Week 10

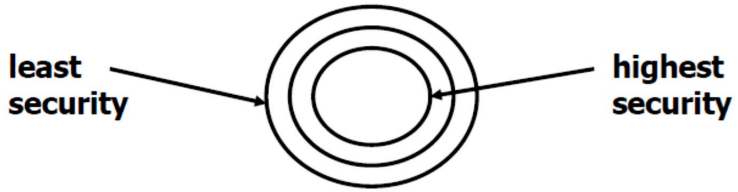
Software & Network Security

THE UNIVERSITY OF SYDNEY

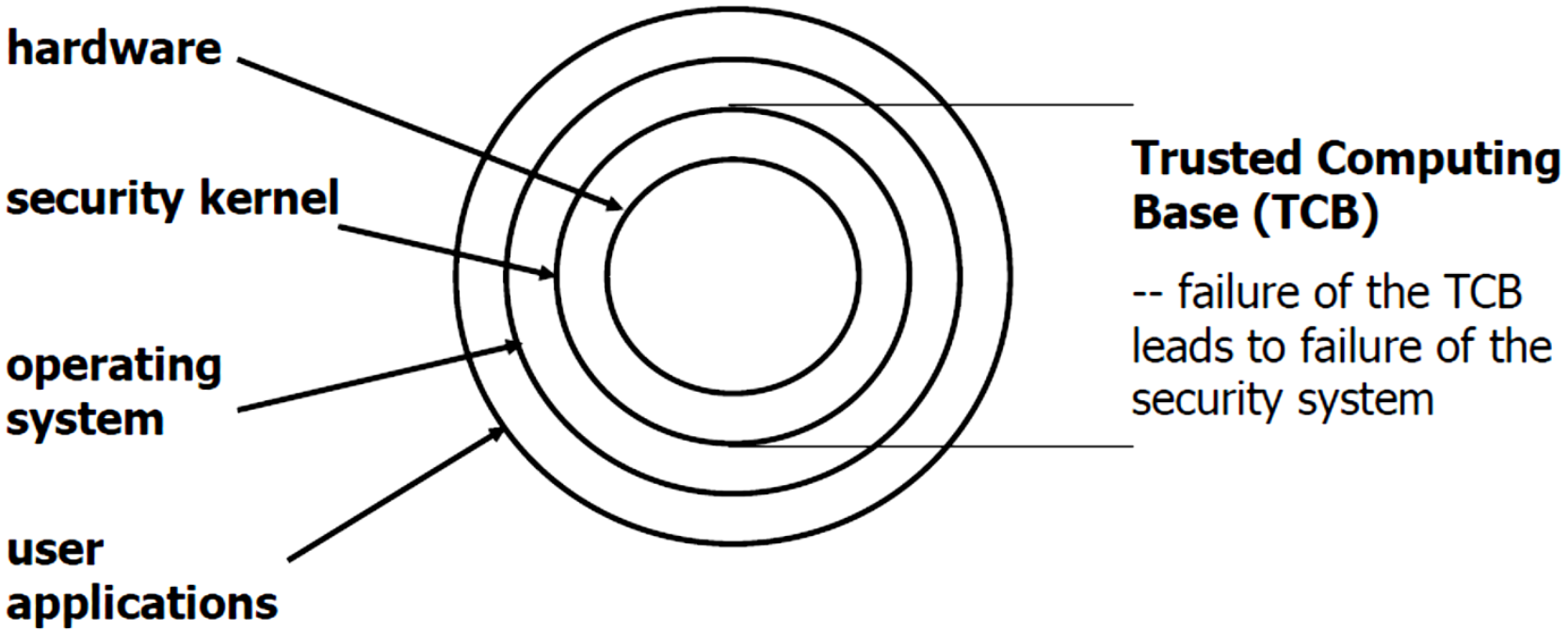
Design of a Secure System

- Top-Down Approach:
  - 1. Threat Model - What are the likely risks?
  - 2. Security Policy - What is the security meant to achieve?
  - 3. Security Mechanisms - How are the policies implemented?

- Design of a secure system follows a ring design
  - Every object has an associated security attribute
  - Every subject has a security clearance
  - The aim is to restrict the interaction between rings



Example: Trusted Operating System



Bell-LaPadula Model

- Developed in the context of military and intelligence data. Also known as "multilevel security" or "MLS systems"

- Simple Security Property:
  - No process may read data at a higher level (no read up - NRU)
  - Agent Alice with "Secret" clearance can't read Top-Secret documents

- The * (Star) property:
  - No process may write data to a lower level (no write down- NWD)
  - Agent Alice with "Secret" clearance can't write Public documents

Covert Channels

- Bell-LaPadula: Prevent subjects with different access rights from communicating.

- Problem: Covert communication channels undetected by the security policy enforcer. Example: File locking
  - High clearance subject frequently locks and unlocks a file.
  - Low clearance subject checks lock status.
  - Using synchronized timer: 1000bit/sec transfer rate.

Security Evaluation: The Orange Book

- Trusted Computer Systems Evaluation Criteria US Department of Defense (1979)
- Ratings System:
  - D: Anyone can get this rating.
  - C1: Discretionary security. Users can disable the security.
  - C2: Controlled access. Per user protection.
  - B1: Labeled protection. Every object is labeled.
  - B2: Structured protection. More OS module verification.
  - B3: Security domains. Modular OS design. Clear security policy.
  - A1: Verified design. Formally verified system design.

Minimal protection

Discretionary protection

Mandatory protection

Verified protection

Common Programming Mistakes

- Poor design choices / assumptions
- Failing to check user supplied input
- Buffer overflows
- Incorrect levels and separation of privileges
- Predictable sources of randomness
- Poor default settings
- Poor failure modes
- Insufficient testing
- Failure to protect secrets properly
- Poor documentation and commenting
- Poor maintenance of legacy code

Buffer Overflows

- Mainly due to poor programming practices, lack of security features in the languages and lack of proper code review.

- Easy to exploit :
  - Find problem in the source code
  - Craft an exploit

- There exists automatic exploit generators that will drop in the right shellcode for a particular processor and operating system.

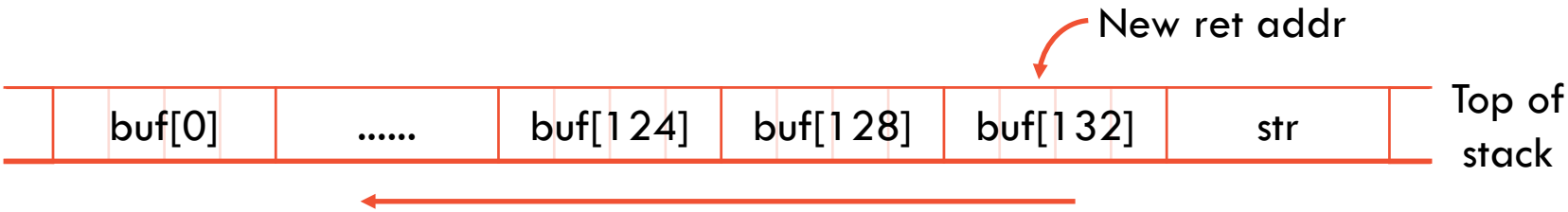
Buffer Overflows Explained

- Suppose a server contains the function **`foo`**:

```
void foo(char *str) {
    char buf[128];
    strcpy(buf, str);
}
```

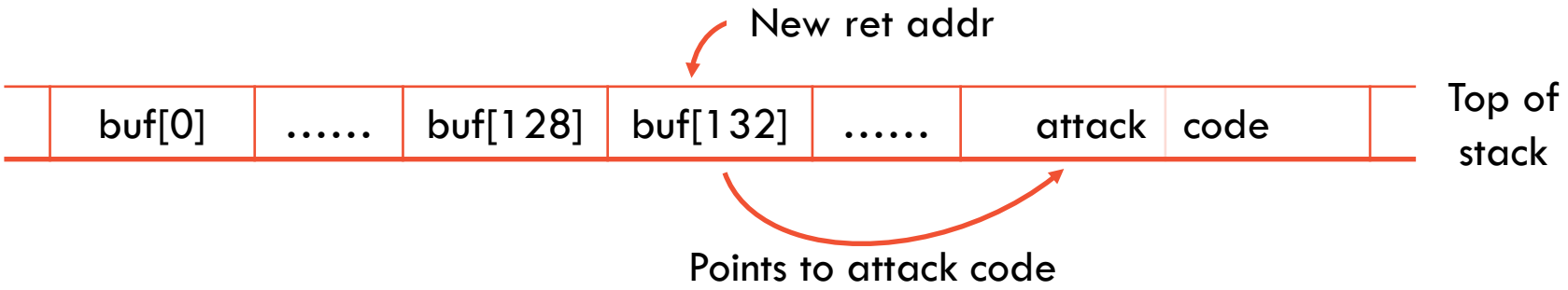
- When the function is invoked the stack in computer memory looks like:

- What if user supplied input pointer **`*str`** is longer than the buffer size, e.g., 136 bytes? After **`strcpy()`**



Basic Stack Exploit

- Main problem: no range checking in `strcpy()`
- Suppose `*str` is such that after `strcpy()` stack looks like:



- Attack code: `execv("/bin/sh", 0)`
- When the function exits, it returns using the new return address and starts executing the attack code - giving a shell.
- Note: In this example, the attack code runs in the stack.

Exploiting Buffer Overflows

- Suppose the above code is from a web server and `foo()`is called with a URL sent by the browser.
- An attacker can create a 200-byte URL to obtain shell on web server.
- Some complications:
  - The attack code should not contain the '`\0`' character.
  - Overflow should not crash program before `foo()` exists.
- Examples of buffer overflows of this type (www.us-cert.gov):
  - TA07-089A: Microsoft Windows Animated Cursor Buffer Overflow (March 2007) A buffer overflow vulnerability in the way Microsoft Windows handled animated cursor files was actively exploited.
  - TA05-362A: Microsoft Windows Metafile Handling Buffer Overflow (December 2005) Microsoft Windows was vulnerable to remote code execution via an error in handling files using the Windows Metafile image format.

More General Exploits

- Basic stack exploit can be prevented by marking the stack segment as non-executable:
  - Various solutions exist but there are many ways to trick
  - Does not block more general overflow exploits.
- General buffer overflow exploits are based on two steps:
  - Arrange for attack code to be present in program space.
  - Cause program to execute attack code.

Inserting the Attack Code

- Inserting attack code:
  - Place code in stack variable (local variables).
  - Place code in a heap variable (malloc'd variables).
  - Place code in static data segment (static variables).

- Using existing code: return-into-libc (exec)
  - Cause function pointer or return address to point to the libc "exec" function.
  - At same time override first argument to be "/bin/sh"

Finding Buffer Overflows

- Hackers find buffer overflows as follows:
  - Run software on local machine.
  - Supply long strings into user input fields. Long strings are of the form "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX".
  - If the software crashes, search the core dump for "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX" to find the overflow location.

- Some automated tools exist. (eEye Retina, ISIC).

Preventing Buffer Overflows

- Main problem:
  - strcpy(), strcat(), sprintf() have no range checking.
  - "Safe" versions strncpy(), strncat() are often misleading
    - strncpy() may leave buffer unterminated.
    - Off by 1 bugs are common in strncpy(), strncat() – forgetting the space required for null terminator.
    - strncpy(dest, src, strlen(src)+1)

- Defenses:
  - 1. Static source code analysis.
  - 2. Run time checking.
  - 3. Black box testing (e.g. eEye Retina, ISIC).

Static Source Code Analysis

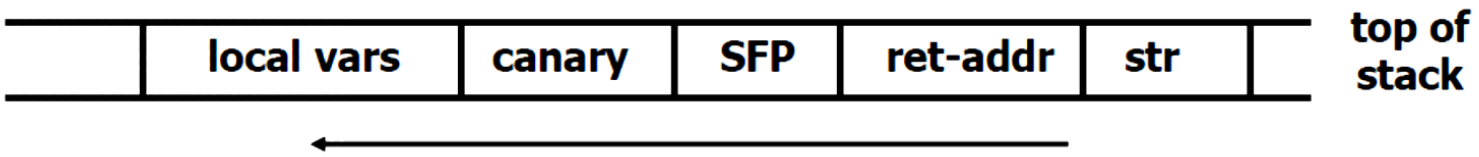
- Statically check source to detect buffer overflows.
  - Internal code reviews
  - Consulting companies ($$$)

- Automated tools
  - Still requires expertise to run and analyse results

Run Time Checking

- Runtime range checking
  - Significant performance degradation.
  - Hard for languages like C and C++.

- Run time tests for stack integrity.
  - Embed "canaries" in stack frames and verify their integrity prior to returning from the function.



Canary Types

- Random canary:
  - Choose random string at program startup.
  - Insert canary string into every stack frame.
  - Verify canary before returning from function.
  - To corrupt random canary attacker must learn current random string.

- Terminator canary:
  - Canary = 0, new line, linefeed, EOF.
  - String functions will not copy beyond terminator.
  - Hence, attacker cannot use string functions to corrupt stack.
  - Other functions still have problems.

Race Conditions

- A race condition attack exploits the fact that multiple instruction transactions are not atomic

- Example: Race condition in the old UNIX "mkdir" command
  - mkdir executed with two stages:
    - Storage allocated
    - Ownership transferred to user
  - Attack:
    - User initiates mkdir
    - User quickly replaces the new directory (the argument) with /etc/passwd
    - mkdir process transfers ownership to user
  - Many such problems have existed with temporary files in /tmp

Race Conditions

- Example: Print server race condition with print quotas
  - User prints file
  - Print server determines cost of file
  - Print server checks account balance and deducts amount
  - Print server spools file for printing

- Attack:

```
lpr smallfile
sleep (1)
cat bigfile > smallfile
```

Timing Attacks

- Timing attacks extract secret information based on the time a device takes to respond ("side channel attack")

- Applicable to:
  - Smartcards
  - Cell phones
  - PCI cards
  - Network software

- Examples:
  - RSA exponentiation
  - Password checking and lengths
  - Inter-keystroke timing (e.g. attack on ssh)

Timing Attacks

- Consider the following password checking code:

```
int password_check(char *inp, char *pwd) {
    if (strlen(inp) != strlen(pwd)) return 0;
    for(i=0; i < strlen(pwd); ++i)
        if (inp[i] != pwd[i])
            return 0;
    return 1;
}
```

- A simple timing attack will expose the length and the password one character at a time.

Timing Attacks

- Correct code to prevent timing attacks:

```
int password_check(char *inp, char *pwd) {
    int oklen = 1;
    if (strlen(inp) != strlen(pwd)) oklen=0;
    for(int ok=1, i=0; i < strlen(pwd); ++i) {
        if (inp[i] != pwd[i])
            ok = ok & 0;
        else
            ok = ok & 1;
    }
    return ok & oklen;
}
```

Further Reading

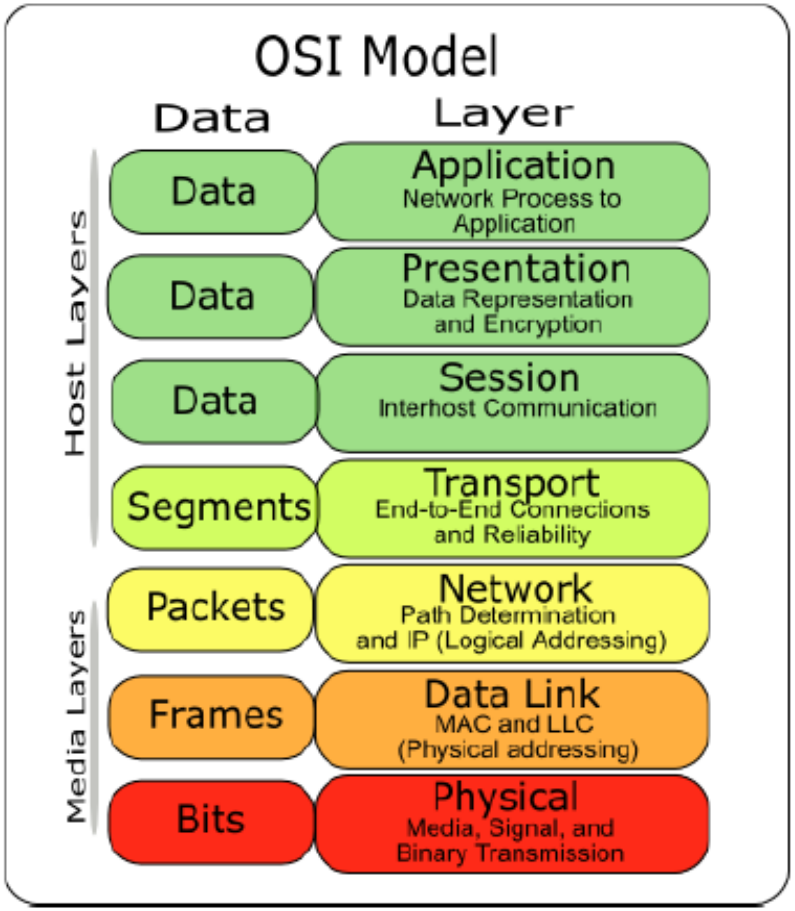
- Smashing the Stack for Fun and Profit (Aleph One)
  - http://www.insecure.org/stf/smashstack.txt
- Timing Analysis of Keystrokes and Timing Attacks on SSH (UC Berkeley)
  - http://www.usenix.org/events/sec01/song.html

Network Protocols (Revision)

THE UNIVERSITY OF SYDNEY

OSI Model

- The OSI Model is a way to conceptualise network systems, whereby each layer is an abstraction from the previous.



The OSI Model - Wikimedia

Layer 1: Physical Layer

- Layer 1 is the physical layer. It deals with bits and how they are modulated, multiplexed, and encoded onto the medium.

(a) Wireless

(b) Copper Twisted Pair

(c) Optic Fibre

Layer 2: Data Link Layer

- Layer 2 is the data link layer. It deals with frames and is concerned with how data is transmitted between devices on the same LAN.
- It provides services such as:
  - Data transfer
  - Error checking
  - Collision detection

A network switch forwards frames

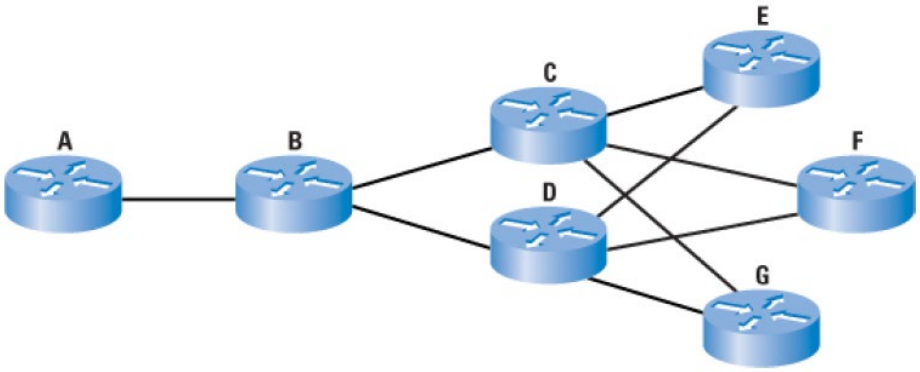
Layer 2: MAC Addresses

- Devices communicate on layer 2 using MAC addresses.
- Every device has a unique MAC address, which is programmed in the factory (nowadays MAC addresses can be reassigned).
- Different manufacturers have different MAC ranges allocated to them, which can allow remote identification of hardware.

- Address Resolution Protocol (ARP)
  - Before sending data on a LAN, a sending device must identify the MAC address of the destination device.
  - If the sender only has an IP address, they can use ARP to determine the corresponding MAC address.

Layer 3: Network Layer

- Layer 3 is the network layer. It deals with packets and is concerned with how data is routed between networks.
- The network layer involves IP addresses and determines how packets can be routed via the most efficient route between two points.



Network routers forward packets

Layer 3: IP Addresses

- An IP address is a 32-bit or 128-bit number, which uniquely identifies a device on a network.
- There are two types:
  - IPv4: 172.16.254.1
    - The original 32-bit addresses commonly seen today.
    - Dot-decimal notation splits the 32-bit number into 4 bytes, each represented as a decimal number between 0-255.
  - IPv6: 2001:db8:0:1234:0:567:8:1
    - A much longer 128-bit address, to cater for the growth of the internet and exhaustion of available IPv4 addresses.
    - Represented as a hexadecimal string of 32 characters.

Layer 3: Subnet Addresses

- In larger networks, it is necessary to break the network into smaller sub-networks.
- When breaking a network up into subnets, we break an IP address into two components:
  - Network Address
    - The beginning part of the IP address
  - Host Address
    - The end part of the IP address
- Where is the beginning and end of the IP address? That is defined by the subnet mask

192.168.000.010
255.255.255.000

or

192.168.000.010/24

Layer 3: Subnet Mask

- The subnet mask is a set of bits that is the same length as an IP address.
- The '1' bits specify the bits of the IP address that apply to the subnet, and the '0' bits define the host. For example:

  192.168.1.150/24

  255.255.255.0

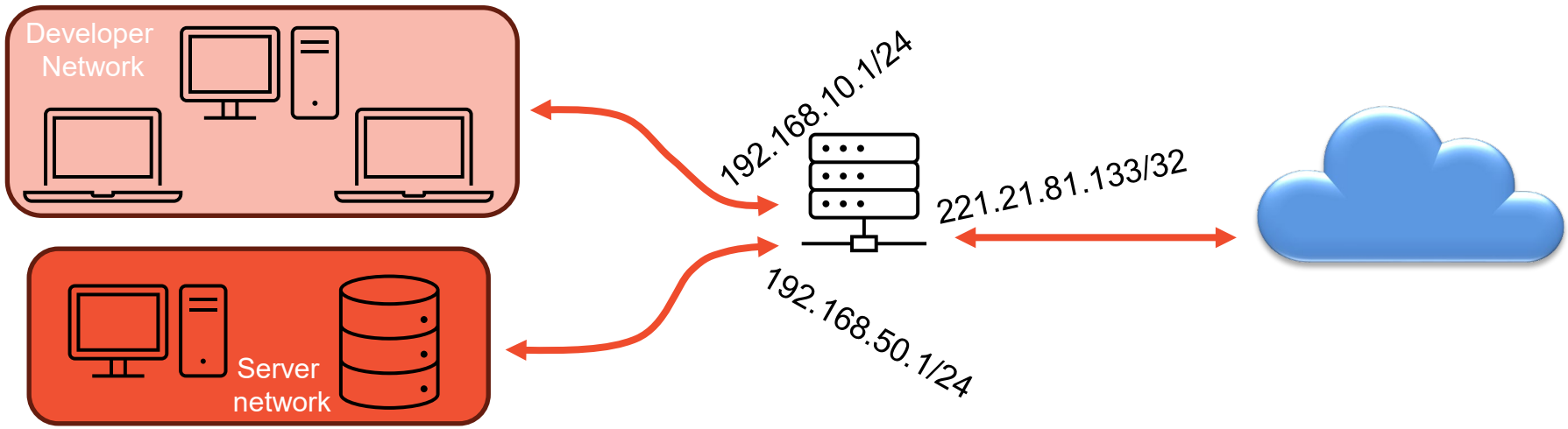
  - The subnet here is: 192.168.1.0/24, and it is the host number 150 in that network.
  - There is no host "0" or "255", since that is reserved for the subnet identifier and broadcast address respectively.
- If you are in one subnet, you cannot talk to hosts on another subnet without having your traffic pass through a router.

Routing

- In order for hosts on different networks (subnets) to communicate, packets must be passed between networks by a router.
- A router is simply a computer with more than one network interface that has rules about forwarding traffic between them.
- Rules are listed in routing tables which can become very complex, potentially going into fine-grained detail like which TCP ports are permitted for communication between particular computers in different networks.

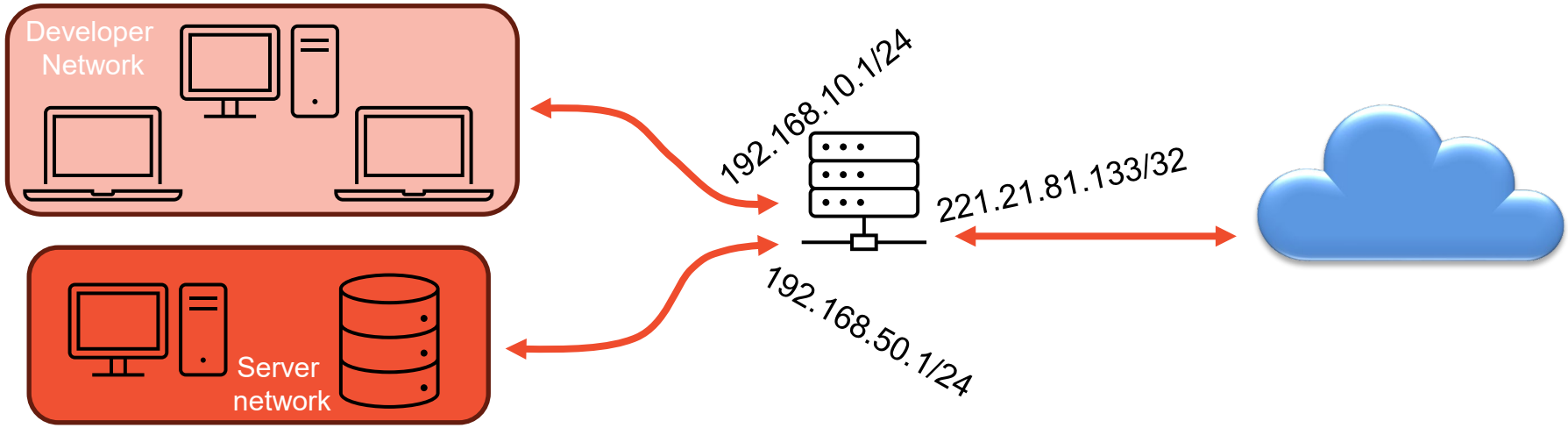
Network Address Translation (NAT)

- Assume, you have a network setup where a router passes traffic between three networks, as shown below.
- The router's public internet-facing IP address is 221.21.81.133. When someone from the developer network, say 192.168.10.105, wants to access Google's public DNS server at 8.8.8.8, they will need to ask their default gateway, which is the router's developer network IP address: 192.168.10.1.



Network Address Translation (NAT)

- For the developer to receive a response from Google, the router needs to rewrite the source IP address (192.168.10.105) to it's public IP (221.21.81.133) so that Google knows where to send the response over the public internet.
- When the router receives the response, the 'destination' in the IP packet will be the router's public IP address. The router remembers the computer that sent the original request, so that it can rewrite the destination address to be the original host (192.168.10.105) and passes Google's response back to the originator.



NAT: Port Forwarding

- In the case that a network service is to be accessible from the internet, there is no originating request to tell the router where the packet should be rewritten to. In this case we need to use NAT's port forwarding feature.

- This may be useful if we have a file server on the "servers" subnet, 192.168.50.20. We may need to configure the router's NAT so that any incoming request on it's public facing IP that arrives on port 21 (FTP), is always forwarded to 192.168.50.20.

- This system is often called a "firewall", though in this case only specifies inbound rules, not outbound rules. It's a kind of implicit firewall whereby incoming packets which don't have an originating request and don't have a port forwarding rule, are simply dropped.

Virtual LANs

- If everyone was connected to the same switch, but were allocated to different subnets, it would be possible for a host to simply change their IP address to assume a different subnet. This arrangement is called a “flat” switch and presents a security vulnerability. To separate network hosts and prevent them from assuming different subnets, we need to connect them to separate switches or set up a Virtual LAN.

- VLANs on Layer 2 Switches
  - In the network switch, we can define different ports to have different VLAN IDs. A different VLAN ID effectively causes a port to be on a different virtual switch, similarly all ports with the same VLAN ID are on the same virtual switch.
  - With VLANs, traffic on certain physical switch ports can be “tagged” as being part of a VLAN or virtual switch. Physical switches will then ensure traffic cannot pass between different VLANs.

Layer 4: Transport Layer

- Layer 4 is the transport layer and defines how packets are sent between two endpoints which are not necessarily on the same network segment (e.g. between two IP addresses on the internet).

- The most common protocols on layer 4 are:
  - UDP: User Datagram Protocol
    - A best-effort connection-less protocol
    - Sends packets to the destination with no acknowledgements, retransmissions, additional checks or delivery guarantees
    - Operates using UDP datagrams
  - TCP: Transmission Control Protocol

Layer 4: Transmission Control Protocol (TCP)

- The Transmission Control Protocol (TCP) is not responsible for addressing or segregation like the layers below it but is instead responsible for delivery of packets.

- Properties of TCP:
  - Establishes a connection using the TCP 3-way handshake
  - Ensures in-order delivery of data for the application layer(s) above it
  - Retransmits lost packets
  - Has a scalable 'sliding window' which is responsible for achieving the fastest possible transfer speed, while avoiding congestion.
  - Operates using TCP segments

Layer 4: TCP Ports

- TCP segments are assigned to particular "ports", which define what application they are intended for.
- Any application can run on any port between 1 - 65535 (16-bit port number).
- Usually there are two TCP port numbers:
  - The destination port: usually a low-numbered standard port number.
  - The source port: usually high-numbered port that handles the response.
- Common Port Numbers
  - HTTP: 80, HTTPS: 443, FTP: 21, SSH: 22, DNS: 53, etc.
  - https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

Network Security

THE UNIVERSITY OF
SYDNEY

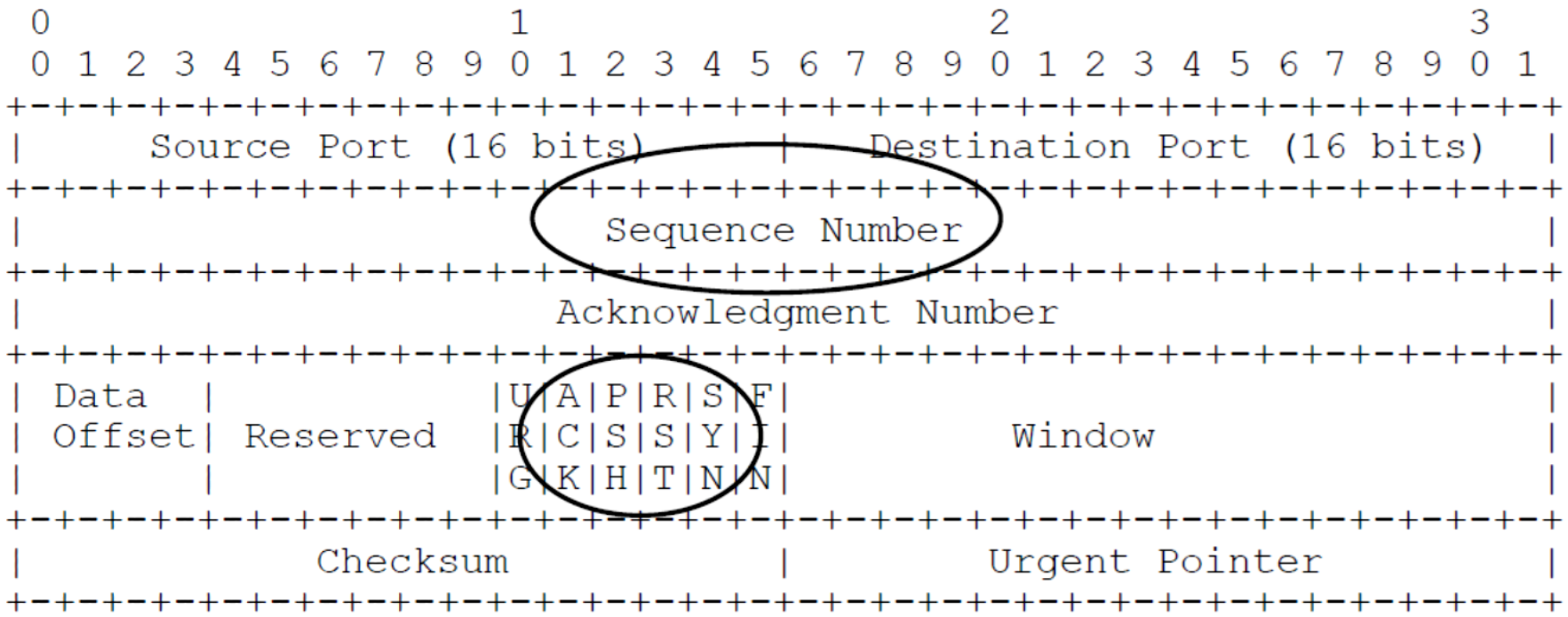
IP

- 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) to provide a connection-orientated circuit.
- IP addresses are translated to and from name addresses (e.g. canvas.sydney.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)

TCP/IP Three Way Handshake

- TCP uses 32-bit sequence numbers 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)
- There are two sequence numbers, one for each direction of the channel.

TCP Header



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

TCP/IP Three Way Handshake

- 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.

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

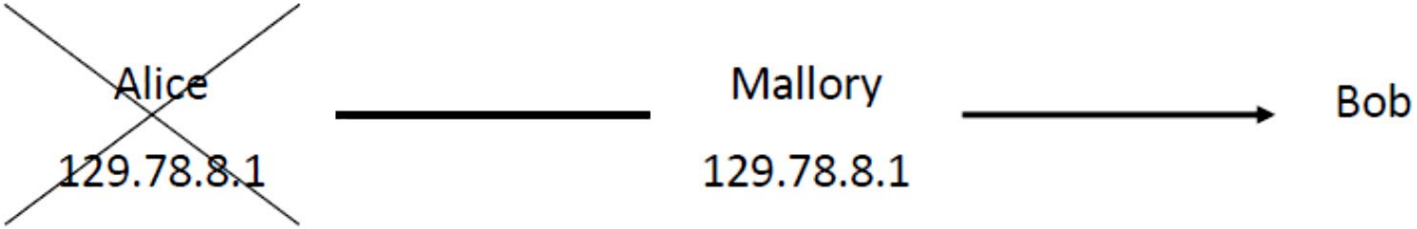
- Packet sniffing is usually confined to LAN protocols (e.g. Ethernet, 802.11, etc.) due to the expense of equipment for sniffing other protocols
  - It gets hard to process packets at higher speeds without specialised hardware

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

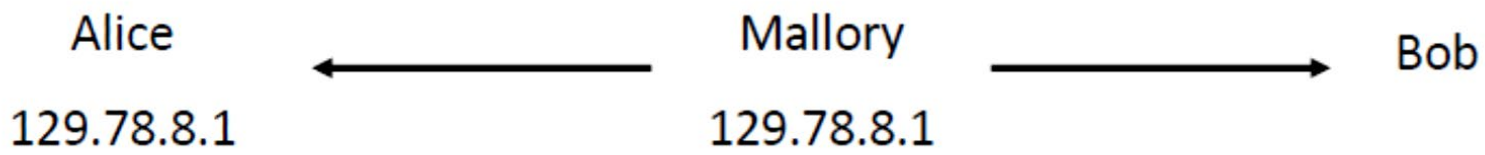
- Say Bob trusts Alice (e.g. through /etc/hosts.equiv)



- Say also Alice is down (e.g. turned off) and 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 a connection

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

- Alice can tear down the connection

- However, Mallory can perform a denial-of-service attack on Alice before Alice sends the TCP RST.

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 (e.g., botnets)
- Choose amplifiers with abundant bandwidth

THE UNIVERSITY OF SYDNEY