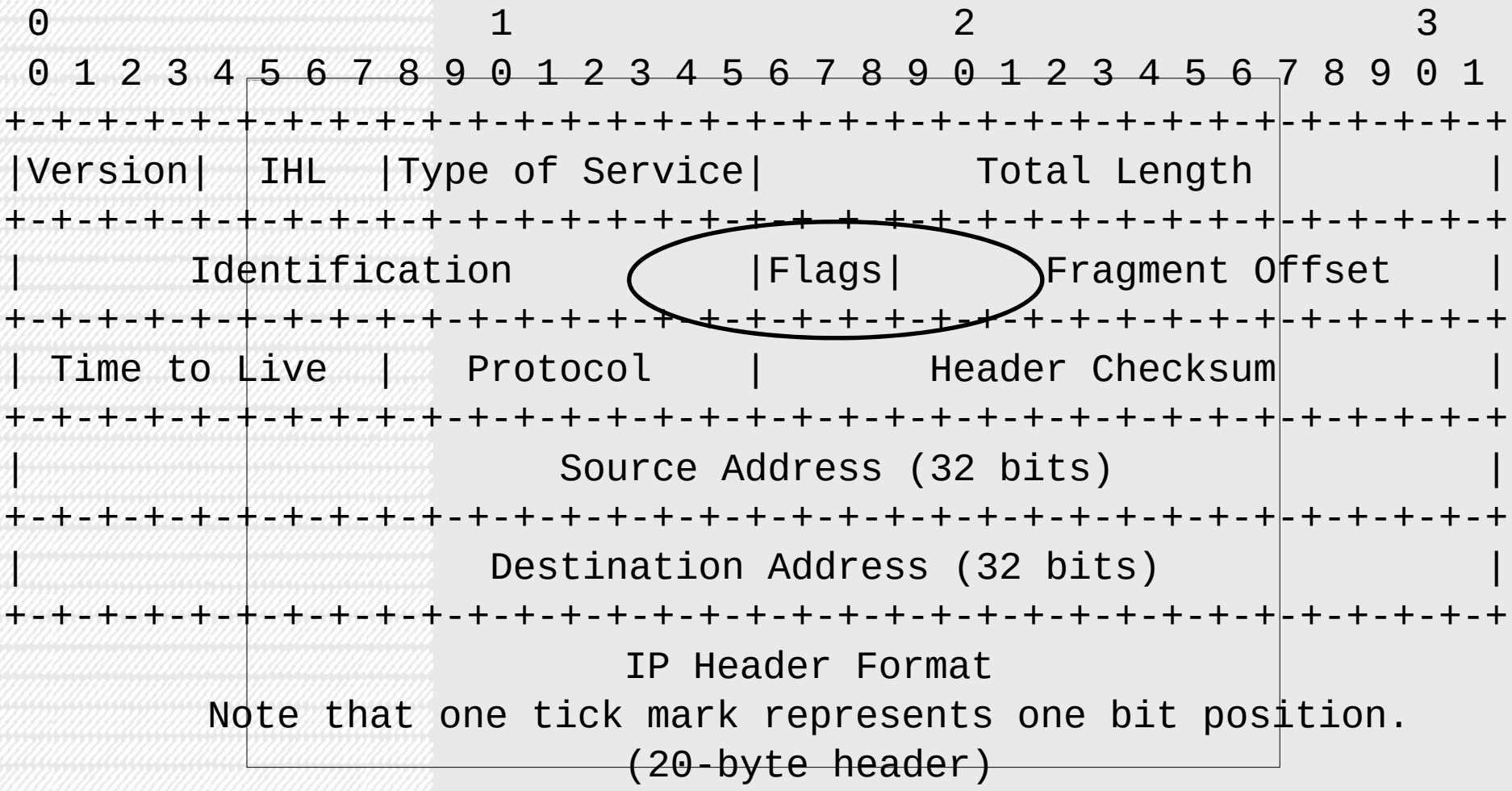
ELEC5616 COMPUTER & NETWORK SECURITY

Lecture 17:
Network Protocols I

IP

- The Internet Protocol (IP) is a stateless protocol that is used to send packets from one machine to another using 32-bit addresses (e.g. 129.78.13.49)
- Many services use the Transmission Control Protocol (TCP) on top of IP (TCP/IP) in order to provide a connection-orientated circuit.
- The other main protocol is UDP which is connectionless.
- IP addresses are translated to and from name addresses (e.g. cassius.ee.usyd.edu.au) using the Domain Name System (DNS)
- Most local networks use Ethernet where machines have unique Ethernet (or MAC) addresses which are mapped to IP addresses using the Address Resolution Protocol (ARP)

IP HEADER

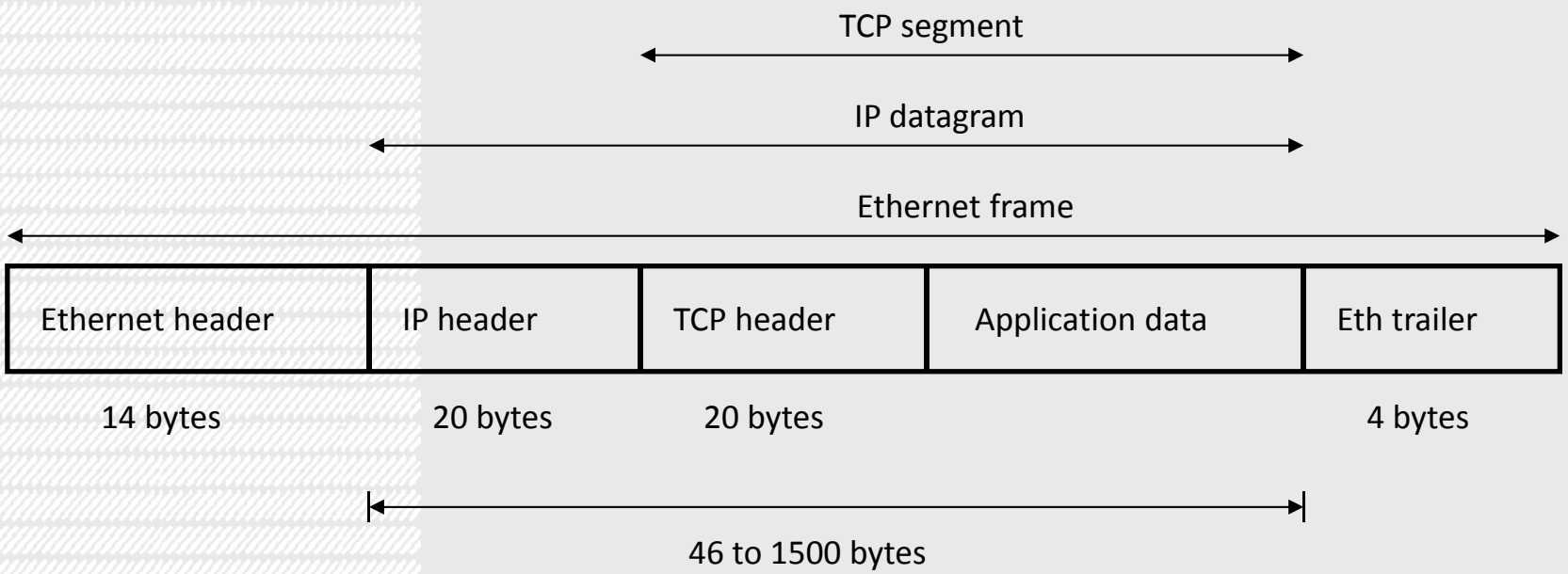


IP Header Format

Note that one tick mark represents one bit position.

(20-byte header)

PROTOCOL ENCAPSULATION



TCP/IP THREE WAY HANDSHAKE

TCP uses 32-bit sequence numbers in order to identify lost packets and rearrange packets received out of order.

Sequence numbers are incremented 128,000 times a second and by 64,000 for each new connection (BSD Unix stack)

Say Alice wants to open a TCP/IP connection to Bob:

Alice → Bob: SYN(ISN_A)

Bob → Alice: ACK(ISN_A + 1), SYN(ISN_B)

Alice → Bob: ACK(ISN_B + 1), PSH(DATA)

Bob → Alice: ACK(ISN_A + k), PSH(DATA)

... data ...

TCP/IP THREE WAY HANDSHAKE

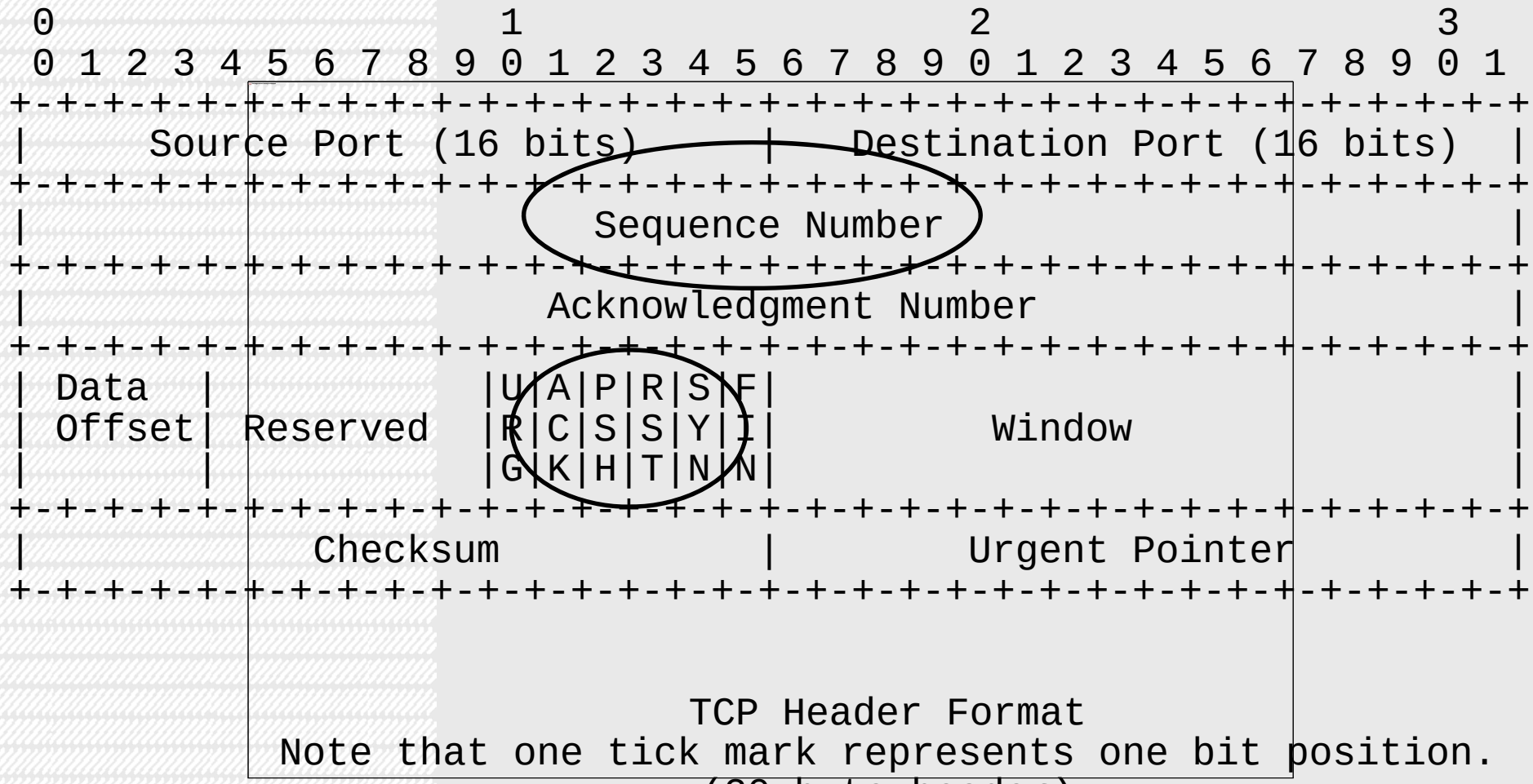
Note there are two sequence numbers (one for each direction of the channel).

As packets can be received out of order, a window exists for valid sequence numbers {sn ... sn + window}

Packets which do not fit within this range are regarded as invalid and dropped.

If the received packet is within this range but greater than the current sequence number + k, the packet is regarded as being received out of order and stored in anticipation of packets in between.

TCP HEADER



TCP Header Format
Note that one tick mark represents one bit position.
(20-byte header)

PACKET SNIFFING

Packet sniffing is the process of listening to raw network traffic (i.e. eavesdropping).

As most of the information flowing across the Internet is unencrypted, packet sniffing on a particular link usually reveals volumes of information

- Logins / passwords
- Email traffic (POP3/IMAP is unencrypted by default, even passwords!)
- Information useful for other attacks (e.g. sequence numbers)

Packet sniffing is usually confined to LAN protocols (e.g. Ethernet, 802.11, etc.) due to the expense of equipment for sniffing other protocols

- It gets hard to process packets at higher speeds without specialised hardware

SPOOFING

Spoofing is the process of forging packets.

Spoofing is typically used to impersonate others or to manipulate protocol or implementation errors.

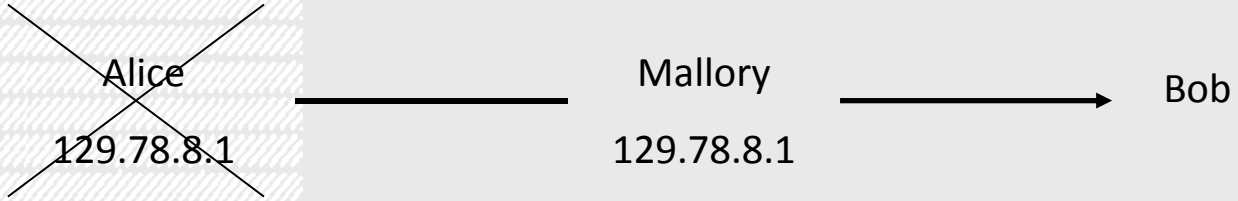
There are two classes of spoofing attacks:

Non-blind spoofing attacks are where an attacker can both inject packets into the network and sniff replies.

Blind spoofing is where an attacker cannot see replies to their spoofed packets.

SIMPLE SPOOFING EXAMPLE

Say Bob trusts Alice (e.g. through /etc/hosts.equiv)



Say also Alice is down (e.g. turned off)

Say Mallory is on the LAN

Mallory only needs to set his IP address to be Alice's address

Bob will believe Mallory is Alice

ANOTHER SPOOFING EXAMPLE

Say Bob trusts Alice (e.g. through /etc/hosts.equiv)

Say this time Alice is alive and Mallory is on the LAN

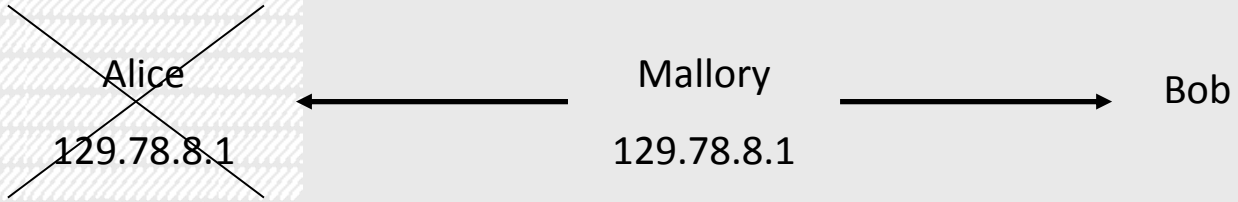
Mallory tries to open an connection

Mallory → Bob: SYN(ISN_A)	**# hi**
Bob → Alice: ACK(ISN_A + 1), SYN(ISN_B)	**# welcome**
Alice → Bob: RST	**# wasn't me!**

Alice will tear down the connection

ANOTHER SPOOFING EXAMPLE

However Mallory can denial-of-service Alice



Mallory → Alice: Denial-of-Service # bye bye
Mallory → Bob: SYN(ISN_A) # hi
Bob → Alice: ACK(ISN_A + 1), SYN(ISN_B) # welcome
Mallory → Bob: ACK(ISN_B + 1), PSH(DATA) # thanks

Mallory can successfully complete the connection

DENIAL OF SERVICE PRINCIPLES

Find a resource (any resource) and use it up

- Bandwidth
- CPU or router processing ability
- Memory, disk space
- File descriptors, sockets (or other OS resources)
- Cognitive limits of humans

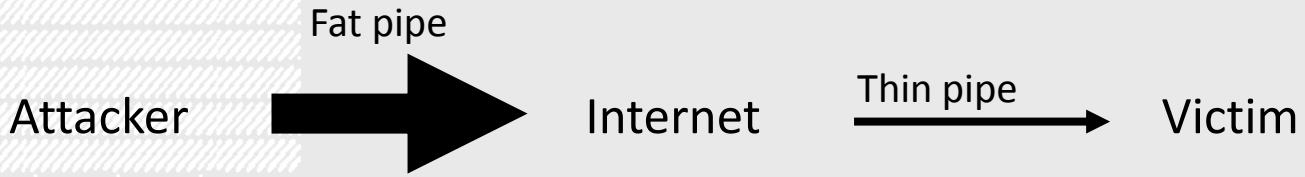
Own as many attackers as possible

Find amplifiers (or post to slashdot.org)

Choose amplifiers with abundant bandwidth

SYN FLOODING

A simple denial-of-service attack on TCP/IP



Attacker's link is faster than the victim's

Alice → Bob: SYN(ISN_A)

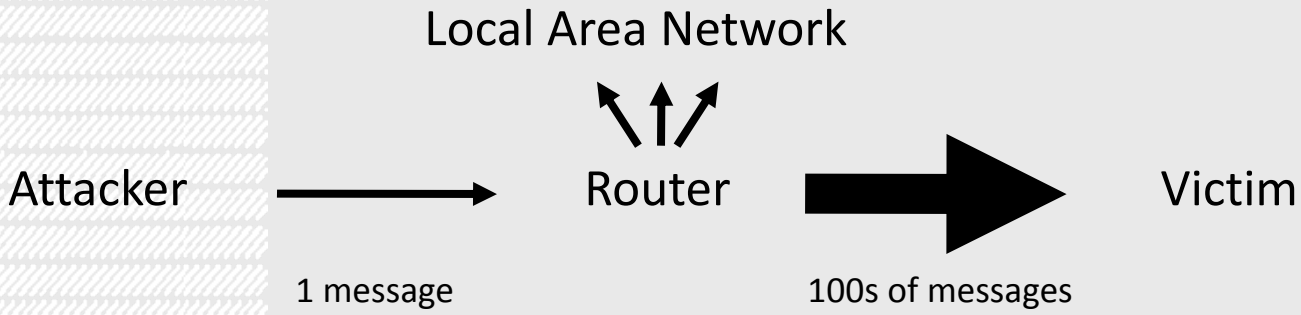
Bob → Alice: ACK(ISN_A + 1), SYN(ISN_B)

Bob allocates resources (memory, a process, a socket) to store details from Alice

If Alice never completes the handshake, eventually all of Bob's resources are used up

SMURFING

Another simple denial-of-service attack



Attacker uses broadcast facility of ICMP echo (i.e. “ping”)

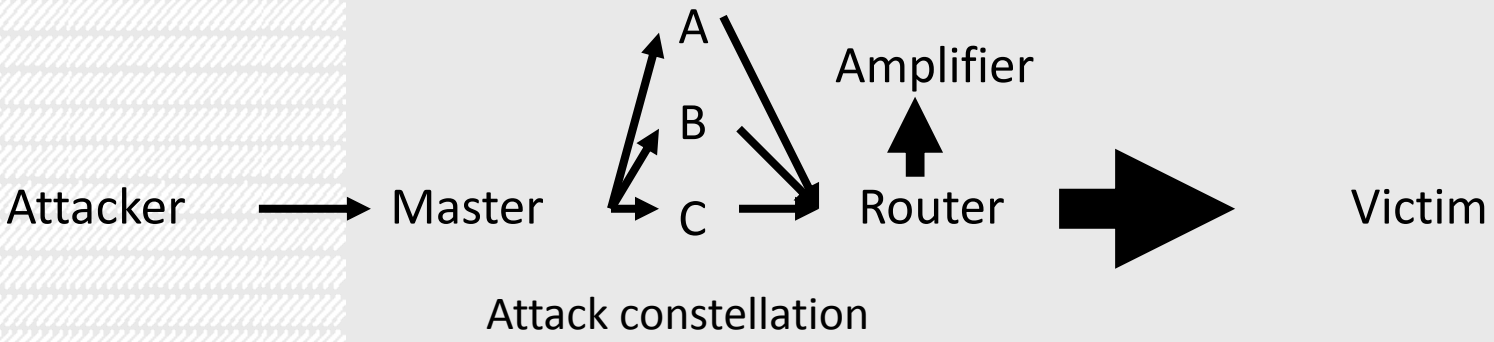
All hosts respond to single message

Attacker forges the source address of the victim

Amplifier machines do not need to be compromised!

DISTRIBUTED DENIAL-OF-SERVICE (DDOS)

Attacker scans 1000s of machines looking for a set of vulnerabilities



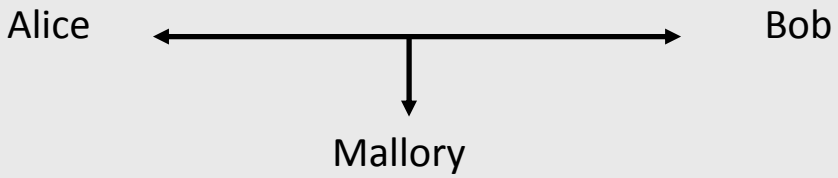
Script scans hundreds of machines that have a problem and installs a drone waiting for time and attack commands

Modern features of DDOS attack tools

- Anonymous encrypted one-way stealth protocols
- Internet Relay Chat (IRC) command and control
- Auto-update

SEQUENCE NUMBER PREDICTION

Say Bob trusts Alice (e.g. through /etc/hosts.equiv)



Say this time Alice is alive and Mallory is remote

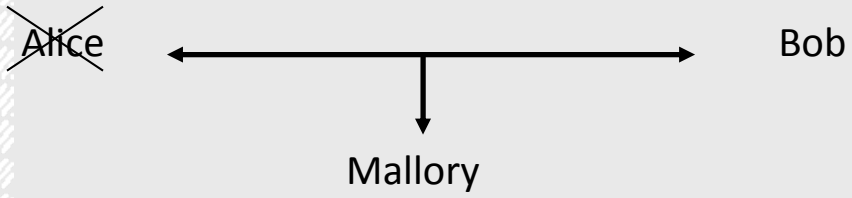
Mallory can't see reply packets

Mallory → Bob: SYN(ISN_A)
Bob → Alice: ACK(ISN_A + 1), SYN(ISN_B)
Alice → Bob: RST # wasn't me!

Alice will tear down the connection

SEQUENCE NUMBER PREDICTION

Say Bob trusts Alice (e.g. through /etc/hosts.equiv)



Mallory → Alice: SYN flood

Mallory → Bob: SYN(ISN_A)

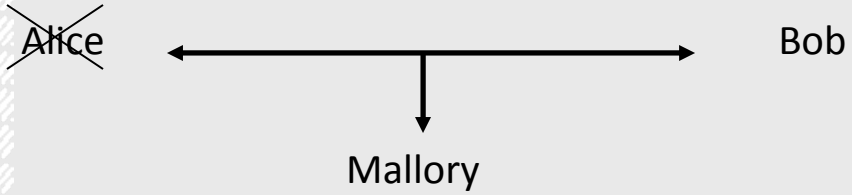
Bob → Alice: ACK(ISN_A + 1), SYN(ISN_B)

Mallory can't see reply packets (he is blind)

Mallory needs to know ISN_B to complete the connection

SEQUENCE NUMBER PREDICTION

Say Bob trusts Alice (e.g. through /etc/hosts.equiv)



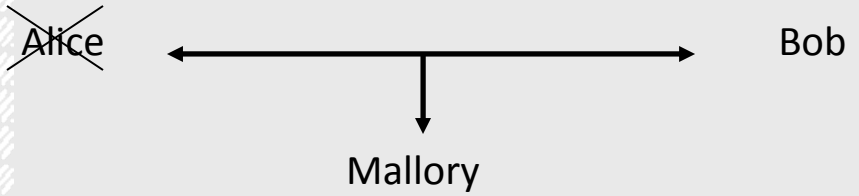
Remember that sequence numbers are incremented 128,000 times per second and by 64,000 every new connection

Mallory can open a connection to Bob earlier to obtain an estimate of the current value of the pointer then guess the current value (or send a flood of guesses)

Mallory can then piggyback data on the final handshake packet even though he is blind and can't see replies

SEQUENCE NUMBER PREDICTION

Say Bob trusts Alice (e.g. through /etc/hosts.equiv)



Mallory → Alice: SYN flood	**# bye bye Alice**
Mallory → Bob: SYN(ISN_M)	**# hi it's Mallory**
Bob → Mallory: ACK(ISN_M + 1), SYN(ISN_X)	**# welcome**
Mallory → Bob: SYN(ISN_A)	**# hi it's Alice**
Bob → Alice: ACK(ISN_A + 1), SYN(ISN_B)	**# welcome**
Mallory → Bob: SYN(ISN_B + 1), PSH(DATA)	**# execute code**

SEQUENCE NUMBER PREDICTION

The problem here is authentication by source IP address

Poor ISN generation also contributes to the problem

Note that it is the granularity that sequence numbers are incremented by that is important - not the average rate

The remarkable thing is that as technology increases, the bounded latency of networks and computer systems becomes more accurate, making this attack easier

SESSION HIJACKING

Session hijacking is where a connection between two parties is hijacked by an attacker (after authentication)

Effectively becoming the man in the middle

In TCP, packets are checked by sequence numbers. i.e. Alice accepts a packet from Bob because it has her IP address and a correct sequence number.

One form of session hijacking can occur is through connection desynchronisation.

SESSION HIJACKING BY DESYNCHRONISATION

Mallory listens for a connection between Alice and Bob.

At an opportune time (say just after Alice enters her password to BlackNet), Mallory sends packets to both Alice and Bob that increment the sequence numbers on each end such that further packets between Alice and Bob will be regarded as old (outside the window).

Mallory is now effectively the man in the middle.

NULL DATA DESYNCHRONISATION

Mallory listens for a connection between Alice and Bob.

Alice → Bob: ACK(SN_B), PSH (DATA)
Bob → Alice: ACK(SN_A), PSH (DATA)

Mallory → Bob: ACK(SN_B + 1), PSH (DATA) # NOP
Mallory → Alice: ACK(SN_A + 1), PSH (DATA) # NOP
[...]

Mallory now has a connection to both Alice and Bob

EARLY DESYNCHRONISATION

Mallory listens for a connection between Alice and Bob.

Alice → Bob: SYN(ISN_A)
Bob → Alice: ACK(ISN_A + 1), SYN(ISN_B)
Mallory → Bob: SYN(ISN_B + 1), RST # goodbye Bob

Mallory → Bob: SYN(ISN_{AM})
Bob → Mallory: ACK(ISN_{AM} + 1), SYN(ISN_{BM})
Mallory → Bob: SYN(ISN_{BM} + 1), PSH(DATA)

Mallory now has a connection to both Alice and Bob

THE MISSING PACKETS (ACK STORM)

When Alice or Bob gets a packet for an invalid connection (e.g. one that Mallory has just closed), they reply with an ACK packet and the expected sequence number.

When the other end gets this packet, they too will reply with an ACK and the expected serial number for the other direction of the connection.

This generates an ACK storm; however ACK packets do not contain data, and hence are not re-sent on loss

Also IP is an unreliable transport medium

It is interesting to note that this attack is self regulating (i.e. the bigger the ACK storm, the more packets are lost due to congestion)

REFERENCES

Papers

Steven Bellovin (http://www.research.att.com/~smb/papers/)

"Security problems in the TCP/IP Protocol Suite"

"Using the Domain Name System for System Break-Ins"

"Strange Attractors and TCP/IP Sequence Number Analysis"

Phase-space analysis of seq-num predictability across different OSes