

Simulate TCP-SYN flooding with IP Spoofing between Vms

Introduction.....	1
Simulation Environment.....	1
Logical network topology.....	2
Comparison between different programming.....	2
Verify the attack tool.....	4
Syn attack example.....	5
Syn attack with IP spoofing and dynamic source port.....	5
Syn attack with IP spoofing and static source port.....	6
verify the attack with netstat.....	7
attack from static ip address.....	7
attack from dynamic source ip and source port.....	7
quantity of tcp syn-rev.....	7
verify the attack effect.....	8

Introduction

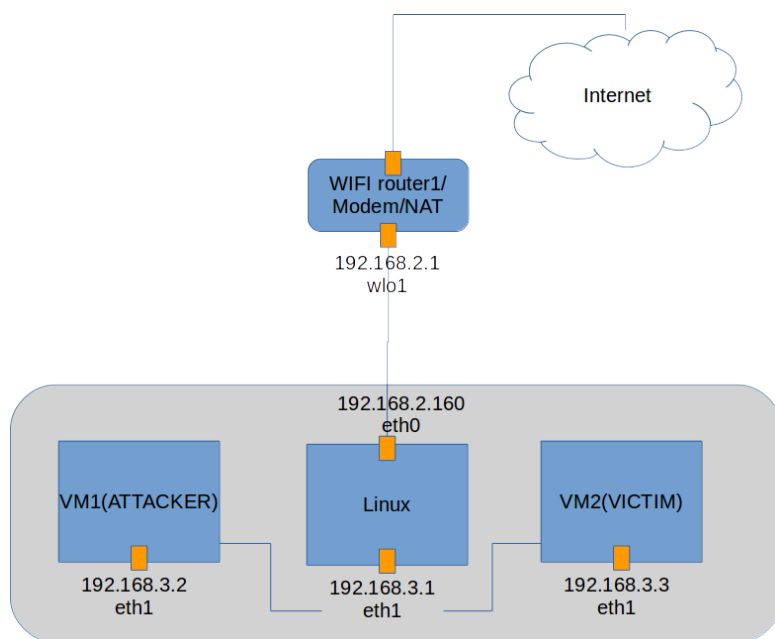
The purpose of this simulation is to understand how TCP-SYN flooding works and its effect.

Simulation Environment

Host machine hardware info				VM Guest machine hardware info			
H/W path	Device	Class	Description	vagrant@victim:~\$ sudo lshw -short			
				H/W path	Device	Class	Description
		system	Computer			system	VirtualBox ()
/0		bus	Motherboard			bus	VirtualBox
/0/0		memory	15GiB System memory	/0		memory	128KiB BIOS
/0/1		processor	Intel(R) Core(TM) i7-4700MQ CPU @ 2.40GHz	/0/0		memory	363MiB System memory
				/0/1		processor	Intel(R) Core(TM) i7-4700MQ CPU @ 2.40GHz
				/0/2			
Host cpu info				Guest cpu info			

<pre> exinton@exinton-HP-ZBook-15:~/OpenStackCookbook\$ lscpu Architecture: x86_64 CPU op-mode(s): 32-bit, 64-bit Byte Order: Little Endian CPU(s): 8 On-line CPU(s) list: 0-7 Thread(s) per core: 2 Core(s) per socket: 4 Socket(s): 1 NUMA node(s): 1 Vendor ID: GenuineIntel CPU family: 6 Model: 60 Model name: Intel(R) Core(TM) i7-4700MQ CPU @ 2.40GHz Stepping: 3 CPU MHz: 3341.812 CPU max MHz: 3400.0000 CPU min MHz: 800.0000 BogoMIPS: 4789.04 Virtualization: VT-x L1d cache: 32K L1i cache: 32K L2 cache: 256K L3 cache: 6144K NUMA node0 CPU(s): 0-7 </pre>	<pre> vagrant@victim:~\$ lscpu Architecture: x86_64 CPU op-mode(s): 32-bit, 64-bit Byte Order: Little Endian CPU(s): 2 On-line CPU(s) list: 0,1 Thread(s) per core: 1 Core(s) per socket: 2 Socket(s): 1 NUMA node(s): 1 Vendor ID: GenuineIntel CPU family: 6 Model: 60 Stepping: 3 CPU MHz: 2394.286 BogoMIPS: 4788.57 Virtualization: VT-x L1d cache: 32K L1i cache: 32K L2 cache: 256K L3 cache: 6144K NUMA node0 CPU(s): 0,1 </pre>
---	---

Logical network topology



Comparison between different programming

I tried Java, C and Python for the TCP flooding client.

Since Java doesn't support raw socket, an alternative way to realize tcp flooding in java is JNI.

My first draft was C, and improved to multi-threading later.

Another version is written in python, I add multi-threading as well.

In order to collect the Syn-received msg easily, I use a ruby script to fetch the data automatically.

Source code file version:

Source File name	Main function	feature	Attack effect
synflood_ipspoofing.c	Initial attack	rawsocket	Longest DoS time
Syn.c	Initial attack	Multi-threading	medium DoS time
Tcp-flooding.py	Initial attack	rawsocket	medium DoS time
Tcp-flooding-multi-threading.py	Initial attack	Multi-threading	Longest DoS time
check_syn_flood.rb	Counter the syn-recv TCP session number		

Test environment preparation:

1. Verify 80 port on virtual machine victim

```
vagrant@victim:~$ netstat -l
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 *:sunrpc                *:*                     LISTEN
tcp        0      0 *:http                  *:*                     LISTEN
tcp        0      0 *:38132                  *:*                     LISTEN
tcp        0      0 *:ssh                   *:*                     LISTEN
tcp6       0      0 [::]:sunrpc             [::]:*                  LISTEN
tcp6       0      0 [::]:ssh                 [::]:*                  LISTEN
tcp6       0      0 [::]:44956               [::]:*                  LISTEN
udp        0      0 *:39738                  *:*                     LISTEN
udp        0      0 *:bootpc                *:*                     LISTEN
udp        0      0 *:bootpc                *:*                     LISTEN
udp        0      0 localhost:849            *:*                     LISTEN
udp        0      0 *:sunrpc                *:*                     LISTEN
udp        0      0 192.168.3.3:ntp         *:*                     LISTEN
udp        0      0 10.0.2.15:ntp           *:*                     LISTEN
udp        0      0 localhost:ntp            *:*                     LISTEN
udp        0      0 *:ntp                   *:*                     LISTEN
udp        0      0 *:805                    *:*                     LISTEN
udp6       0      0 [::]:sunrpc             [::]:*                  LISTEN
udp6       0      0 lp6-localhost:ntp       [::]:*                  LISTEN
udp6       0      0 fe80::a00:27ff:fe80:ntp [::]:*                  LISTEN
udp6       0      0 fe80::a00:27ff:fe80:ntp [::]:*                  LISTEN
udp6       0      0 [::]:ntp                 [::]:*                  LISTEN
udp6       0      0 [::]:53016               [::]:*                  LISTEN
udp6       0      0 [::]:805                 [::]:*                  LISTEN
Active UNIX domain sockets (only servers)
Proto RefCnt Flags   Type       State       I-Node  Path
unix   2      [ ACC ] STREAM    LISTENING   7436    /var/run/dbus/system_
bus_socket
unix   2      [ ACC ] STREAM    LISTENING   7167    @/com/ubuntu/upstart
unix   2      [ ACC ] STREAM    LISTENING   8264    /run/rpcbind.sock
unix   2      [ ACC ] STREAM    LISTENING   9373    /var/run/apache2/cgls
ock.933
unix   2      [ ACC ] SEQPACKET LISTENING   1531    /run/udev/control
vagrant@victim:~$
```

2. set up vm via vagrant

```
exinton@exinton-HP-ZBook-15: ~/workspace/python-tcp-flooding
exinton@exinton-HP-ZBook-15:~/workspace/python-tcp-flooding$ vagrant global-status
id      name      provider  state    directory
-----
e055d31 controller virtualbox poweroff  /home/exinton/openstack
e4f91ba default   virtualbox poweroff  /home/exinton/mininet
a2f22b6 default   virtualbox running   /home/exinton/Project/tcpsynflood
c2453ad default   virtualbox running   /home/exinton/Project/victim
d73abb0 default   virtualbox poweroff  /home/exinton/Project/mysql

The above shows information about all known Vagrant environments
on this machine. This data is cached and may not be completely
up-to-date. To interact with any of the machines, you can go to
that directory and run Vagrant, or you can use the ID directly
with Vagrant commands from any directory. For example:
"vagrant destroy 1a2b3c4d"
exinton@exinton-HP-ZBook-15:~/workspace/python-tcp-flooding$
```

3.open 80 port on victim vm

```
vagrant@victim: ~
vagrant@victim:~$ netstat -l
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp    0      0 *:sunrpc                 *:*                     LISTEN
tcp    0      0 *:http                   *:*                     LISTEN
tcp    0      0 *:38132                  *:*                     LISTEN
tcp    0      0 *:ssh                     *:*                     LISTEN
tcp6   0      0 [::]:sunrpc              [::]:*                  LISTEN
tcp6   0      0 [::]:ssh                  [::]:*                  LISTEN
tcp6   0      0 [::]:44956                [::]:*                  LISTEN
udp    0      0 *:39738                  *:*                     LISTEN
udp    0      0 *:bootpc                 *:*                     LISTEN
udp    0      0 *:bootpc                 *:*                     LISTEN
udp    0      0 localhost:849             *:*                     LISTEN
udp    0      0 *:sunrpc                 *:*                     LISTEN
udp    0      0 192.168.3.3:ntp          *:*                     LISTEN
udp    0      0 10.0.2.15:ntp            *:*                     LISTEN
udp    0      0 localhost:ntp             *:*                     LISTEN
udp    0      0 *:ntp                    *:*                     LISTEN
udp    0      0 *:805                     *:*                     LISTEN
udp6   0      0 [::]:sunrpc              [::]:*                  LISTEN
udp6   0      0 ip6-localhost:ntp        [::]:*                  LISTEN
udp6   0      0 fe80::a00:27ff:fe88:ntp  [::]:*                  LISTEN
udp6   0      0 fe80::a00:27ff:fe80:ntp  [::]:*                  LISTEN
udp6   0      0 [::]:ntp                 [::]:*                  LISTEN
udp6   0      0 [::]:53016               [::]:*                  LISTEN
udp6   0      0 [::]:805                  [::]:*                  LISTEN
```

Verify the attack tool

In order to verify the attack works well, several tests are needed. The simplest way is to initial an syn message to victim from a real IP address, and capture the reply from the victim.

The to verify the attack tool is that sometimes the wrong checksum in TCP part may lead to victim dropping the syn message. The Checksum part in IP header doesn't matter since Linux kernel will fill it automatically.

In the following picture, the victim(192.168.3.2) replies syn,ack as response, which proofs that the attack tool works perfect.

```
19 0.000194 192.168.3.3 192.168.3.2 TCP 58 80→5678 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0
  Frame 19: 58 bytes on wire (464 bits), 58 bytes captured (464 bits)
  Ethernet II, Src: CadmusCo_f0:84:31 (08:00:27:f0:84:31), Dst: CadmusCo_9f:f7:74 (08:00:27:9f:f7:74)
  Internet Protocol Version 4, Src: 192.168.3.3 (192.168.3.3), Dst: 192.168.3.2 (192.168.3.2)
    Version: 4
    Header Length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable))
    Total Length: 44
    Identification: 0x0000 (0)
    Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (6)
    Header checksum: 0xb376 [validation disabled]
    Source: 192.168.3.3 (192.168.3.3)
    Destination: 192.168.3.2 (192.168.3.2)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
  Transmission Control Protocol, Src Port: 80 (80), Dst Port: 5678 (5678), Seq: 0, Ack: 1, Len: 0
    Source Port: 80 (80)
    Destination Port: 5678 (5678)
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 0 (relative sequence number)
    Acknowledgment number: 1 (relative ack number)
    Header Length: 24 bytes
    .... 0000 0001 0010 = Flags: 0x012 (SYN, ACK)
    Window size value: 14600
    [Calculated window size: 14600]
    Checksum: 0x8774 [validation disabled]
    Urgent pointer: 0
    Options: (4 bytes), Maximum segment size
    [SEQ/ACK analysis]
```

Syn attack example

Syn attack with IP spoofing and dynamic source port

This attack dynamically changes its source port to prevent detection.

As the following diagram indicates, the source port is different at each syn attack.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.3.107	192.168.3.3	TCP	60	55793->80 [SYN] Seq=0 Win=53270 Len=0
2	0.000208	192.168.3.54	192.168.3.3	TCP	60	50645->80 [SYN] Seq=0 Win=53270 Len=0
3	0.000413	192.168.3.222	192.168.3.3	TCP	60	50746->80 [SYN] Seq=0 Win=53270 Len=0
4	0.000627	192.168.3.80	192.168.3.3	TCP	60	59721->80 [SYN] Seq=0 Win=53270 Len=0
5	0.000838	192.168.3.241	192.168.3.3	TCP	60	37371->80 [SYN] Seq=0 Win=53270 Len=0
6	0.001050	192.168.3.215	192.168.3.3	TCP	60	40240->80 [SYN] Seq=0 Win=53270 Len=0
7	0.001261	192.168.3.110	192.168.3.3	TCP	60	50139->80 [SYN] Seq=0 Win=53270 Len=0
8	0.001595	192.168.3.92	192.168.3.3	TCP	60	48269->80 [SYN] Seq=0 Win=53270 Len=0
9	0.001835	192.168.3.250	192.168.3.3	TCP	60	39011->80 [SYN] Seq=0 Win=53270 Len=0
10	0.001847	192.168.3.228	192.168.3.3	TCP	60	55421->80 [SYN] Seq=0 Win=53270 Len=0
11	0.002134	192.168.3.161	192.168.3.3	TCP	60	49675->80 [SYN] Seq=0 Win=53270 Len=0

Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)	
Ethernet II, Src: CadmusCo 9f:f7:74 (08:00:27:9f:f7:74), Dst: CadmusCo f0:84:31 (08:00:27:f0:84:31)	
Internet Protocol Version 4, Src: 192.168.3.107 (192.168.3.107), Dst: 192.168.3.3 (192.168.3.3)	
Version: 4 Header Length: 20 bytes Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport)) Total Length: 40 Identification: 0xd431 (54321) Flags: 0x00 Fragment offset: 0 Time to live: 255 Protocol: TCP (6) Header checksum: 0x5fdf [validation disabled] Source: 192.168.3.107 (192.168.3.107) Destination: 192.168.3.3 (192.168.3.3) [Source GeoIP: Unknown] [Destination GeoIP: Unknown]	
Transmission Control Protocol, Src Port: 55793 (55793), Dst Port: 80 (80), Seq: 0, Len: 0	
Source Port: 55793 (55793) Destination Port: 80 (80) [Stream index: 0] [TCP Segment Len: 0] Sequence number: 0 (relative sequence number) Acknowledgment number: 0 Header Length: 20 bytes ... 0000 0000 0010 = Flags: 0x002 (SYN) Window size value: 53270 [Calculated window size: 53270] Checksum: 0x7dc0 [validation disabled] Urgent pointer: 0	

Syn attack with IP spoofing and static source port

The simplest syn attack which has fixed combination of source ip address and ip ports.

```
vagrant@victim: /vagrant
06:34:37.771804 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.771937 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.772122 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.772340 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.772483 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.772735 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.785658 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.785696 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.785706 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.785714 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.785720 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.785727 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.785735 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.785741 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.785746 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.785751 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.785756 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.785768 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.785774 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.785782 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.785789 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.785796 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.785859 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.785974 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.785981 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.785988 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.785996 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.786003 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.786010 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.786018 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.786025 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
06:34:37.786032 IP 192.168.3.12.5678 > 192.168.3.3.ssh: Flags [S], seq 0, win 65535, length 0
```


verify the attack with netstat

attack from static ip address

the netstat indicates that there is only one syn-recv since the ip address and ports combination is static.

```
vagrant@victim: /vagrant

vagrant@victim:/vagrant$ netstat -t
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp      0      0 192.168.3.3:ssh         192.168.3.12:5678      SYN_RECV
tcp      0      96 10.0.2.15:ssh           10.0.2.2:46992        ESTABLISHED
vagrant@victim:/vagrant$
```

attack from dynamic source ip and source port

the netstat indicates that coming ip address and ip ports changed dynamically.

```
vagrant@victim: ~

tcp      0      0 192.168.3.3:ssh         192.168.3.112:35183    SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.205:36504    SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.214:38437    SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.246:51527    SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.193:48623    SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.10:40397     SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.51:51062     SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.184:58096    SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.63:58523     SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.184:58096    SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.63:58523     SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.159:59046    SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.44:53507     SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.120:38122    SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.10:40851     SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.243:55215    SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.203:58349    SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.189:51554    SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.167:57936    SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.219:46007    SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.145:37678    SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.155:58755    SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.67:52353     SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.27:58228     SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.239:57449    SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.236:39860    SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.162:39899    SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.58:59063     SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.121:50811    SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.91:41980     SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.128:45087    SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.158:42133    SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.166:40388    SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.184:53675    SYN_RECV
tcp      0      0 192.168.3.3:ssh         192.168.3.102:55031    SYN_RECV
tcp      0      0 10.0.2.15:ssh           10.0.2.2:46992        ESTABLISHED
```

quantity of tcp syn-recv

With the help of ruby , we get the average of hanging tcp session is 60 when launching dynamic source ip attack.

```
vagrant@victim: ~  
vagrant@victim:~$ sudo ruby ./check_syn_flood.rb -w 500 -c 1000  
SYN Count: 64  
vagrant@victim:~$ sudo ruby ./check_syn_flood.rb -w 500 -c 1000  
SYN Count: 63  
vagrant@victim:~$ sudo ruby ./check_syn_flood.rb -w 500 -c 1000  
SYN Count: 62  
vagrant@victim:~$ sudo ruby ./check_syn_flood.rb -w 500 -c 1000  
SYN Count: 60  
vagrant@victim:~$ sudo ruby ./check_syn_flood.rb -w 500 -c 1000  
SYN Count: 59  
vagrant@victim:~$ sudo ruby ./check_syn_flood.rb -w 500 -c 1000  
SYN Count: 63  
vagrant@victim:~$ sudo ruby ./check_syn_flood.rb -w 500 -c 1000  
SYN Count: 58  
vagrant@victim:~$ sudo ruby ./check_syn_flood.rb -w 500 -c 1000  
SYN Count: 65  
vagrant@victim:~$ sudo ruby ./check_syn_flood.rb -w 500 -c 1000  
SYN Count: 67  
vagrant@victim:~$ sudo ruby ./check_syn_flood.rb -w 500 -c 1000  
SYN Count: 71  
vagrant@victim:~$ sudo ruby ./check_syn_flood.rb -w 500 -c 1000  
SYN Count: 69  
vagrant@victim:~$ sudo ruby ./check_syn_flood.rb -w 500 -c 1000  
SYN Count: 72  
vagrant@victim:~$ sudo ruby ./check_syn_flood.rb -w 500 -c 1000  
SYN Count: 74  
vagrant@victim:~$ sudo ruby ./check_syn_flood.rb -w 500 -c 1000  
SYN Count: 76  
vagrant@victim:~$
```

verify the attack effect

the average time of wget a file before syn flooding is about 0.04s.

```
2016-02-17 07:15:05 (32.9 MB/s) - 'uptime.log.20' saved [2316828/2316828]  
vagrant@attacker:/vagrant$ wget http://192.168.3.3/uptime.log  
--2016-02-17 07:15:05-- http://192.168.3.3/uptime.log  
Connecting to 192.168.3.3:80... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 2316828 (2.2M)  
Saving to: 'uptime.log.21'  
100%[=====] 2,316,828 --.-K/s tn 0.04s  
2016-02-17 07:15:06 (55.6 MB/s) - 'uptime.log.21' saved [2316828/2316828]  
vagrant@attacker:/vagrant$ wget http://192.168.3.3/uptime.log  
--2016-02-17 07:15:07-- http://192.168.3.3/uptime.log  
Connecting to 192.168.3.3:80... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 2316828 (2.2M)  
Saving to: 'uptime.log.22'  
100%[=====] 2,316,828 --.-K/s tn 0.05s  
2016-02-17 07:15:06 (43.2 MB/s) - 'uptime.log.22' saved [2316828/2316828]  
vagrant@attacker:/vagrant$ wget http://192.168.3.3/uptime.log  
--2016-02-17 07:15:07-- http://192.168.3.3/uptime.log  
Connecting to 192.168.3.3:80... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 2316828 (2.2M)  
Saving to: 'uptime.log.23'  
100%[=====] 2,316,828 --.-K/s tn 0.04s  
2016-02-17 07:15:07 (53.1 MB/s) - 'uptime.log.23' saved [2316828/2316828]  
vagrant@attacker:/vagrant$ wget http://192.168.3.3/uptime.log  
--2016-02-17 07:15:08-- http://192.168.3.3/uptime.log  
Connecting to 192.168.3.3:80... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 2316828 (2.2M)  
Saving to: 'uptime.log.24'  
100%[=====] 2,316,828 --.-K/s tn 0.05s  
2016-02-17 07:15:08 (40.5 MB/s) - 'uptime.log.24' saved [2316828/2316828]  
vagrant@attacker:/vagrant$ wget http://192.168.3.3/uptime.log  
--2016-02-17 07:15:09-- http://192.168.3.3/uptime.log  
Connecting to 192.168.3.3:80... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 2316828 (2.2M)  
Saving to: 'uptime.log.25'
```

the average time of wget a file during syn flooding is about 1s.


```

2016-02-17 07:12:46 (7.56 MB/s) - 'uptime.log.12' saved [2316828/2316828]

vagrant@attacker:/vagrant$ wget http://192.168.3.3/uptime.log
--2016-02-17 07:12:47-- http://192.168.3.3/uptime.log
Connecting to 192.168.3.3:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2316828 (2.2M)
Saving to: 'uptime.log.13'

100%[=====] 2,316,828 --.-K/s ln 0.08s

2016-02-17 07:12:47 (27.7 MB/s) - 'uptime.log.13' saved [2316828/2316828]

vagrant@attacker:/vagrant$ wget http://192.168.3.3/uptime.log
--2016-02-17 07:12:48-- http://192.168.3.3/uptime.log
Connecting to 192.168.3.3:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2316828 (2.2M)
Saving to: 'uptime.log.14'

100%[=====] 2,316,828 --.-K/s ln 0.09s

2016-02-17 07:12:48 (25.6 MB/s) - 'uptime.log.14' saved [2316828/2316828]

vagrant@attacker:/vagrant$ wget http://192.168.3.3/uptime.log
--2016-02-17 07:12:49-- http://192.168.3.3/uptime.log
Connecting to 192.168.3.3:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2316828 (2.2M)
Saving to: 'uptime.log.15'

100%[=====] 2,316,828 1.06M/s ln 2.1s

2016-02-17 07:12:51 (1.06 MB/s) - 'uptime.log.15' saved [2316828/2316828]

vagrant@attacker:/vagrant$ wget http://192.168.3.3/uptime.log
--2016-02-17 07:12:53-- http://192.168.3.3/uptime.log
Connecting to 192.168.3.3:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2316828 (2.2M)
Saving to: 'uptime.log.16'

100%[=====] 2,316,828 2.26M/s ln 1.0s

2016-02-17 07:12:54 (2.26 MB/s) - 'uptime.log.16' saved [2316828/2316828]

vagrant@attacker:/vagrant$ wget http://192.168.3.3/uptime.log
--2016-02-17 07:12:58-- http://192.168.3.3/uptime.log
Connecting to 192.168.3.3:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2316828 (2.2M)
Saving to: 'uptime.log.17'

100%[=====] 2,316,828 2.22M/s ln 1.0s

2016-02-17 07:12:59 (2.22 MB/s) - 'uptime.log.17' saved [2316828/2316828]

```

the top commands during syn flooding

you can also noticed the softirq load increase a lot because it has to handle the tcp socket.

```

top - 17:10:57 up 39 min, 1 user, load average: 0.00, 0.01, 0.05
Tasks: 94 total, 2 running, 92 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.0%us, 0.2%sy, 0.0%ni, 86.0%id, 0.0%wa, 0.0%hi, 13.8%si, 0.0%st
Mem: 372268k total, 350244k used, 22024k free, 17080k buffers
Swap: 786428k total, 340k used, 786088k free, 201300k cached

```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3	root	20	0	0	0	0	S	0	0.0	0:00.19	ksoftirqd/0
7	root	20	0	0	0	0	R	0	0.0	0:00.25	rcu_sched
1	root	20	0	24452	2180	1236	S	0	0.6	0:00.62	init
2	root	20	0	0	0	0	S	0	0.0	0:00.00	kthreadd
5	root	0	-20	0	0	0	S	0	0.0	0:00.00	kworker/0:0H

the top commands before syn flooding

```

top - 17:12:42 up 40 min, 1 user, load average: 0.00, 0.01, 0.05
Tasks: 94 total, 1 running, 93 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.0%us, 0.0%sy, 0.0%ni, 99.8%id, 0.0%wa, 0.0%hi, 0.2%si, 0.0%st
Mem: 372268k total, 350560k used, 21708k free, 17100k buffers
Swap: 786428k total, 340k used, 786088k free, 201300k cached

```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
8	root	20	0	0	0	0	S	0	0.0	0:00.47	rcuos/0
32	root	20	0	0	0	0	S	0	0.0	0:00.92	kworker/0:1
52	root	20	0	0	0	0	S	0	0.0	0:00.47	kworker/u4:1
1	root	20	0	24452	2180	1236	S	0	0.6	0:00.62	init
2	root	20	0	0	0	0	S	0	0.0	0:00.00	kthreadd

summary

Syn flooding is very cheap. With several lines of code, you can slow down the victim's network performance a lot. In terms of programming, the multi-threading doesn't help to boost the syn flood effect mainly because the bottle neck is not the socket system which needs to be shared between different threads.