CSB Notes

The CIA Triad

Confidentiality	Assures that information cannot be viewed by unintended recipients
Integrity	Assures that information is free from deliberate or inadvertent unauthorized manipulation
Availability	Assures that systems work promptly and authorized users won't be denied service

OSI Security Architecture Definitions

Security Attack	Any action that compromises the security of information owned by an organization
Security Mechanism	A process (or a device) that is designed to detect, prevent, or recover from a security attack
Security Service	A service that is intended to enhances the security of data processing systems, making use of one or more security mechanisms

The AAA Triad

Authentication: Who are you?
- Knowledge factor: Something you know (Passwords)
- Possession factor: Something you have (Keys)
- Inherence factor: Something you are (Biometrics and behaviours)

Authorisation: What can you do?
- Users should not have access to private information that isn't relevant to them

Accountability: Who did what?
- Ensure any action done by any user is *logged* and *traceable*

Types of Attacks

Passive Attacks

Gathering data via monitoring transmissions

Eavesdropping	Act of secretly listening to the private communications of others without their consent.
Traffic analysis	Process of intercepting and examining messages in order to deduce information from patterns

Active Attacks

Involves some modification of an existing data stream or creation of a new fake stream

Active attacks

Masquerade	Pretending to be someone else
Replay	The caputure and retransmission of a message (whilst supressing the original message)
Modification	When a legitimate stream is altered, delayed or reordered
Denial of Service	An attack that inhibits legitimate users from accessing a computer or network.

Threats

Threats to Confidentiality
- Eavesdropping
- Traffic Analysis

Threats to Integrity
- Modification
- Masquerading
- Replaying
- Repudiation: An attack in which you cannot prove that a transaction took place between two entities.

Threats to Availability
- Denial of Service

Nonrepudiation
- Prevents either sender or reciever from denying a transmitted message
- Receiver can verify the sender of a message
- Sender can verify the reciever of a message

Hash Functions Requirements
- $h(m)$ can be applied to any size m
- $h(m)$ produces a fixed length output
- It is computationally infeasible:
  - **Preimage resistant:** To find m given $h(m)$
  - **Weak collision resistant:** To find another message m' such that $h(m) = h(m')$ and $m \neq m'$
  - **Strong collison resistant:** To find two messages $m \neq m'$ such that $h(m) = h(m')$

Message Authentication Code (MAC)
- A MAC function takes a secret key (k) and a data block (m) as input
- It produces the MAC by applying a hash function to the data ($h(m)$) and then ciphering the output using the secret key
- If the integrity of the message needs to be checked, the MAC function can be applied to the message and the result compared with the associated MAC value

MAC (Continued)

An attacker who alters the message won't know the secret key used in the MAC function, and therefore cannot alter a valid message.

Cryptographic Algorithms

Keyless	Cryptographic hash function, Pseudo-random number generator
Single-Key	Block/Stream cipher symmetric encryption, MAC
Two-Key	Asymmetric encryption, Digital signature, Key exchange, User authentication

Security Definitions

Unconditionally secure:

Encryption is secure because not enough information is given to decrypt it (e.g. Vernam/Lorenz)

Computationally secure:

- The cost of breaking the cipher exceeds the value of the encrypted information
- The time required to break the cipher exceeds the useful lifetime of the information

Block Cipher:

A symmetric cipher which encrypts a fixed-length (typically 64/128) group of bits to ciphertext. The encryption algorithm takes a key as input and uses it to produce ciphertext on the corresponding block.

Stream Cipher:

A symmetric cipher where plaintext is encrypted one byte at a time. This works using a bit stream generation algorithm, which takes a key as input, to produce a ciphertext stream.

History / Case Studies

Data Encryption Standard (DES):

- Issued in 1977 by the National Bureau of Standards (now NIST)
- Most widely used encryption scheme until AES in 2001
- Algorithm itself referred to as Data Encryption Algorithm (DEA)
- Encrypts 64-bit blocks with a 56-bit key (symmetric)
- The algorithm transforms 64-bit input in a series of steps into a 64-bit output

1988 Morris Worm (Worm):

- The first computer worm
- Before any real information/network security existed
- Duplicated itself 14% of the time even if the computer was already infected
- Result similar to that of a fork bomb

Fork bomb: A denial-of-service attack wherein a process continually replicates itself to deplete available system resources, slowing down or crashing the system due to resource starvation.

1986 The Brain (Virus):

- The first computer virus
- No serious malicious intent
- Overwrote first sector of boot sector with message: "Beware of this virus… Contact us for vaccination…"

2018 SamSam ransomware attack (DNS):

Hackers targeted several US healthcare organizations, using DNS hijacking to redirect traffic from legitimate websites to malicious domains that delivered ransomware payloads.

2019 Cloudflare DNS hijacking (DNS):

Hackers used a vulnerability in Cloudflare's DNS infrastructure to redirect traffic from several websites, including coinbase.com, to a malicious domain that delivered a cryptocurrency mining payload.

2017 Exim vulnerability exploit (DNS):

Hackers exploited a vulnerability in the Exim mail server software to gain control over the DNS records of several hosting providers, allowing them to redirect traffic from legitimate websites to malicious domains.

2022 Google armor attack (DDOS):

- Google stifled what would have been the largest DDoS attack in the world
- The attack peaked at 46 million requests per second (Total daily requests to Wikipedia in 10 seconds)
- The attack lasted about 30 minutes and consisted of more than 5,000 devices from 132 countries.

2020 AWS attack (DDOS):

- The attack saw as many as 2.3 terabits per second (Tbps) coming into its servers.
- The hackers hijacked user directories on Connection-less Lightweight Directory Access Protocol (CLDAP) servers to flood the AWS servers with massive amounts of information.

2018 GitHub attack (DDOS):

- Hackers used a caching system known as *Memcached* to manually send 1.3 Tbps of information to the GitHub servers
- The Memcached servers made it so the hackers could amplify their attack by 50,000 times.

2016 Dyn attack (DDOS):

The Marai botnet used an Internet of Things (IoT) army, comprised of almost much every IoT device (i.e., Smart TVs, Printers, Cameras etc.) to overload the Dyn servers.

2013 Spamhaus Attack (DDOS):
- Spamhaus, as an anti-spam organization, is regularly attacked and had DDoS protection services already in place
- However, this attack estimated at 300 gigabits of traffic per second — was large enough to knock its website and part of its email services offline.

2000 Mafiaboy attacks (DDOS):
- Multiple attacks that took down the websites of companies like Dell, E-Trade, eBay, and Yahoo
- Mafiaboy built a network of bots by compromising university computer networks and using them for DDoS attacks
- The effects of Mafiaboy's attacks reached as far as the stock market and led to the creation of current cybercrime laws.

Networking

Network Security: Protection of networks and their services from unauthorized modification, destruction, or disclosure, and provision of assurance that the network performs its critical functions correctly and there are no harmful side effects

Data routing in a network:

Circuit switching	Establishes a dedicated communication path between the sender and the receiver
Packet switching	Transfer of small pieces of data (packets) across various networks

Multiplexing: Allocating fractions of bandwidth
Mostly for Circuit Switching

FDM	Each user gets a fixed channel to communicate on
TDM	Each user gets allotted time slots during which they can communicate

Attack on each layer:

Layer	Attack
Application	Repudiation, Data Corruption, Spyware
Transport	Session hijacking, SYN Flooding, SEQ number prediction, Smurf attacks, Routing attacks
Network	IP smurfing, Address spoofing, Resource Consumption
Data Link	MAC/ARP sniping, Address spoofing, Traffic analysis
Physical	Jamming, Interception, Eavesdropping
Multi-layer	DOS, Impersonation, Replay, Man-In-The-Middle, Data alteration

OSI model:

Protocol Data Unit	Layer	Responsibility
Data	Application	Where applications can access network services
Data	Presentation	Data representation and encryption
Data	Session	Inter-host communication
Segments	Transport	End-to-End connections and reliability
Packets	Network	Path determination and routing between networks using IP (Logical addressing)
Frames	Data Link	Data transfer within the same network, MAC and LLC (Physical addressing)
Bits	Physical	Data / Signal Transmission

Packets
- Packets are data units within the network layer created by software
- Each packet contains logical addressing information such as IP addresses, the transport protocol used as well as data (encapsulated from higher layer protocols)
- Internet Protocol (IP) is often described as transmitting packets

Frames
- A chunk of data created by network communication hardware such as Network Interface Cards and router interfaces
- A unit of data used in the data link layer.
- A frame contains more information about the transmitted message than a packet.
- Frames contain frame delimiters, hardware (MAC) addresses as well as all the aforementioned data stored in a packet.

Protocols

Application:
- Post-Office Protocol (POP3) [port 110]
- Simple Mail Transfer Protocol (SMTP) [port 587]
- Domain Name System (DNS) [port 53]
- File Transfer Protocol (FTP) [port 21]
- HTTP / HTTPS [port 80 / 443]
- Telnet [port 23]
- Secure Shell (SSH) [port 22]

Presentation:
- Network Data Representation (NDR)
- Lightweight Presentation Protocol (LPP)

Session:
- NetBIOS [port 137]

Transport:
- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)

Network:
- Internet Protocol (IP)
- Address Resolution Protocol (ARP)
- Internet Control Message Protocol (ICMP)

Data link:
- Serial Line Internet Protocol (SLIP)
- Point-to-Point Protocol (PPP)

Physical:
- IEEE 1394
- Digital Subscriber Line (DSL)
- Integrated Services Digital Network (ISDN)

Network Attack Categories
- Intrusion
- Blocking
- Malware

DNS
DNS translates human readable domain names to machine readable IP addresses (port 53)

User → DNS resolver → DNS server → User

Weaknesses of DNS:
- TCP/UDP on port 53 (mostly UDP)
- Unencrypted
- Easily monitored
- Easily redirected
- Can be blocked
- Can be forged (if DNSSEC not used)

Threats to DNS:
- Corrupted host platforms
- Wireline and middleware inspection and interception
- Resolvers that leak queries
- Servers that leak queries

Attacks on DNS

DNS Cache Poisoning:
Attackers can poison DNS caches by impersonating DNS nameservers
1. Attacker makes a request to a DNS resolver
2. The attacker forges the reply when the DNS resolver queries a nameserver
3. The response is saved in the DNS resolver caches
4. When a query is made by a user they are directed to a malicious website instead

DNS Amplification Attack:
1. Attacker spoofs a victim's IP address
2. Attacker requests large amounts of data from DNS servers
3. DNS servers send the requested data to the victim's IP
4. Victim essentially recieves a DDoS from the DNS servers

DNS-over-TLS (DoT)
- Released in 2016, first established DNS encryption solution
- Uses secure TLS channel on port 853 instead of common port 53
- Prevents attackers seeing or manipulating DNS requests

DNS-over-HTTPS (DoH)
- Introduced in 2018
- Uses TLS, like DoT, but does so via HTTPS, opening a new port (443) for secure communication
- The DNS query request and reply are both encrypted

Attacks on HTTP

HTTP Session Hijacking / Cookie Stealing:
1. Attacker injects script onto server
2. Victim authenticates on server
3. Victim's browser sends the session cookie to the attacker
4. Attacker can hijack the user's session

Session Side Jacking:
1. Attacker sniffs packets on local network (often unsecured hotspots)
2. If a session isn't entirely encrypted with SSL/TLS, a victim's session key might be contained within packets
3. The attacker can use the session key to hijack the session and impersonate the victim

Solution: Use HTTPS

Attacks on TCP

TCP Handshake:

SYN	seq = client_seq	Client → Server

SYN-ACK	seq = server_seq
	ack = client_seq+1
		Client ← Server

ACK	seq = client_seq+1
	ack = server_seq+1
		Client → Server

It's very difficult to intercept a TCP connection that's already established

TCP Session Hijacking:
Possible when an attacker is on the same network segment as the target machine

1. Attacker can sniff all back/forth tcp packets and know the seq/ack numbers
2. Attacker can inject a packet with the correct seq/ack numbers with the spoofed (victim's) IP address

IP spoofing needs low-level packet programming, OS-based socket programming cannot be used!

SYN Flooding Attack:

1. An attacker sends a large number of SYN requests to a target's system
2. Target uses too much memory and CPU resources to process these fake connection requests
3. Target's bandwidth is overwhelmed

Usually SYN flood packets use spoofed source IPs

- No TCP connection is set up (unlike TCP hijacking)
- Hides the attacking source
- Make it difficult for the target to decide which TCP SYNs are malicious and which are from legitimate users

Potential Solutions for TCP:

Ingress Filtering	Drop all packets that aren't from expected destination
uRPF Checks	Only accept packets from interface if forwarding table entry for source IP address matches ingress interface (only works on symmetric routing)

Symmetric routing: A single route for incoming and outgoing network traffic

SYN Flood Defence: SYN Cookie

1. Client sends SYN to server
2. Server responds with SYN-ACK cookie
3. Honest client responds with ACK
4. Server checks response
5. If matches SYN-ACK, establishes connection

Note: server_seq = f(src addr, src port, dest addr, dest port, rand)

IP Spoofing

1. In the most basic IP spoofing attack, the hacker intercepts the TCP handshake before the source manages to send its SYN-ACK message
2. The hacker sends a fake confirmation including their device address (MAC address) and a spoofed IP address of the original sender
3. Now the receiver thinks that the connection was established with the original sender, but they're actually communicating with a spoofed IP

IP address spoofing is most often used to bypass basic security measures such as firewalls that rely on blacklisting

IP Spoofing (continued)

Denial of service
An attacker can send out millions of requests for files with a spoofed IP addresses, causing all of the responses to be sent to the victim's device

Man-in-the-middle attacks
If you're browsing an insecure HTTP address, an attacker can use IP spoofing to pretend they're both you and the service you're speaking to, thereby fooling both parties and gaining access to your communications

Attacks on ARP

Address Resolution Protocol (ARP):

- Each IP node (Host, Router) on LAN has an ARP table
- The ARP table contains mappings from IP to MAC ***<IP address; MAC address; TTL>***
- **TTL = Time To Live:** Time after which the mapping will be forgotten
- Works by broadcasting requests and caching responses for future use

ARP Spoofing:

- ARP table is updated whenever a response is recieved
- Requests are not tracked
- ARP announcements are not authenticated
- A rogue machine can use this to spoof other machines

ARP Spoofing Countermeasures:

- Using static entries (hard to manage)
- Check for multiple occurences of the same MAC
- Software detection solutions (Anti-arpspoof, Xarp, Arpwatch)

MITM Attacks

ARP cache poisoning: The aim is to associate the attacker's MAC address with the IP address of another host, such as the default gateway, causing any traffic meant for that IP address to be sent to the attacker instead.

DNS spoofing: An attack where corrupt Domain Name System data is introduced into the DNS resolver's cache, causing the name server to return an incorrect result record.

IP spoofing: IP spoofing is the creation of Internet Protocol (IP) packets which have a modified source address in order to either hide the identity of the sender, to impersonate another computer system, or both.

Rogue WiFi access point: An access point that has been installed on a secure network without explicit authorization from a local network administrator that allows wireless devices to connect to and communicate with the network.

SSL spoofing: In an SSL hijacking MITM attack, the attacker generates fake certificates for the domains of HTTPS sites the victim attempts to visit.

Radio Jamming Attack

By creating a noisy radio signal, we can cause enough interference to disrupt legitimate communication

Common Types of DDoS Attacks

- **Application layer attacks:** Generate huge amounts of HTTP requests
- **Protocol attacks:** Attacks exploiting weaknesses in the protocols, e.g. SYN flooding (TCP connection), Spoofed IP addresses
- **Volumetric attacks:** Attacks that attempt to consume all of the target's available bandwidth e.g. DNS Amplification

Top Web Security Threats

- Phishing
- Ransomware
- Viruses / worms
- Spyware
- Code injection

SQL Injection

Used to manipulate operations on databases, with the eventual goal of having complete control over it

Cross-site Scripting (XSS)

Can be used for session stealing

1. Attacker injects malicious scripts into web applications
2. Script will run on victim's devices when they use the app

Firewalls

- Filters traffic between a protected network and the outside
- Usually runs on a dedicated device, as performance is critical
- Usually runs on a minimal and proprietary OS to reduce attack sites
- Provides a focal point for monitoring
- Provides a central point for access control
- Limits the potential damage from a network security issue
- **Doesn't** protect against malicious insiders
- **Doesn't** protect connections that do go through
- **Doesn't** completely protect against new threats
- **Doesn't** protect against viruses, trojans, etc.

Generic Techniques for Enforcing Policy:

- **Service Control:** Determine the types of Internet services that can be accessed
- **Direction Control:** Determine the direction in which particular service requests are allowed
- **User Control:** Controls access to a service according to which user is attempting to access it

Types of Firewall:

- **Packet Filtering Firewall** (Works at Network layer, IP)
- **Circuit-level Gateway** (Works at Transport layer, TCP)
- **Stateful Inspection Firewall**
- **Application Level Gateway** (Works at higher layers)

Packet Filtering:

- Simple and effective, uses **packet addresses** and **transport protocol type** to determine policy
- Works at most up to **Transport layer**, but on packet level
- Stateless
- Fast processing
- Doesn't supported advanced user authentication schemes
- Lack of granularity, cannot differentiate between traffic from one application

Application-level Gateway (AKA Application Proxy):

- Acts as a relay for **Application-level** traffic
- Tends to be more secure than packet filters
- Large processing overhead as all traffic must be forwarded

Stateful Inspection Firewall:

- Maintains state from one packet to another in the input stream
- Good at detecting attacks split across muliple packets

Circuit-level Gateway (AKA Circuit-level proxy):

- Can be stand-alone, or can be performed by an application-level gateway for specific applications
- Does not permit end-to-end TCP connections
- Traffic will appear as if it's coming from the gateway itself

Personal Firewalls:

- Useful to compensate for lack of regular firewall
- Can generate logs of accesses

Intrusion Detection System (IDS)

Typically a dedicated device on a system that monitors for malicious or suspicious events on a network

- Monitors user and system activiy
- Audits system config for vulnerabilities and misconfigurations
- Assessing integrity of critical systems and data
- Recognizing known attack patterns in system activity
- Installs and operates traps to record information about intruders

Types of IDS

Signature-Based Intrusion Detection:

Performs simple pattern-matching corresponding to known attacks, such as lots of incoming TCP SYN packets on many ports (*SYN flooding*)

- Cannot detect attack patterns that aren't yet part of their attack pattern database
- Attacks will try to modify basic attacks to not match common attack signatures
- Often uses lots of statistical analysis

Heuristic Intrusion Detection:

Instead of looking for specific patterns, looks for odd behaviour

- Builds a model of acceptable behavior and flag exceptions to that model
- e.g. One specific user may not often use many admistrator utilities. If they suddenly try to access lots of sensitive management utilities, an attacker may have gained access to their account

IDS: Response to Alarms

- Monitor and collect data about the situation
- Act to protect the system, like locking certain resources
- Alert a human to the situation

Effectiveness:

- IDSs can't be perfect. The degree of false positives and false negatives represents the sensitivity of the IDS, which can usually be tuned by a system administrator
- **The Detection Rate** (*DR*) is calculated by $\frac{\text{TP}}{\text{TP}+\text{FN}}$
- **The Precision** is calculated by $\frac{\text{TP}}{\text{TP}+\text{FP}}$

Where:

- TP: True positive
- FN: False negative
- FP: False positive
- TN: True negative

Note: Ideally one would like to have 0 FP and 0 FN

Instrusion Prevention System (IPS): IDS + Firewall

Advanced Encryption Standard (AES): The AES Encryption algorithm (also known as the Rijndael algorithm) is a symmetric block cipher algorithm with a block/chunk size of 128 bits. It converts these individual blocks using keys of 128, 192, and 256 bits. Once it encrypts these blocks, it joins them together to form the ciphertext.

Cryptographic Systems:

Symmetric Encryption:

Advantages	**Disadvantages**
• Fast ciphering/ deciphering • Longer key, more robust encryption	• Secure key exchange • The message's origin and validity cannot be guaranteed.

Examples:

- Triple-DES
- AES (Advanced Encryption Standard)

Asymmetric Encryption:

Advantages	**Disadvantages**
• More secure • Distribution problem is eliminated • Enables use of digital signatures • Allows for nonrepudiation	• Up to 100x slower than symmetric encryption

Examples:

- RSA (Rivest, Shamir, Adleman),
- ECC (Elliptic Curve Cryptography), NTRU

SSL/TLS

- Developed by Netscape Communications
- Security layer between the **Transport** and **Application** layers to protect data exchanges
- Ensures the protection of TCP-based applications
- Secure applications are renamed: HTTPS (443), Telnets (992), FTPS (21)

SSL version 3.0:

- Last SSL version released in 1996
- Integrated in Netscape Navigator and Microsoft Internet Explorer
- Broadly used over Internet to protect exchanges to online web services (bank, electronic commerce…)
- SSLv3 deprecated by Internet Engineering Task Force (IETF) standard organisation in June 2015 (RFC 7568) as non sufficiently secure

Transport Layer Security (TLS):

Similar to SSL 3.0 but with a few adjustments:

- HMAC construction considered by IPsec is adopted
- Key exchange mechanism based on open-source Data Security Standard

TLS organized into 2 parts:

- TLS record protocol: User data protection
- 4 sub-protocols TLS: Establishment and management of TLS sessions

Initialization phase:

- Server must authenticate to client with public key certificate
- Client can optionally authenticate itself to server
- Negotiation of security services and mechanisms
- Establishment of a secret (master) key
- Phase implemented by the TLS Handshake Protocol

Data Protection Phase (TLS 1.0-1.2):

- Data confidentiality
- Data integrity/authentication
- Usage of symmetric encryption to protect this phase
- Phase implemented by the TLS Record Protocol

TLS Record Protocol:

1. Fragmentation	Data blocks are split into smaller fragments
2. Compression	Data fragments are compressed
3. Encryption	Data fragments are encrypted
4. MAC Introduction	Ciphered fragments are appended with a MAC

TLS Sub-Protocols:

1. **Alert:** Alarms transmissions through the Record Protocol
2. **Change Cipher-Spec:** Move to a new security context by the sender
3. **Application Data:** Direct data communication to the Record Protocol
4. **Handshake:** Authentication and security parameters established

TLS Handshake Protocol:

This sub-protocol enables the server and client to:

- Agree on the TLS version
- Agree on security parameters (compression + encryption algorithms)
- Authenticate each other
- Exchange master keys
- Perform replay detection
- Detect message integrity problems

Key Exchange Methods:

- RSA
- DH-DSS
- DH_RSA
- DHE_DSS
- DHE_RSA
- DH_anon

Ciphering Algorithms:

- RC4_128
- 3DES_EDE
- AES_128_CBC
- AES_256_CBC

Hash Functions:

- MD5
- SHA-1
- SHA-256

IP Security (IPsec)

Used to protect IP traffic between two remote networks

Initialization Phase:

- Both entities must authenticate each other (public key + shared secret)
- Negotiation of security services and mechanisms
- Establishment of a secret key
- Phase implemented by the Application level module IKE (Internet Key Exchange)

Data Protection Phase:

- Data confidentiality
- Data integrity/authentication
- Usage of symmetric encryption to protect this phase
- Phase implemented by the IPsec sub-protocol: AH (Authentication Header) or ESP (Encapsulating Security Payload)
- Possibility to create a protected tunnel or to secure an IP packet flow

IPsec Security Sub-Protocols

AH (Authentication Header):
- **Integrity** and **Authentication** of data origin
- Replay Detection (optional)
- Protection over the packet content and part of the header (port 51)

ESP (Encapsulating Security Payload):
- Data **Confidentiality** (optional)
- **Integrity** and **Authentication** of data origin
- Replay Detection (optional)
- Protection over packet content only

IPsec Protection modes:

1. Transport	Only the content of the packet and some fields are protected. Usable only between ends of connection
2. Tunnel	All fields of the packet are protected before being encapsulated in another packet

Uses of modes:

End-to-end Protection	Tunnel/Transport
Protection over Network Segments	Tunnel
Access from a nomad	L2TP Tunnel protected by Transport mode IPsec

Security Association (SA):
A set of parameters for secure communication, e.g.:
- Choice of IPsec protocol mode (Tunnel or Transport)
- Choice of security protocol (AH or ESP)
- Choice of algorithm for encryption and hashing
- SA lifetime
- Key and security association management done by IKE (Internet Key Exchange) protocol (UDP port 500)
- Services offered by IKE:
  - Mutual authentication between IPsec modules
  - Negotiation of the IPsec security associations (enciphering algorithms, key length)
  - Generation of a symmetric encryption key

VPN (Virtual Private Network)

Primary goal: To facilitate communications between companies and their partners, internal communications of a geographically distributed company, or remote communications between a mobile and its company

How it works: Establishing an IP tunnel to exchange data through *(Security is optional to protect the tunnel)*

IP tunnel
Encapsulating an IP packet into another IP packet
- The two remote local networks are virtually forming the same local network thanks to the tunnel
- The packet is going out a private network and is getting into another private network through the tunnel

Security protocols and VPN

Strong need to introduce some security services:
- Data confidentiality
- Data integrity
- Data origin authentication

Services implemented by the security protocols:
- IPsec (IP Security)
- TLS (Transport Layer Security)

Objectives of security protocol: To protect any communication over a network

Two successive phases:
- **Initialization phase:** authentication of entities, negotiation of security services, exchange of encryption keys
- **Data protection phase:** activation of security services over data flows

IPsec and TLS VPN

2 main usages:
- **Interconnecting LANs**
- **Nomad access to a distant network**

Digital nomads are people who travel freely while working remotely using technology and the internet.

IPsec VPN

Interconnecting remote sites:
1. Getting the certificate of the remote gateway
2. Initialization phase with negotiation of a security association
3. Protection of data ensured by each IPsec gateway introducing security headers to encrypt, authenticate their originator, check their integrity

IPsec gateways authenticate each other based on pre-shared key as well as a public key certificate

IPsec users authenticate using their login credentials

IPsec Using a Tunnel (L2TP)

Layer Two Tunneling Protocol
- Known as the standard protocol of tunneling for switched access
- Concurrent proprietary protocol:
  - PPTP (Point-to-Point Tunneling Protocol) from Microsoft with data encryption

IPsec Using a Tunnel (Continued)

L2TP Role:
- Tunnel between a nomad and private network
- No services to ensure data protection

L2TP entities:
- L2TP client (within the device)
- LNS server: L2TP Network Server responsible for L2TP tunnels management, and located within the company's IPsec gateway

IPsec VPN Connection in Detail

1. First establishing an IPsec session IPsec enabling:
    - Protection of IP packets being exchanged between the device and the gateway
    - Authentication of the nomad's equipment
2. Then, establishing an L2TP tunnel:
    - Getting a private address for the device when establishing the L2TP tunnel
    - Authentication of the nomadic user
3. Finally, the user accesses to company's resources

TLS VPN

Interconnecting remote sites:
- TLS communications between the nomad and the TLS VPN gateway
- TLS VPN gateway being used as an interface between the device and the applications within the private network

Clientless solution:
- Creates a secure, remote-access VPN tunnel to an ASA (Adaptive Security Appliance) using a web browser without requiring a software or hardware client.

Clientless solution:

Advantages	Disadvantages
• No need to install specific clients • Easy management • Lower costs	-Restricted access to web applications

Client-based solution:
- Created a secure, remote-access VPN tunnel between a single user and a remote network.

Advantages	Disadvantages
• Access to private network similar to local connection • Easy management • Lower costs	• Sometimes need to install and manage an IPsec/VPN client within the nomad

Comparison of Security Protocols

IPsec

Advantages	Disadvantages
• Reduced key negotiation overhead and simplified maintenance by supporting the IKE protocol • Good compatibility. You can apply IPsec to all IP-based application systems and services without modifying them • Encryption on a per-packet rather than per-flow basis.	-Heavy to manage (application-level IKE module)

TLS

Advantages	Disadvantages
• The most common solution • Largely used for nomad's remote access protection (VPN)	• High Latency • Susceptible to MITM attacks

Software Security

Memory corruption bugs:
Property of memory unsafe languages. Can lead to:
- Arbitrary read
- Arbitrary write
- Control flow hijack
- Control flow corruption

Pointers:
- Allow you to refer to arbitrary memory addresses
- To introduce a bug: Get a pointer pointing somewhere it shouldn't

Memory safety	State of being protected from various software bugs and security vulnerabilities when dealing with memory access
Spatial safety violation	Attempting to access out-of-bounds memory
Temporal safety violation	Attempting to access memory that was previously freed / deallocated

Memory Safety Violations:

Spatial Safety

Bounds Violation (BV)	`int arr[100];`
	`int *p;`
	`p = &(arr[101]);`
	`...*p... // <-BV`

Uninitialized Pointer (UP)	`int *p;`
	`... *p // <-UP dereference`

`NULL` Pointer (NP)	`int *p;`
	`p = NULL;`
	`... *p // <-NP dereference`

Manufactured Pointer (MP)	`int *p;`
	`p = (int*) 42;`
	`... *p // <-MP dereference`

Temporal Safety

Dangling Stack Pointer (DSP)	`int *p;`
	`...`
	`void f() {`
	`  int x;`
	`  p = &x;`
	`}`
	`...`
	`void g() {`
	`  f();`
	`  ...*p... /* <-DSP dereference */`
	`}`

Dangling Heap Pointer (DHP)	`int *p, *q;`
	`p = (int*) malloc(10*sizeof(int));`
	`q = p;`
	`...`
	`free(p);`
	`... *q ... /* <-DHP dereference */`

Multiple Deallocations (MD)	`int *p, *q;`
	`p = (int*) malloc(10*sizeof(int));`
	`q = p;`
	`...`
	`free(p);`
	`free(q); /* <-MD */`

A **dangling pointer** is a pointer to storage that is no longer allocated.

How do we fix these:

Short term:

- Do not teach programmers unsafe practice
- Listen to your compiler

Longer term:

- Maybe we should make it harder to do dangerous things?
- Language standard, compilers, and tools evolve

Memory layout

Command line arguments	High address: 0xFFFFFF
Stack	
↓	
↑	
Heap	
Uninitialised Data segment	
Initialised Data segment	
Code / Text segment	Low address: 0x000000

1. **Code / Text segment:** Contains machine code of the compiled program.
2. **Initialized Data segment:** Stores all global, static, constant, and external variables that are initialized beforehand.
3. **Uninitialized Data segment:** Data in this segment is initialized to arithmetic 0 before the program starts executing.

(LIFO) Stack layout:

Parameters	High address
ret	Return address / pointer
sp + fp	Stack + Frame pointer
Local variables	
↓	Low address

- Stack memory is divided into successive frames where each time a function is called, it allocates itself a fresh stack frame.

Stack pointer: Points to the location of the last item that was stored on the stack.

Frame pointer: Points to the start of the stack frame.

Buffer Overflow:

- Program writes data to a buffer beyond the buffer's allocated memory, overwriting adjacent memory locations. Leading to memory corruption

Arrays are fixed-size blocks of contiguous memory. It's very easy to fall out of the allocated region.

Example using strcopy()

```
char* strcpy(char* destination, char* source);
```

```
char buffer[4];
char tmp = "qqqqqqqqqqqqqqqqqqqqqqqqq";
char *r;
r = strcpy(buffer,tmp);
```

Writes past length of buffer

Consequences:
- Modification / corruption of memory
- Arbitrary code execution

Countermeasures:
- Modern CPUs don't allow you to write and execute regions of memory at the same time
  - This can still be circumvented with attacks like a "return to libc" attack
- **Stack canaries**
  - By inserting a number before the return address, we can validate that it hasn't been corrupted later
- **Shadow stacks**
  - Keep a second stack that contains only return addresses
  - Check for consistency with main stack
  - Not implemented everywhere
- **Use safer functions**
  - strnpcy is better than strcpy
    - strcpy does not check buffer lengths
  - fgets is better than gets
    - gets does not check the buffer length

Format String Errors:
- Formatted output functions consist of a **format string** and a **variable number of arguments**
- By manipulating the format string, we can control execution of the formatted output
- If there are more placeholders than arguments in the format string, the result is undefined
- Arguments expected to be passed on the stack, so printf starts reading values from the stack
- Can be exploited for writing to memory that is out-of-bounds

Safer functions:

Function	Safer Alternative	Use
sprintf()	snprintf()	Store the string and format it as needed
vsprintf()	vsnprintf()	Composes a string with the same text that would be printed if format was used on printf

printf()
fprintf()
	Hardcode the format string. Never let it come directly from any user's input.	Writes output to stream specified

n in function name indicates the maximum number of bytes (*n*, which is taken in as input) (including the terminating null byte ('0')) to be written to a buffer.

C Language Data Types

Type	Length	Format Specifier
char	1 byte	%c
int	4 bytes	%d
long	8 bytes	%l
float	4 byte	%f
double	8 byte	%lf
pointer	4 bytes(x32) 8 bytes(x64)	%p

Operating Systems
A computer program that:
- Multiplexes hardware resources
- Implements resource abstractions
- **Multiplexing:** Allows multiple people or programs to use the same set of hardware resources – processors, memory, disks, network connection – safely and efficiently
- **Abstractions:** processes, threads, address spaces, files, and sockets Simplify the usage of hardware resources by organizing information or implementing new capabilities

Protection Boundaries
- Multiple privilege levels different for different groups. e.g. Admin, User
- Different software can run with different privileges
- Processors provide at least two different modes
  - User space: How "regular" programs run
  - Kernel mode: How the kernel runs
- The mode determine a number of things
  - What instructions may be executed
  - How addresses are translated
  - What memory locations can be accessed

Example: Intel

Ring 0	Most privileged level, where kernel runs
Ring 1/2	Designed to run less privileged code such as device drivers or vm monitors. More restrictions than the kernel, often ignored
Ring 3	Where "normal" processes live

Memory is divided into segments:
- Each segment has a privilege level (0 to 3)
- Processor maintain a current privilege level (CPL)
- Can read/write in segment when CPL $\geq$ segment privilege
- Cannot directly call code in segment where CPL < segment privilege

Example: MIPS

User mode	Can access CPU registers; flat uniform virtual memory address space
Kernel mode	Can access memory mapping hardware and special registers

Changing Protection Level

Sleeping beauty approach	Wait for something to happens to wake up the kernel. These include: System Calls, Interrupts or a Trap
Alarm clock approach	Set a timer that generate an interrupt when it finishes

System Call

- Way for userspace programs to interact with the kernel
- Defined by the API of the operating system
  - The API provides an interface between a process and operating system to allow user-level processes to request services of the operating system.

Trap

- An unintended request for kernel service triggered by an exception or error in a process when executing an application , e.g.
  - Attempt to divide by 0
  - Attempt to access invalid memory
  - A breakpoint
- Any response directly affects the program that generated the trap
- After a trap / exception / error has been handled, the processer returns to it's previous activity

Interrupts

- A hardware or software signal that demands attention from the OS, e.g. I/O completion
- Handled independently of any user program, unlike a trap

Abstractions

Abstractions simplify applications design by:

- Hiding undesirable properties;
- Adding new capabilities;
- Organizing information

Abstractions provide an interface to programmers that separates **policy** – what the interface commits to accomplish – from the **mechanism** – how the interface is implemented.

Processes	Abstract what is being done
Threads	Abstract the CPU
Address Space	Abstract the memory
Files	Abstract the disk

Files

- What **undesirable properties** file systems **hide**?
  - Disks are slow!
  - Chunk of storage are distributed all over the disk
  - Disk storage may fail
- What new **capabilities** do files add?
  - Growth and shrinking
  - Organization into directories
- What **information** files help to organize?
  - Ownership and permission
  - Access time, modification time, type etc.

Processes

- Processes are not tied to a hardware component
- They contain and organize other abstractions

Processes vs Threads

- Both described as "running"
- Processes require multiple resources: CPU, memory, files
- Threads abstract the CPU
- **A process contains threads, threads belong to a process**
  - Kernel threads do not belong to a user space process
- A process is running when one or more of its threads are running

Process Example: Firefox

- Firefox has **multiple threads**. What do they do?
  - Waiting and processing interface events
  - Redrawing the screen as necessary
  - Loading elements in web pages
- Firefox is using **memory**. For what?
  - The executable code itself
  - Shared libraries: web page parsing, TLS/SSL etc.
  - Stacks storing local variables for running threads
  - A heap storing dynamically allocated memory
- Firefox has **files** open. Why?
  - Fonts
  - Configuration files

Memory Layout

Stack	Scratch space for a thread of execution. When a function is called, a block is reserved on the top of the stack. When that function returns, the block becomes unused and can be used the next time a function is called
Heap	Used for dynamic allocation, unlike the stack, there is usually only one shared heap

Process as a protection boundary

- The OS is responsible for isolating processes from each other. Processes should not directly affect other processes
- Intra-process communication (between threads) is application responsibility
  - Shared address space
  - Shared file descriptors

Physical Memory

- Maximum amount of addressable physical memory is 2^P, where the physical address is P bits long
- OS161′s MIPS is 32 bits = 2^{32} physical addresses = maximum of 4GB memory
- Modern CPU support large amount of addressable memory
- x86_64 supports 52-bit physical addressing, 48-bit virtual addressing
- Needs to be shared between all processes
- Needs to be carefully managed to avoid processes interfering with one another

Virtual Memory

- The kernel provides virtual memory for each process
- If virtual memory addresses are V bits, the amount of addressable virtual memory is 2^V
  - On OS161/MIPS: V=32
- Running processes only see virtual memory
- Each process is isolated in its virtual memory and cannot address the virtual memory of other processes
- May be larger than physical memory

Why virtual memory?

- Isolate processes from each other
- To support virtual memory larger than physical memory
  - Provide greater support for multiprocessing

Memory-Mapping Unit (MMU)

- Is a piece of hardware
- Maps virtual memory addresses to physical addresses
- Only configurable by a privileged process (i.e. the kernel)
- Virtual addresses are what a process uses
- Physical addresses is what the CPU presents to the RAM

Base + Bound:

- Associate virtual address with base and bound register
- Base: where the physical address space start
- Bound: the length of the address space (both virtual and physical)

Base + Bound

Pros:
- Allow each virtual address space to be of different size
- Allow each virtual address space to be mapped into any physical RAM of sufficient size
- Straightforward isolation: just ensure no overlap!

Cons:
- Wastes physical memory if the virtual address space is not fully used
- Same privilege everywhere (read/write/execute)
- Sharing memory can only happen by overlapping top and bottom of two spaces (if need to be shared by more than 2?)

Segmentation:

A single address space has multiple logical segments
- **Code**: read/execute, fixed size
- **Static data**: read/write, fixed size
- **Heap**: read/write, dynamic size
- **Stack**: read/write, dynamic size

Each segment is associated with privilege + base + bound

Segmentation

Pros:
- Can share memory at the segment granularity
- Waste less memory (i.e. hole between heap and stack doesn’t need to be mapped)
- Enables segment granularity memory protection

Cons:
- Segments may be large
  - Need to map the whole segment into memory even to access a single byte
  - Cannot map only the part of the segment that is utilized
- Need to find free physical memory large enough to accommodate a segment
- Explicit segment management is not very elegant (better with partitioned address)

Paging

Paging is a function of memory management where a computer will store and retrieve data from a device’s secondary storage to the primary storage.
- Seperates virtual memory into fixed-size units called pages (with no more bounds)

Paging

Pros:
• Can allocate virtual address space with fine granularity • Only need to bring small pages that the process needs into the RAM
Cons:
• Bookkeeping becomes more complex • Lots of small pages to keep track of

Fine granularity: Segmented level of access to small chunks / pages

- Access the exact small bit you need

Page Tables: A data structure used by a virtual memory system in a computer operating system to store the mapping between virtual addresses and physical addresses.

Single-level page table

- The most straightforward approach would simply have a single linear array of page-table entries (PTEs).
- Each PTE contains information about the page, such as its **physical page number** (a pointer to the associated **physical address**) as well as **status bits**, such as whether or not the page is valid.
- Need to keep around a mapping between virtual page and physical page
- Suppose 32bits addresses:
  - 12 bits offset (4kb per page)
  - 20 bits for page number (~1 million entries)
  - 4 bytes for a PTE (=> Table size: 4Mbytes)
- **For 100 running processes:**
  - Each process needs its own table
  - 400 Mbytes of RAM just to hold all the page tables

Two-level page table

- Multi-level page tables are tree-like structures to hold page tables
- Virtual address encodes a directory number, a page number, and an offset
- Directory number is used to locate a page table, rest functions like a single-level page table
- Suppose 32bits addresses:
  - 12 bits offset (4kb per page)
  - Virtual address:
    - 10 bits for the directory (L0) index
    - 10 bits for the page (L1) index
    - 12 bits for the offset within a page.
- The entries of the **L0 page table** are pointers to entries in the **L1 page table**, and the entries of the L1 page table are PTEs

If we have one valid page in our process, now our two-level page table only consumes:

$$\left(2^{10}\text{ L0 entries}\right)\cdot\left(2^{2}\text{ bytes/entry}\right)+$$
$$1\cdot\left(2^{10}\text{ L1 entries}\right)\cdot\left(2^{2}\text{ bytes/entry}\right)=$$
$$2\cdot 2^{12}\text{ bytes}=8\text{ kbytes}$$

Swapping pages

- Physical RAM may be oversubscribed
  - Total virtual pages > number of physical pages
- Swapping is moving virtual pages from physical RAM to a swap device (any storage device)

Page Faults

- When a process tries to access a page not in memory
  - MMU detects this and raise an exception
- The kernel's job on page fault is to:
  - Swap the page from secondary storage to memory, evicting another page if necessary
  - Update the Page Table Entry
  - Return from the exception so the application can try again
- Page faults are slow
  - Milliseconds to swap from hard drive
  - Microseconds to swap from SSD
  - Getting a page hit from RAM only takes nanoseconds

Page Swapping

First In First Out (FIFO)	Remove the page that has been in memory the longest
MIN	Replace the page that will not be referenced for the longest
Least Recently Used (LRU)	Remove the page that has been used the least recently (temporal locality)
Clock	(See next paragraph)

Clock

- Add a "used" bit to PTE
  - Set by MMU when page accessed
  - Can be cleared by kernel

Pseudocode:

Inputs: victim (v), page (p), number of frames (n), use bit (.ub), frames (f)

```
if page exists in frames:
  set page.ub = 1;

while victim.ub = 1:
  set victim.ub = 0;
  victim = (victim + 1) % n;

evict victim;
set frames[victim] = page;
set page.ub = 1;
victim = (victim + 1) % n;
```

Operating Systems History:

1940-1955

- Computer are exotic experimental equipment
- Program in machine language
- Use plugboard to direct computer
- Program manually loaded via card decks
- Goal: churn table of numbers (e.g. accounting)

1955-1970

- Move the users away from the computer, give them terminal
- OS is a simple batch-processing program that:
  - Loads a user job
  - Runs it
  - Moves to the next job
- Program fails? Record memory, save it, move on
- Efficient use of hardware (less “downtime”)
- Hard to debug

Problems:

- Utilization is low (one job at a time)
- No protection between different jobs
- Short jobs get stuck behind long ones

Solutions:

- Seperate code and data for better memory protection
- Multiprogramming: many users share the system
- Scheduling: let short jobs finish quickly

The First OS:

- OS/360 introduced in 1963
- Lecture notes seem to suggest it didn’t really work until 1968 but I don’t have any other source for that
- Written in assembly code

1970-1980

- Interactive timesharing: many people use the same machine at once
- Terminals are cheap: everyone gets one!
- Emergence of the file systems
- Try to give reasonable response time
- Compatible Time-Sharing System (CTSS)
  - Developed at MIT
  - The first general-purpose time-sharing system
  - Pioneered much of the work on scheduling
  - Motivation for MULTICS
- MULTICS
  - Joint development by MIT, Bell Labs, General Electric
  - One computer for everyone, people will buy computing as they buy electricity
  - Many influential ideas: hierarchical file systems, devices as files

UNIX

- Ken Thompson (worked on MULTICS) wanted to use an old computer available at Bell Labs
- He and Dennis Ritchie built a system built by programmers for programmers
- Originally in assembly, later rewritten in C
- Universities got the code for experimentation
- Berkeley added virtual memory support
- DARPA selected UNIX as its networking platform (ARPANET)
- UNIX becomes a commercial OS
- Important ideas popularized by UNIX
  - OS written in a high-level language
  - OS is portable across hardware platforms
  - Mountable file systems

1980-1990

- Put a computer in each terminal!
- CP/M first personal computer operating system
- IBM needed an OS for its PC, CP/M behind schedule
- Approached Bill Gates (Microsoft) to build one
- Gates approached Seattle Computer Products, bought 86-DOS, and created MS-DOS
- Goal: run Control Program/Monito (CP/M) programs and be ready quickly
- Market is broken horizontally
  - Hardware
  - OSes
  - Applications

1990s-Today

- Connectivity is main priority
- Networked applications propel industry
  - Email
  - Web
- Protection and multiprogramming less important for individual machines
- Protection and multiprogramming more important for servers
- New network-based architectures
  - Clusters
  - Grids
  - Distributed Operating Systems
  - Cloud
- Linux everywhere! (except in workstations)