Course Companion

AP Cybersecurity

Every topic handout in one volume

全部专题讲义合集

exercises.lol

Contents 目录

---

1. Introduction to Security 2

2. Securing Spaces 10

3. Securing Networks 20

4. Securing Devices 29

5. Securing Applications and Data 41

Introduction to Security

AP Cybersecurity

Understanding Social Engineering

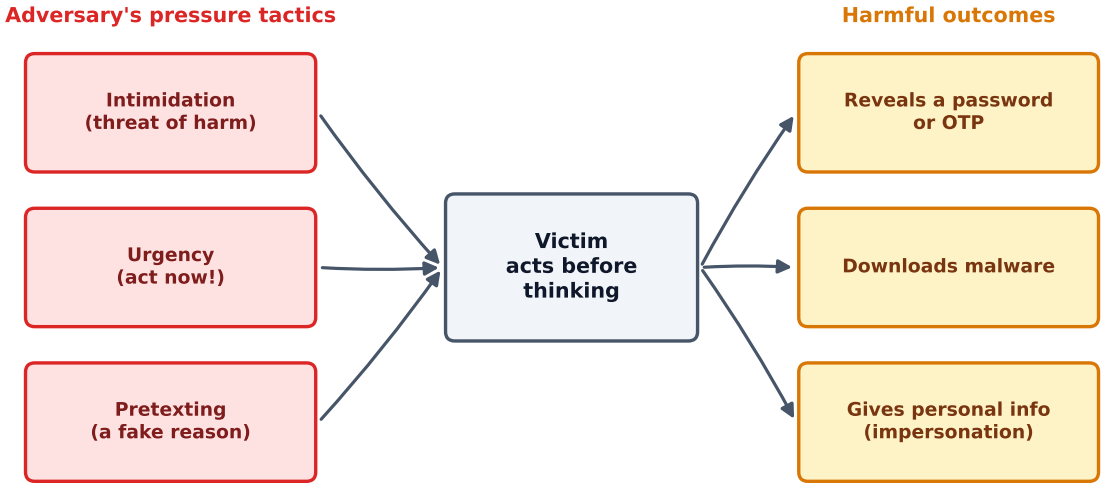
In July 2020 attackers telephoned staff at Twitter and pretended to be from the company's own IT team. Some staff believed them, and the attackers used what they learned to take over famous accounts and post a Bitcoin scam. No clever code broke the system first: a phone call did. This unit starts with attacks you can see in daily life, and then asks how AI changes both sides.

The weakest part of any computer system is often the **human** using it. **Social engineering** 社会工程学 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** 对手) 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. One goal is **elicitation** 套取信息 - getting sensitive information out of someone without them realising.

Adversaries lean on two powerful feelings:

- **Intimidation** 恐吓 - the adversary threatens a bad result if you do not obey. Fear pushes you to act.
- **Urgency** 紧迫感 - 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.



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

The **impact** 影响 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** 安全问题 and **impersonate** 冒充 them. They might hand over a **one-time password (OTP)** 一次性密码, letting the adversary log in as them. Or they might download **malware** 恶意软件 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** 密码攻击 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** 弱 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** 字典 of likely passwords from information gathered about you and let an automated tool try each one.

To make **authentication** 身份验证 stronger:

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

Another good choice is a long **passphrase** 密码短语: several random words joined together. It is long, and easier to remember than random symbols.

> **Worked example: read the log.** A website keeps this login record for one account. Which signs of an online password attack do you see?

Time	Device	Password result
Mon 08:02	usual laptop	success
Tue 03:14:01	unknown	fail
Tue 03:14:03	unknown	fail
Tue 03:14:05	unknown	fail
Tue 03:14:07	unknown	success

All three warning signs appear: repeated failures two seconds apart, an unusual hour, and an unknown device. The timing alone does not prove automation or an attack. The successful login could mean the account is **compromised**: verify it with the user through a trusted channel, investigate subsequent activity, and contain suspected compromise by revoking sessions, resetting the password and enabling MFA.

> **Worked example: why patterns are weak.** How many guesses does each password need, at most?

1. `Summer24!` follows the word + year + symbol pattern. Suppose a dictionary has 10,000 words, 100 two-digit years and 10 common symbols: $10000 \times 100 \times 10 = 10^7$ guesses. An automated tool can finish that.
2. Eight random lowercase letters: $26^8 \approx 2 \times 10^{11}$ guesses, about 20,000 times more.
3. Four words picked at random from a list of 7,776: $7776^4 \approx 3.7 \times 10^{15}$ guesses, and still easy to type.

Length and randomness give strength. Adding personal facts makes the attacker's dictionary smaller, not larger. If the same password is used on two sites, a leak from one site opens the other: this is why each one must be **unique**.

Best Practices for Public Networks

Not all adversaries are the same. We classify them by **skill**: low-skilled attackers buy ready-made tools online and reuse known **exploits** 漏洞利用, while high-skilled attackers write their own tools and can discover brand-new holes called **zero days** 零

日漏洞. Their **motivation** 动机 varies too - greed, revenge, politics, or belief.



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

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

- **Evil twin** 双胞胎恶意热点 - the adversary sets up a fake **access point** 接入点 with a name (**SSID** 服务集标识符) copied from the real network. Victims connect to the fake one, and the adversary captures their traffic. Valid **encrypted** 加密的 HTTPS protects the content in transit; this does not prove the hotspot belongs to the school or make every destination trustworthy.
- **Jamming** 干扰攻击 - the adversary floods the air with a strong radio signal so no one can connect. This is one kind of **denial of service (DoS)** 拒绝服务 attack.
- **War driving** 驾车无线网络探测 - the adversary drives around detecting wireless networks and where their signal leaks outside a building.

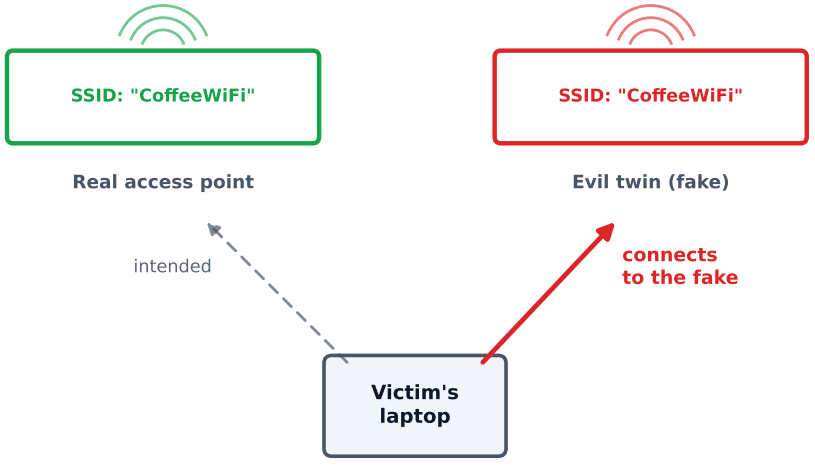
Try it – classify three separate incidents. A school confirms that (A) an unauthorised access point copies its SSID and captures students' traffic, (B) an adversary's transmitter emits a strong signal on the network frequency and prevents legitimate communication, or (C) an adversary drives around recording network beacons and where the signal reaches outside, without connecting. Identify the attack in each case and cite its mechanism.

Worked reasoning. A is an evil twin: it imitates the network's identity to attract connections. B is jamming: the interfering radio signal prevents access, producing denial of service. An outage alone would not prove jamming; the transmitter is part of the given evidence. C is war driving: it gathers wireless-network information without needing to join. Similar observations can be part of an authorised survey, so the purpose and permission matter.

Transfer. A student sees the correct SSID and says, "The name matches, so it must be

the school's network. With HTTPS, nothing can go wrong." Explain both overclaims and suggest a separate identity check.

Worked answer. An evil twin can copy the name, so an exact name match is a check rather than proof of ownership. Follow trusted school connection instructions and confirm uncertainties with staff through a separate trusted channel. Valid encrypted HTTPS protects content in transit from someone merely capturing traffic; it does not authenticate the hotspot or make every destination trustworthy. A fake network can still interfere with access or direct users toward a phishing site.



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, while remembering that this alone does not prove its identity; prefer valid encrypted HTTPS, and consider a **virtual private network (VPN)** 虚拟专用网络, which encrypts traffic carried through its tunnel to the VPN operator.

A VPN moves your trust; it does not remove it. For traffic carried through the tunnel, the local network and Internet service provider see the VPN connection rather than the final destinations. The VPN provider can see outgoing destinations and unencrypted content; valid HTTPS still protects message content between your browser and the website. Think about how sensitive your data are before joining an unencrypted Wi-Fi network: even when websites use HTTPS, information such as **DNS** queries may still be exposed.

AI-Based Cybersecurity Attacks

Artificial intelligence gives adversaries powerful new tools. With enough voice and image samples, an adversary can build a **deepfake** 深度伪造 avatar to impersonate someone on a call. **Large language models (LLMs)** 大语言模型 let them write convincing **phishing** 钓鱼 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.

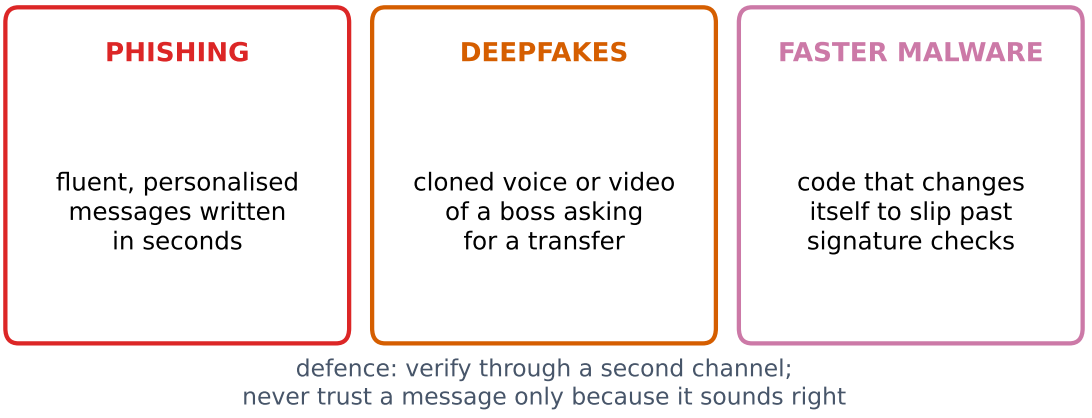
You can defend against many AI-augmented attacks: agree on a **shared secret** 共享秘密 word with close contacts to verify identity, enable MFA (so a cloned voice alone cannot log in), never type sensitive data into a chatbot, and always double-check AI output against reliable, non-AI sources.

Worked example: match the defence to the attack.

AI-augmented attack	Why it works	Defence
A cloned voice of your brother asks for money	the voice sounds real	ask for your **shared secret** word
A cloned voice is used to pass voice login	the voice is the only factor	MFA adds a second factor
Staff paste private files into a chatbot	some tools train on user input, so prompts may later extract it	never enter sensitive data
False web pages get into training data	the model repeats what it learned	check output against reliable non-AI sources

A good answer gives the mechanism as well as the name of the defence.

AI writes code, and that cuts both ways. Adversaries use **AI-enhanced coding tools** 人工智能辅助编程工具 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** 漏洞 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.



How attackers use AI: phishing, deepfakes and self-changing malware

Leveraging AI in Cyber Defense

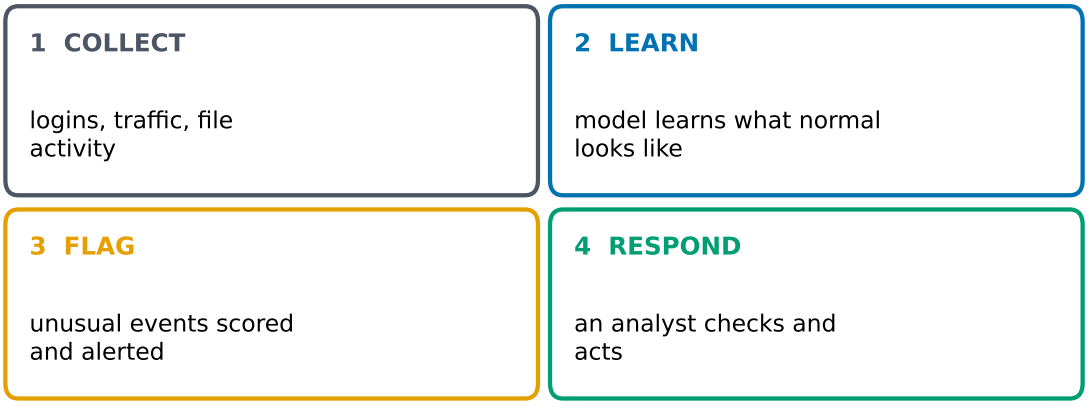
The same technology defends us. AI tools can analyse an application's own source code, identify **vulnerabilities** in it and **recommend mitigations**; they can also review firewall rules and access settings and **recommend** safer options - though a human expert must always check the advice before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

> **Worked example: how large is the problem?** A network records 5 million events a day. If an analyst could read one event every 10 seconds for 8 hours, that is $\frac{8 \times 3600}{10} = 2880$ events a day. It would take about $\frac{5000000}{2880} \approx 1700$ analysts. No team can do this, so AI sorts the events first and people check what is left.

That scale is what makes **threat detection and response** 威胁检测与响应 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.



AI sorts huge volumes quickly but raises false alarms, so people stay in the loop

AI in defence learns normal behaviour and flags anomalies for a human to check

Exam tips

- When a question asks you to rank risks, weigh likelihood and impact together. An open internal port may expose sensitive systems; Wi-Fi reaching a car park is not automatically a lower risk. Use the stated protections and consequences.
- 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.
- **Valid encrypted HTTPS protects content on an evil twin** from passive capture; it does not authenticate the hotspot. Say what is protected and what risk remains.
- 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

In a 2016 study, researchers left nearly 300 USB drives around a university campus. Almost all were picked up, and people opened files on at least 45% of them. No one broke a lock, and no firewall was involved: curiosity did the work. This unit is about the people and places around computers, and how to measure and reduce risk there.

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** 信息安全三要素 - the three goals of security:

- **Confidentiality** 保密性 - only authorised people can read the data.
- **Integrity** 完整性 - the data is accurate and trustworthy, with changes made only as authorised. Legitimate updates do not violate integrity.
- **Availability** 可用性 - authorised users can access the data and services when needed.

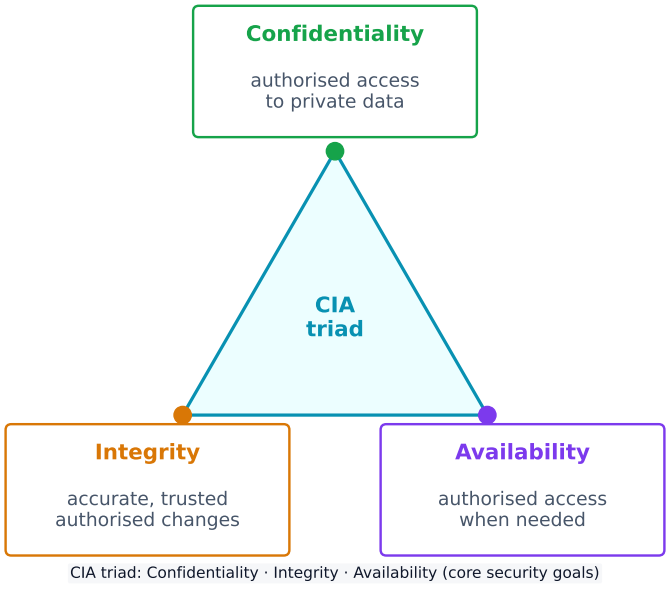
Try it – identify the primary goal. A school has a private records portal. In separate incidents: (A) a private record is copied to a public page, while the original is unchanged and the portal works; (B) an unauthorised person changes a mark from 19 to 91, while the record stays private and accessible; (C) the portal is offline when authorised staff need it, with no evidence of copied or changed records. Name the primary CIA goal harmed and cite the stated evidence.

Worked reasoning. A harms confidentiality because unauthorised people can read private data. B harms integrity because an unauthorised change makes the record untrustworthy. C harms availability because authorised staff cannot access the ser-

vice when needed. Access permissions and encryption can support confidentiality; restricted editing and trusted change records support integrity; tested recovery arrangements support availability. Explain the mechanism, not just a control's name.

Transfer – more than one goal. An adversary copies private records, changes some marks, and makes the portal unusable. A manager proposes an offline backup. Identify evidence for all three CIA goals. What can a clean, tested backup help repair? Can restoration make the copied records private again?

Worked answer. Copying the records harms confidentiality, changing marks harms integrity, and preventing access harms availability. A clean backup can help restore correct records and recover the service, supporting integrity and availability. A stored copy alone does not guarantee timely access: working recovery procedures and testing matter. Restoration does not remove the adversary's copies, so the confidentiality loss still needs a response. One incident can harm several goals and one control can support several goals.



The CIA triad: use the goals to explain what a control protects

Attacks come from different adversaries, classified by their access, skill or goals. A **script kiddie** 脚本小子 reuses tools built by others for greed or recognition; a **hacktivist** 黑客活动分子 acts for a political, social, or personal cause; an **insider** 内部人员 already holds legitimate access and may act from revenge or greed; a **cyberterrorist** 网络恐怖分子 disrupts critical infrastructure like a power grid or water plant; and **transnational criminal organisations** 跨国犯罪组织 chase money through ransomware and stolen data.

Most attacks unfold in **phases** 阶段: **reconnaissance** 侦察 (gathering information, often from public **OSINT** 公开来源情报 sources), initial access, persistence, **lateral movement** 横向移动 (extending access to other systems, sometimes using stolen

credentials or additional privileges), taking action on the goal, and evading detection. Naming the phase an attacker has reached helps a defender choose the right response.



The six phases; a real attack may skip some of them

Phase	What the adversary does	Example at a school
Reconnaissance	collects facts, often from public sources	reads staff names on the school website
Initial access	gets a foothold, often by social engineering or weak credentials	phishes a teacher's password
Persistence	keeps access without having to regain it, often with a **command and control (C2)** 命令与控制 link	malware on the laptop waits for orders
Lateral movement	extends access to other systems; higher **privileges** 权限 are not required	uses a stolen login to access another school computer
Taking action	collects and removes data (**exfiltration** 数据外泄), disrupts services or destroys data	copies student records out
Evading detection	removes or edits logs and deletes planted files	deletes the login records

■ Social engineering: the seven tactics

Most attacks begin not with code but with **social engineering** 社会工程学 - 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 借口	inventing a believable reason to make contact ("I'm from IT, verifying your account")
Authority 权威	posing as someone powerful, or relaying "the boss's" instructions
Intimidation 恐吓	threatening negative consequences if a demand is not met
Consensus 从众	claiming everyone else is already doing it, to create social pressure
Scarcity 稀缺	inventing limited availability ("only 2 left")
Familiarity 熟悉	pretending to be, or to know, someone close to the target
Urgency 紧迫感	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** 风险 appears when a **threat** 威胁 can exploit a **vulnerability** 漏洞 to compromise an **asset** 资产 (anything valuable - data, money, hardware, reputation). We **assess** risk by weighing two things: the **likelihood** 可能性 of an attack and the **severity** 严重性 of the damage.

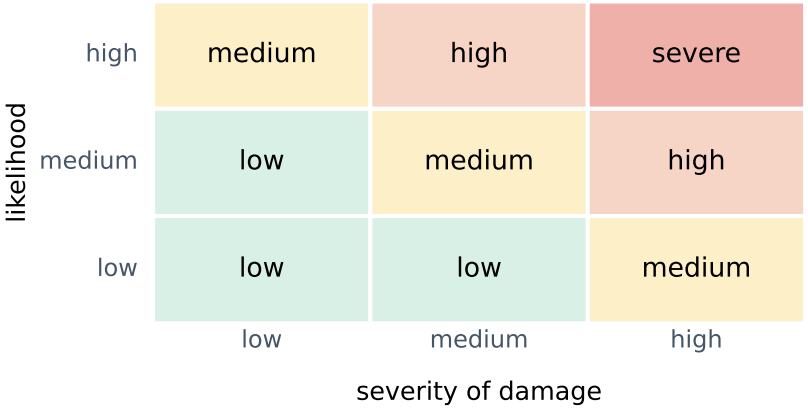
Likelihood itself depends on the **value** of the target (adversaries chase what looks worth stealing), the **skill** needed to exploit the vulnerability (a well-documented exploit needs little skill, so more adversaries can use it), and the **motivation and capability** of likely adversaries. Severity 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** 定量 - a number: a score on a scale (e.g. 1-10), or a money value (e.g. "a $10,000 annual risk").
- **qualitative** 定性 - a label: *low / medium / high / severe*, or a grid such as *likely-high-impact* vs *unlikely-low-impact*.

A written **risk assessment** 风险评估 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.

exercises.lol exercises.com.cn



One common way to turn likelihood and severity into a qualitative rating

Worked example: a number for risk. A school expects 2 laptops to be stolen each year, and each loss costs \$1,000. The risk is $2 \times \$1000 = \$2000$ a year. Cable locks for the room cost \$300 a year, and should stop nearly all of the thefts. They are **cost-effective** 具有成本效益的: they cost less than the **expected loss** 预期损失. A guard costing \$30,000 a year also stops the thefts, but costs far more than the loss, so it is not cost-effective for this risk alone.

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

- **Avoid** 规避 - stop the risky activity (only possible if it isn't essential).
- **Transfer** 转移 - shift the burden to someone else, such as an insurer.
- **Mitigate** 缓解 - add controls to lower the likelihood or impact.
- **Accept** 接受 - live with the leftover **residual risk** 剩余风险, because perfect security is impossible.

Worked example: which response?

Decision	Response
A shop stops storing card numbers on its own computers	avoid
A hospital buys cyber insurance	transfer
A school adds MFA to staff accounts	mitigate
A club decides a small risk costs too little to act on	accept

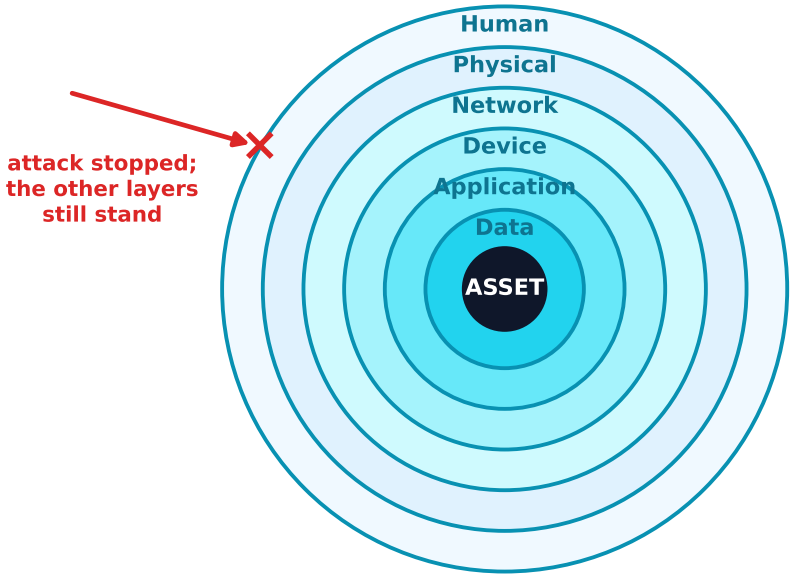
A hospital cannot avoid keeping patient records, because records are part of its work: avoidance is possible only when the activity is not critical.

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

Control	Type	Function
Camera	physical	detective
Intrusion detection system (IDS) or SIEM	technical	detective
Password policy, access reviews	managerial	preventative
Encryption	technical	preventative
Vulnerability patch, intrusion prevention system (IPS)	technical	corrective
Repairing a broken card reader	physical	corrective

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** 纵深防御 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** 物理攻击 often begin with social engineering:

- **Piggybacking** 尾随（获许可）- tricking an authorised person into holding a door open (for example, by carrying a heavy box).
- **Tailgating** 尾随（未察觉）- slipping through a secured door behind someone without their knowledge.
- **Shoulder surfing** 肩窥 - watching someone type a password or read sensitive information.
- **Dumpster diving** 翻垃圾搜集情报 - searching a target's trash for useful information.
- **Card cloning** 门禁卡复制 - 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** 键盘记录器. 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** 立足点 to reach other resources, and **low** when the asset is worthless and unlikely to be attacked.

Threats are not only human: fires, floods and storms can also destroy computers and stop services. Physical access is dangerous because it can bypass many technical controls: someone who can reach a device does not need to get past the firewall.

Worked example: rate the physical risk.

Situation	Rating	Reason
A customer-data server sits in an unlocked room off an unwatched corridor	high	sensitive system, no controlled access
A receptionist's computer has open USB ports and joins the internal network	moderate	the desk is not sensitive, but it is a foothold
Laptops with no sensitive data stay on desks in a badge-only office over lunch	low	low value, 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** 清桌政策), and using privacy screens.

Physical controls then harden the building: fences, gates, and **bollards** 防撞柱 deter access; locks protect doors and cabinets; **card readers** 读卡器 log and restrict entry; an **access control vestibule** 门禁前室 (a two-door airlock) stops piggybacking; disabling **USB ports** blocks malware drives; and an **uninterruptible power supply (UPS)** 不间断电源 keeps devices running through an outage. Organisations prioritise these by matching the cost of a control to the severity of the risk.

A **turnstile** 闸机 does the same job as a vestibule at a busy entrance. A **surge protector** 浪涌保护器 protects devices from a sudden rise in power, and generators can power a whole building.

> **Worked example: choose the control.** To find a control, ask how the adversary would use the weakness, and whether you can prevent, detect or correct it.

Weakness	Control	What it does
Strangers follow staff through a door	access control vestibule	one person at a time passes through
Malware could arrive on a USB drive	disable USB ports	prevents loading
Papers are left out overnight	clean desk policy	removes what could be stolen



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** 运动传感器 alert staff to movement; and employees themselves often notice an intruder first.

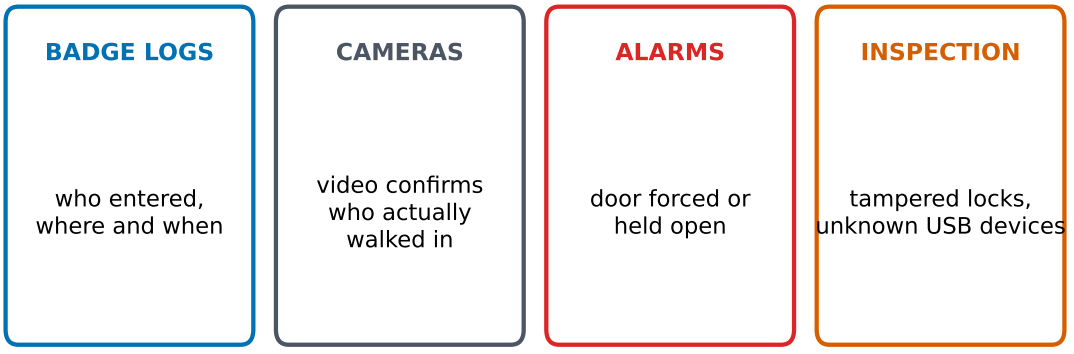
Placement matters. Cameras belong at **points of ingress and egress** 出入口 (entrances and exits). Motion sensors work best in low-traffic areas like server rooms - put them in a busy hallway and constant false alarms make everyone ignore them. Stationary guards protect a fixed high-value point, while 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.

Controls also work better together. A motion sensor raises an alarm, and a camera lets a guard check at once whether it is real. Think about a camera's coverage and angle, and whether someone could reach it and tamper with it.

Worked example: read the door log. A server-room door records each badge and how long the door stayed open.

Time	Badge	Door open (s)
09:02	Liu	5
11:30	Park	4
13:47	Liu	31
16:15	Park	5

The 13:47 entry stands out: about six times longer than the others, with one badge. Two people may have passed on one badge, so this could be piggybacking or tailgating. It is **evidence, not proof**: Liu may have been carrying a box. The next step is to check the camera recording for that time.



tailgating shows up as one badge swipe but two people on camera

Detecting physical attacks by comparing badge logs, cameras, alarms and inspections

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

In October 2016 the Mirai botnet flooded a major DNS provider with traffic, and many popular websites became hard to reach for hours. Mirai's army was not made of powerful servers. It was made of home cameras and routers that still had their factory passwords. Small, forgotten devices on a network can do great harm, so this unit is about how networks are attacked, divided, filtered and watched.

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** 地址解析投毒 - the **address resolution protocol (ARP)** 地址解析协议 pairs IP addresses with hardware **MAC addresses** 物理地址. An adversary sends fake ARP messages so traffic meant for the target flows to the adversary instead. This is an **on-path attack** 中间人攻击 (also called man-in-the-middle): the adversary secretly sits between two parties, reading and even altering their messages.
- **MAC flooding** 物理地址泛洪 - many fake source addresses may exhaust a **switch** 交换机 forwarding table. Some switches then flood unknown-destination frames to other ports, which may enable **eavesdropping** 窃听. Known-destination traffic and configured protections can behave differently; capturing all traffic is not guaranteed.
- **DNS poisoning** 域名投毒 - planting a fake record on a **domain name system (DNS)** 域名系统 server redirects users to a malicious site to steal credentials (**credential harvesting** 凭据收集).
- **Smurf attack** - sending **ICMP** echo requests to a network's broadcast address with the victim's address forged (spoofed) as the source, so every device on that network replies to the victim. It is a **denial of service (DoS)** 拒绝服务 attack; when many machines attack at once it becomes a **distributed denial of service (DDoS)** 分布式拒绝服务.

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** 非法接入点 that may bypass the intended perimeter firewall. 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** 自动漏洞扫描器: 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** 缓解措施. Fixing the highest-severity items first is a core part of managing network risk.

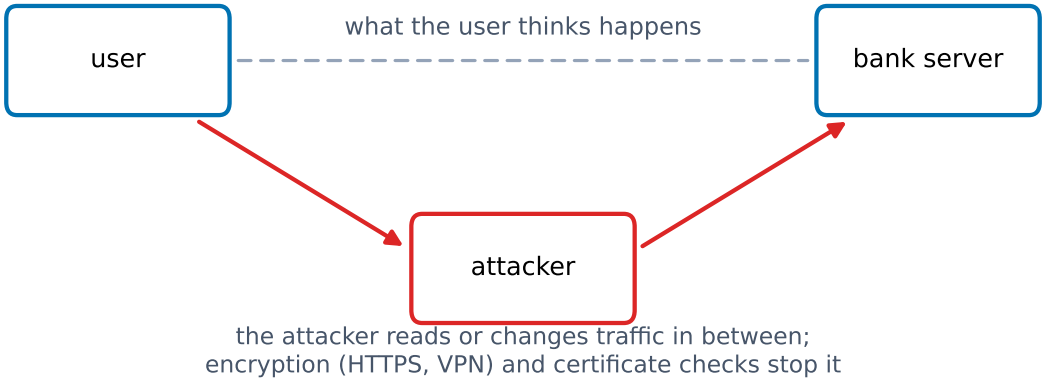
Worked example: rate the network risk. A network vulnerability is a high risk when it lets an adversary easily have a big effect, a moderate risk when it mainly gives information, and a low risk when it is hard to use and does little harm. Many network exploits need advanced skill, which lowers their likelihood.

Finding	Rating	Reason
An open port in a public hallway, with no port security	high	anyone can plug in, spoof a device or launch a DoS
A network scan can list which devices exist	moderate	it gives information about the network
Wi-Fi reaches the car park but uses strong encryption and authentication	low	hard to exploit, with little effect

Worked example: read the log. Two lines from a network log:

```
10:01:14 ARP Reply: 10.0.0.1 is-at 3c:22:fb:10:aa:01
10:42:09 ARP Reply: 10.0.0.1 is-at 7a:9e:04:cc:51:0f
```

One IP address, the gateway's, is now advertised with a **different MAC address**. This can indicate **ARP poisoning** and an on-path attack, but legitimate gateway replacement or failover can also change the mapping. Check the approved gateway configuration and packet evidence before concluding that traffic reaches an adversary. By contrast, a sudden flood of many **new** MAC addresses on one switch port points to MAC flooding, and **port security** stops that by limiting the addresses allowed on each port.



A man-in-the-middle attack sits between a user and a server

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** 分离隧道; and a **wireless security policy** requires strong encryption and authenticated access.

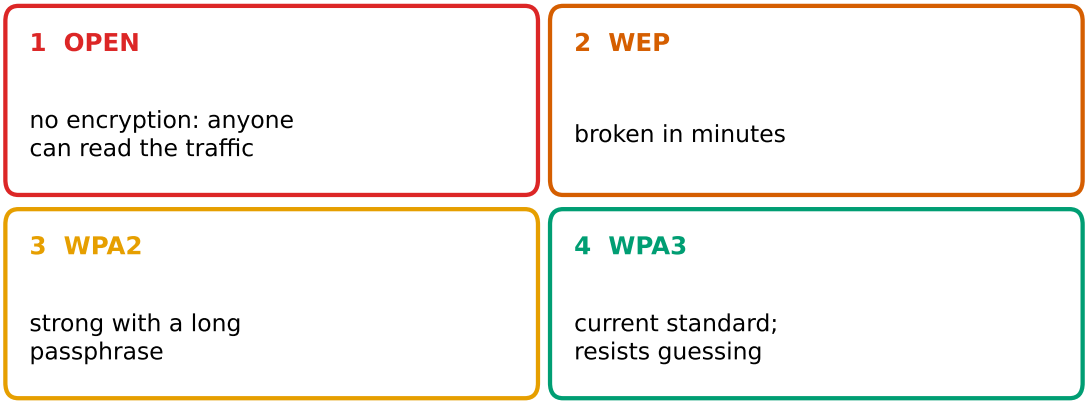
What each policy typically requires, and why:

- **Router policy.** No local user accounts: every login goes through an approved **authentication server**, so accounts can be checked and removed in one place. **Disable unnecessary services**, such as **Telnet**, a remote-login service that sends everything, passwords included, as plain text; any service left running is one more way in, so use an encrypted alternative (SSH) or turn it off. Require a **firewall**, which may be a separate device from the router.
- **Switch policy.** No local accounts; **port security** enabled, so a port accepts only the devices it expects; **MAC filtering**.
- **VPN policy.** Which job roles may use the VPN; how users must authenticate (a public/private key system or MFA); and no **split tunneling**, where a device on the VPN also has a direct connection to the internet, because traffic from that open side could reach the internal network through the device.
- **Wireless policy.** Users authenticate through **EAP** (extensible authentication protocol) to an approved authentication server; all wireless traffic is encrypted with **AES** at a minimum key length; the AP CED lists disabling beacon-frame broadcasting as a discovery-limiting measure. Hiding an SSID does not disable the beacon frames or provide access security.

> **Worked example.** A school's router still accepts Telnet logins, and staff connect to the school VPN from home with split tunneling allowed. Identify two weaknesses and a control for each. (1) Telnet sends the administrator's password in plain text, so anyone capturing traffic on the path can read it and log in to the router. Control: the router policy disables Telnet and allows only SSH, through the authentication server. (2) With split tunneling, a home laptop is joined to the school network and to the open internet at the same time, so malware or an attacker reaching the laptop from the internet has a route into the school network. Control: the VPN policy bans split tunneling, so all the laptop's traffic goes through the VPN and the school's firewall.

For wireless networks, use authenticated access and suitable encryption, such as **WPA3** Wi-Fi 保护接入第三代 where supported; avoid obsolete WEP and original WPA. Hiding the network name does not stop beacon transmissions or reliably conceal the network. Signal-strength limits and MAC filtering are supplementary measures: radio reception can extend beyond a building, and an observed device

address can be spoofed.

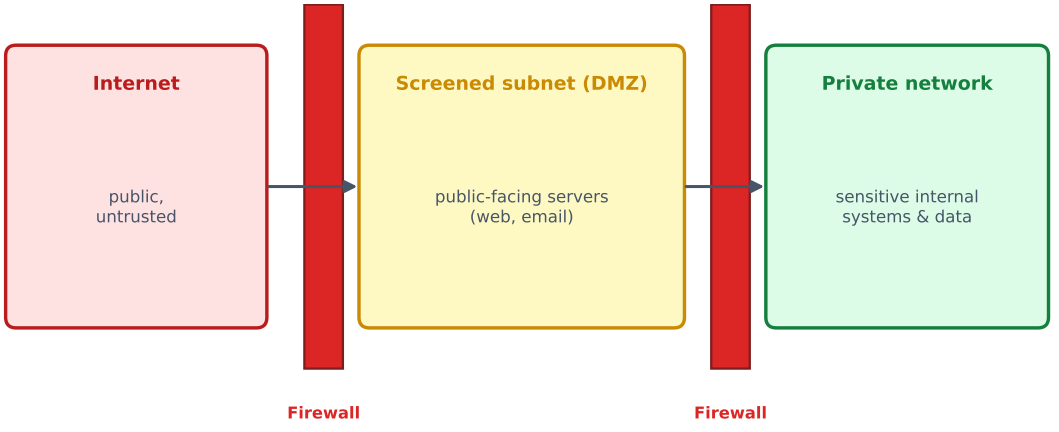


Wireless security protocols from weakest to strongest

Protecting Networks: Segmentation

Network segmentation 网络分段 divides a network into controlled sections, often using **subnets** 子网. Enforced filtering between sections can limit access and incident spread; an address boundary alone does not guarantee isolation.

A key pattern is the **screened subnet** 屏蔽子网 (also called a **DMZ** 隔离区). 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** 虚拟局域网 (logically separating devices on the same switch). Each segment can then get its own security policy - higher-security and lower-security zones.



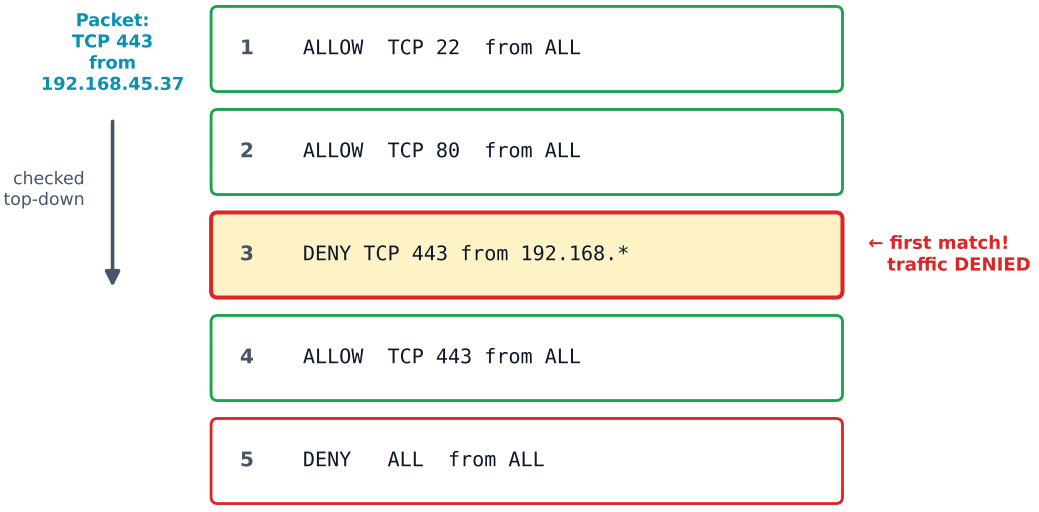
Server racks: network segmentation isolates systems so one breach does not open everything

Protecting Networks: Firewalls

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

- **Stateless** 无状态 - filters on packet headers alone (IP, port, protocol).
- **Stateful** 有状态 - 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)** 访问控制列表 - 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

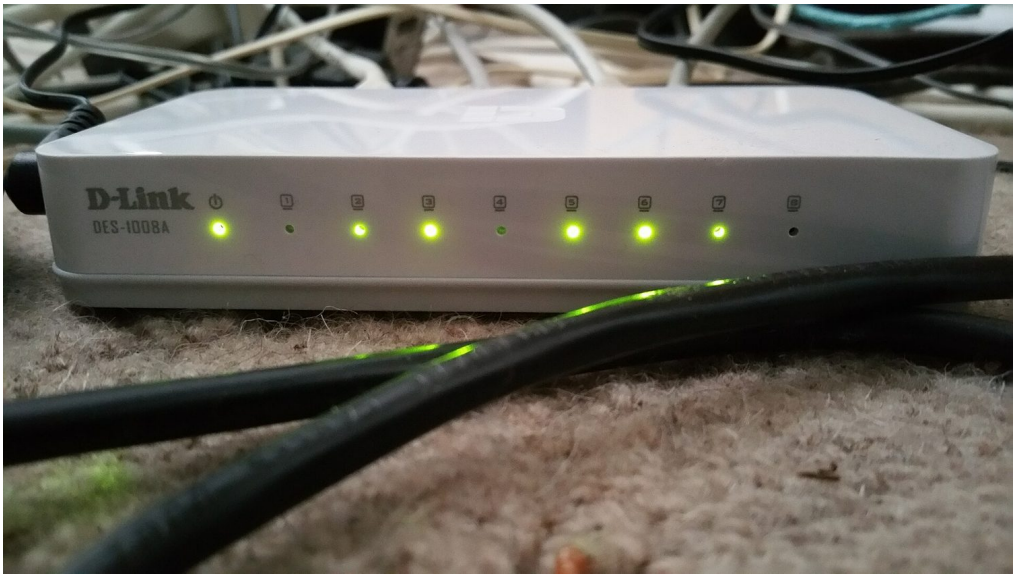
Worked example. A firewall has Rule 3: `DENY TCP 443 from 192.168.*`, and lower down Rule 7: `ALLOW TCP 443 from ALL`. A user at `192.168.45.37` cannot reach port 443 - even though Rule 7 would allow them - because Rule 3 matches first, and the *first match wins*. If this specific host should have HTTPS access, add an ALLOW for `192.168.45.37` **above** Rule 3 while retaining the broader DENY. Moving ALLOW-from-ALL above that DENY would allow the whole range. Choose rule order and scope to match the intended policy.

Worked example: write the rules. Requirements for a web server at `10.0.0.5`: everyone may reach HTTPS on port 443; only the administrator's computer at `10.0.1.20` may use SSH on port 22; all other traffic to the server is denied.

Rule	Action	Service	From	To
1	ALLOW	TCP 443	ALL	`10.0.0.5`
2	ALLOW	TCP 22	`10.0.1.20`	`10.0.0.5`
3	DENY	ALL	ALL	`10.0.0.5`

Test each requirement by reading from the top. A visitor on port 443 matches rule 1. A student trying port 22 misses rules 1 and 2 and stops at rule 3. Now put rule 3 first: every packet matches it, and the server is cut off completely. **The most specific rules go first, and the broad deny goes last.**

Firewalls belong at every point where data crosses between zones - at each network segment and at every gateway to the public internet. Each firewall can be set separately, so a segment holding student records can have stricter rules than a guest network.



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** 日志文件 that record network activity:

- a **network intrusion detection system (NIDS)** 网络入侵检测系统 analyses traffic and **raises an alert**, but does not block;
- a **network intrusion prevention system (NIPS)** 网络入侵防御系统 can also **stop** an attack by closing ports or blocking addresses;
- a **security information and event management (SIEM)** 安全信息与事件管理 system gathers data from many sources to spot patterns.

There are two detection methods. **Signature-based** 基于特征 detection compares traffic to a database of known attack **signatures** - often efficient, but may miss activity without a recognised pattern. **Anomaly-based** 基于异常 detection compares traffic to a normal **baseline** 基线 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** 网络入侵指标 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.

Wireless attacks leave their own signs. Regular scans for names that look like a real

SSID can reveal an **evil twin**, and signal triangulation can find where it is. If no device in one area can connect and a scan shows strong radio noise, suspect **jamming**.

■ 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** 概率的: a detector can report a **percentage** score for how suspicious an event appears. The course describes this as a likelihood; a reported percentage should not automatically be treated as a perfectly calibrated probability.

The organisation then sets a **threshold** 阈值 - 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.

Try it – apply an explicit threshold. In this teaching dataset, higher scores mean more suspicious. Alert when score is greater than or equal to the threshold. Investigation has established the outcomes.

Event	Score	Outcome
A	90%	attack
B	70%	normal
C	60%	attack
D	40%	normal
E	30%	attack
F	10%	normal

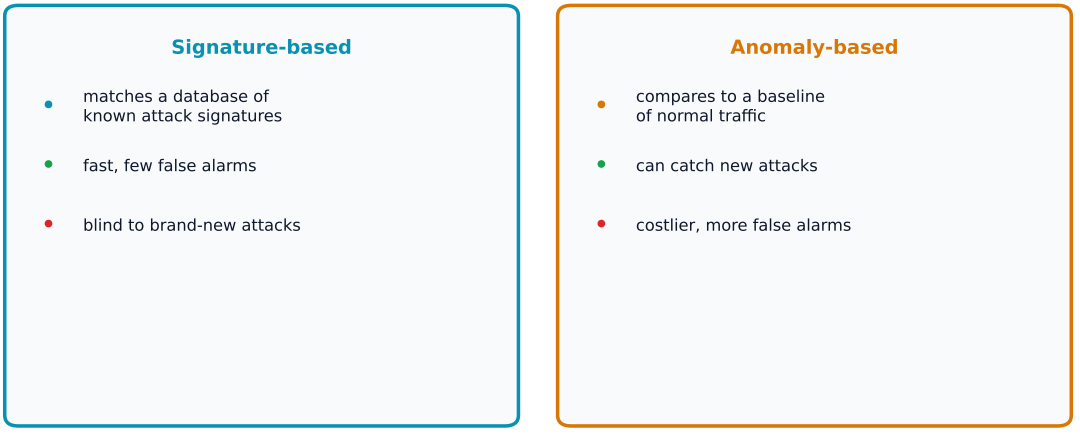
At thresholds of 70% and 30%, list the alerted events, count normal events alerted, and count attacks missed. Include scores equal to the threshold.

Worked reasoning. At 70%, A and B alert: B is one normal event alerted, while attacks C and E are missed. At 30%, A, B, C, D and E alert: B and D are two normal events alerted, and no attack in this sample is missed. Lowering the threshold catches C and E, but adds normal D to the review queue. Even a high threshold can produce a false alert. These scores and investigated outcomes are not measured detector accuracy or a guarantee about future events.

Capacity transfer. Suppose the team can review only two alerts in the period. The 70% setting fits that capacity but misses two known attacks; the 30% setting produces five alerts and exceeds capacity, although it catches all three attacks in this sample. Neither is automatically best. Weigh the harm of misses, the cost of false alerts and investigation capacity. Priorities and other controls matter too; a threshold alone is not a complete defence.

Too many false alerts cause **alert fatigue** 警报疲劳: 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 often has fewer false alarms, while anomaly-based detection can flag unfamiliar activity but may add false alarms. Actual results depend on the signatures, baseline, thresholds and traffic; neither method guarantees detection.



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

Exam tips

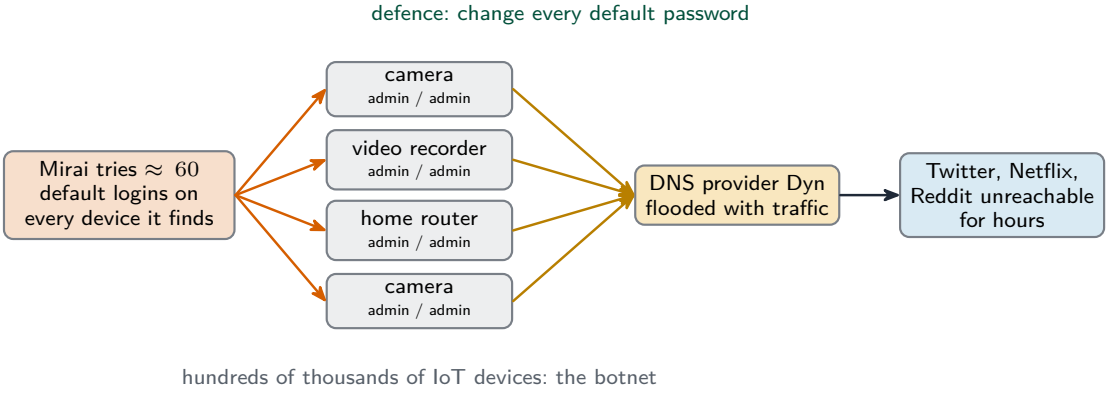
- For firewall-ACL questions, **read the rules top-to-bottom and stop at the first match** - a Deny rule above an Allow blocks the traffic even though the Allow exists lower down.
- Look for attack indicators: unexpected ARP mappings, a surge of new source MAC addresses, or DNS answers that differ from a trusted record. These suggest ARP poisoning, MAC flooding or DNS poisoning respectively; investigate context rather than treating a traffic drop as proof.
- 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** matches known indicators; **anomaly-based** compares a baseline. Discuss coverage, false alerts and resource costs in the scenario; neither guarantees detection.
- 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

Why it matters. On 21 October 2016, many of the most popular websites in the United States, including Twitter, Netflix and Reddit, became unreachable for hours. The attack came from the **Mirai** botnet: hundreds of thousands of internet cameras, video recorders and home routers that still used their factory usernames and passwords. Mirai's code simply tried a list of about sixty default logins, took over every device that accepted one, and used them all to flood the DNS company Dyn with traffic. No clever exploit was needed - just devices nobody had secured. This unit is about how devices are attacked, and how to protect and monitor them.



Mirai, October 2016: default passwords turned home devices into a botnet

A **device** is any computer - a server, a personal laptop, a smartphone, or an **embedded computer** 嵌入式计算机 built into a machine. Network-connected everyday devices with embedded sensors or actuators are called **Internet of Things (IoT)** 物联网 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	desktops and laptops	general purpose, so they run anything the user installs
handheld computers 手持计算机 (also called mobile computers or information appliances)	smaller than a PC and running on **battery power** – smartphones, tablets, smart watches and other **wearable technology**	easily lost or stolen, and often carried across untrusted networks
embedded computers	a computer that is **part of a machine** – a car's engine controller, a thermostat, a medical pump	performs specialised tasks, often using a general-purpose instruction set, 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** 恶意软件 - malicious software. Learn the types:

- **Virus** 病毒 - attaches to a host file or program and runs when that host executes; opening an infected file is one possible trigger.
- **Worm** 蠕虫 - spreads by itself, with no human action.
- **Trojan** 木马 - hides inside software that looks safe; a **remote access trojan (RAT)** 远程访问木马 gives the adversary remote control.
- **Ransomware** 勒索软件 - encrypts your files and demands payment for the key.
- **Spyware** 间谍软件 - secretly tracks what you do.
- **Keylogger** 键盘记录器 - records every keystroke to steal passwords.
- **Logic bomb** 逻辑炸弹 - 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** 无文件恶意软件 is different: it often runs in **RAM** 内存 and abuses legitimate programs already on the device instead of a standalone malicious executable. It can leave traces or persist in settings such as the

registry; memory and behaviour monitoring can still detect it.

Adversaries exploit **unpatched software** 未打补丁的软件, 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**.

UNPATCHED

known flaws left open:
exploits work for years

DEFAULT PASSWORDS

cameras, routers taken
over in bulk

MALWARE

a downloaded file installs
ransomware or spyware

LOST DEVICE

no lock, no encryption:
the data is read directly

Common device vulnerabilities and the attacks that use them

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** 密码散列函数 - a one-way maths algorithm that turns any input into a fixed-length string called a **hash** 散列值 (or digest). Hashes have three vital properties: they are **collision**

resistant 抗碰撞 (hard to find two inputs with the same output), have **pre-image resistance** 抗原像 (recovering an input from a hash is computationally infeasible in general, although weak passwords can be guessed), and are **repeatable** (the same input always gives the same hash).

Try it – read real hash results. These inputs use UTF-8 with no final newline. Only the first 16 of each full SHA-256 hash's 64 hex characters are displayed.

Input	Hash prefix	Full hex length
`cat`	`77af778b51abd4a3`	64
`cat again`	`77af778b51abd4a3`	64
`cats`	`d936608baaacc6b7`	64

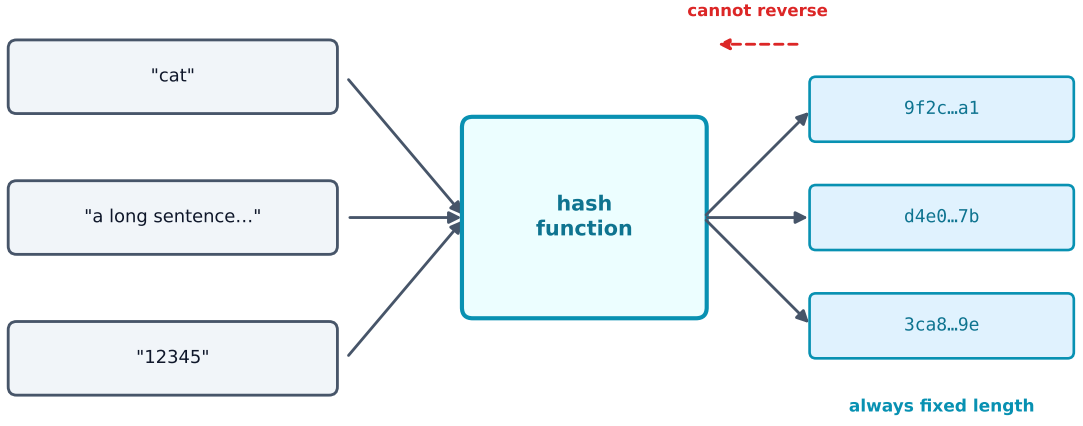
Which rows illustrate repeatability? What stays fixed when input length changes? Do these results prove that collisions are impossible?

Worked reasoning. The two identical `cat` inputs give the same full hash. Both `cat` and `cats` have outputs of 64 hex characters. Each hex character represents 4 bits:

$$b = 4h = 4 \times 64 \text{ bits} = 256 \text{ bits}$$

The changed output for `cats` illustrates sensitivity to an input change; three examples do not prove collision resistance. Matching prefixes alone do not establish that full hashes match. The full outputs in this worked example were checked.

Why collisions exist. An n-bit output has $N = 2^n$ possible values. A toy 4-bit hash has $N = 2^n = 2^4 = 16$ outputs. With 17 different inputs, at least two must share an output. This toy is not secure. Real cryptographic hashes have much larger output spaces, but collisions still exist; the goal is to make finding them computationally infeasible.



Schematic: the digest labels are illustrative, not computed SHA-256 values. There is no direct decryption step; guesses can still be hashed and compared.

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** 弃用 (retired from secure use). **MD5** and **SHA-1** are the classic deprecated examples – never rely on them to protect data today.

A secure service should not store your plaintext password. It stores a salted password **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** 盐值 are added before hashing, so identical passwords do not normally produce identical stored hashes. Salt is stored with the hash and does not need to be secret.

> **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 random salt - shortened here to `x7` and `q2` for readability - and the service hashes `sunshinex7` and `sunshineq2` instead. The two stored hashes now look completely different, so the adversary must attack each account separately. Salt prevents reuse of one unsalted lookup table across accounts, but does not make a weak password strong. This simple concatenation illustrates the idea; real password storage uses a purpose-built slow password-hashing scheme with a suitably sized unique random salt, not raw SHA-256.

Transfer – one-way does not mean unguessable. A stolen record has salt `x7` and a SHA-256 hash. Its displayed prefix is `f9325402a7a1542a`. An adversary guesses `rainbow` and `sunshine`, appends the salt, hashes the results, and compares them with the record. The `rainbowx7` prefix is `2a0e6ce47f2c701d`; `sunshinex7` begins `f9325402a7a1542a`. Which guess should be checked in full? Was the hash decrypted? Would live account lockout stop these computations?

Worked answer. The `sunshine` guess should be checked in full because its displayed prefix matches. The prefix alone cannot establish a full match; the `rainbow` prefix already rules that guess out. This is guessing, hashing and comparing, not decryption. A known salt does not prevent guessing a weak password. Account lockout controls live login attempts; it does not stop computations on a stolen database. The guesses generate no login events at the service, although database theft or later use of the credentials may leave evidence.

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

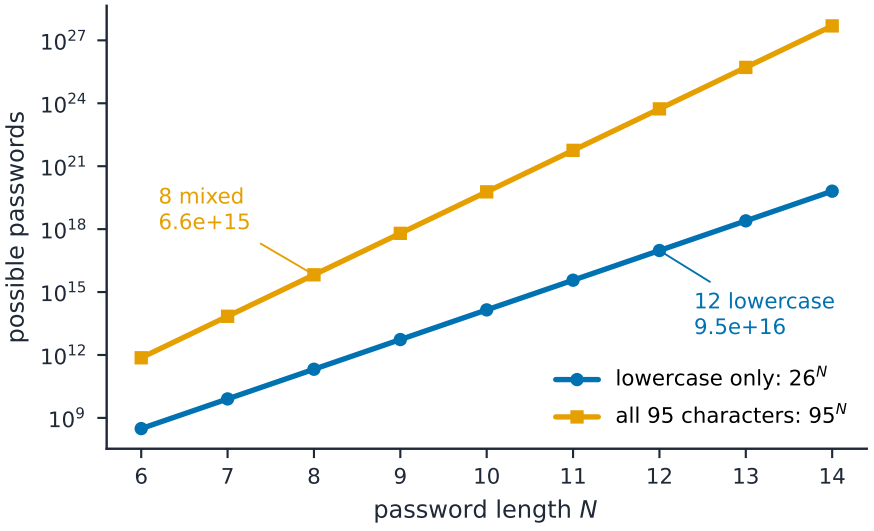
- **brute force** 暴力破解 - an automated tool tests all candidates in a chosen search space. Exhaustive search finds a match if the space includes the password and every candidate can be tested, but the cost may be impractical. For length L with A choices per position, the candidate count is $N = A^L$. Real attacks often try likely choices first.

- a **dictionary attack** 字典攻击 - the tool tries a list of common words and known passwords first, because most people pick guessable ones.
- **password spraying** 密码喷洒 - one common password against many accounts (this dodges lockout, which counts failures per account).
- **credential stuffing** 撞库 - reusing stolen or default credentials, exploiting that people reuse passwords across sites.
- a **rainbow table** 彩虹表 - precomputed chains of hash and reduction operations that trade storage for recovery work on supported unsalted password candidates; it is not simply a complete list of every password and hash.

■ Password policy settings

The course names several login-policy settings. Describe their mechanisms on an exam without assuming that enabling every setting is current best practice:

Setting	What it does	The attack it slows
complexity 复杂度	require a character from each set (upper, lower, digit, special)	brute force / dictionary
minimum length 最小长度	require N characters - length matters more than anything	brute force (grows exponentially)
maximum age 最长有效期	sets a time limit after which a password must change; a policy mechanism, not a universal recommendation	limits how long a stolen password is useful
password history 密码历史	store the last 5-10 hashes, block reuse	stops recycling an old (possibly leaked) password
lockout 锁定	lock the account after 3-5 wrong tries	brute force / online guessing



Length beats complexity: each added character multiplies the search space

Distinguish a setting from a recommendation. NIST SP 800-63B-4 says not to require periodic password changes or character-class composition rules. Change a password when compromise is suspected or known, and block common/compromised choices. Routine expiry can encourage predictable patterns like `PasswordFall2028`. A **password manager** 密码管理器 solves the real problem - it generates and stores a long, unique password per site, so none is ever reused or guessable.

Authentication factors fall into categories: something you **know** (a password), something you **have** (a token), or something you **are** (a **biometric** 生物特征). **Multi-factor authentication (MFA)** 多因素身份验证 combines distinct factor categories. The course also discusses location ("somewhere you are"); under NIST, location is context for assessing risk, not a substitute for an independent authentication factor.



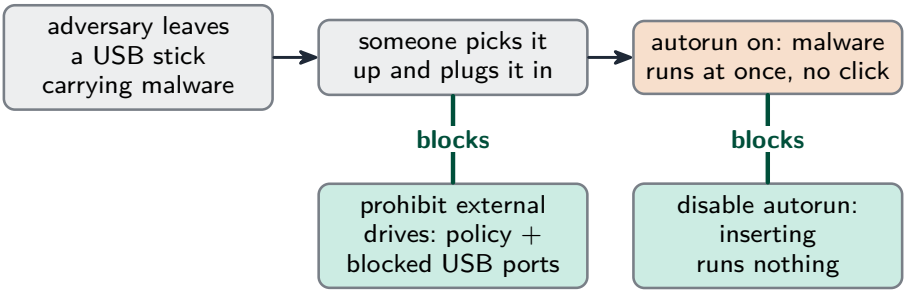
A hardware security key proves who you are with something you physically hold – a strong second factor

■ 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** 自动运行 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.

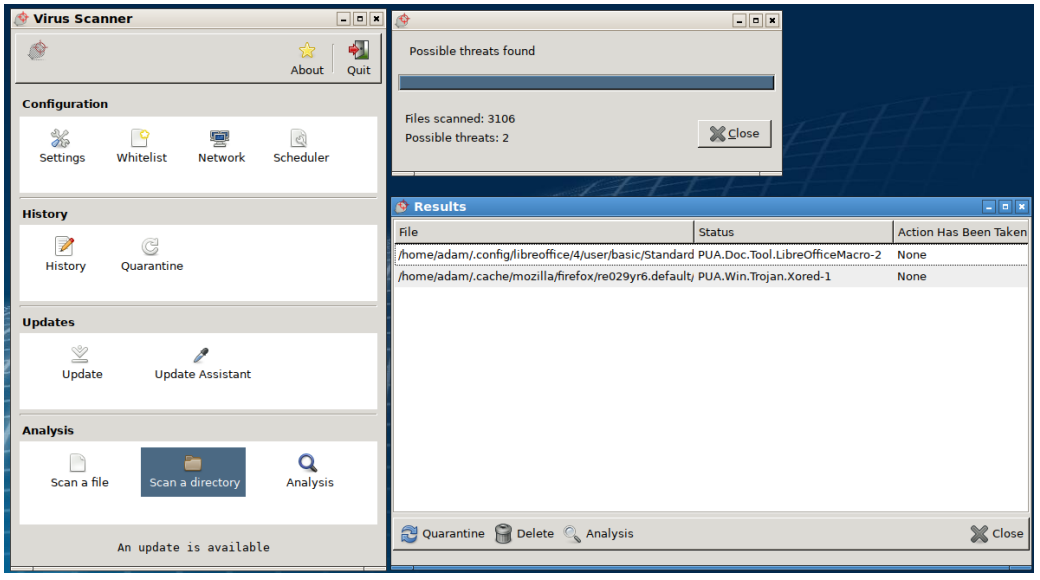


Each of the two controls breaks a different link in the chain

Protecting Devices

Managerial controls set the rules: an **acceptable use policy** 可接受使用政策 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** 反恶意软件 keeps a database of malware **signatures** and quarantines any file that matches. Keeping the operating system and applications **updated** - installing each **patch** 补丁 - closes known holes before adversaries can use them. A **host-based firewall** 主机防火墙 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

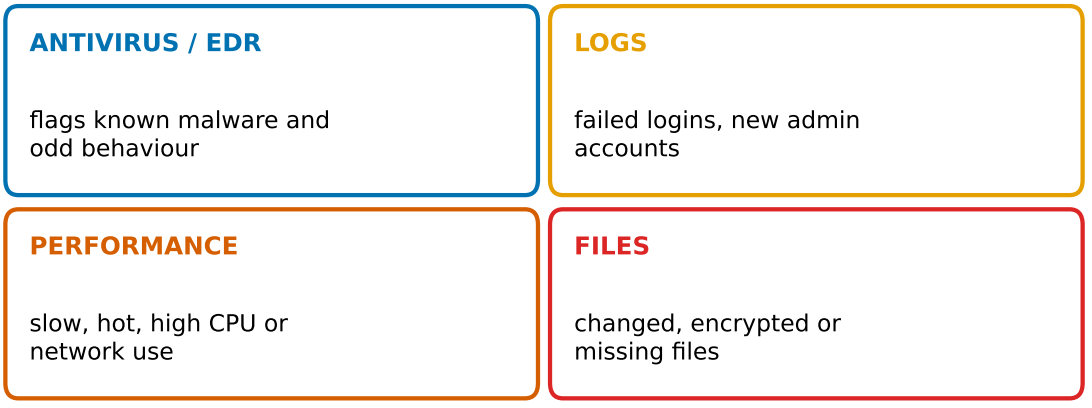
Detecting Attacks on Devices

Devices log logins, file changes, and processes, and these logs reveal an **indicator of compromise (IoC)** 入侵指标 - evidence suggesting possible intrusion, to investigate in context. **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)** 端点检测与响应 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 in quick succession is an indicator of **password spraying**; the AP course document also treats rapid default-credential attempts as an indicator of **credential stuffing**. These patterns

need context; one shared IP address does not by itself prove an attack. Offline guesses create no live login events because they run on the adversary's copy; the theft or later use of the credentials may still be detected.

Speed is itself a security factor. Signature-based detection compares what it sees against a list of known-bad patterns, so it is often cheaper to run than **anomaly-based** detection, which compares events with a learned model of normal behaviour. Anomaly-based detection can catch attacks that have no signature yet, but may need more processing power; actual costs depend on the implementation and workload – 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.



Signs of a compromised device

Worked example: reading an authentication log. Two excerpts from a school server's log:

Time	User	Source IP	Result
02:14:01	`mchen`	203.0.113.7	failed
02:14:02	`mchen`	203.0.113.7	failed
02:14:03	`mchen`	203.0.113.7	failed
… 200 more in 4 minutes …	`mchen`	203.0.113.7	failed

Time	User	Source IP	Result
03:00:10	`adiaz`	198.51.100.20	failed
03:00:40	`bkumar`	198.51.100.20	failed
03:01:10	`cliu`	198.51.100.20	failed
03:01:40	`dpatel`	198.51.100.20	**success**

- **Log 1** suggests a **brute-force** or dictionary attack on one account: hundreds of failures, one user, one source, seconds apart at an unusual hour. Investigate the source and account activity. **Account lockout** can limit online guesses, and the source IP can be blocked.
- **Log 2** is consistent with **password spraying**: attempts span several accounts without repeated failures on one account. The log does not show the passwords, so it cannot establish that one common password was used; credential stuffing is another possibility. The success on `dpatel` warrants checking with the user and reviewing later activity. Contain suspected compromise by disabling access or revoking sessions and resetting credentials. MFA can block password-only access; blocking common passwords reduces spraying risk.

The pattern to look for is *what repeats*: the same user (brute force), the same source across many users (spraying), or a success right after a run of failures.

Exam tips

- Know each **malware type by its defining trait**: a worm self-spreads, a virus attaches to a host and runs when the host executes, ransomware encrypts for money, a RAT gives remote control, a rootkit hides.
- A hash is **one-way and fixed-length**; a unique **salt** changes the hashed input even when passwords match. A secure service should store a **salted password hash**, using a purpose-built slow scheme rather than raw SHA-256.
- 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: cite failed-login entries, timing and users. These are indicators, not proof. The AP CED includes default-credential attempts under **stuffing**.
- Recognise the course's **know / have / are / where** labels, while distinguishing location context from an independent NIST authentication factor. **MFA** combines distinct factors, not two passwords.
- **Offline guessing creates no live login events**: the computations happen on the adversary's copy. Theft or later account use may still leave evidence.

Course terminology and implementation boundary. The AP CED 4.2.B.5 includes default-credential attempts under credential stuffing. OWASP's narrower definition focuses on replaying stolen username/password pairs across services. Use the AP definition for AP questions, while identifying the actual evidence and mechanism. The course's named general-purpose hashes illustrate hashing; they are not a recommendation to store passwords with plain SHA-256. See OWASP password-storage guidance and NIST hash properties.

Common mistakes.

- Calling every malicious program a virus. Name the type by its trait: a worm spreads by itself; a virus needs a host program to run.
- Thinking hashing is encryption. A hash cannot be reversed with a key; it is checked by hashing the guess again.
- Treating a password and a security question as two factors. Both are something you know, so it is still one factor.
- Assuming signature-based detection catches new attacks. It only matches known patterns.
- Leaving default credentials or skipping patches because a device "only" controls a camera or a thermostat.

Practice.

1. Name the attack shown by each log pattern: (a) 500 failed logins to one account from one IP address in two minutes; (b) one failed login each to 300 accounts, all with the password `Spring2024!`; (c) many accounts tried once each with username-password pairs leaked from another site.
2. Explain why a unique salt for each user defeats a rainbow table.
3. Classify each authentication factor: a PIN, a code from an authenticator app on a phone, a fingerprint.
4. An attack uses a vulnerability no vendor knows about yet. Which kind of detection is more likely to notice it - signature-based or anomaly-based? Explain.
5. A toy hash has 4-bit outputs. Explain why any 17 different inputs must include a collision.
6. Give two steps that would have stopped Mirai from taking over a home camera.

Answers.

1. (a) Brute force; (b) password spraying; (c) credential stuffing.
2. A rainbow table is precomputed for unsalted inputs. A unique random salt changes every input, so the attacker would need a new table for every salt, which is not practical.
3. Something you know; something you have; something you are.
4. Anomaly-based, because it flags behaviour that differs from normal; signature-based needs a known pattern, which does not exist yet for a new attack.
5. There are only $2^4 = 16$ possible outputs, so with 17 inputs at least two must share an output (the pigeonhole principle).
6. Change the default password to a strong unique one, and install firmware updates (patches); also block remote login from the internet if not needed.

Securing Applications and Data

AP Cybersecurity

Application and Data Vulnerabilities and Attacks

Why it matters. In March 2017 a security fix was published for a flaw in Apache Struts, a widely used web-application framework. The credit agency Equifax did not apply it. In May, attackers used the flaw to get into an Equifax dispute website, moved through its systems for about eleven weeks, and took names, birth dates and identity numbers of about 147 million people. The attack was not new or clever; the patch had been available for two months. This unit covers how applications and data are attacked, and the controls - patching, access control, encryption and monitoring - that protect them.

Applications 应用程序 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** 管理性 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** 注入攻击. **Data validation** 数据验证 (checking input meets expected rules) is one layer of defence; SQL, browser output and file paths also need protections suited to their destination. Key attacks:

- **SQL injection** SQL 注入 - inserting **SQL** commands into an input field to read or change a database.
- **Cross-site scripting (XSS)** 跨站脚本 - 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 conventionally written in capital letters** (SQL itself ignores their case, so `select` works too) – `SELECT`, `FROM`, `WHERE`, `IN`, `OR`, `AND`. An unsafe login form may build 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.** In the password field of the unsafe query shown, entering `' OR '1'='1` changes its logic to `(name = 'alice' AND password = '') OR '1'='1'`, which is true for every row. The same text in the name field does not automatically bypass the remaining password check.

- **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** 参数化查询 (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** 缓冲区溢出 - sending more data than a memory **buffer** 缓冲区 can hold, so it overflows into nearby memory and may run the adversary's code.
- **Directory traversal** 目录遍历 - using `../` sequences in a URL to reach files outside the intended folder, such as `/etc/passwd`.

Rate data risk using both likelihood and impact. Sensitivity, exposure, access controls, encryption and credible threats matter; low-value data or a short key alone does not determine a fixed risk rating.

> **Worked example.** Rate two data-confidentiality risks. (A) A clinic's patient records, which health-privacy laws protect, sit unencrypted in a database behind a public web form, and the form's software has a published SQL injection exploit. (B) A public bus timetable, stored without encryption on the same kind of server. For A, the **impact** is high: the data is highly sensitive and governed by law, so a breach brings fines, lawsuits and harm to patients. The **likelihood** is high too: the form is exposed to the internet, and a published exploit needs little skill. High likelihood and high impact give a **high** (or severe) risk, to be fixed first, for example with parameterised queries, encryption and patching. For B, the server could be attacked just as easily, but the data is already public, so its theft does little harm: a **low confidentiality** risk. Changing the timetable or taking the service offline could still harm integrity or availability; those risks need a separate assessment. The same weakness gives different confidentiality risks because the **value of the data** differs.

```
login form input:  ' OR '1'='1
```

the query the program builds by pasting the input in:

```
SELECT * FROM users WHERE pw = '' OR '1'='1'
```

'1'='1' is always true, so every row matches: the login is bypassed

fix: parameterised queries and input validation keep data from becoming code

SQL injection: input pasted into a query changes the query itself

Protecting Applications and Data: Managerial Controls and Access Controls

Data is classified by its **state** - **at rest** 静态数据 (stored on a drive), **in transit** 传输中数据 (moving between devices), and **in use** 使用中数据 (being processed). Data at rest and in transit can be **encrypted** so a thief cannot read it; conventional processing normally decrypts data in use, so **access controls** and protection of memory and execution are also needed.

Some data types are **regulated** 受监管 - the law dictates how they must be stored, transmitted and handled - so an organisation must achieve **compliance** 合规 by matching its controls to the rules. The course pairs these data types with example U.S. legal frameworks or industry standards. Their scope matters; these are not universal one-to-one rules:

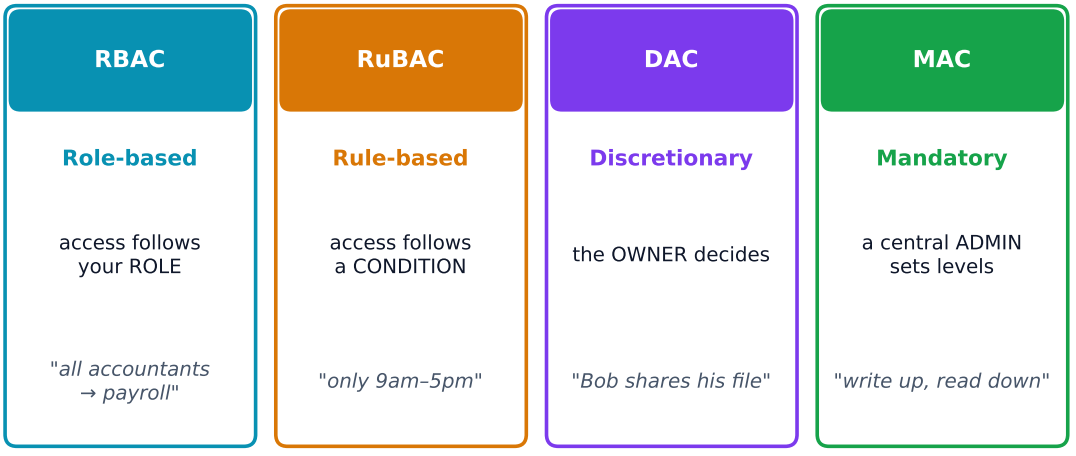
Data type	What it is	Example framework and scope
personally identifiable information (PII) 个人身份信息	information identifying or linked to a person	Privacy Act: U.S. federal-agency records; COPPA: covered online collection from children under 13
protected health information (PHI) 受保护健康信息	identifiable health information in covered contexts	**HIPAA**: covered entities and business associates, with defined scope and exclusions
payment card information (PCI) 支付卡信息	payment-card account and authentication data	**PCI DSS**: an industry security standard for relevant card-data environments

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:

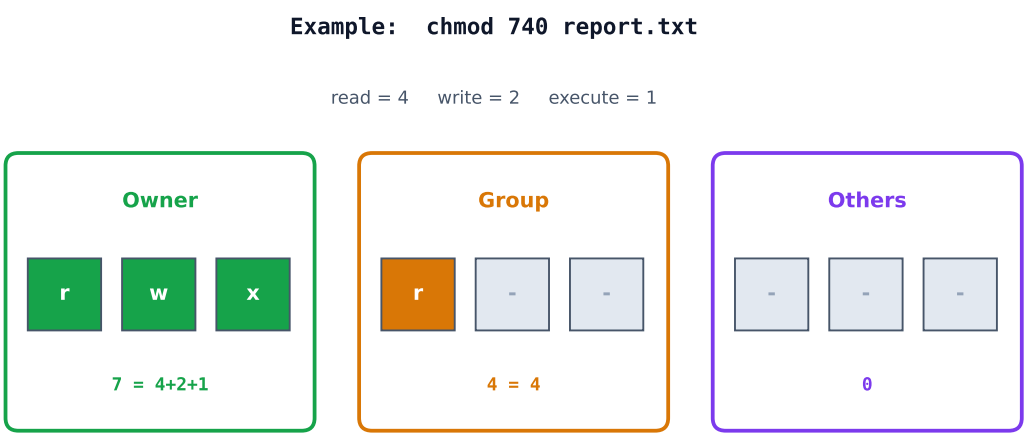
- **Role-based (RBAC)** 基于角色的访问控制 - access follows your **role** (all "accountants" reach the payroll software).
- **Rule-based (RuBAC)** 基于规则的访问控制 - access follows **conditions** (only during business hours), layered on another model.
- **Discretionary (DAC)** 自主访问控制 - the **owner** of a file decides who else may use it.
- **Mandatory (MAC)** 强制访问控制 - a central administrator sets strict levels; the **Bell-LaPadula** model summarises it as "write up, read down".



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

A guiding idea across all models is the **principle of least privilege** 最小权限原则 - 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).



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 密码学 uses mathematical techniques to protect information and sup-

port checks such as authenticity and integrity. An **encryption** algorithm combines the **plaintext** 明文 with a **key** 密钥 to produce **ciphertext** 密文; **decryption** reverses it. The **keyspace** 密钥空间 is the set of possible keys. A fully variable **n-bit** key has 2^n possible values. A larger space can increase exhaustive-search work, but practical security also depends on the algorithm and implementation.

Symmetric encryption 对称加密 uses the **same key** to encrypt and decrypt. The standard is **AES** 高级加密标准, a **block cipher** 分组密码 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.

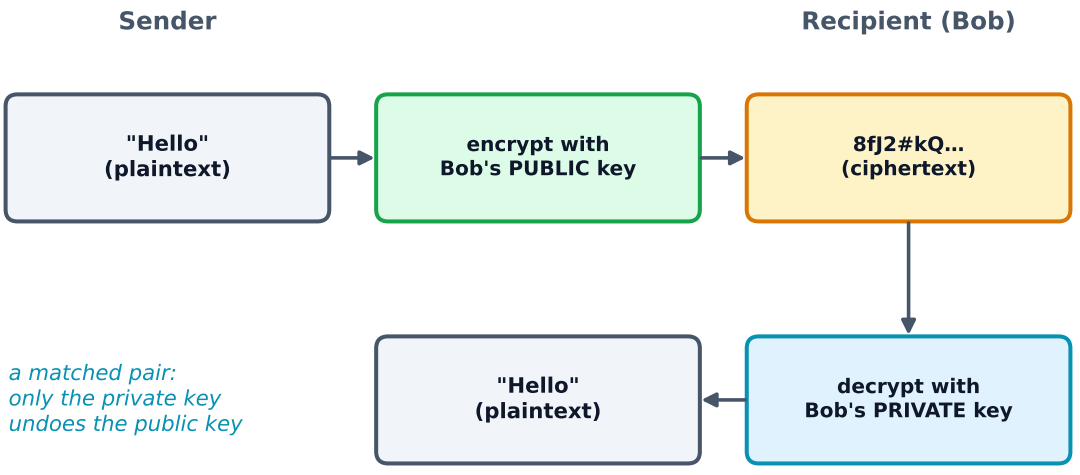


The Enigma machine scrambled messages with rotors – an early, breakable example of encryption

Asymmetric Cryptography

Asymmetric encryption 非对称加密 solves the key-sharing problem with a **key pair** 密钥对 - a **public key** 公钥 anyone may see and a **private key** 私钥 kept secret. Each scheme specifies how its keys are used. For public-key encryption, I encrypt with **your public key** and you decrypt with **your private key**. Digital signatures use a private signing key and public verification key; signing is not generally "encrypting with the private key". The sender must also establish that the public key really belongs to the intended recipient.

exercises.lol exercises.com.cn



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)** 椭圆曲线密码学, 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.



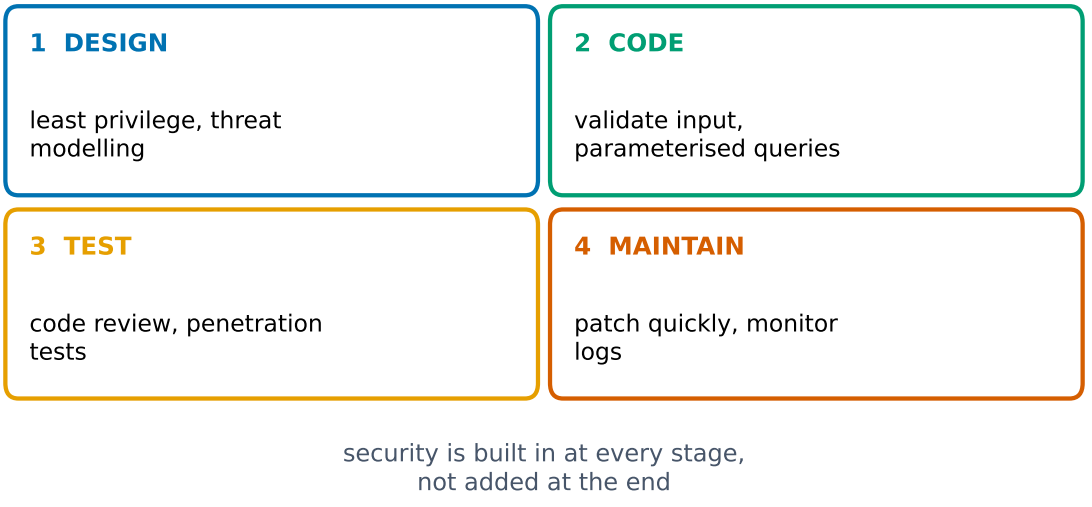
A padlock: cryptography locks data so only someone with the matching key can open it

Protecting Applications

Two design principles keep applications safe from the start. **Secure by design** 安全设计 builds security into every phase of development, not as an afterthought. **Secure by default** 默认安全 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.

Input sanitization 输入清理 transforms input for a specific destination; validation checks an allowed type, length, range or format. Quotes and semicolons are printable **special characters** 特殊字符, not necessarily control characters. A legitimate name such as `O'Neill` should remain intact. A single blacklist is not a universal defense: SQL injection needs **parameterised queries** with separately bound values; XSS needs safe text rendering or output encoding for the destination context (and a suitable HTML sanitizer when HTML is allowed); directory traversal needs resolved paths confined to an allowed directory. For example, `0 OR 1=1` can change an unsafe numeric SQL expression without using quotes or semicolons. Use fixed SQL such as `SELECT name FROM pupils WHERE name = ?` and bind the entered name separately.



Building security into every stage of an application's life cycle

Detecting Attacks on Data and Applications

To detect data attacks, systems perform **accounting** 审计记录 - 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** 蜜罐 - a fake file that looks valuable; unexpected access gives an alert to investigate. Administrators or scanners may also access a decoy, so an alert alone is not proof 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)** 数据泄露防护 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. A change normally gives a different digest, but collisions exist. A match to the publisher's trusted full digest is strong evidence that the bytes match, rather than mathematical proof. 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.

△ Comparing a hash checks **integrity**; an untrusted hash does not establish **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**.
- Match input defenses to the destination: parameter binding for SQL, context-specific output protection for XSS, and allowed-directory checks for paths. **Input sanitization** is not one universal character filter; a **honeypot** is a detective control.

Common mistakes.

- Thinking input validation alone stops SQL injection. Parameterised queries (binding) are the main defence; validation is an extra layer.
- Encrypting a message to a friend with your own private key. To keep it secret, use the **recipient's public key**; your private key is for signing.
- Confusing hashing (integrity, one-way) with encryption (confidentiality, reversible with a key).
- Reading `chmod 750` as decimal. Each digit is a sum of 4 (read), 2 (write) and 1 (execute) for owner, group and others.
- Giving every user administrator rights "to make things easier", which breaks least privilege.

Practice.

1. A web log shows a login request with the username `admin' OR 1=1 --`. Name the attack and the main defence.
2. Write the permissions for `chmod 640` in words, and the number for owner `rwx`, group `r-x`, others `r--`.
3. Name the access-control model in each case: (a) nurses can see patient records because they are nurses; (b) a file's creator decides who can open it; (c) logins are allowed only between 07:00 and 19:00.
4. Alice wants to send Bob a secret message and prove it came from her. Which keys does she use, and for what?
5. Why is AES, not RSA, used to encrypt a large file, even when RSA is used in the same system?
6. Give two controls that would have prevented or limited the Equifax breach.

Answers.

1. SQL injection; use parameterised queries (bound parameters) so the input is treated as data, not code.
2. Owner read and write, group read, others nothing; `754`.
3. (a) Role-based (RBAC); (b) discretionary (DAC); (c) rule-based (RuBAC).
4. She encrypts with Bob's public key (only Bob's private key can decrypt) and signs with her own private key (anyone can check it with her public key).
5. Symmetric encryption is much faster for large data; RSA is used to share the AES key safely.

exercises.lol exercises.com.cn

6. Apply security patches promptly; limit access between systems (least privilege, network segmentation) and monitor for unusual data leaving the network.

exercises.lol exercises.com.cn