Cybersecurity Lab Report – 02

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.

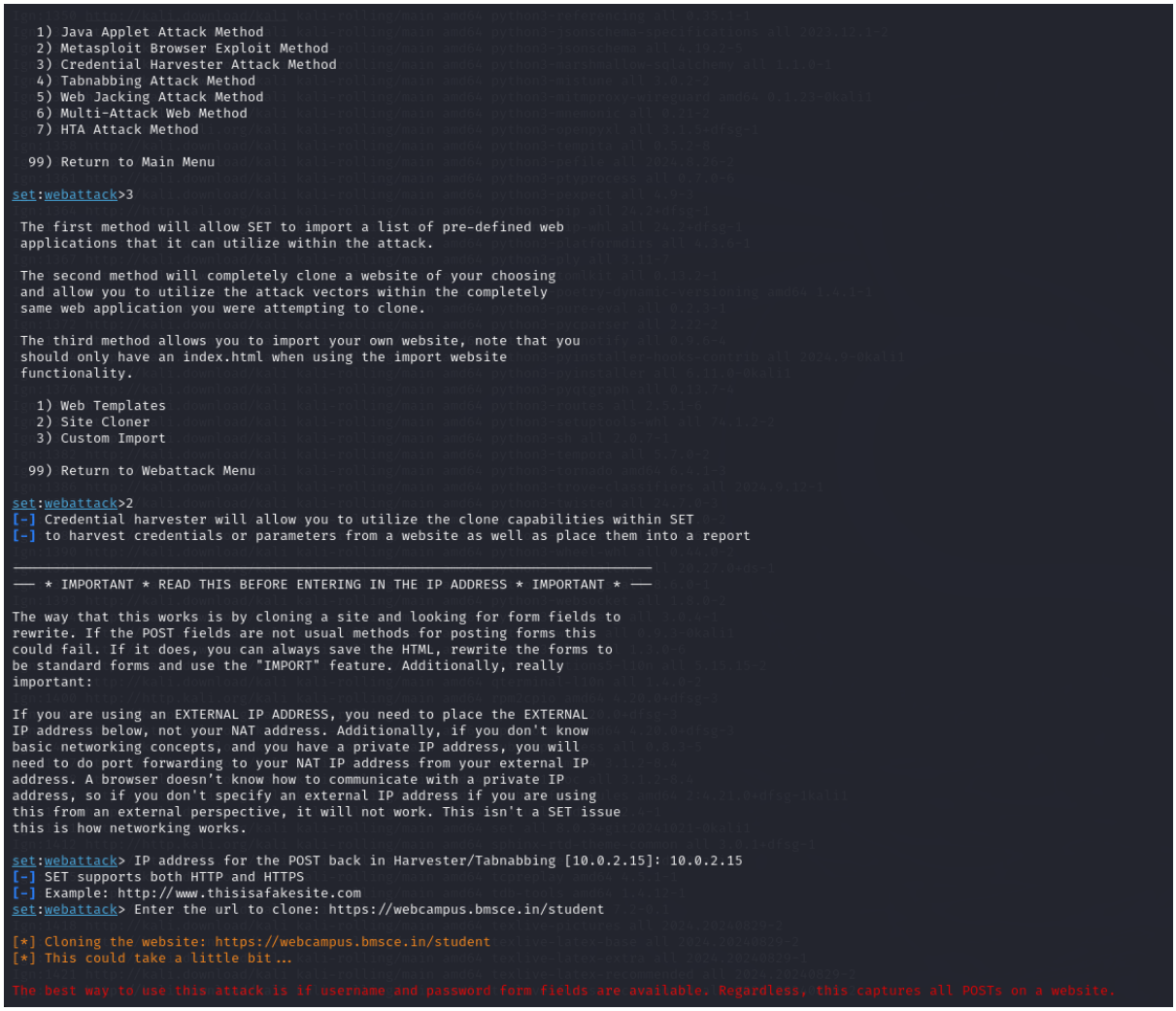
2) Afterward, it redirects the user to the real website, making it difficult for the victim to realize they've been compromised.

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

sudo setoolkit

The output was as follows:

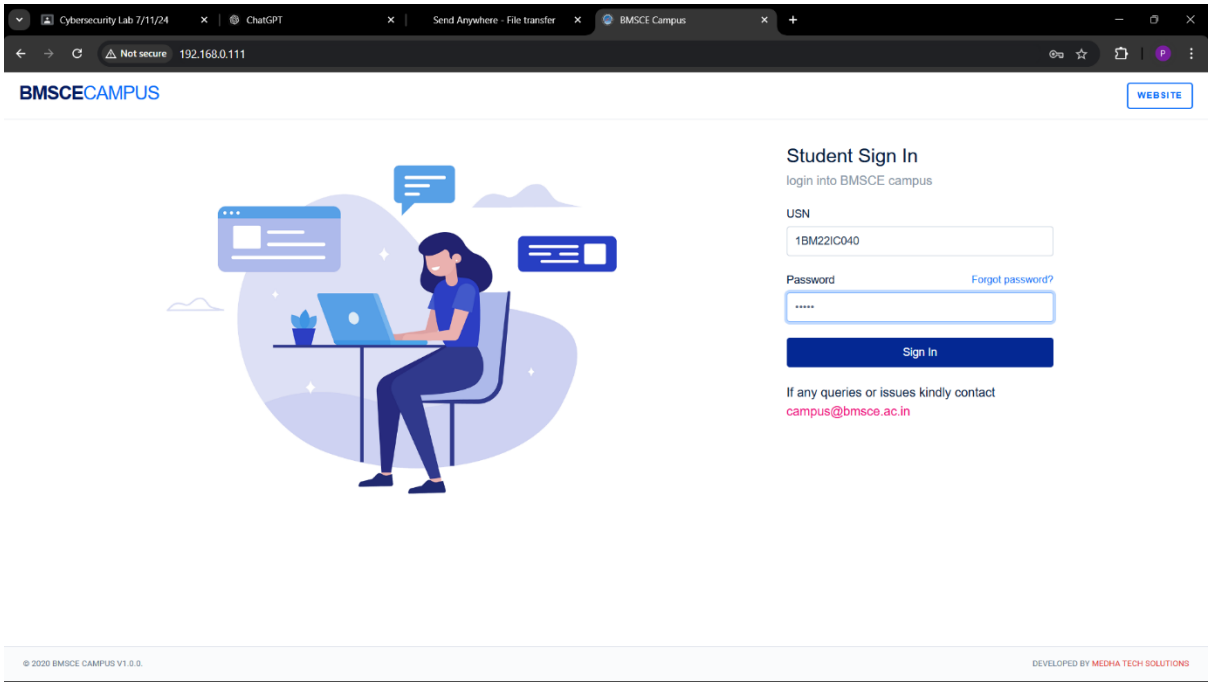
In the screenshot above, I was prompted to select the attack vector I wanted to use. Since I opted for a website-based attack, I chose option 2.



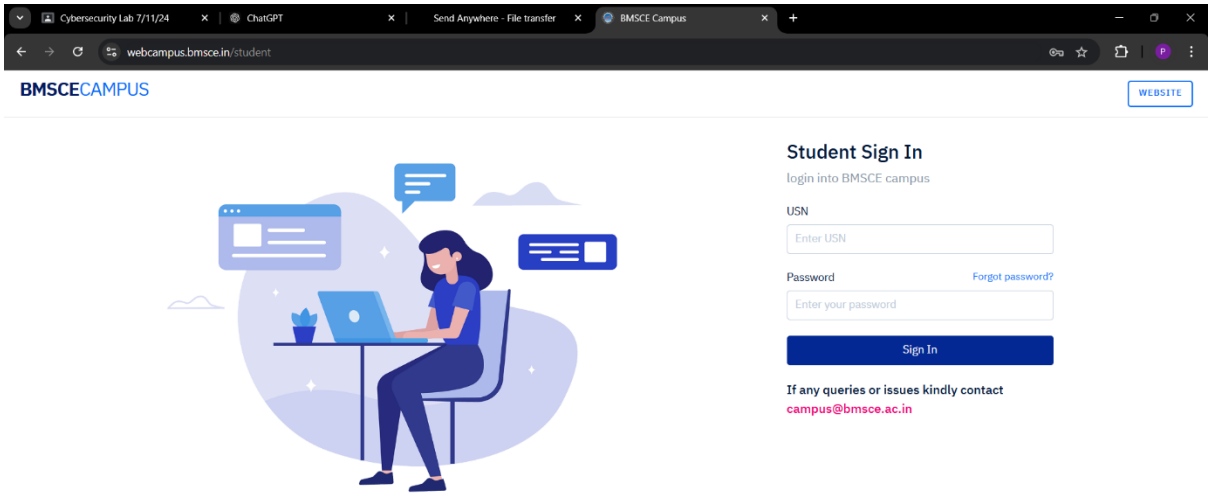
I was then asked to select the type of attack I wanted to perform. Since I intended to do a credential stuffing attack, I chose option 3. Next, I was asked which method to use, and because I was cloning a website, I selected option 2.

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.

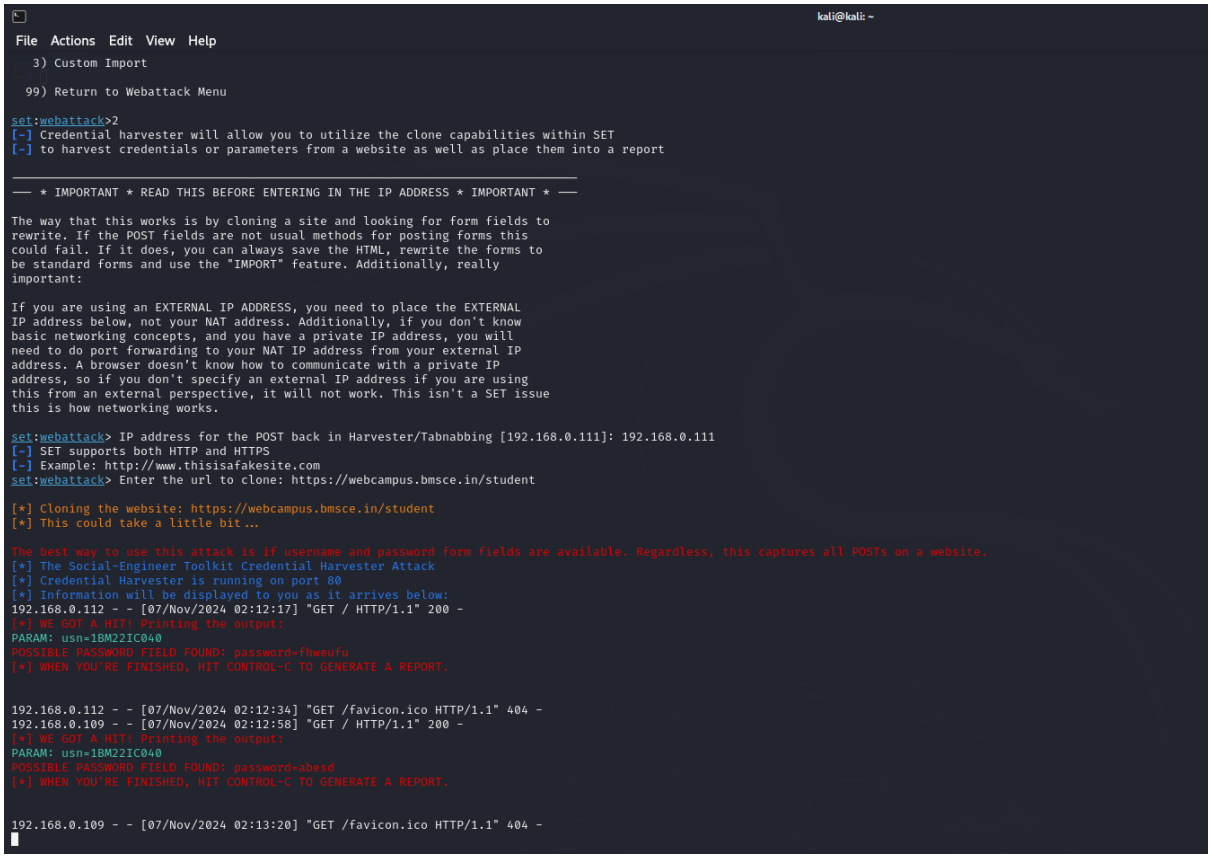
Finally, I went to my host machine, typed the IP address of the Kali Linux VM into my browser, and proceeded to enter my credentials, as shown in the results below.



here I was redirected to the original website



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



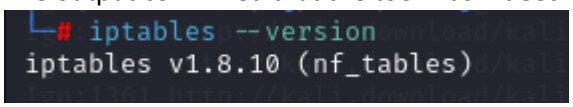
This concludes the setoolkit part

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.



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

sudo iptables -L -v -n

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 ipables -L -v -n and got the following output

```
┌──(root㉿kali)-[/home/kali]
└─# sudo iptables -P INPUT DROP

┌──(root㉿kali)-[/home/kali]
└─# sudo iptables -L -v -n
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

┌──(root㉿kali)-[/home/kali]
└─# sudo iptables -P FORWARD DROP

┌──(root㉿kali)-[/home/kali]
└─# 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

┌──(root㉿kali)-[/home/kali]
└─# sudo iptables -P OUTPUT ACCEPT

┌──(root㉿kali)-[/home/kali]
└─# sudo iptables -L -v -n
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
```

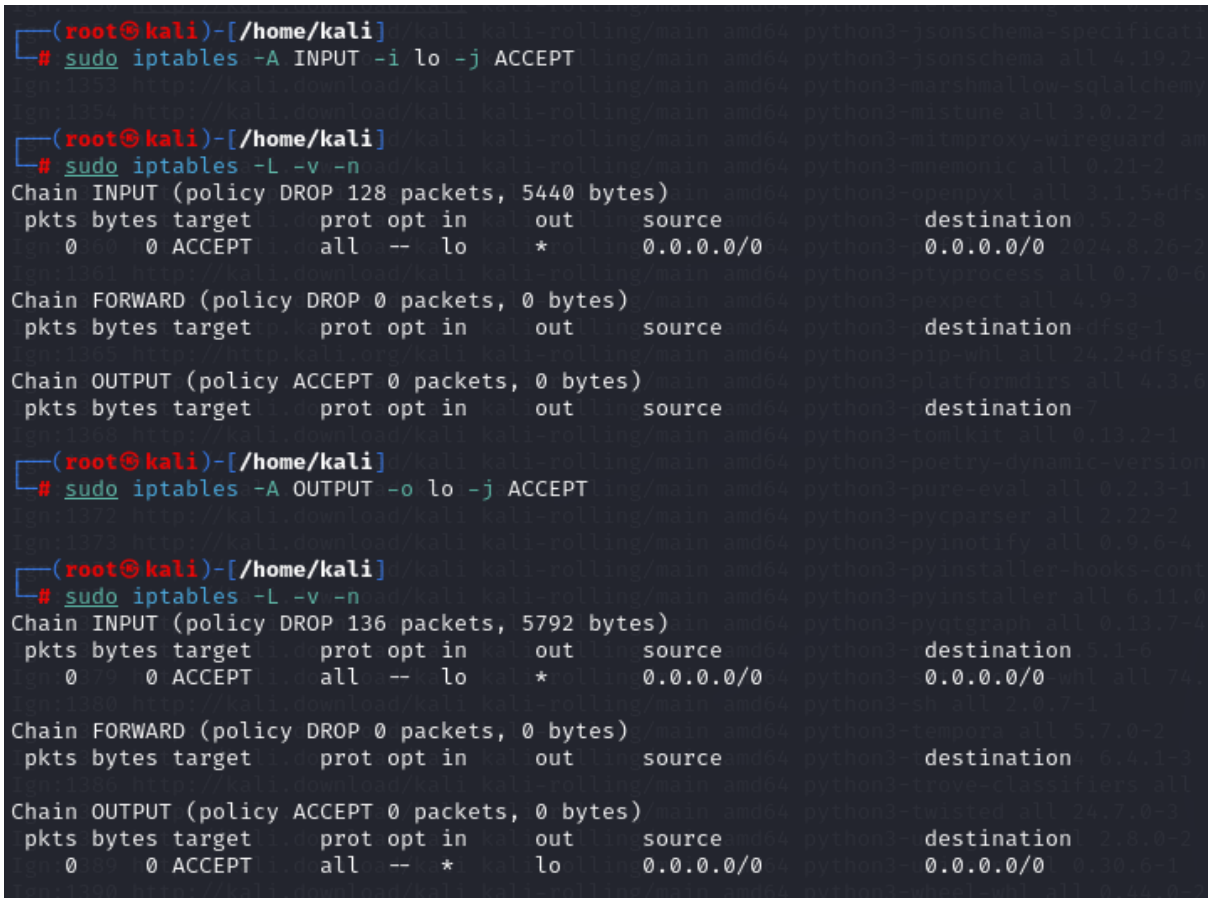
We observed that the packet drop count increased when the DROP INPUT and DROP FORWARD commands were executed. To enable the loopback interface, I ran the following commands and received the corresponding firewall rules:

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



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

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)-[/home/kali]
└─# sudo iptables -A INPUT -p tcp --dport 22 -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
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:22

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

┌──(root㉿kali)-[/home/kali]
└─# sudo iptables -A INPUT -p tcp --dport 80 -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
    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

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

┌──(root㉿kali)-[/home/kali]
└─# sudo iptables -A INPUT -p tcp --dport 443 -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
    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

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 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 respong 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 146 packets, 6224 bytes)
 pkts bytes target     prot opt in     out     source               destination
    6   504 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
    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           icmptype 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  --  *      lo      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.