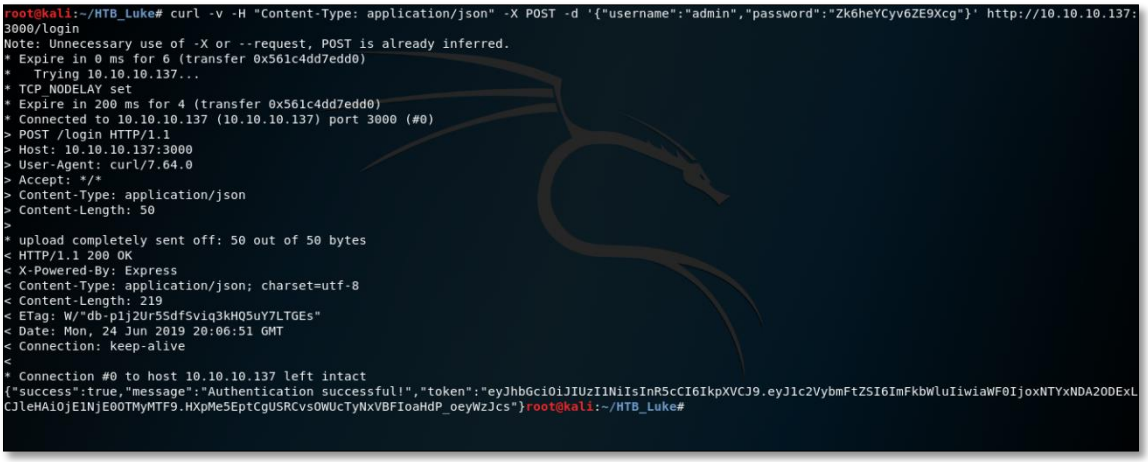

Ilustración 29: Petición POST con usuario admin al endpoint login.

Como bien se mostró anteriormente el usuario *root* no existe, por lo que si se realizaba la petición a la API con dicho usuario daba error. Pero con el usuario *admin* la API nos devolvía un token de autentificación, lo que implica que ya se podría tener acceso a la ruta de ese usuario haciendo peticiones con el token obtenido.

Para ejecutar correctamente la petición POST con el token de autentificación se tomo como referencia la fuente: https://auth0.com/learn/token-based-authentication-made-easy/

Ya que la clave para tener éxito en la petición era añadir: Authorization: `Bearer {TOKEN}`

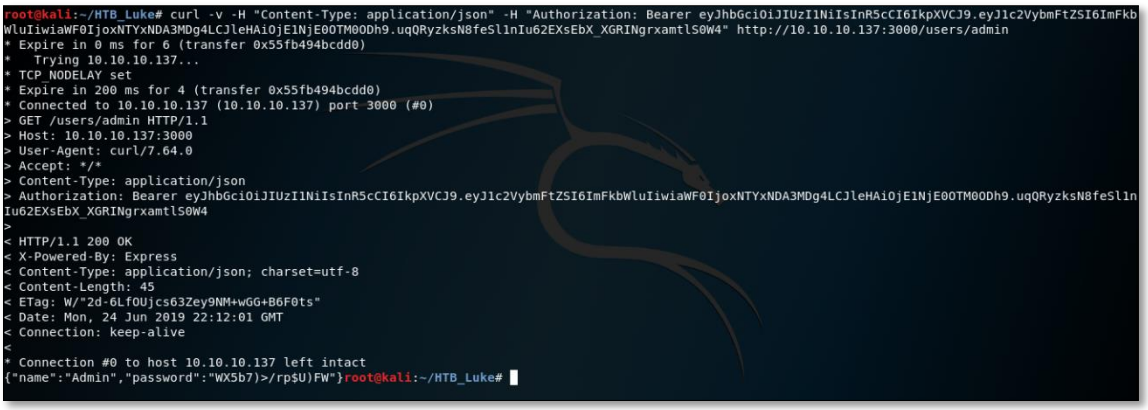

Ilustración 30: Petición POST con el token de autentificación en la ruta del usuario admin.

La API proporcionó una contraseña para el usuario *Admin*, que fue probada en todos los paneles de inicio de sesión conocidos hasta ese momento. En ninguno de ellos se consiguió acceder con esa combinación de usuario y contraseña.

Sabiendo que existe otro usuario llamado *Derry*, se ejecutó otra petición POST con el mismo token que se había obtenido, pero esta vez a la ruta del usuario mencionado:

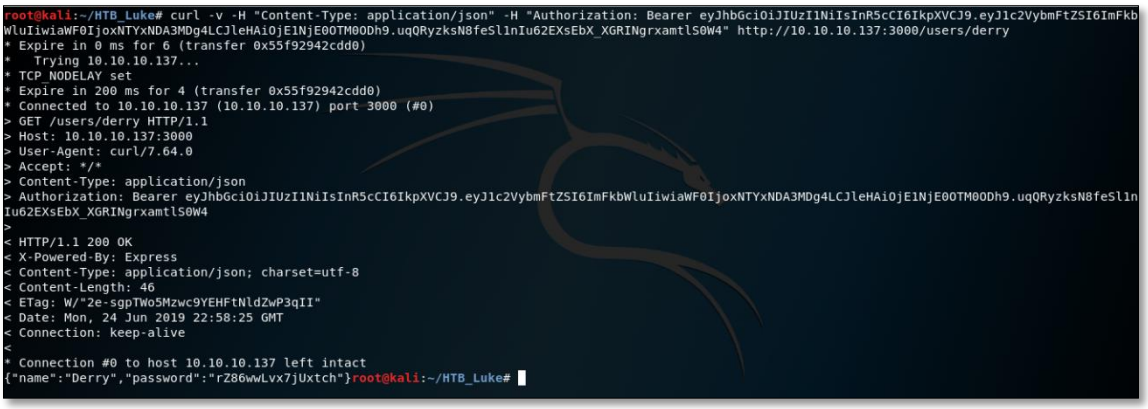

Ilustración 31: Petición POST con el token de autentificación en la ruta del usuario derry.

La contraseña obtenida para el usuario *Derry*, fue probada en todos los paneles de inicio de sesión conocidos, en ninguno se consiguió validar tal combinación excepto en http://10.10.10.137/management:

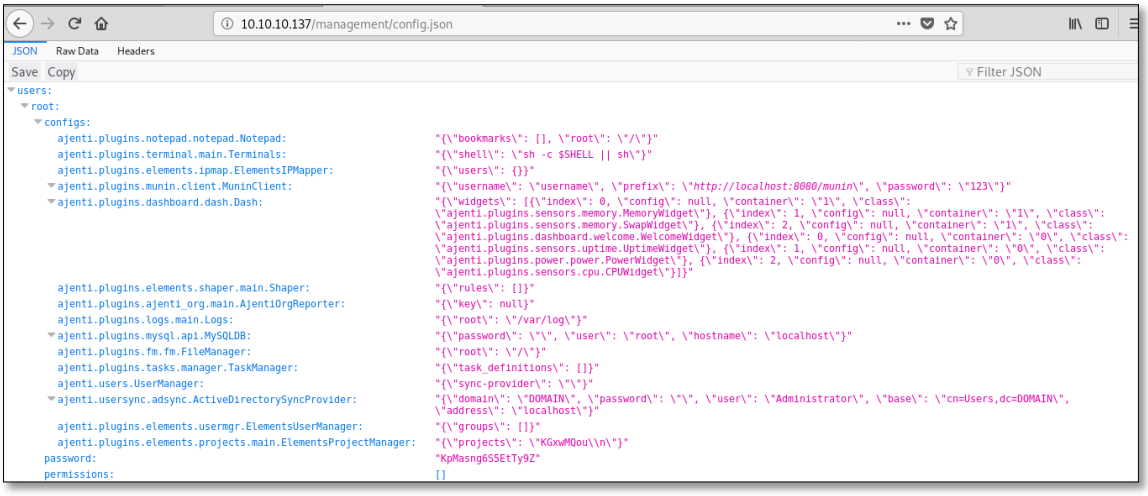

Ilustración 32: Fichero config.json.

Al introducir las credenciales del usuario *Derry* se tenía acceso a un fichero con extensión JSON en el cuál existía una contraseña para un usuario *root*.

Nuevamente se volvió a probar dicha combinación de usuario y contraseña en todos los paneles de login que se conocía, funcionando únicamente en la web que daba servicio en el puerto 8000 (*Ajenti*).

La posible explicación de que ahora exista un usuario *root* cuando antes la API no lo devolvía como un usuario existente, es que la única web que haga uso de ésta sea la que se encuentra en el puerto 80 y no la del 8000.

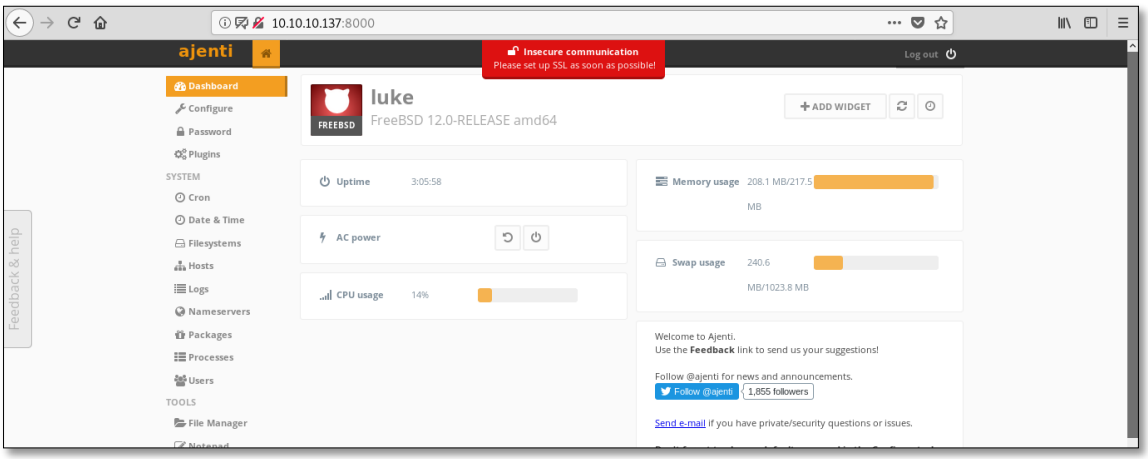

Ilustración 33: Acceso como administrador a Ajenti.

Una vez dentro fue fácil acceder a los ficheros del sistema y obtener las *flags* ya que existía una herramienta denominada “File Manager” que permitía navegar por los directorios de la máquina.

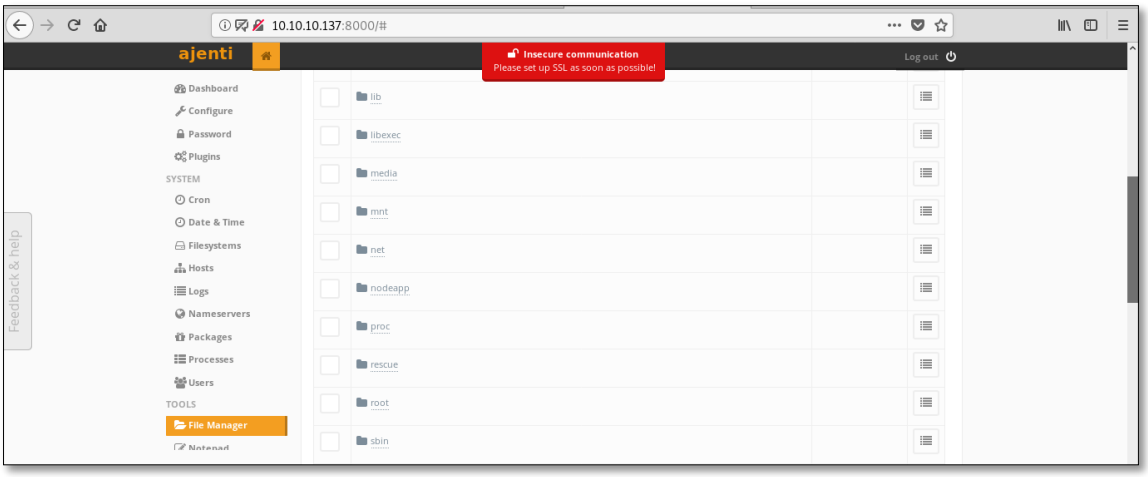

Ilustración 34: Directorios de la máquina.

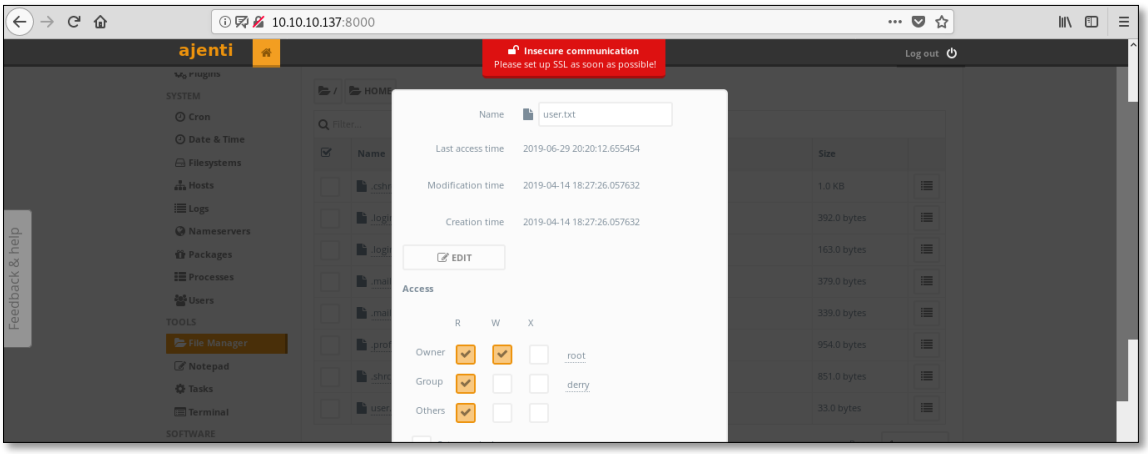

Ilustración 35: Abriendo fichero user.txt.

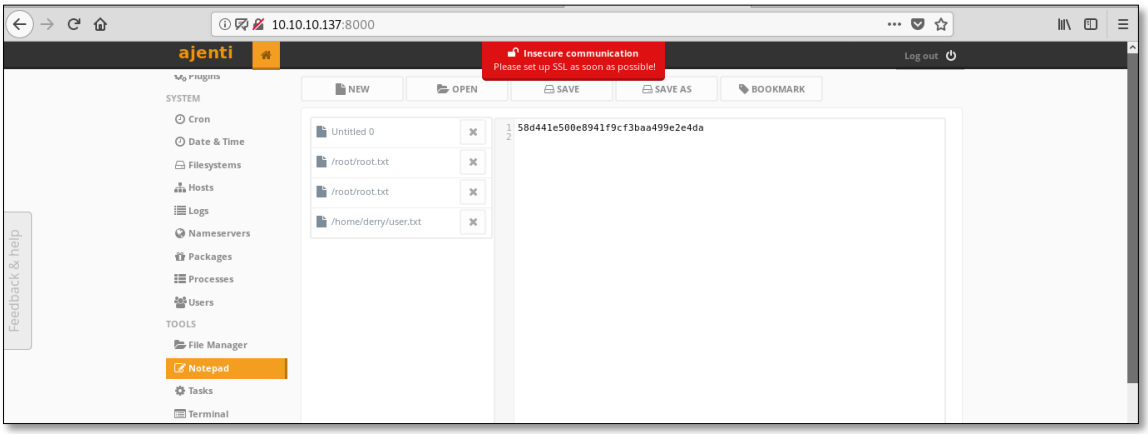

Ilustración 36: Flag del user.

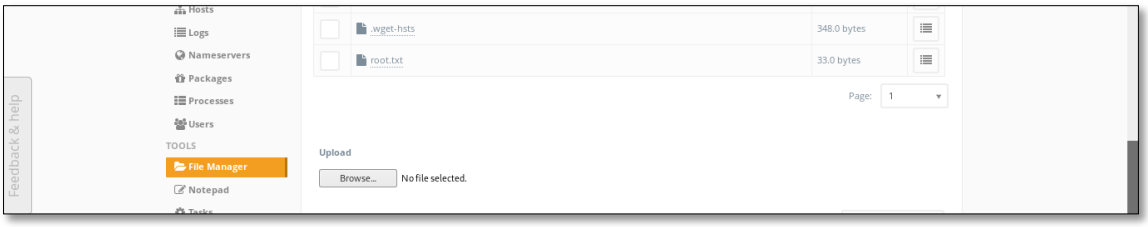

Ilustración 37: Fichero root.txt

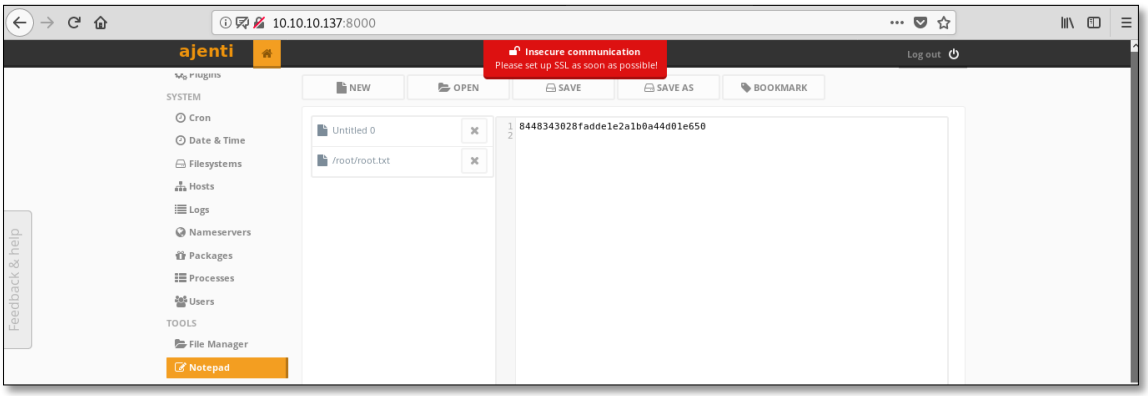

Ilustración 38: Flag del usuario root.

Además, también había otra herramienta que abría una consola como administrador en el sistema, pudiendo navegar por el árbol de directorios más fácilmente:

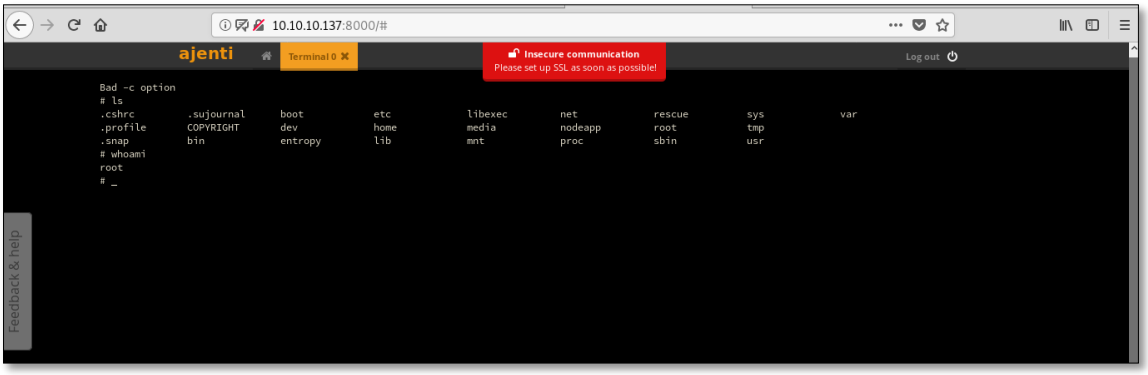

Ilustración 39: Consola web.

Como conclusión se podría decir que la dificultad de la máquina está en enumerar la gran cantidad de paneles de inicio de sesión que existen, así como comprobar de una manera ordenada todos los usuarios y contraseñas que se van obteniendo. Porque una vez se tiene acceso a la web que está en el puerto 8000 es muy fácil obtener las *flags*.