

Course Companion

AP Cybersecurity

Every topic handout in one volume

exercises.lol

Contents

1. Introduction to Security.....	2
2. Securing Spaces.....	7
3. Securing Networks.....	14
4. Securing Devices.....	20
5. Securing Applications and Data.....	27

Introduction to Security

AP Cybersecurity

Understanding Social Engineering

The weakest part of any computer system is often the **human** using it. **Social engineering** is the art of tricking people into breaking security - giving away a password, opening a bad file, or clicking a bad link. The attacker (we call them an **adversary**) does not need to break the code; they only need to fool a person.

Most social engineering happens by **email**, text message, or social media, though it can also happen in person or by phone. The goal is **elicitation** - getting sensitive information out of someone without them realising.

Adversaries lean on two powerful feelings:

- **Intimidation** - the adversary threatens a bad result if you do not obey. Fear pushes you to act.
- **Urgency** - the adversary invents a deadline (“reply in the next hour or your account closes”). When we feel rushed, we stop thinking carefully about whether an action is safe.

Adversary's pressure tactics



AP Cybersecurity · U1 · social engineering: pressure tactics -> victim action

Social engineering uses psychological pressure to make a victim act before they think

The **impact** on a victim can be serious. They might reveal personal details (name, address, pet's name, birthday) that are later used to answer security **challenge questions** and **impersonate** them. They might hand over a **one-**

time password (OTP) , letting the adversary log in as them. Or they might download **malware** that steals data from their browser.

Worked example. A phishing email reads: “Over 90% of staff have already verified their account - confirm yours in the next hour or lose payroll access.” Two tactics are stacked here. “In the next hour” is **urgency** (a deadline that rushes you), and “over 90% of staff have already” is **consensus** (social pressure to follow the crowd). Naming each tactic - not just calling the email “suspicious” - is exactly what an exam answer needs.

Suspicious Website Logins



A hardware security key: strong authentication reduces damage when a password is phished

A **password attack** is any attempt to log in using guessed or stolen passwords. In an **online** password attack the adversary tries passwords against a real login page. The warning signs are visible in the logs:

- many failed logins in a short time,
- login attempts at unusual hours,
- login attempts from unknown devices.

Adversaries succeed because people choose **weak** passwords. Common patterns include a word plus a two-digit year plus a special character (like **Summer24!**), or a pet's or family member's name. Because these patterns are so common, an adversary can build a **dictionary** of likely passwords from information gathered about you and let an automated tool try each one.

To make **authentication** stronger:

- Create passwords that are **long, random, and unique** - a **password manager** can generate and store them for you.
- Avoid names, dates, and meaningful words.
- Turn on **multifactor authentication (MFA)** , which asks for extra proof (like a texted code) on top of the password.

Best Practices for Public Networks

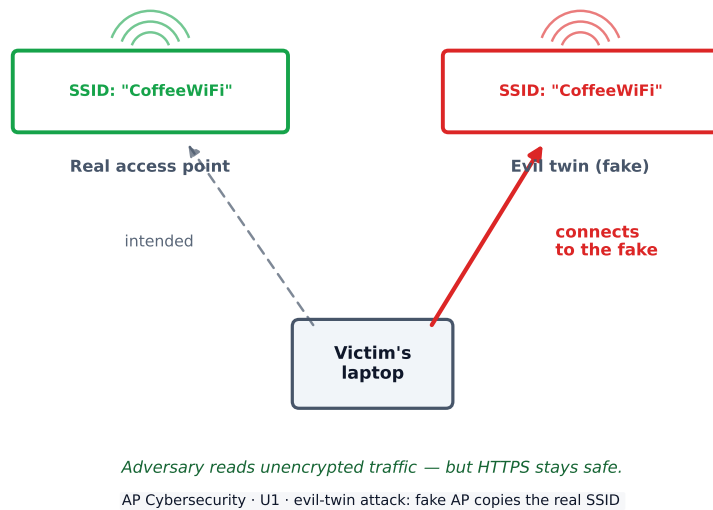


A security token: one-time codes and tokens stop password-only logins from being enough

Not all adversaries are the same. We classify them by **skill**: low-skilled attackers buy ready-made tools online and reuse known **exploits** , while high-skilled attackers write their own tools and can discover brand-new holes called **zero days** . Their **motivation** varies too - greed, revenge, politics, or belief.

Public **Wi-Fi** is a favourite hunting ground. Three wireless attacks you must know:

- **Evil twin** - the adversary sets up a fake **access point** with a name (**SSID**) copied from the real network. Victims connect to the fake one, and the adversary reads their traffic (though **encrypted** sites like HTTPS stay safe).
- **Jamming** - the adversary floods the air with a strong radio signal so no one can connect. This is one kind of **denial of service (DoS)** attack.
- **War driving** - the adversary drives around detecting wireless networks and where their signal leaks outside a building.



An evil-twin access point copies the real network's name so victims connect to the attacker

To protect yourself on public networks: check that the network name **exactly** matches the one you intend to join, prefer encrypted sites, and consider a **virtual private network (VPN)**, which encrypts all of your traffic to the VPN operator.

AI-Based Cybersecurity Attacks

Artificial intelligence gives adversaries powerful new tools. With enough voice and image samples, an adversary can build a **deepfake** avatar to impersonate someone on a call. **Large language models (LLMs)** let them write convincing **phishing** emails in perfect, native-sounding language - removing the clumsy wording that once gave scams away.

AI also helps adversaries **on the back end**: crafting prompts that pull secret data out of an LLM, planting false information on websites so it poisons an LLM's training data, scanning the internet to gather facts about a target, and even writing new malware.

You can defend against many AI-augmented attacks: agree on a **shared secret** word with close contacts to verify identity, enable MFA (so a cloned voice alone cannot log in), never type sensitive data into a chatbot, and always double-check AI output against reliable, non-AI sources.

AI writes code, and that cuts both ways. Adversaries use **AI-enhanced coding tools** to write new **malware** faster than they could by hand, to modify existing application code so that it performs malicious activity, and to scan a codebase for **vulnerabilities** to attack. The skill barrier falls: someone who

could not previously write an exploit can now ask for one, so the number of capable attackers rises even when no new technique is invented.

Leveraging AI in Cyber Defense

The same technology defends us. AI tools can analyse an application's own source code, identify **vulnerabilities** in it and **recommend mitigations**; they can also review firewall rules and access settings and **recommend** safer options - though a human expert must always check the advice before applying it. AI can scan application code for weaknesses and suggest detection rules.

△ **A recommendation is not a fix.** The CED is explicit that the advice must be reviewed and implemented by a **knowledgeable programmer**: an AI tool can be confidently wrong about whether a flaw is exploitable, and applying a suggested patch without understanding it can introduce a new fault of its own.

Its biggest advantage is **scale**. A medium network produces millions of events every day - far too many for people to read. AI can quickly sort the harmless events from the likely-malicious ones, **alert** human staff, or take an automatic action. This lets defenders catch an attack and respond in seconds instead of days, preventing loss and damage.

That scale is what makes **threat detection and response** possible in practice: an AI system flags malicious activity as it happens, so the response team can **intervene quickly** enough to prevent loss, harm, or destruction of digital infrastructure — rather than reading the logs days later and finding out what was taken.

Exam tips

- When a question asks you to rank risks, remember **high risk = high impact AND easy to exploit**. A parking-lot Wi-Fi leak matters less than an **open internal port that lets an adversary spoof a device**.
- Learn the social-engineering tactics by name - **intimidation, urgency, pretexting, authority, consensus, scarcity, familiarity** - and be ready to spot which one an email is using.
- **Encryption still protects you on an evil twin**: the adversary sees your traffic but cannot read HTTPS. Say *what* is exposed, not just “it's unsafe”.
- For “how to make authentication stronger”, **MFA** is almost always part of the answer, plus long/unique passwords from a manager.
- AI is **dual-use**: the same tool (LLMs, code analysis) appears on both the attack and the defense side. Read the question carefully to see which side it asks about.

Securing Spaces

AP Cybersecurity

Cyber Foundations

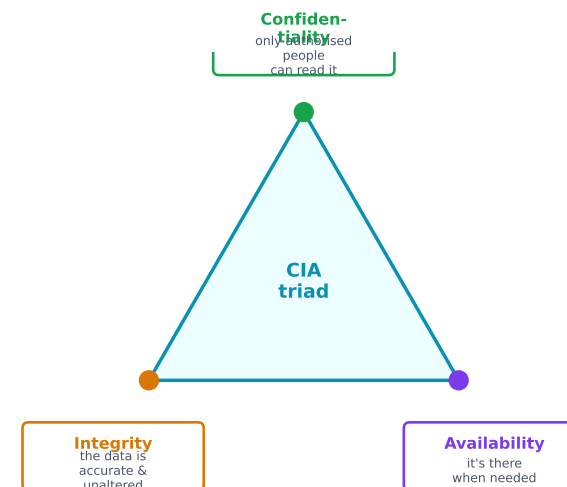


A monitor wall: network monitoring and logging help detect intrusions

Before defending a system, you need a shared language. This section builds it.

Every security control protects at least one part of the **CIA triad** - the three goals of security:

- **Confidentiality** - only authorised people can read the data.
- **Integrity** - the data is accurate and unaltered.
- **Availability** - the data and services are there when needed.



CIA triad: Confidentiality · Integrity · Availability (core security goals)

The CIA triad: the three goals every security control supports

Attacks come from different adversaries, classified by their goals. A **script kiddie** reuses tools built by others for greed or recognition; a **hacktivist** acts for a political, social, or personal cause; an **insider** already holds legitimate access and may act from revenge or greed; a **cyberterrorist** disrupts critical infrastructure like a power grid or water plant; and **transnational criminal organisations** chase money through ransomware and stolen data.

Most attacks unfold in **phases**: **reconnaissance** (gathering information, often from public **OSINT** sources), initial access, persistence, **lateral movement** (spreading to more systems by escalating privileges), taking action on the goal, and evading detection. Naming the phase an attacker has reached helps a defender choose the right response.

Social engineering: the seven tactics

Most attacks begin not with code but with **social engineering** - psychological tricks that manipulate a person into doing what the adversary wants. The exam names **seven** tactics, and expects you to identify which one a scenario shows:

Tactic	The trick
Pretexting	inventing a believable reason to make contact (“I’m from IT, verifying your account”)
Authority	posing as someone powerful, or relaying “the boss’s” instructions
Intimidation	threatening negative consequences if a demand is not met
Consensus	claiming everyone else is already doing it, to create social pressure
Scarcity	inventing limited availability (“only 2 left”)
Familiarity	pretending to be, or to know, someone close to the target
Urgency	imposing a tight deadline so the target acts before thinking

the common thread is that all seven bypass a target’s judgement by triggering an automatic emotional response - fear, trust, haste, or the wish to fit in. The defence is the same each time: **verify through a separate, trusted channel** before acting.

A **risk** appears when a **threat** can exploit a **vulnerability** to compromise an **asset** (anything valuable - data, money, hardware, reputation). We **assess** risk by weighing two things: the **likelihood** of an attack and the **severity** of the damage.

Likelihood itself depends on the **value** of the target (adversaries chase what looks worth stealing), the **skill** needed to exploit the vulnerability (a well-documented exploit needs little skill, so more adversaries can use it), and the **motivation and capability** of likely adversaries. Severity is usually measured in **financial** cost, but includes **reputational** and **operational** damage too.

The final rating can be written two ways, and the exam wants you to tell them apart:

- **quantitative** - a number: a score on a scale (e.g. 1-10), or a money value (e.g. "a \$10,000 annual risk").
- **qualitative** - a label: *low / medium / high / severe*, or a grid such as *likely-high-impact* vs *unlikely-low-impact*.

A written **risk assessment** should record, for each risk: the vulnerable asset and its value, the likely threats, how the specific vulnerability would be exploited, the severity if it were compromised, and a final quantitative or qualitative rating.

Once a risk is measured, an organisation has four ways to **manage** it:

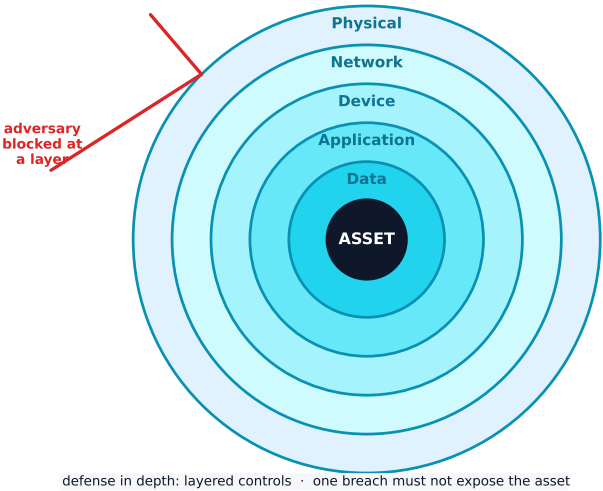
- **Avoid** - stop the risky activity (only possible if it isn’t essential).
- **Transfer** - shift the burden to someone else, such as an insurer.

- **Mitigate** - add controls to lower the likelihood or impact.
- **Accept** - live with the leftover **residual risk**, because perfect security is impossible.

Security controls are grouped two ways. By **type**: **physical** (locks, fences, guards), **technical** (firewalls, anti-malware, encryption), and **managerial** (policies and procedures). By **function**: **preventative** (stop an attack, like a lock), **detective** (spot an attack, like a camera), and **corrective** (fix and restore, like patching).

Worked example. A hospital stores patient records on an unencrypted server in an unlocked room. Rate the risk: the asset is highly sensitive (patient data, protected by law) *and* the vulnerability is easy to exploit (no encryption, no access control), so this is a **high** risk. Now classify one fix - a door lock: by **type** it is a *physical* control, and by **function** it is *preventative* (it stops entry before an attack even begins).

The best strategy layers many controls - a **defense-in-depth** approach. If an adversary bypasses one layer, another still stands. Layers include human, physical, network, device, application, and data.



Defense in depth: many layers so one breach does not expose the asset

Physical Vulnerabilities and Attacks

Digital security means nothing if an adversary can simply walk in. Common **physical attacks** often begin with social engineering:

- **Piggybacking** () - tricking an authorised person into holding a door open (for example, by carrying a heavy box).
- **Tailgating** () - slipping through a secured door behind someone without their knowledge.
- **Shoulder surfing** - watching someone type a password or read sensitive information.
- **Dumpster diving** - searching a target's trash for useful information.
- **Card cloning** - copying an access card to enter restricted areas.

With physical access, an adversary can cut power, steal or copy data, or plug in a **keylogger**. We rate physical risk as **high** when sensitive systems sit in a space without controlled access, **moderate** when an unimportant area could act as a **foothold** to reach other resources, and **low** when the asset is worthless and unlikely to be attacked.



A padlock on a chain: physical security is the first layer — locks, doors and barriers matter

Protecting Physical Spaces

Managerial controls come first: **security-awareness training** teaches staff not to badge strangers in, and a **workstation security policy** requires locking devices, clearing desks (a **clean desk policy**), and using privacy screens.

Physical controls then harden the building: fences, gates, and **bollards** deter access; locks protect doors and cabinets; **card readers** log and restrict entry; an **access control vestibule** (a two-door airlock) stops piggybacking;

disabling **USB ports** blocks malware drives; and an **uninterruptible power supply (UPS)** keeps devices running through an outage. Organisations prioritise these by matching the cost of a control to the severity of the risk.



A dome security camera: physical controls and monitoring protect spaces as well as networks

Detecting Physical Attacks

Some controls **detect** attacks rather than prevent them. **Cameras** record activity and help after-incident investigations; **security guards** respond to what they see; **motion sensors** alert staff to movement; and employees themselves often notice an intruder first.

Placement matters. Cameras belong at **points of ingress and egress** (entrances and exits). Motion sensors work best in low-traffic areas like server rooms - put them in a busy hallway and constant false alarms make everyone ignore them. Stationary guards protect a fixed high-value point, while patrolling guards are harder for an adversary to plan around. Reviewing **door-open times** in entry logs can even reveal piggybacking, because a door held open too long is suspicious.

Exam tips

- Memorise the **CIA triad** and be ready to say *which* goal a control protects - encryption serves **confidentiality**, a hash checks **integrity**, a backup restores **availability**.

- Know the **four risk responses** (avoid, transfer, mitigate, accept) and the two ways to **classify controls** (by type: physical/technical/managerial; by function: preventative/detective/corrective).
- Distinguish **piggybacking** (with consent, tricked) from **tailgating** (without the person's knowledge) - exam questions test this exact pair.
- For risk-rating questions, **high risk needs both high value AND easy exploitation**; a “foothold to other systems” is the classic **moderate** risk.
- **Defense in depth** is the model answer whenever a question asks why one control is not enough.

Securing Networks

AP Cybersecurity

Network Vulnerabilities and Attacks

A **network** connects devices so they can share data - and every connection is a possible way in. You must know the classic network attacks and the tricks behind them.

- **ARP poisoning** - the **address resolution protocol (ARP)** pairs IP addresses with hardware **MAC addresses**. An adversary sends fake ARP messages so traffic meant for the target flows to the adversary instead. This is an **on-path attack** (also called man-in-the-middle): the adversary secretly sits between two parties, reading and even altering their messages.
- **MAC flooding** - flooding a **switch** with fake MAC addresses forces it into broadcast mode, so the adversary can capture all traffic. This is a form of **eavesdropping**.
- **DNS poisoning** - planting a fake record on a **domain name system (DNS)** server redirects users to a malicious site to steal credentials (**credential harvesting**).
- **Smurf attack** - flooding a network with **ICMP** requests aimed at the broadcast address, so every device replies to the victim. It is a **denial of service (DoS)** attack; when many machines attack at once it becomes a **distributed denial of service (DDoS)**.

Adversaries exploit weak networks to flood, map, or spoof devices. A physical data port with no **port security** lets an attacker plug in; an open port lets them install a **rogue access point** that bypasses the firewall entirely. We rate network risk by impact and by how much skill the exploit needs.

To find weaknesses **before** an adversary does, organisations run an **automated vulnerability scanner**: a tool that checks networks, devices, and applications against a database of **known** vulnerabilities, then produces a report listing each one found, how **severe** it is, and a recommended **mitigation**. Fixing the highest-severity items first is a core part of managing network risk.

Protecting Networks: Managerial Controls and Wireless Security

Good network security starts with written **policies** that set a minimum standard: a **router security policy** and **switch security policy** ban local accounts and require

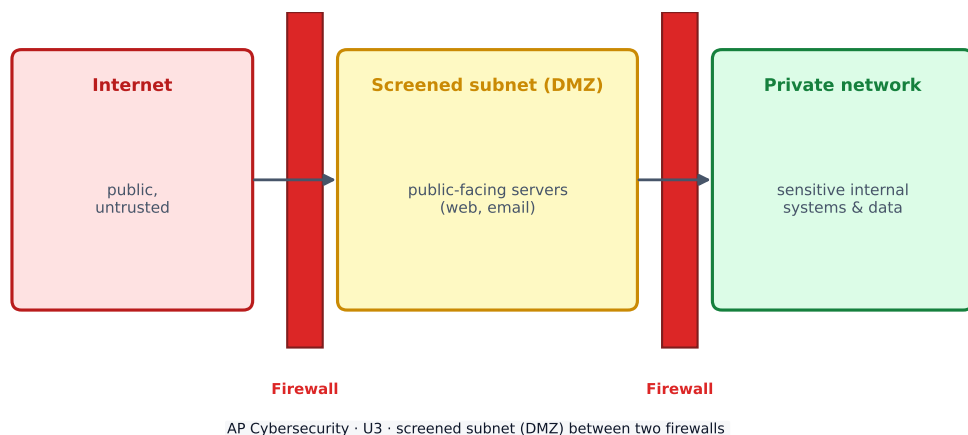
port security; a **VPN policy** sets authentication rules and forbids **split tunneling**; and a **wireless security policy** requires strong encryption and authenticated access.

For wireless networks specifically, organisations **disable beacon frames** so the network is harder to find, **control signal strength** so it does not leak outside the building, **enable strong encryption - WPA3** Wi-Fi is the current strongest, while old **WEP** and the original **WPA** are broken - and use **MAC filtering** to allow only known devices.

Protecting Networks: Segmentation

Network segmentation divides one network into smaller, isolated pieces (**subnets**). If one subnet is breached, the damage is **contained** and cannot spread.

A key pattern is the **screened subnet** (also called a **DMZ**). It sits between the public internet and the private internal network, holding an organisation's public-facing servers in a lower-security zone - separated from the sensitive internal systems.



A screened subnet (DMZ) puts public servers between two firewalls, away from the private network

Segments can also be built with **subnetting** (by IP address) or **VLANs** (logically separating devices on the same switch). Each segment can then get its own security policy - higher-security and lower-security zones.



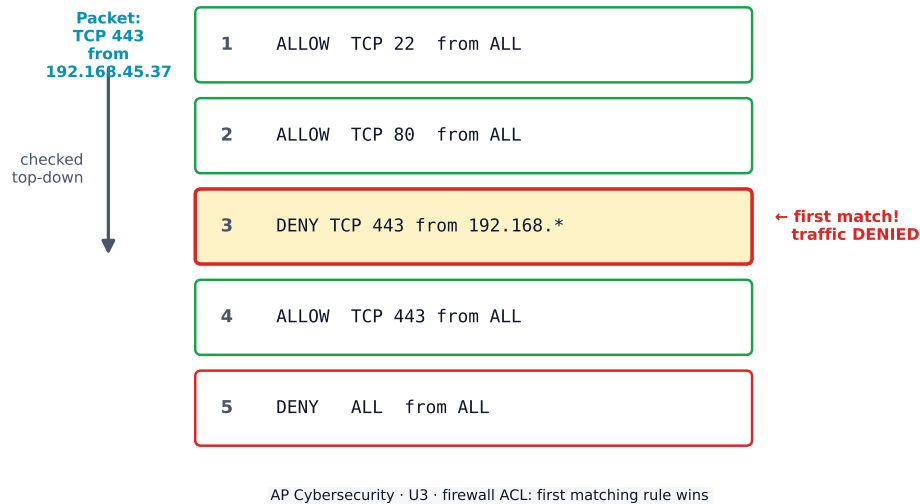
Server racks: network segmentation isolates systems so one breach does not open everything

Protecting Networks: Firewalls

A **firewall** allows or denies traffic entering or leaving a network. There are several kinds:

- **Stateless** - filters on packet headers alone (IP, port, protocol).
- **Stateful** - also tracks the **state** of each connection for finer control.
- **Next-generation (NGFW)** - adds advanced features like intrusion prevention and deep packet inspection.

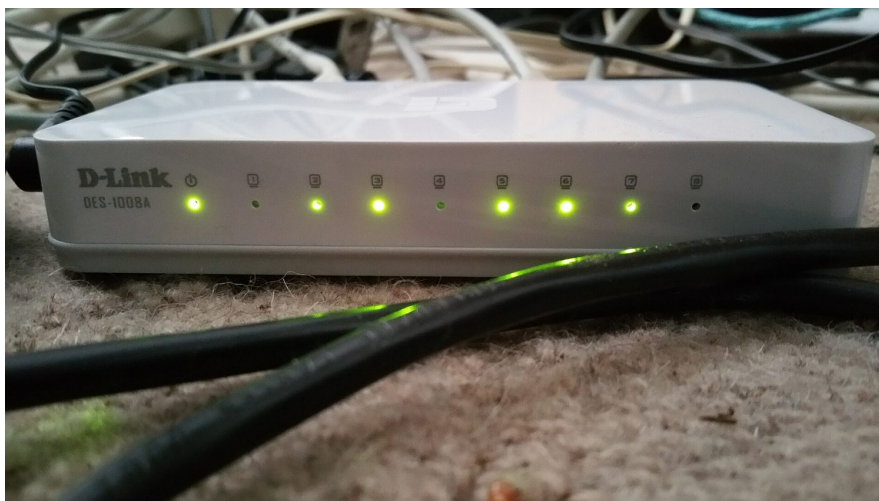
A firewall follows an **access control list (ACL)** - an ordered set of rules. Rules are checked **in order**, and the **first match wins**, so the order of rules changes which traffic gets through. Each rule specifies a direction, a thing to filter by (IP, port, service), and an action (permit or deny).



A firewall checks its ACL top to bottom; the first matching rule decides

Worked example. A firewall has Rule 3: DENY TCP 443 from 192.168.*, and lower down Rule 7: ALLOW TCP 443 from ALL. A user at 192.168.45.37 cannot reach port 443 - even though Rule 7 would allow them - because Rule 3 matches first, and the *first match wins*. The fix is to move the ALLOW rule **above** the DENY. This is why rule order, not just rule content, decides what traffic gets through.

Firewalls belong at every point where data crosses between zones - at each network segment and at every gateway to the public internet.



Real network hardware: a firewall is a device (or software) sitting where these cables meet the outside world

Detecting Network Attacks

When prevention fails, **detection** takes over. Automated tools read the **log files** that record network activity:

- a **network intrusion detection system (NIDS)** analyses traffic and **raises an alert**, but does not block;
- a **network intrusion prevention system (NIPS)** can also **stop** an attack by closing ports or blocking addresses;
- a **security information and event management (SIEM)** system gathers data from many sources to spot patterns.

There are two detection methods. **Signature-based** detection compares traffic to a database of known attack **signatures** - fast and low on false alarms, but blind to brand-new attacks. **Anomaly-based** detection compares traffic to a normal **baseline** and flags anything unusual - it can catch novel attacks but needs more resources and raises more false alarms. A **hybrid** approach combines both.

Examining captured traffic (**packet-capture** files), analysts hunt for **network-based indicators of compromise** in the source and destination IP addresses, ports, and protocols. Four common ones: connections to **known-malicious IP addresses**, **unauthorized network scans** (an outsider probing your ports), unusual **spikes or slowdowns** in traffic, and **mismatched port-application traffic** (for example, non-web traffic flowing over port 80). These complete the host-, file-, and behaviour-based indicators a single device logs.

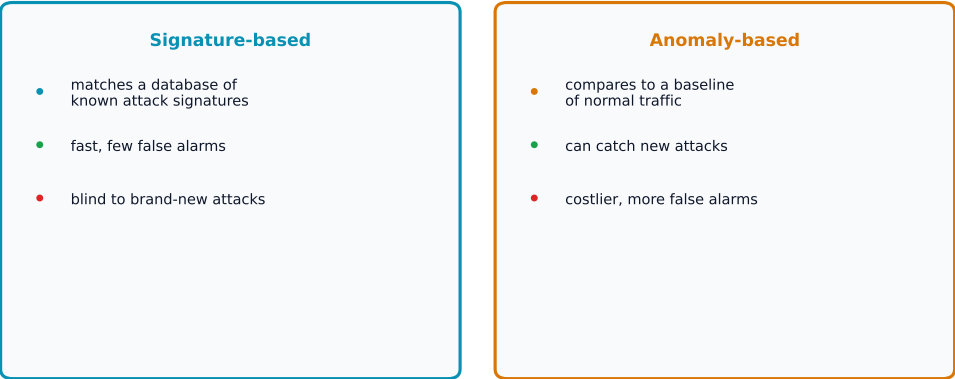
AI, thresholds, and alert fatigue

A medium network logs **millions** of events a day - far more than any team can read - so organisations train **AI** models to sort likely-malicious patterns from normal ones. These models are **probabilistic** : rather than a yes/no, each event gets a **percentage** likelihood of being malicious.

The organisation then sets a **threshold** - the likelihood at which an alert fires - and that choice is a genuine trade-off:

- set the threshold **too high** and real attacks slip through **undetected**;
- set it **too low** and the team is **overwhelmed** with false alerts.

Too many false alerts cause **alert fatigue** : responders get so used to false positives that they start assuming an alert is false *before* investigating it - so a real attack, when it finally comes, is waved away. This is exactly why a **low false-positive rate matters**: signature-based detection has almost none, while anomaly-based and hybrid detection trade a higher false-positive rate for the ability to catch novel attacks.



AP Cybersecurity · U3 · signature-based vs anomaly-based detection

Signature-based detection matches known attacks; anomaly-based detection flags deviations from normal

Exam tips

- For firewall-ACL questions, **read the rules top-to-bottom and stop at the first match** - a Deny rule above an Allow blocks the traffic even though the Allow exists lower down.
- Pair each attack with its **tell-tale sign**: ARP poisoning = one IP with two MAC addresses; MAC flooding = a surge of new MAC addresses; DNS poisoning = an unexplained drop in web traffic.
- Read **packet captures** for network-based IoCs: known-malicious IPs, unauthorized scans, traffic spikes/slowdowns, and mismatched port-application traffic.
- Run **vulnerability scanners** to find *known* weaknesses proactively, and fix the highest-severity findings first.
- **Signature-based** = fast, few false positives, misses new attacks (more false negatives); **anomaly-based** = catches new attacks, costs more, more false positives. Memorise this trade-off.
- A **screened subnet** / **DMZ** holds public-facing servers between the internet and the private network - name it whenever a question separates public services from internal data.
- **WPA3** is the strong wireless encryption; **WEP** and original **WPA** are insecure.

Securing Devices
AP Cybersecurity

Device Vulnerabilities and Attacks

A **device** is any computer - a server, a personal laptop, a smartphone, or an **embedded computer** built into a machine. Everyday devices with embedded computers are called **Internet of Things (IoT)** devices, and they run everything from water pumps to washing machines.

The four classes of device, and why the class matters

Class	What it is	Security consequence
servers	shared machines running services for many users	the highest-value target; one compromise reaches everyone
personal computers handheld computers (also called mobile computers or information appliances) embedded computers	desktops and laptops smaller than a PC and running on battery power — smartphones, tablets, smart watches and other wearable technology a computer that is part of a machine — a car’s engine controller, a thermostat, a medical pump	general purpose, so they run anything the user installs easily lost or stolen, and often carried across untrusted networks has a specialised instruction set for interfacing with its components, and tends to be slower, cheaper and to have minimal storage , so security features are often left out and updates are rare

That last row is the reason embedded and IoT devices appear so often in attack scenarios: the constraints that make them cheap are the same constraints that make them hard to defend.

The main threat to a device is **malware** - malicious software. Learn the types:

- **Virus** - must be activated by a user opening a file.
- **Worm** - spreads by itself, with no human action.

- **Trojan** - hides inside software that looks safe; a **remote access trojan (RAT)** gives the adversary remote control.
- **Ransomware** - encrypts your files and demands payment for the key.
- **Spyware** - secretly tracks what you do.
- **Keylogger** - records every keystroke to steal passwords.
- **Logic bomb** - triggers only when a condition is met (a date, a version).
- **Rootkit** - deeply hides in the operating system and can even make itself invisible.

Most malware is a file, but **fileless malware** is different: it lives only in **RAM** and abuses legitimate programs already on the device, leaving no file for a scanner to find.

Adversaries exploit **unpatched software**, weak passwords, unprotected **BIOS/UEFI** startup settings, and open ports. We rate device risk by the value and criticality of the device - a hospital's unpatched email server is **high** risk, while an employee's laptop with one unused open port is **low**.

Authentication

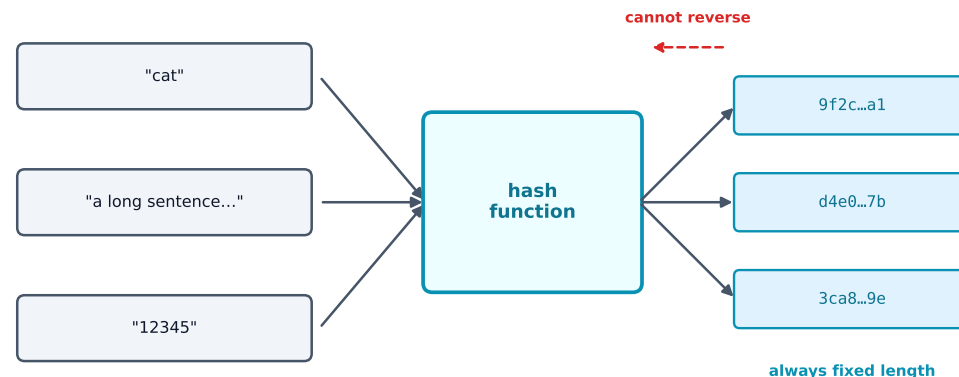


A fingerprint scanner: biometric authentication checks something you ARE, which is much harder for an attacker to steal or guess than a password

To store passwords safely, systems use a **cryptographic hash function**

- a one-way maths algorithm that turns any input into a fixed-length string called a **hash** (or digest). Hashes have three vital properties: they are **collision resistant** (hard to find two inputs with the same output), have **pre-image resistance** (you cannot work backwards to the input), and are **repeatable**

(the same input always gives the same hash).



AP Cybersecurity · U4 · a one-way hash function -> fixed-length digest

A hash function turns any input into a fixed-length digest, and cannot be reversed

Real hash functions have names. The **Secure Hash Algorithm (SHA)** family – SHA-256 and SHA-512 – is today's standard. Adversaries attack a hash function by trying to **force a collision** (two different inputs with the same hash); once an efficient collision attack exists, that function is **deprecated** (retired from secure use). **MD5** and **SHA-1** are the classic deprecated examples – never rely on them to protect data today.

A service never stores your plaintext password. It stores the **hash**; when you log in, it hashes what you typed and compares. To stop two identical passwords producing identical hashes, a few random bits called **salt** are added before hashing, so every stored hash is unique.

Worked example. Two users both choose the password **sunshine**. Without salt, both stored hashes would be identical, so cracking one instantly cracks the other. Give each user a unique salt - say **x7** and **q2** - and the service hashes **sunshinex7** and **sunshineq2** instead. The two stored hashes now look completely different, so the adversary must attack each account separately. This is why a stolen hash database is far less dangerous when the hashes are salted.

Adversaries fight back with **password attacks**. **Online** attacks guess against a live login; **offline** attacks steal the hash database and crack it on their own machine (which bypasses any account-lockout protection). Techniques include:

- **brute force** - an automated tool tries *every* possible password in turn; guaranteed to work eventually, but slow, and it grows explosively with password length.

- a **dictionary attack** - the tool tries a list of common words and known passwords first, because most people pick guessable ones.
- **password spraying** - one common password against many accounts (this dodges lockout, which counts failures per account).
- **credential stuffing** - reusing stolen or default credentials, exploiting that people reuse passwords across sites.
- a **rainbow table** - a precomputed table of passwords and their hashes, sorted by hash, so a captured hash can be looked up instead of recomputed.

Password policy settings

An administrator hardens accounts by **configuring login settings** - and the exam expects you to name them and say what each defends against:

Setting	What it does	The attack it slows
complexity	require a character from each set (upper, lower, digit, special)	brute force / dictionary
minimum length	require N characters - length matters more than anything	brute force (grows exponentially)
maximum age	force a change every ~90-120 days	limits how long a stolen password is useful
password history	store the last 5-10 hashes, block reuse	stops recycling an old (possibly leaked) password
lockout	lock the account after 3-5 wrong tries	brute force / online guessing

One subtlety worth a mark: some national standards now advise **against** forced expiry, because regular changes push users into predictable patterns like PasswordFall12028. A **password manager** solves the real problem - it generates and stores a long, unique password per site, so none is ever reused or guessable.

Authentication factors prove who you are, and fall into categories: something you **know** (a password), something you **have** (a token or phone), something you **are** (a **biometric** like a fingerprint or retina scan), and somewhere you **are** (a location factor). Using two or more is **multifactor authentication (MFA)** - far stronger than a password alone.



A hardware security key proves who you are with something you physically hold — a strong second factor

Removable media, and the autorun problem

An adversary can load malware onto an **external drive** — a USB stick, a portable disc — and leave it where someone will pick it up. If **autorun** is enabled, the device runs a program from that drive **the moment it is inserted**, with no click required, so the malware executes before the user has decided to trust anything.

Two controls answer this, and the exam wants both named:

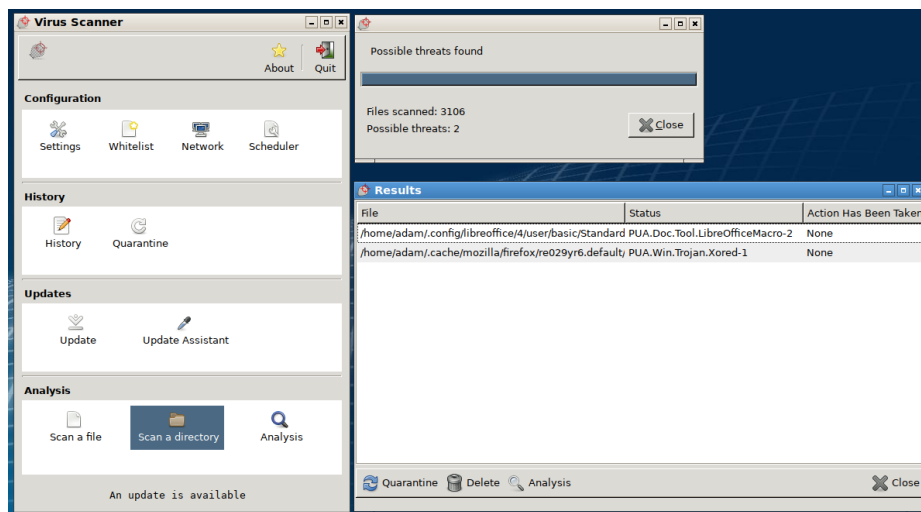
- **Disable autorun**, so inserting a drive never runs anything by itself.
- **Prohibit users from connecting external drives or media** at all — enforced by policy and by a technical control that blocks the USB ports — which is why so many secure environments physically or logically disable them.

Protecting Devices

Managerial controls set the rules: an **acceptable use policy** lists what users may and may not do, a **password policy** sets length and reuse rules, and a **software installation policy** controls what can be installed.

Technical controls do the work. **Anti-malware software** keeps a database of malware **signatures** and quarantines any file that matches. Keeping the operating system and applications **updated** - installing each **patch** - closes known holes before adversaries can use them. A **host-based firewall**

controls traffic in and out of one single device, blocking ports and services it does not need.



Anti-malware software scans files against a signature database and quarantines any matches — this scan has flagged two threats

Detecting Attacks on Devices

Devices log logins, file changes, and processes, and these logs reveal an **indicator of compromise (IoC)** - evidence that an adversary got in. **Host-based** IoCs show up as unexpected processes or changed settings; **file-based** IoCs are files whose hash matches known malware; **behaviour-based** IoCs are things like many failed logins or unusual login times.

Choosing a detection method means weighing **performance** (signature-based is lighter, better for weak devices), **cost** (an **endpoint detection and response (EDR)** service is powerful but expensive), and how sensitive the device is. Reading **authentication logs** exposes password attacks: many wrong passwords for one user signals a guessing attack; many users failing from one IP signals **password spraying**; a burst of default credentials signals **credential stuffing**. Offline attacks, though, cannot be detected - they happen on the adversary's own computer.

Speed is itself a security factor. **Signature-based** detection compares what it sees against a list of known-bad patterns, so it is **faster** than **anomaly-based** detection, which must first learn what normal looks like and then measure every event against that model. Anomaly-based detection catches attacks that have no signature yet, but it costs far more processing power — and on a device that lacks it, the effect

compounds: the detection runs slowly, the device degrades, and the method ends up not being implemented effectively at all.

Exam tips

- Know each **malware type by its defining trait**: a worm self-spreads, a virus needs a user, ransomware encrypts for money, a RAT gives remote control, a rootkit hides.
- A hash is **one-way and fixed-length**; **salt** makes identical passwords hash differently. Never say a service “stores the password” - it stores the **salted hash**.
- Name real algorithms: **SHA-256/SHA-512** are current; **MD5** and **SHA-1** are **deprecated** because efficient collision attacks exist.
- Match the password attack to its log signature: one user + many wrong passwords = guessing; many users + one IP = **spraying**; default credentials = **stuffing**.
- Sort authentication factors into **know / have / are / where**, and remember **MFA** combines two or more - a fingerprint plus a password, not two passwords.
- **Offline password attacks cannot be detected** because the cracking happens on the adversary's machine - a favourite exam “gotcha”.

Securing Applications and Data

AP Cybersecurity

Application and Data Vulnerabilities and Attacks

Applications are the programs that run on computers, and **data** is what they process - both are prime targets. If files are stored **unencrypted**, anyone with access to the drive can read them. If a normal user is given **administrative** privileges, an adversary who steals that account gains sweeping power.

The biggest application danger is bad **user input**. When a program does not check what a user types, an adversary can slip in commands - an **injection attack**. **Data validation** (checking input meets expected rules) is the defense. Key attacks:

- **SQL injection** SQL - inserting **SQL** commands into an input field to read or change a database.
- **Cross-site scripting (XSS)** - injecting malicious script into a website that runs in another user’s browser.

What a SQL injection actually looks like

SQL is a language for querying a database, and its **control words are always written in capital letters** — SELECT, FROM, WHERE, IN, OR, AND. A login form usually builds a query by pasting what you typed into one:

```
SELECT * FROM users WHERE name = 'alice' AND password = 'secret'
```

An attacker types SQL into the field instead of a name. Two tricks do most of the damage:

- **A condition that is always true.** Entering ' OR '1'='1 makes the WHERE clause true for every row, so the database returns every user.
- **A double dash, which begins a comment in SQL.** Entering admin' -- ends the name string and comments out the whole rest of the line, including the password check, so the query becomes ... WHERE name = 'admin' and the attacker is logged in as the administrator without a password.

The defence is **not** to filter for the word SELECT. It is to stop the input being treated as code at all: use **parameterised queries** (also called prepared statements), where the database is given the query and the values separately and never mixes them, and add **input validation** to reject characters the field has no reason to contain.

- **Buffer overflow** - sending more data than a memory **buffer** can hold, so it overflows into nearby memory and may run the adversary’s code.
- **Directory traversal** - using ../ sequences in a URL to reach files outside the intended folder, such as /etc/passwd.

We rate data risk by sensitivity: unencrypted military plans are **high** risk; customer data with a weak key is **moderate**; low-value data with short keys is **low**.

Protecting Applications and Data: Managerial Controls and Access Controls

Data is classified by its **state** - **at rest** (stored on a drive), **in transit** (moving between devices), and **in use** (being processed). Data at rest and in transit can be **encrypted** so a thief cannot read it; data in use must be decrypted, so **access controls** guard it instead.

Some data types are **regulated** - the law dictates how they must be stored, transmitted and handled - so an organisation must achieve **compliance** by matching its controls to the rules. The exam expects you to pair each data type with its governing law:

Regulated data	What it is	Governing law
personally identifiable information (PII)	anything identifying a person: name, address, SSN, biometrics, date of birth	The Privacy Act (1974); COPPA for under-13s
protected health information (PHI)	health, treatment and healthcare-payment records	HIPAA (1996)
payment card information (PCI)	card number, expiry, CVV, cardholder name	PCI-DSS

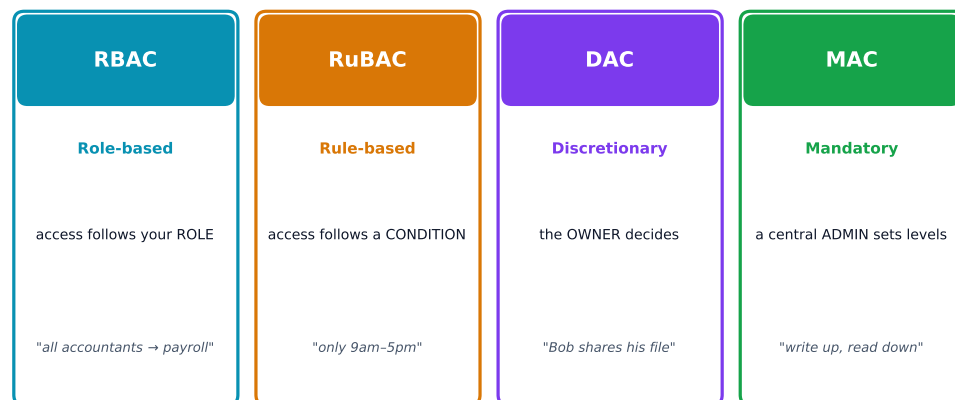
An organisation that collects regulated data must **label** it and hold **policies** that keep its storage, transmission and handling compliant - the higher the sensitivity, the higher the required degree of security.

Access control decides which **subjects** (users) may perform which **operations** on which **objects** (files). Four models:

- **Role-based (RBAC)** - access follows your **role** (all “accountants” reach the payroll software).
- **Rule-based (RuBAC)** - access follows **conditions** (only

during business hours), layered on another model.

- **Discretionary (DAC)** - the **owner** of a file decides who else may use it.
- **Mandatory (MAC)** - a central administrator sets strict levels; the **Bell-LaPadula** model summarises it as “write up, read down”.



AP Cybersecurity - U5 - four access-control models by their 'decider'

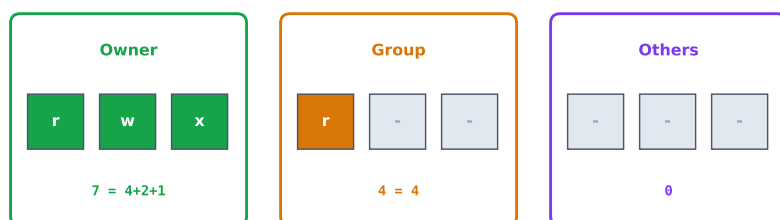
Four access-control models decide who reaches which object, and how

A guiding idea across all models is the **principle of least privilege** - give each entity exactly the access it needs and no more.

On a **Linux** system, each file has three permissions - **read (r)**, **write (w)**, **execute (x)** - for three groups: the **owner**, the **group**, and **others**. The **chmod** command sets them with numbers, adding 4 (read) + 2 (write) + 1 (execute). So **chmod 640** means owner read+write (6), group read (4), others nothing (0).

Example: **chmod 740 report.txt**

read = 4 write = 2 execute = 1



AP Cybersecurity - U5 - Linux file permissions (rwx for owner/group/other)

Linux file permissions: read/write/execute for owner, group, and others

Worked example. A principal wants only herself to read *and* edit a file, her staff group to read it, and no one else to touch it. Read+write = 4+2 = **6** for the owner, read = **4** for the group, nothing = **0** for others, giving **chmod 640 file**. The listing then shows **-rw-r-----**. To also let the owner run the file as a program you would add execute (7 = 4+2+1), giving **chmod 740**.

Protecting Stored Data with Cryptography



An Enigma machine: cryptography protects stored and transmitted data from eavesdroppers

Cryptography hides information. An **encryption** algorithm combines the **plaintext** with a **key** to produce **ciphertext**; **decryption** reverses it. The **keyspace** is the number of possible keys - the bigger it is, the longer an adversary needs to guess. An **n-bit** key has a keyspace of 2^n .

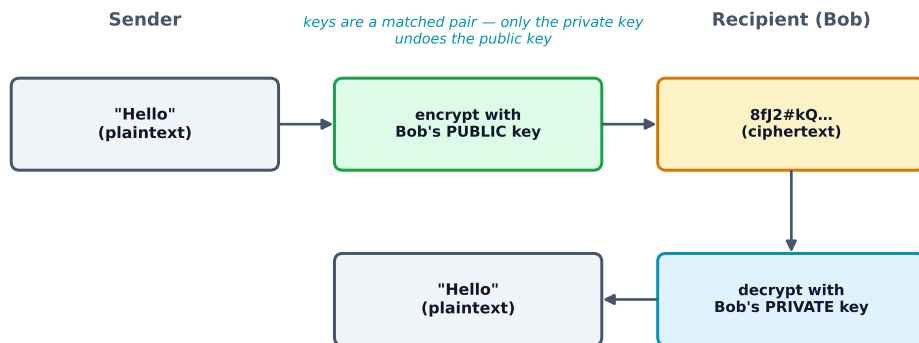
Symmetric encryption uses the **same key** to encrypt and decrypt. The standard is **AES**, a **block cipher** that works on 128-bit blocks and secures Wi-Fi, browsing, and stored files. Because both sides need the same secret key, sharing that key safely is the challenge.



The Enigma machine scrambled messages with rotors — an early, breakable example of encryption

Asymmetric Cryptography

Asymmetric encryption solves the key-sharing problem with a **key pair** - a **public key** anyone may see and a **private key** kept secret. The keys are mathematical inverses: whatever one locks, only the other unlocks. To send you a secret, I encrypt with **your public key**, and only **your private key** can decrypt it - so we never had to share a secret in advance.



AP Cybersecurity · U5 · asymmetric encryption: public key locks, private key unlocks

Asymmetric encryption: encrypt with the public key, decrypt with the private key

Longer keys mean larger keyspaces and more security, but slower encryption. Common

asymmetric algorithms are **RSA** and **elliptic curve cryptography (ECC)**, used in digital signatures and certificates. Remember: you can only compare key lengths **within the same algorithm** - an RSA 4096-bit key is not directly comparable to an AES 256-bit key.



A padlock: cryptography locks data so only someone with the matching key can open it

Protecting Applications

Two design principles keep applications safe from the start. **Secure by design** builds security into every phase of development, not as an afterthought. **Secure by default** means the product ships with its security features already **enabled** - safe straight out of the box.

Secure by design rests on three principles a company must adopt: (1) **take ownership of its customers' security outcomes** rather than shifting blame onto users, (2) embrace **radical transparency and accountability** - sharing security-relevant news and updates quickly so everyone becomes safer, and (3) build the **organisational structure and leadership** that makes security a first-class goal.

The key defense against injection attacks is **input sanitization**. Certain **special characters** - the single quote, double quote, and semicolon - can be used to manipulate a system, so a good program removes or rejects them before processing. Sanitization protects against SQL injection, XSS, and directory-traversal attacks alike.

Detecting Attacks on Data and Applications

To detect data attacks, systems perform **accounting** - logging who accessed what and when. But logs are huge, so **log analysis** must be **automated** to run at a useful speed; a human reading raw logs is far too slow. A clever complement is a **honeypot** - a fake file that looks valuable; since no one has a real reason to open it, any access is a clear, **near-instantaneous** sign of an attack. Watch especially for attempts to **delete or copy sensitive files**. **Cryptographic hashes** also help: re-hash a file and compare - if the digest changed, the file was altered.

Choosing detective controls means weighing **cost** (honeypots are cheap; a **data loss prevention (DLP)** service is powerful but pricey) against the sensitivity of the data. To read a specific attack from logs, look for its signature: SQL injection shows **OR 1=1** and **--**; XSS shows **<script>** tags; directory traversal shows **../** sequences; a buffer overflow shows unusually long input strings.

Checking that a file has not been altered

A **cryptographic hash** turns a file of any size into a short fixed-length value. Change one byte of the file and the hash changes completely, so comparing a downloaded file's hash with the one the publisher lists proves the file arrived intact. You do this at the command line:

Shell	Command
BASH (Linux, and most servers)	sha256sum testfile
zsh , the usual terminal on Apple computers	shasum -a 256 testfile

Both print the SHA-256 hash of **testfile**. If it differs from the published value by even one character, the file has been altered — by corruption in transit, or by an attacker who replaced it.

△ A hash proves **integrity**, not **authenticity**. An attacker who can replace the file on a web page can usually replace the published hash beside it too; that is why a signed hash, or one fetched over a separate trusted channel, is stronger evidence.

Exam tips

- Match each application attack to its **evidence in a log**: **OR 1=1** / **--** = **SQL injection**; **<script>** = **XSS**; **../** = **directory traversal**; very long input = **buffer overflow**.
- Learn the four access-control models by their **decider**: RBAC = your role, RuBAC = a condition, DAC = the file's owner, MAC = a central admin. **Least privilege** underlies them all.

- Read Linux permissions by adding **4+2+1** per group - **chmod 750** = owner rwx (7), group r-x (5), others none (0). Practice converting both ways.
- **Symmetric** = one shared key (fast, AES); **asymmetric** = a public/private key pair (solves key sharing, RSA/ECC). Encrypt with the **recipient's public key**.
- **Input sanitization** is the single best answer for preventing injection attacks; a **honeypot** is the classic cheap detective control.