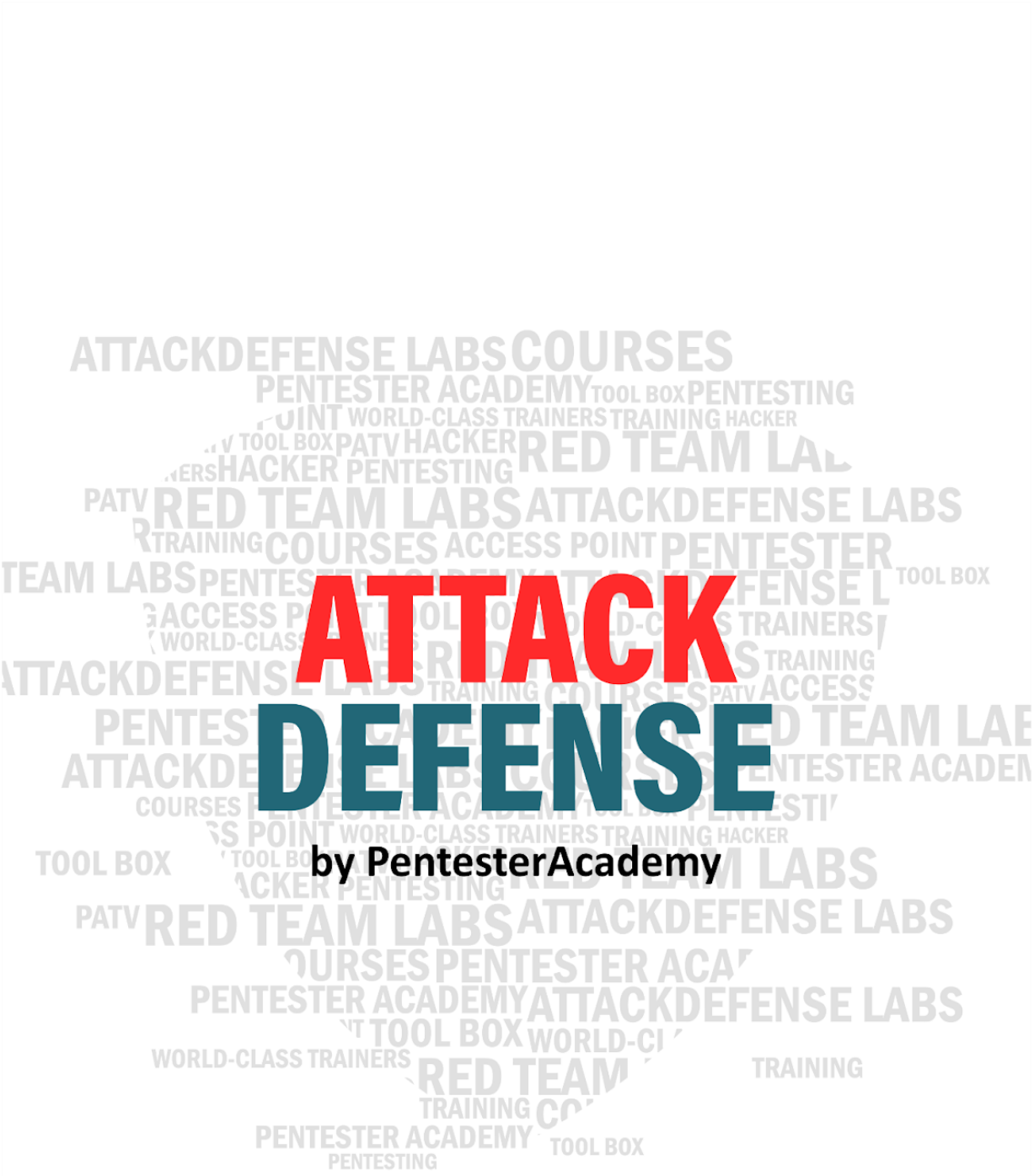

ATTACK
DEFENSE
by PentesterAcademy

Name	Credentials Recovery
URL	https://attackdefense.com/challengedetails?cid=1456
Type	Docker Security : Docker Forensics

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. Also, note that the IP addresses and domain names might be different in your lab.

Objective: Retrieve the credentials passed by the user to "authenticator" process!

Solution:

Step 1: Check all running and stopped containers

Command: docker ps -a

```
root@localhost:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                       PORTS               NAMES
86b773dbfe1c        lamp-wordpress      "./run.sh"          11 minutes ago      Exited (137) 10 minutes ago                      wordpress
root@localhost:~#
```

A wordpress container is stopped on the host.

Step 2: Start the container and check if "authenticator" process is running.

Commands:
docker start wordpress
ps -ef | grep authenticator

```
root@localhost:~# docker start wordpress
wordpress
root@localhost:~# ps -ef | grep authenticator
root      1480  1332  0 08:34 pts/0    00:00:00 grep --color=auto authenticator
root@localhost:~#
```

Authenticator process is not running on the machine.

Kindly note that all the processes running in the container are also visible on host machine, hence one can directly check the running process list on host instead of the container.

Step 4: Check checkpoints available for this container

Commands: docker checkpoint ls wordpress

```
root@localhost:~# docker checkpoint ls wordpress
CHECKPOINT NAME
checkpoint2-initial
root@localhost:~#
```

A checkpoint is available for this container.

Step 5: Stop the running container.

Command: docker stop wordpress

```
root@localhost:~# docker stop wordpress
wordpress
root@localhost:~#
```

Step 6: Start it again with available checkpoint and verify that the container has started.

Command: docker start --checkpoint checkpoint2-initial wordpress

```
root@localhost:~# docker start --checkpoint checkpoint2-initial wordpress
root@localhost:~#
root@localhost:~#
root@localhost:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
86b773dbfe1c        lamp-wordpress      "./run.sh"          14 minutes ago      Up 10 seconds       80/tcp              wordpress
root@localhost:~#
```

Step 7: Check if “authenticator” process is running. Please remember that the container will take some time to start all the processes and operate normally

Commands: ps -ef | grep “authenticator”

```
root@localhost:~# ps -ef | grep authenticator
root      2016  2014  0 08:37 ?        00:00:00 authenticator
root      2118  1332  0 08:38 pts/0    00:00:00 grep --color=auto authenticator
root@localhost:~#
```

The process is running with PID 2016. Please remember that the process is running inside the wordpress container.

Step 8: GDB is installed on the machine and can be used to take memory dump of the process. Check the process memory map at cat /proc/[pid]/maps

Command: cat /proc/2016/maps

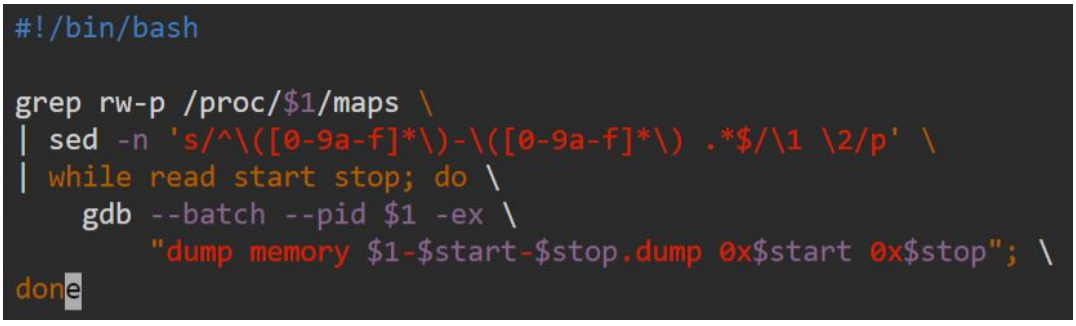
```
root@localhost:~# cat /proc/2016/maps
55cf77416000-55cf77417000 r-xp 00000000 08:00 26251                      /usr/bin/authenticator
55cf77616000-55cf77617000 r--p 00000000 08:00 26251                      /usr/bin/authenticator
55cf77617000-55cf77618000 rw-p 00001000 08:00 26251                      /usr/bin/authenticator
7f39e08c6000-7f39e0a81000 r-xp 00000000 08:00 140690                     /lib/x86_64-linux-gnu/libc-2.19.so
7f39e0a81000-7f39e0c80000 ---p 001bb000 08:00 140690                     /lib/x86_64-linux-gnu/libc-2.19.so
7f39e0c80000-7f39e0c84000 r--p 001ba000 08:00 140690                     /lib/x86_64-linux-gnu/libc-2.19.so
7f39e0c84000-7f39e0c86000 rw-p 001be000 08:00 140690                     /lib/x86_64-linux-gnu/libc-2.19.so
7f39e0c86000-7f39e0c8b000 rw-p 00000000 00:00 0
7f39e0c8b000-7f39e0cae000 r-xp 00000000 08:00 140668                     /lib/x86_64-linux-gnu/ld-2.19.so
7f39e0ea4000-7f39e0ea7000 rw-p 00000000 00:00 0
7f39e0ea9000-7f39e0ead000 rw-p 00000000 00:00 0
7f39e0ead000-7f39e0eae000 r--p 00022000 08:00 140668                     /lib/x86_64-linux-gnu/ld-2.19.so
7f39e0eae000-7f39e0eaf000 rw-p 00023000 08:00 140668                     /lib/x86_64-linux-gnu/ld-2.19.so
7f39e0eaf000-7f39e0eb0000 rw-p 00000000 00:00 0
7fff4d30f000-7fff4d330000 rw-p 00000000 00:00 0                          [stack]
7fff4d377000-7fff4d37a000 r--p 00000000 00:00 0                          [vvar]
7fff4d37a000-7fff4d37b000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0                  [vsyscall]
root@localhost:~#
```

There are multiple batches of memory. One can either dump one of these selectively or dump them all iteratively. Use the following to do the latter.

Script:

```
#!/bin/bash

grep rw-p /proc/$1/maps \
| sed -n 's/^\([0-9a-f]*\)-\([0-9a-f]*\) .*$/\1 \2/p' \
| while read start stop; do \
    gdb --batch --pid $1 -ex \
        "dump memory $1-$start-$stop.dump 0x$start 0x$stop"; \
done
```



Source: https://serverfault.com/questions/173999/dump-a-linux-processs-memory-to-file

Save this script as dump-memory.sh

Step 9: Make this script executable and then execute it to dump the memory batches.

Commands:

```
chmod +x dump-memory.sh
./dump-memory.sh 2016
```

```
root@localhost:~# chmod +x dump-memory.sh
root@localhost:~# ./dump-memory.sh 2016

warning: Target and debugger are in different PID namespaces; thread lists and other data are likely unreliable.  Connect to gdbserver inside t
he container.
0x00007f39e0986f20 in nanosleep () from target:/lib/x86_64-linux-gnu/libc.so.6

warning: Target and debugger are in different PID namespaces; thread lists and other data are likely unreliable.  Connect to gdbserver inside t
he container.
0x00007f39e0986f20 in nanosleep () from target:/lib/x86_64-linux-gnu/libc.so.6
```

GDB will throw some warnings which can be ignored.

Step 10: Check the dump files.

Commands: ls -l

```
root@localhost:~# ls -l
total 204
-rw-r--r-- 1 root root   4096 Dec  2 08:39 2016-55cf77617000-55cf77618000.dump
-rw-r--r-- 1 root root   8192 Dec  2 08:39 2016-7f39e0c84000-7f39e0c86000.dump
-rw-r--r-- 1 root root  20480 Dec  2 08:39 2016-7f39e0c86000-7f39e0c8b000.dump
-rw-r--r-- 1 root root  12288 Dec  2 08:39 2016-7f39e0ea4000-7f39e0ea7000.dump
-rw-r--r-- 1 root root  16384 Dec  2 08:39 2016-7f39e0ea9000-7f39e0ead000.dump
-rw-r--r-- 1 root root   4096 Dec  2 08:39 2016-7f39e0eae000-7f39e0eaf000.dump
-rw-r--r-- 1 root root   4096 Dec  2 08:39 2016-7f39e0eaf000-7f39e0eb0000.dump
-rw-r--r-- 1 root root 135168 Dec  2 08:40 2016-7fff4d30f000-7fff4d330000.dump
-rwxr-xr-x 1 root root    224 Dec  2 08:39 dump-memory.sh
root@localhost:~#
```

Step 11: Run strings command on these dumps and look for the keyword “password”.

Commands: strings *.dump | grep -i PASSWORD

```
root@localhost:~# strings *.dump | grep -i PASSWORD
 Username:  Password:  USERNAME=root&PASSWORD=ironsmith
USERNAME=root&PASSWORD=ironsmith
root@localhost:~#
```

And, the credentials are present in one of the dumps.

Username: root
Password: ironsmith