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

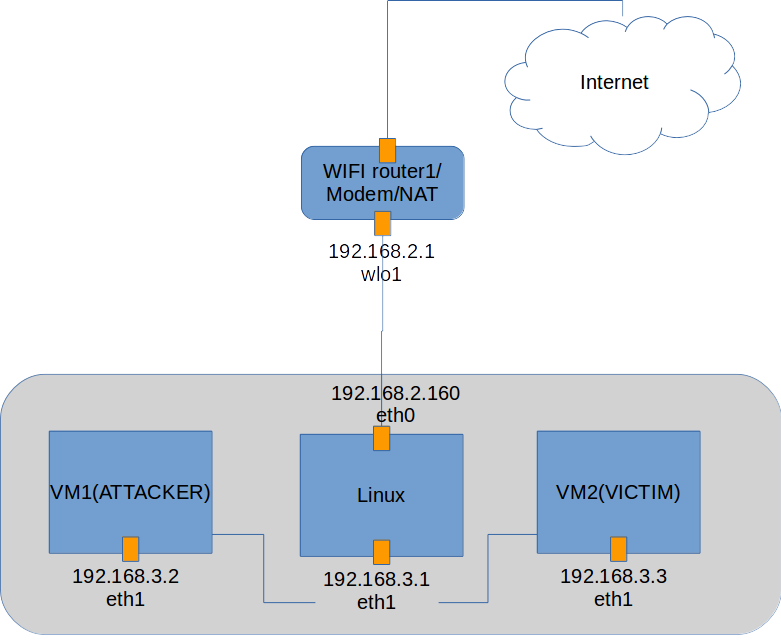
# Introduction

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 |
|  |  |
| Host cpu info | Guest cpu info |
|  |  |

# 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:

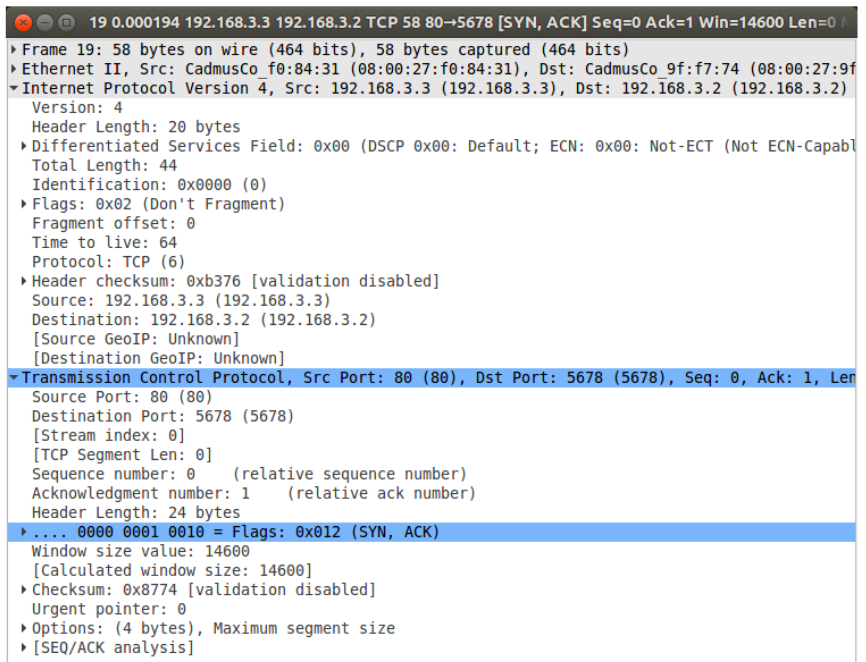
|  |  |
| --- | --- |
| 1. Verify 80 port on virtual mahcine victim |  |
| 2. set up vm via vagrant |  |
| 3.open 80 port on victim vm |  |
|  |  |
|  |  |
|  |  |

# 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.

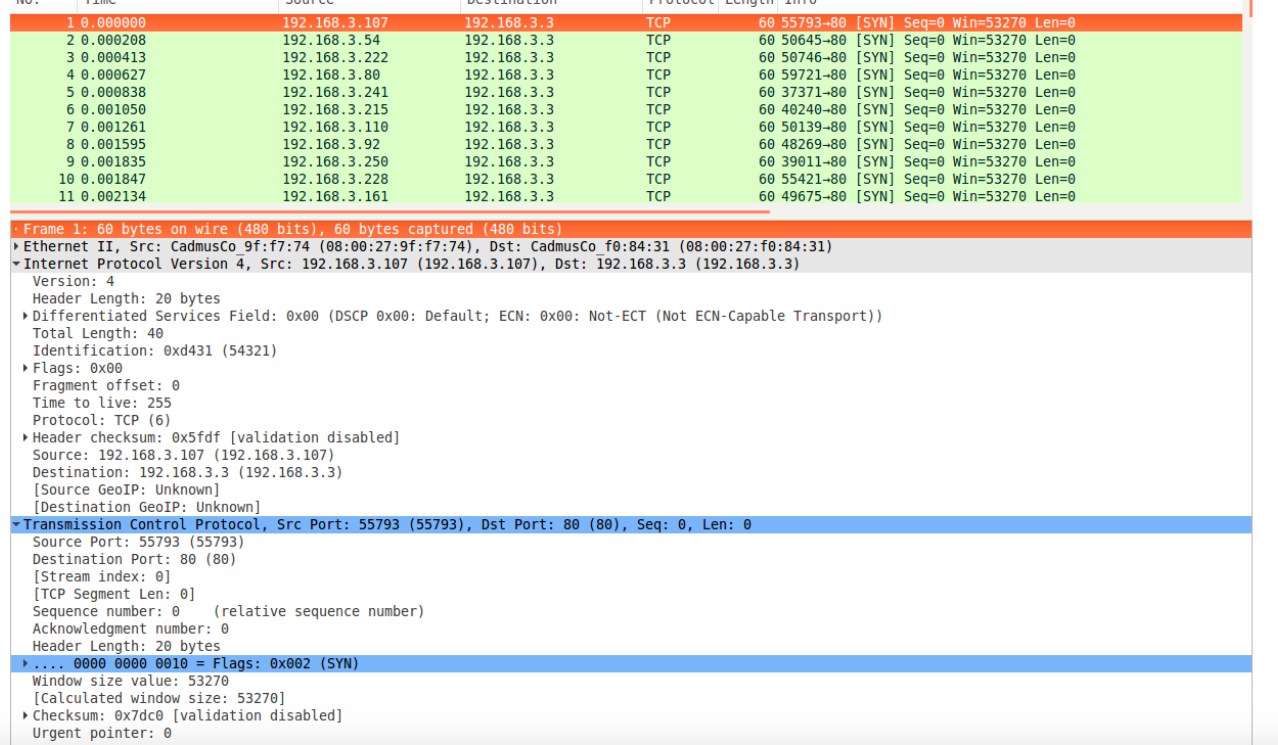


# Syn attack example

## Syn attack with IP spoofing and dynamic source port

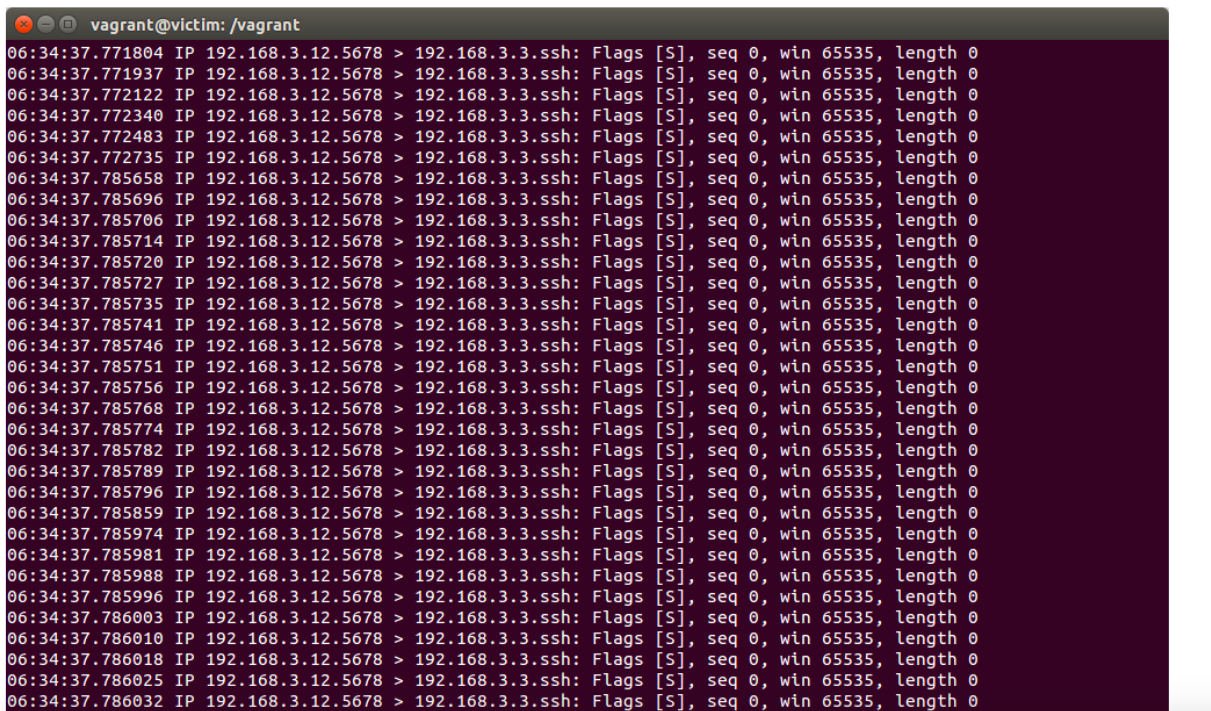
This attack dynamically changes its source port to prevent detection.

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



## Syn attack with IP spoofing and static source port

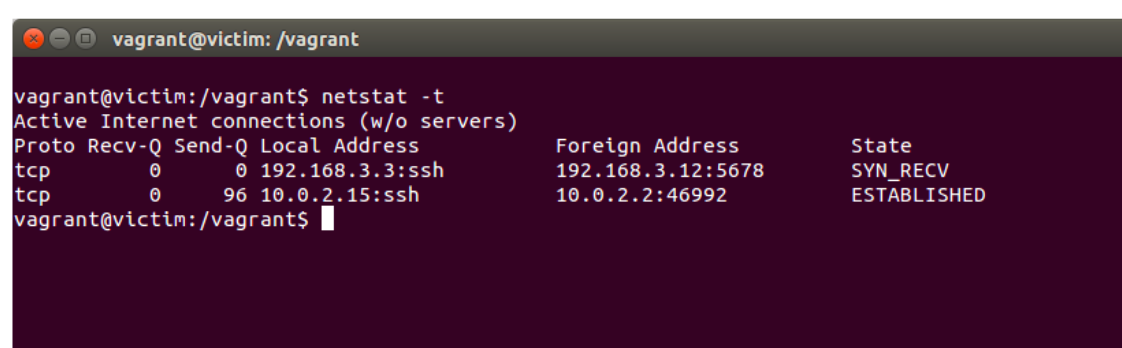
The simplest syn attack which has fixed combination of source ip address and ip ports.



## verify the attack with netstat

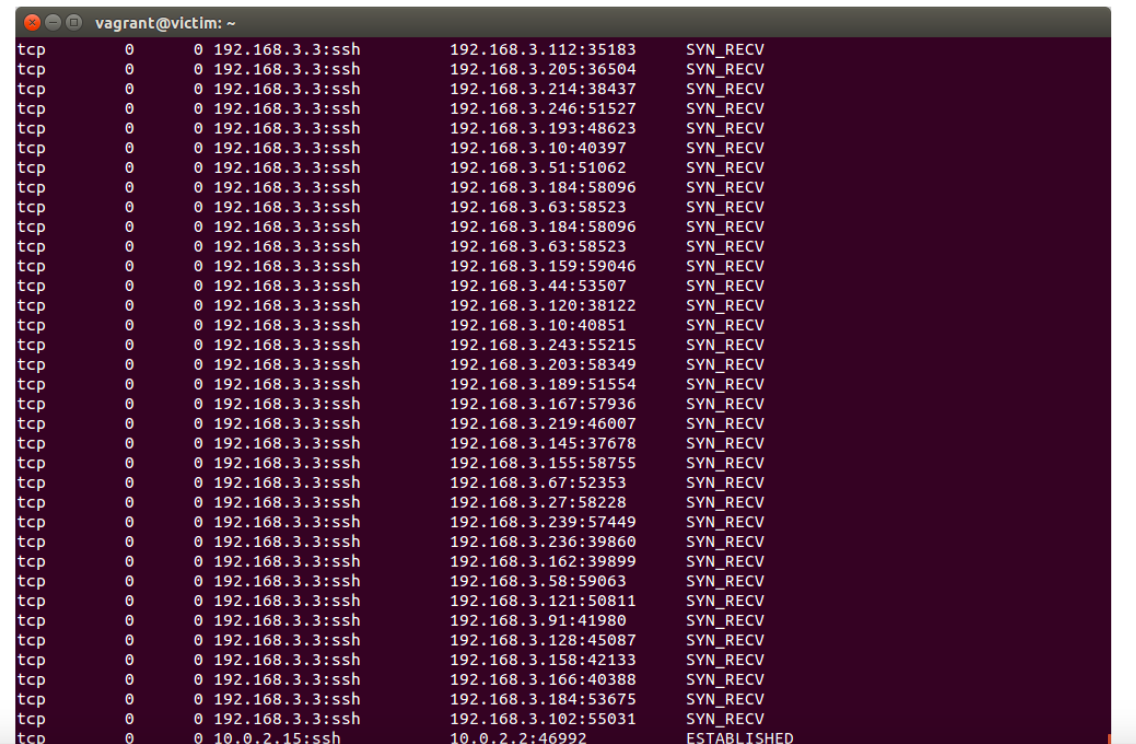
### attack from static ip address

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



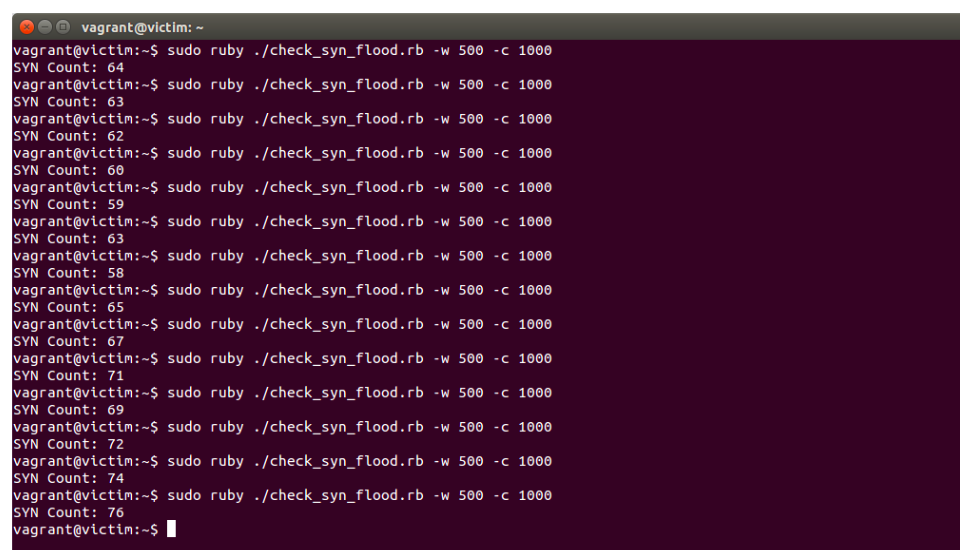
### attack from dynamic source ip and source port

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



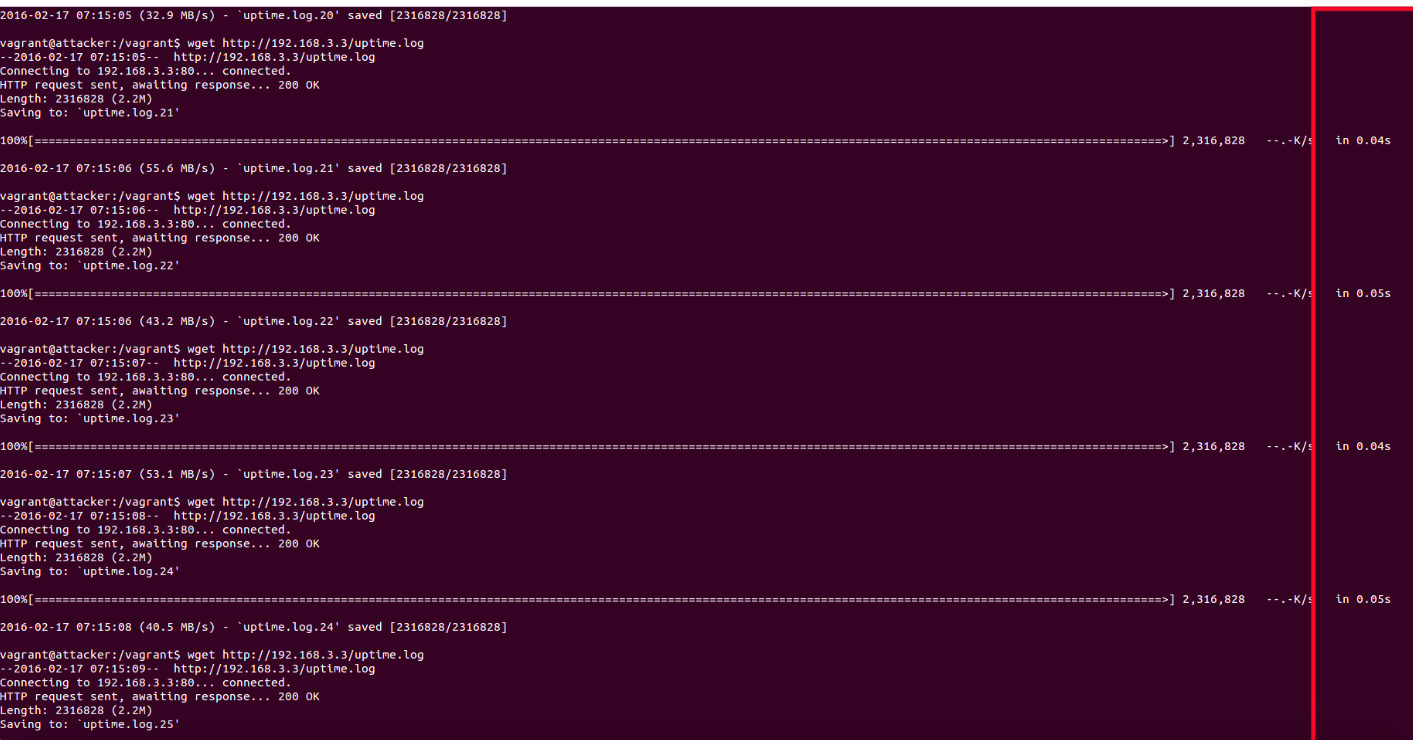
### quantity of tcp syn-rev

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

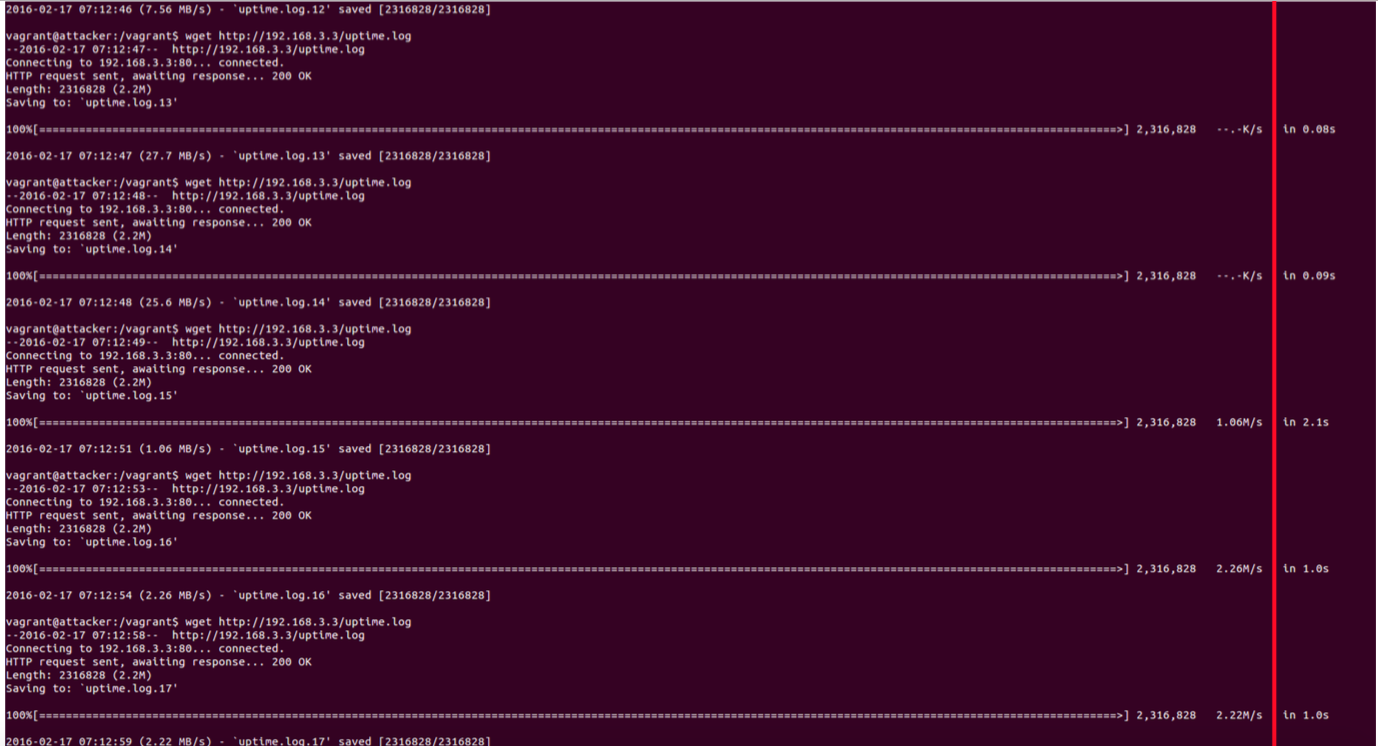


# verify the attack effect

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

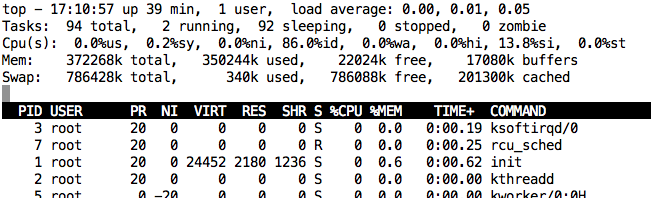


the average time of wget a file during syn flooding is about 1s.

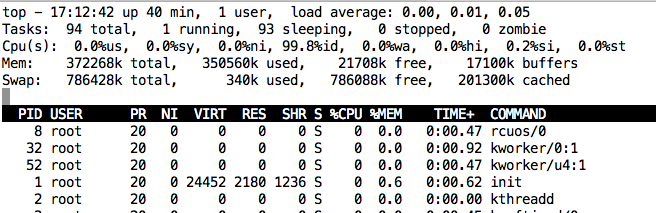


the top commands during syn flooding

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



the top commands before syn flooding



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.