Professor Messer Security+ SY0-701: Study Notes

Free study notes for the Professor Messer's free CompTIA Security+ SY0-701 course, one section per exam objective. Written by HoverNotes in our own words as a companion to the videos: watch the objective's videos, then use these to review and test yourself. Not affiliated with Professor Messer or CompTIA.

Course guide, study plan and reviews: https://hovernotes.io/courses/professor-messer-security-plus

Made with HoverNotes, the notes panel that sits beside any video. Version 2026-11-02.

1.1 Security controls

Exam domain: 1. General Security Concepts · **Video:** Security Controls (12 min)

Key ideas

- A **security control** is anything that stops an incident, limits its damage, or helps you recover. It protects more than data: buildings, equipment and people count too.
- Every control has two labels. The **category** says *how* it's carried out; the **type** says *what it does*. The exam asks for one or both.
- **Four categories:**
  - **Technical:** enforced by systems. OS policies, firewalls, antivirus.
  - **Managerial:** decided on paper. Security policies, standard operating procedures, onboarding rules.
  - **Operational:** carried out by people. Guards, awareness training, posters, lunch-and-learn sessions.
  - **Physical:** limits physical access. Fences, locks, badge readers, a guard shack.
- **Six types:**
  - **Preventive:** blocks access before anything happens (a firewall rule, a locked door).
  - **Deterrent:** doesn't block, but makes an attacker think twice (a login warning banner, warning signs, the threat of being fired).
  - **Detective:** spots and records an incident while or after it happens (log review, motion sensors, a patrol).
  - **Corrective:** acts after detection to undo or limit the damage (restoring from backup after ransomware, a fire extinguisher).
  - **Compensating:** a substitute when the proper fix isn't possible yet (a firewall rule until the vendor ships a patch, a generator during a power cut).
  - **Directive:** tells people what to do, and relies on them doing it. The weakest type (a policy to keep sensitive files in an encrypted folder, an "authorized personnel only" sign on an unlocked door).

One example per square

Type	Technical	Managerial	Operational	Physical
Preventive	Firewall rule	Onboarding policy	Guard checks IDs	Door lock
Deterrent	Login splash screen	Threat of demotion	Reception desk	Warning signs
Detective	System logs	Reviewing login reports	Property patrol	Motion detector
Corrective	Restore from backup	Policy for reporting issues	Calling law enforcement	Fire extinguisher
Compensating	Block traffic instead of patching	Separation of duties	Several guards on shift at once	Backup generator
Directive	File storage policy	Compliance policy	Security training	"Authorized personnel only" sign

These are examples, not the only right answers. Most controls can be argued into more than one square, so learn the reasoning, not the table.

Remember for the exam

- **Category = who or what enforces it; type = when and how it acts.** A guard is operational (category) and can be preventive or detective (type), depending on the job.
- **Deterrent vs preventive:** a deterrent discourages; a preventive control physically or technically stops. A sign is a deterrent, a lock is preventive.
- **Compensating** means "not the real fix": it's temporary or a substitute while the primary control is missing.
- **Directive** controls depend on people following instructions, which is why they're the weakest.
- **Detective** controls don't stop anything; they tell you it happened.

Check yourself

1. A vendor won't have a patch for two weeks, so you block the vulnerable port at the firewall. Which control type? *(Compensating: it stands in for the patch.)*
2. A sign reads "Trespassers will be prosecuted." Which type and category? *(Deterrent, physical.)*
3. A security analyst reviews login failure reports every Monday. Which type and category? *(Detective, managerial: a reviewing process, not a system alert.)*
4. Why is a laptop backup restore a corrective control and not a preventive one? *(It acts after the incident to recover from it.)*

Do next

Draw the 6 × 4 grid from memory and fill one example of your own into each square. Then move on to 1.2, which builds the vocabulary (CIA, AAA, zero trust) used in every later objective.

1.2 Fundamental security concepts

Exam domain: 1. General Security Concepts · **Videos:** The CIA Triad (5 min) · Non-repudiation (8 min) · Authentication, Authorization, and Accounting (9 min) · Gap Analysis (7 min) · Zero Trust (10 min) · Physical Security (8 min) · Deception and Disruption (5 min)

Key ideas

The CIA triad

- **Confidentiality:** only the right people can see the data. Encryption, access controls and two-factor authentication support it.
- **Integrity:** data hasn't been altered without anyone noticing. Hashing, digital signatures, certificates and access controls support it.
- **Availability:** systems and data are reachable when needed. Redundancy, fault tolerance and patching support it.

Non-repudiation

- **Non-repudiation** means a person can't credibly deny they did something, such as sending a message or approving a payment.
- **Proof of origin** shows who sent something. A **digital signature** gives it: the sender signs with a private key, and anyone can check the signature with the matching public key.
- **Proof of integrity** shows the content wasn't changed. A **hash** of the message provides it: if one bit changes, the hash changes.
- A signature is a hash that has been signed, so it delivers both. This is how the exam expects you to link authenticity, integrity and non-repudiation.

AAA: authentication, authorization, accounting

- **Authentication** proves a user or device is who it claims to be (password, certificate, biometric).
- **Authorization** decides what the proven identity may do. Access is usually granted by role or group.
- **Accounting** records what happened: logins, time spent, data sent, resources used. The logs support auditing and billing.
- An **AAA server** (RADIUS and TACACS+ are the usual ones) sits in the middle so devices don't each hold their own accounts.
- **Authenticating people** can use a username and password, or a one-time code. **Authenticating systems** often uses a **device certificate**: a signed digital certificate is stored on the device, and a trusted authority vouches for it.
- An **authorization model** defines how rights are granted. A common approach maps users to roles and gives each role only what it needs.

Gap analysis

- A **gap analysis** compares where you are with where you want to be, then lists what's missing.
- Start by choosing a **baseline**: a standard such as a published framework, or your own target. Then **evaluate** current systems against it, usually through a mix of interviews, document review and technical tests.
- The result is a **report** with the gaps and a plan to close them, including time, staffing and cost.
- Example: a framework says all remote access needs multifactor authentication, but your VPN uses passwords alone. The gap is the missing second factor.

Zero trust

- **Zero trust** treats every request as untrusted, whether it comes from inside or outside the network. Each person, device and process must prove itself every time.
- It splits a network into two logical parts. The **control plane** decides; the **data plane** carries the traffic.
- **Control plane parts:**
  - **Adaptive identity** judges each request using the context around it, such as location, device or past behavior.
  - **Threat scope reduction** limits how much of the network a given identity can reach.
  - **Policy-driven access control** applies the written rules.
  - The **policy administrator** issues or revokes access based on the engine's decision, and the **policy engine** makes that decision.
- **Data plane parts:**
  - The **subject/system** is the person or device asking.
  - The **policy enforcement point** is the gatekeeper that allows or blocks. Everything has to pass through it to reach the **implicit trust zone**, where the protected resources live.
- Result: a user inside the office doesn't get free movement. Every resource is still checked.

Physical security

- **Barriers:** bollards and fences stop vehicles and people. An **access control vestibule** (a two-door entry space) lets one person through at a time, preventing tailgating.
- **Badge readers, biometrics and guards** check who is entering. A **guard** can also be paired with a second person for a **two-person integrity** check, and an **access list** says who's allowed.
- **Lighting** makes intruders easier to see, and **cameras (CCTV)** record events. Cameras can use **motion detection**, and some can run object recognition.
- **Sensors: infrared** (heat), **pressure**, **microwave** and **ultrasonic** sensors detect movement or presence.

Deception and disruption

- A **honeypot** is a fake system built to attract attackers so defenders can watch their methods. A **honeynet** is a whole fake network.

- Most of what hits a honeypot is automated, so the point is to learn which tools and targets attackers go for while they waste time on systems that aren't in production. It's an arms race: attackers try to spot fakes, defenders make them more convincing.
- A **honeyfile** is a bait file, such as one named like a password list, that raises an alert when opened. Nobody legitimate should touch it, so any access is a signal.
- A **honeytoken** is traceable fake data, such as made-up API keys left on a public share or a fake email address. If it shows up somewhere else, you know where it leaked from.

Remember for the exam

- **Confidentiality / integrity / availability** map to: encryption, hashing, redundancy.
- **Non-repudiation = digital signature = hash + signing with a private key.** It proves origin and integrity.
- **Authentication = who you are; authorization = what you can do; accounting = what you did.**
- **Control plane decides, data plane acts.** The policy engine/administrator are control plane; the enforcement point is data plane.
- A **honeypot** is a single bait system; a **honeynet** is a bait network; a **honeytoken** is bait data.
- A **vestibule** controls entry one person at a time; **bollards** stop vehicles.

Check yourself

1. A bank needs proof that a customer can't later deny approving a transfer. What should it use? *(A digital signature, which gives non-repudiation.)*
2. You compare your firewall configuration against a framework and list what's missing. What is this? *(A gap analysis.)*
3. In zero trust, which component decides whether a request is allowed? *(The policy engine in the control plane; the enforcement point carries out the decision.)*
4. An attacker opens a file named "passwords.xlsx" on a server and the security team gets an alert. Which control? *(A honeyfile.)*
5. A user logs in successfully and then sees only the folders their role permits. Which AAA step limits what they see? *(Authorization.)*

Do next

Without notes, sketch the zero trust control plane and data plane and label each part. Then move on to 1.3, which covers how changes to systems are approved and made without breaking these protections.

1.3 Change management

Exam domain: 1. General Security Concepts · **Videos:** Change Management (11 min) · Technical Change Management (11 min)

Key ideas

Business processes that govern change

- **Change management** is the formal process for deciding whether, when and how a system gets modified. In a large environment one change can touch thousands of systems, so unplanned edits cause outages and security gaps.
- **Typical steps:** fill in a **change request** that states the reason; define the **scope** (which systems); plan the date, time and affected services; run a **risk analysis**; have the change approved or rejected by a **change control board** (committee); implement it; then test and document.
- **Owner:** the person or team that owns the application or data. They usually start the request and, afterward, confirm it works. They don't normally do the technical work.
- **Stakeholders:** everyone affected by the change. They can be hard to spot. Updating label-printing software may seem to concern only the shipping team, but accounting reports and revenue timing may depend on it.
- **Impact analysis:** estimates how much damage a change could cause if it goes wrong. Risk is rated, for example, high, medium or low. Consider the risk of acting (a patch breaks something) and the risk of not acting (an unpatched vulnerability stays exposed).
- **Test results:** try the change in a **sandbox** that mirrors production before touching real systems, so mistakes cost nothing.
- **Backout plan:** a documented way to return to the previous state if the change fails. Test it too, because an untested rollback may not work. Keep a current **backup** as the last resort.
- **Maintenance window:** the agreed period for making the change, usually outside business hours. Some organizations also declare a **freeze** during their busiest season, when no changes are allowed.
- **Standard operating procedure (SOP):** the written steps for doing the change, so everyone follows the same method.

Technical implications

- **Allow list / deny list:** an allow list permits only named applications and blocks everything else. A deny list permits everything except named applications. Antimalware works like a large deny list. Changing either list is itself a change that needs approval.
- **Restricted activities and scope:** the approved request defines what the technician may change. They shouldn't use the window to do unrelated work. If completing the task needs a small extra edit, policy should say who can authorize it.
- **Downtime:** changes may interrupt service. Tell users in advance through email or a shared change calendar. In always-on environments, update a standby system, switch users to it, and keep the original as a quick fallback.

- **Service restart / application restart:** a change may need a full reboot, or only restarting one service or application. The restart also proves the system comes back correctly.
- **Legacy applications:** old software that nobody wants to touch is risky, because the vendor may no longer support it. Documenting it brings it back into normal support.
- **Dependencies:** one update can require another first, sometimes on a different system. For example, firewall management software may need every firewall updated to a new version before it will install.
- **Documentation:** update network diagrams, IP lists and procedures after each change so they don't go stale.
- **Version control:** track versions of configurations, patches and code so you can see what changed and revert to an earlier version.

Remember for the exam

- The **change control board** approves; the **owner** requests and verifies; the **technician** implements.
- A **backout plan** is required, and it should be tested. A **backup** is separate and covers the case where the backout also fails.
- **Allow list = default deny; deny list = default allow.**
- **Sandbox testing** happens before production, not after.
- Documentation and **version control** are part of the change, not an optional extra.
- A **freeze** period or after-hours window reduces business impact.

Check yourself

1. Which team member approves a change that affects several departments? *(The change control board.)*
2. A patch worked in testing, but production fails. What should already exist? *(A backout plan, backed by a current backup.)*
3. A company wants only approved programs to run on workstations. Allow list or deny list? *(An allow list.)*
4. A firewall manager won't install until the firewalls run newer code. What complication is this? *(A dependency.)*
5. Why might a retailer ban all changes in December? *(A change freeze protects its busiest period from the risk of disruption.)*

Do next

Write a one-page change request for patching a web server, with a reason, scope, risk level, test step, backout plan and window. Then move on to 1.4, where cryptographic tools show up in many of these protections.

1.4 Cryptographic solutions

Exam domain: 1. General Security Concepts · **Videos:** Public Key Infrastructure (9 min) · Encrypting Data (10 min) · Key Exchange (4 min) · Encryption Technologies (7 min) · Obfuscation (8 min) · Hashing and Digital Signatures (10 min) · Blockchain Technology (2 min) · Certificates (15 min)

Key ideas

Public key infrastructure and key management

- **Public key infrastructure (PKI)** is the set of policies, people and systems that create, distribute, store and revoke digital certificates and keys.
- **Symmetric encryption** uses one shared secret key. It is fast, so it handles bulk data, but both sides must get the key safely. **Asymmetric encryption** uses a key pair: a **public key** anyone may have and a **private key** kept secret. What one key encrypts, only the other can decrypt. It is slower, so it's used to protect small items.
- **Key escrow:** a trusted third party holds a copy of a key so data can be recovered if the original is lost, which also raises trust and legal questions.
- **Key management:** keys must be generated, stored, rotated and retired carefully. Storing keys with the data they protect defeats the purpose.

Encrypting data

- **Levels of encryption: full-disk** and **partition/volume** encryption protect everything on a drive. **File** encryption protects single files. **Database** encryption works at the whole-database level, or on a column or **record**.
- **Transport/communication encryption:** protects data in transit, for instance with **TLS** or a **VPN** using IPsec.
- **Hardware support:** a **Trusted Platform Module (TPM)** is a chip that stores keys and can check system integrity. A **hardware security module (HSM)** is a dedicated, tamper-resistant device (or card) for generating and storing many keys. A **key management system** manages keys in a central place. A **secure enclave** is an isolated processor area for sensitive operations.
- **Exam pairing:** TPM = on a single device; HSM = centralized, for servers and many keys.

Key exchange

- Sharing a symmetric key over an open network is a problem. One answer is **out-of-band** exchange, such as delivering it by another route.
- Another is **in-band** exchange protected by asymmetric encryption: the sender encrypts the symmetric (session) key with the recipient's public key. The recipient uses the private key to recover it.
- Combining both gives a **session key**: a one-time symmetric key, created for each conversation. Algorithms like **Diffie-Hellman** let two parties derive a shared secret without sending it.

Encryption technologies and algorithms

- **Key stretching:** repeatedly processes a weak password so each guess takes longer to attack. **bcrypt** and **PBKDF2** do this.
- **Key length:** longer keys make brute force harder, and **AES-256** is stronger than AES-128. Keep the algorithm's key size in mind when comparing.
- **Salting:** random data added to a password before hashing so identical passwords produce different hashes, defeating pre-computed rainbow tables.
- **Common algorithms: AES** (symmetric, block cipher, 128/192/256-bit keys) is the standard; **3DES** is older; **RSA** and **ECC** are asymmetric.

Obfuscation

- **Steganography** hides data inside another file, such as an image or audio file, so no one knows a message exists. The content may or may not be encrypted.
- **Tokenization** swaps sensitive data (such as a card number) for a random token. The real value is stored elsewhere. It's common in payments.
- **Data masking** hides part of a value, for example showing only the last four digits of a card number.

Hashing and digital signatures

- A **hash** turns any input into a fixed-length digest. It is one-way (you can't recover the input), and any change to the input changes the output.
- Uses: confirm **integrity** of downloads and files, store **passwords**, and support digital signatures.
- A **collision** is when two different inputs give the same hash. A good algorithm makes this very unlikely. **MD5** is broken; **SHA-1** is deprecated; **SHA-256** is a standard choice.
- A **digital signature** is a hash encrypted with the sender's private key. The recipient decrypts it with the public key and compares the result with their own hash. It provides **authentication**, **integrity** and **non-repudiation**.

Blockchain

- A **blockchain** is a shared ledger of linked blocks, each holding a hash of the previous block. It's stored on many computers, which makes it hard to alter without everyone noticing. Cryptocurrency is one use; others include digital identity and records.

Certificates

- A **digital certificate** binds a public key to an identity and is signed by a **certificate authority (CA)**. It carries the key owner, issuer, validity dates and serial number.
- **Root of trust:** a CA at the top is trusted by browsers and operating systems. Below it, an **intermediate CA** may sign on its behalf, forming a **chain of trust**.
- **Certificate signing request (CSR):** you generate a key pair and send the public key and identity details to a CA, which returns a signed certificate.
- **Certificate types:** a **wildcard** certificate covers a domain and its subdomains (`*.example.com`). A **SAN** (subject alternative name) lists several names in one certificate. A **self-signed** certificate is not

trusted by default, but it's common internally. A **third-party** CA is trusted by default.

- **Revocation:** a **certificate revocation list (CRL)** lists cancelled certificates. **OCSP** checks one certificate online, and **OCSP stapling** has the server include a fresh response so the client doesn't have to ask the CA.
- **Expiry and renewal** need tracking. An expired certificate causes browser warnings.
- A **registration authority (RA)** can verify identities for the CA.

	Symmetric	**Asymmetric**
Keys	One shared key	Public and private pair
Speed	Fast	Slow
Typical use	Bulk data	Key exchange, signatures
Examples	AES, 3DES	RSA, ECC

Remember for the exam

- **Encrypt with the recipient's public key; sign with your own private key.**
- **TPM** is on one device; **HSM** is centralized.
- **Hash** = one-way, integrity; **encryption** = reversible, confidentiality; **signature** = hash + private key.
- **Salting** defeats rainbow tables; **key stretching** slows brute force.
- **Tokenization** replaces data; **masking** hides part; **steganography** hides existence.
- **CRL** is a list; **OCSP** is a live query; **stapling** shifts that query to the server.
- **Wildcard** covers subdomains; **SAN** lists several names.

Check yourself

1. You want only Alice to read a message. Which key do you encrypt with? *(Alice's public key.)*
2. A payment system stores a random substitute for card numbers. What is it? *(Tokenization.)*
3. A laptop's drive keys are held in a chip on the motherboard. What is it? *(A TPM.)*
4. A browser needs to know if a certificate was cancelled without calling the CA each time. What helps? *(OCSP stapling.)*
5. Two users have the same password but different stored hashes. Why? *(Each was salted with different random data.)*

Do next

Draw the exchange of a session key between two people, labeling every key used. Then move on to domain 2 (objective 2.1), where the threats these tools defend against begin.

2.1 Threat actors and motivations

Exam domain: 2. Threats, Vulnerabilities, and Mitigations · **Video:** Threat Actors (10 min)

Key ideas

- A **threat actor** is whoever causes a security event. To judge an attack, you ask who did it and why, because motive points to the likely goal and the defense.
- **Attributes** describe an actor:
  - **Internal or external:** from inside the organization or outside it.
  - **Resources and funding:** how much money, people and infrastructure they have.
  - **Sophistication and capability:** from running someone else's tool blindly to building custom tools.
- **Motivations** include data exfiltration, espionage, service disruption, blackmail, financial gain, philosophical or political belief, revenge, and war.
- **Nation-state:** a government or its security arm. External, very well funded, highly sophisticated. Often runs long campaigns known as **advanced persistent threats (APTs)**, with targets such as military systems, utilities and finance. Motives include espionage, political goals and disruption. Stuxnet is the textbook example.
- **Unskilled attacker:** runs ready-made scripts without understanding them. Typically external, with little money or skill. Motives are mixed: disruption, data theft, or simple curiosity. Also called a **script kiddie**.
- **Hacktivist:** a hacker driven by a cause. Usually external, skilled, with modest funds (though fundraising can help). Typical acts are **denial of service**, **website defacement** and leaking private documents.
- **Insider threat:** a current or former employee or contractor misusing access they already have. Medium sophistication, but they know where the data is and how to get around controls. Motives are revenge or financial gain. Background checks and access limits help.
- **Organized crime:** professional groups, external, well funded, highly sophisticated, driven by profit. They often work like a company, with specialists for hacking, exploit development, selling data and even customer support for ransomware victims.
- **Shadow IT:** departments that set up their own systems or cloud services without involving IT. Internal, limited funds and skills. No malice is required, but they skip backups, change control and security, which creates risk.

Actor	Internal/external	Resources	Sophistication
Nation-state	External	Extensive	Very high
Unskilled attacker	Usually external	Limited	Low
Hacktivist	Usually external	Modest	High
Insider threat	Internal	Moderate	Medium
Organized crime	External	Extensive	High
Shadow IT	Internal	Limited	Low

Remember for the exam

- **Nation-state + APT** means long-term, well-funded espionage or disruption.
- **Hacktivist** = cause-driven; think defacement and DoS.
- **Organized crime** = money; ransomware and data sales.
- **Insider threat** = revenge or profit using legitimate access.
- **Shadow IT** is internal, usually well-meaning, and bypasses IT rules.
- An **unskilled attacker** uses tools they don't understand.

Check yourself

1. A marketing team buys a cloud storage service with a credit card and never tells IT. Which actor? *(Shadow IT.)*
2. A group defaces a company's site to protest its policies. Which actor? *(Hacktivist.)*
3. A long-running, sophisticated campaign against a country's power grid. Which actor? *(Nation-state, an APT.)*
4. A laid-off administrator copies customer records. Which actor and motive? *(Insider threat; revenge or financial gain.)*

Do next

Make a table from memory with each actor's location, funding, sophistication and motive. Then move on to 2.2, which covers the paths those actors use to reach you.

2.2 Threat vectors and attack surfaces

Exam domain: 2. Threats, Vulnerabilities, and Mitigations · **Videos:** Common Threat Vectors (17 min) · Phishing (7 min) · Impersonation (6 min) · Watering Hole Attacks (4 min) · Other Social Engineering Attacks (3 min)

Key ideas

Common threat vectors

Each **threat vector** is a route an attacker can take to reach a target. The **attack surface** is the sum of all such routes.

- **Message-based:** email, SMS and instant messages carry malicious links or attachments. **Image-based** attacks hide code inside an image file such as an SVG, which can contain scripts. **File-based** attacks use documents like PDFs or office files that run malicious content.
- **Voice call:** attackers phone targets and pretend to be someone trusted (see vishing below).
- **Removable device:** a USB drive plugged into a computer can run malware or act as a fake keyboard. Attackers leave drives in parking lots to tempt curiosity.
- **Vulnerable software:** unpatched flaws in programs. **Client-based** software is installed on each device, so a flaw sits on every machine that runs it until each one is patched. **Agentless** software (a web app, for example) runs on a server you connect to, so one compromise of that server reaches every user, and one fix protects them all.
- **Unsupported systems and applications:** products that no longer get security updates are open to known flaws forever.
- **Unsecured networks:** open or weakly protected **wireless** networks, **wired** networks where anyone can plug in, and **Bluetooth** connections with weak pairing.
- **Open service ports:** every listening service is a possible way in, so close what isn't needed.
- **Default credentials:** devices that still use the vendor's username and password are trivially compromised.
- **Supply chain:** attackers compromise a **vendor, supplier or managed service provider (MSP)** and use that trust to reach customers. Hardware, software and services are all exposed.

Phishing and related attacks

- **Phishing:** a fake message designed to trick someone into giving up credentials or running something. Often uses a convincing link to a fake site.
- **Typosquatting (URL hijacking):** registering a domain that looks like a real one with a typing mistake, hoping people land there.
- **Smishing:** phishing by SMS. **Vishing:** phishing by voice call, sometimes with a spoofed caller ID.
- **Spear phishing:** targets a specific person using personal details. **Whaling:** targets executives.
- **Pretexting:** inventing a believable scenario (a made-up story) to gain trust and information.
- **Business email compromise (BEC):** a message that looks like it's from a senior person or supplier, asking for a payment or data change.

- Warning signs: a link whose real address doesn't match the text, an urgent tone, or an unexpected request.

Impersonation

- **Impersonation** pretends to be a trusted person. The attacker may know your organization and use authority or urgency.
- **Eliciting information** is gathering details through casual conversation.
- **Identity fraud** uses someone else's personal information, for example to open accounts.
- Defense: verify the person through a separate, known channel before acting, and don't hand over data on the strength of a call alone.

Watering hole attack

- The attacker infects a **third-party site** the target group regularly visits, then waits. It works when direct routes (email, USB) are blocked by training.
- Some campaigns serve malicious code only to specific IP ranges, so ordinary visitors see nothing wrong.
- No single fix exists. Use **defense in depth**: patching, monitoring, antimalware and network controls.

Other social engineering

- **Misinformation / disinformation:** factually false information, not just a different opinion; disinformation is spread on purpose. **Influence campaigns** (sometimes run by foreign states) push it on social media: fake accounts post it, like and share it, the platform's algorithm amplifies it, real users pass it on, and eventually news outlets may repeat it.
- **Brand impersonation:** attackers build many fake sites carrying a well-known brand's name and look, get them into search results, and use them to push "you've won" pop-ups or malicious downloads that show ads, track browsing or steal data.
- The common thread in all of these is persuading a person instead of attacking technology.

Remember for the exam

- **Smishing = SMS; vishing = voice; whaling = executives; spear phishing = one targeted person.**
- **Typosquatting** uses a misspelled domain.
- **Watering hole** targets a site the victim visits, not the victim directly.
- **Supply chain** attacks go through vendors, suppliers or MSPs.
- **Default credentials, open ports and unsupported software** are common weaknesses to fix.
- **Pretexting** is the invented story; **impersonation** is the false identity.

Check yourself

1. A text message says your parcel is held and links to a fake site. Which attack? *(Smishing.)*

2. A message to the CFO appears to come from the CEO, asking for an urgent wire. Which attack? *(Business email compromise, aimed at an executive (whaling).)*
3. Attackers compromise a site your staff visit for lunch orders. Which attack? *(Watering hole.)*
4. A router still uses its factory username and password. Which vector? *(Default credentials.)*
5. A hacker compromises your IT contractor to reach you. Which vector? *(Supply chain, through an MSP.)*

Do next

List the vectors above and, for each, write one control that reduces it. Then move on to 2.3, which covers the vulnerabilities these attacks exploit.

2.3 Types of vulnerabilities

Exam domain: 2. Threats, Vulnerabilities, and Mitigations · **Videos:** Memory Injections (3 min) · Buffer Overflows (4 min) · Race Conditions (5 min) · Malicious Updates (6 min) · Operating System Vulnerabilities (4 min) · SQL Injection (5 min) · Cross-site Scripting (9 min) · Hardware Vulnerabilities (6 min) · Virtualization Vulnerabilities (5 min) · Cloud-specific Vulnerabilities (4 min) · Supply Chain Vulnerabilities (9 min) · Misconfiguration Vulnerabilities (7 min) · Mobile Device Vulnerabilities (3 min) · Zero-day Vulnerabilities (3 min)

Key ideas

A **vulnerability** is a weakness that an attacker can exploit. This objective walks through the main places weaknesses live: in memory, in applications, in operating systems, in hardware, in virtual and cloud platforms, in the supply chain, in configuration and in mobile devices.

Memory injection

- Every program has to be loaded into memory to run, so malware has to get into memory too. It can run as its own process or **inject** itself into a process that is already running.
- Injected code gains two advantages: it hides inside a legitimate process (a scanner hunting for a malicious process finds nothing odd), and it inherits that process's rights, which can mean **privilege escalation**.
- **DLL injection** is the common form. A **DLL** (dynamic link library) is shared code that many Windows programs load. The attacker places a malicious DLL on storage the system can reach, then plants a reference to its path in the target process. When the process reaches that reference, it loads the malicious DLL into its own memory.

Buffer overflow

- A **buffer** is a fixed-size block of memory. A **buffer overflow** happens when more data is written than the block holds, and the surplus spills into the neighboring memory.
- Developers prevent it with **bounds checking**: refusing input longer than the space reserved for it.
- Overflows are hard to exploit. Most just crash the program. The attacker wants one that is repeatable and overwrites a specific value on purpose.
- Example: a nine-character string is written into an eight-byte field. The ninth character lands in the next variable, which holds the user's permission level. That one extra byte changes the value, and a guest becomes an administrator without any credentials.
- Related: an **out-of-bounds write** puts data in memory the user has no right to touch, which can lead to a crash or remote code execution.

Race conditions

- A **race condition** appears when two events happen almost at once and the program assumes they can't. The outcome depends on timing.

- **Time-of-check to time-of-use (TOCTOU)** is the classic form. The program checks a value, and something changes that value before the program acts on it. The check is no longer true when it is used.
- Example: two users each see a balance of $100 in an account and both withdraw $50. If withdrawals aren't reflected for everyone immediately, both succeed against the same stale balance and the account ends up lower than either user sees.
- Real cases include a 2023 hacking-contest attack on a car's infotainment system that used a race condition to reach root.

Malicious updates

- Patching quickly is the standard advice, but an update is also new software being installed, so it can carry malware.
- Habits that reduce the risk: **back up** before changing a system so you can roll back, and take updates only from **trusted sources**. A browser-update pop-up that appears after you click a search result is not trustworthy; the app's own update function or the vendor's official site is.
- **Digital signatures** let the operating system check that an update came from the named publisher. Built-in updaters carry high trust because they verify the source automatically.
- Trust is not a guarantee. In the **SolarWinds Orion** compromise, attackers got into the vendor's build environment, inserted code, and the poisoned update was signed and distributed through the normal channel. It reached thousands of customers, including large organizations and government agencies, and went unnoticed for months.

Operating system vulnerabilities

- Everyone runs an operating system and the code base is huge, with tens of millions of lines. More code means more flaws, and many are not yet discovered.
- Microsoft ships its monthly security updates on **Patch Tuesday**, the second Tuesday of the month. One release can fix dozens of flaws across several categories, such as elevation of privilege, security feature bypass and remote code execution.
- Once a flaw is public, attackers reverse-engineer the patch to build exploits, so deploy quickly.
- Good practice: always plan to update; **test** patches before rolling them out to a large environment; expect some patches to require a reboot; and keep a **backup** so you can revert if a patch breaks something.

SQL injection

- A **code injection** attack puts attacker-written code into input that the application passes on. Variants include HTML, XML and SQL injection.
- **SQL injection (SQLi)** targets the database layer. The application builds a query from what the user types, for example "select everything from users where name equals the typed value." If input isn't checked, the attacker types extra SQL that changes the query's meaning.
- A tell-tale payload is a condition that is always true, such as `OR 1=1`, which makes the query return every row. If you see that in logs, suspect injection.

- It needs no special tools, only a browser and an input field, yet it can let the attacker read, change or delete the whole database. The fix is **input validation** and parameterized queries.

Cross-site scripting (XSS)

- **XSS** abbreviates the name that way because CSS is already taken by style sheets. It lets an attacker run script, usually JavaScript, in a victim's browser in the context of a site the browser trusts.
- **Non-persistent (reflected):** the attacker sends a link containing the script. The victim clicks, the trusted site reflects the input back, and the script runs and can send cookies or session IDs to the attacker. The victim is the one who runs the code.
- **Persistent (stored):** the attacker posts the script somewhere that keeps it, such as a social media post or comment. Everyone who views the page runs it, and it can even spread itself.
- A stolen **session ID or token** gives the attacker the victim's access. A token that never expires makes the damage worse.
- Defenses: don't click untrusted links, keep the browser updated, consider limiting JavaScript, and, for developers, validate every input field.

Hardware vulnerabilities

- Many devices on a network hide their operating system: thermostats, door controllers, time clocks, appliances and other **IoT** devices. The embedded operating system is **firmware**, and usually only the manufacturer can update it.
- Manufacturers vary in how quickly they fix flaws. A thermostat maker once took roughly a year to ship the first patch for a reported weakness. The device stayed exposed all that time.
- **End of life (EOL)** means the vendor stops selling a product, though support may continue. **End of service life (EOSL)** means no more security patches. EOSL is the date that matters, and the device should then be replaced.
- **Legacy** devices and software (old operating systems, outdated middleware) may be too critical to switch off. If you must keep them, compensate: tighter firewall rules limiting who can reach them, extra IPS signatures, and a plan to replace them.

Virtualization vulnerabilities

- Virtual machines are created and destroyed constantly and are configured differently, so keeping track of their security posture is harder than for a few physical PCs. They still run Windows or Linux, so the usual hardening applies. Privilege escalation, command injection and information disclosure all occur in VMs.
- **VM escape:** the attacker breaks out of one virtual machine and reaches the **hypervisor** or other VMs on the same host. One hypervisor can hold many VMs, so the payoff is large. A hacking-contest chain once went from a browser bug, to the guest kernel, to a flaw in VMware's hardware emulation.
- **Resource reuse:** the hypervisor shares and reallocates memory, storage and other resources between guests. If it has a bug, one VM may read data left in memory that another VM used. Patching the hypervisor fixes it.

Cloud-specific vulnerabilities

- Cloud applications are reachable by anyone on the internet, so they attract attackers constantly. Common weak points: no multifactor authentication on the cloud console, large amounts of unpatched code, weak or misconfigured authentication, and exposure to **DoS/DDoS**.
- Flaws also include **directory traversal** (browsing outside the web root because of misconfiguration), **remote code execution** on unpatched software (Log4j is the well-known example), **cross-site scripting** when input isn't validated, **out-of-bounds writes**, and **SQL injection** against cloud databases.
- Many of these flaws score 7 or more on **CVSS**, the 0–10 severity scale (7.0–8.9 is high, 9.0 and above critical).

Supply chain vulnerabilities

- The **supply chain** runs from raw materials through suppliers, manufacturers, distributors and customers. An attacker who compromises any link can reach everyone downstream.
- **Service providers** often hold access to your systems. Build the right to audit them into contracts. The 2013 Target breach began at an HVAC contractor whose access reached the store network, and the payment systems were not separated from it. Over 40 million cards were taken.
- **Hardware:** treat new equipment as untrusted until verified. Buy from trusted vendors, follow acquisition and setup procedures, and watch for **counterfeit** gear. Fake network equipment sold with real-looking branding has been found in the field.
- **Software:** check **digital signatures**, be careful with automatic updates, and remember that open source gives everyone, including attackers, the ability to propose changes. SolarWinds is again the standard example.

Misconfiguration vulnerabilities

- **Open storage:** data left in a cloud bucket with no authentication. A third party once exposed millions of customer records this way. Attackers actively scan for such buckets.
- **Unsecured administrator accounts:** the root (Linux) or administrator (Windows) account, collectively the superuser. Weak passwords invite brute force. Best practice is to disable direct login to it, sign in as a normal user, and elevate with `sudo` or "run as administrator." Keep the number of privileged accounts small.
- **Insecure protocols:** Telnet, FTP, SMTP and IMAP, as well as plain HTTP, send data unencrypted, so a packet capture reveals credentials. Use SSH, SFTP, HTTPS and the secure versions of the mail protocols.
- **Default settings:** devices ship with a known username and password. The **Mirai** botnet scans for IoT devices that still use factory credentials. Change them at first setup.
- **Open ports:** each listening service is a way in. Close what you don't need, and **audit firewall rules** regularly because large rule sets make accidental exposure easy.

Mobile device vulnerabilities

- Phones are small, always moving, always connected and full of personal and company data, which makes them hard to control.
- **Jailbreaking** (iOS) and **rooting** (Android) replace or bypass the built-in operating system restrictions. This defeats the security that **mobile device management (MDM)** enforces.
- **Sideloading** installs apps from outside the approved app stores, which bypasses vetting and can introduce malicious apps.
- Policy tools: MDM restrictions on where apps may come from, and an **acceptable use policy (AUP)** that forbids these practices, with discipline as the consequence.

Zero-day vulnerabilities

- A **zero-day** is a flaw that the vendor does not know about, so no patch exists. When attackers exploit it, that is a zero-day attack.
- Attackers search for these quietly so that they can use them before a fix appears. Defenders have little warning, so layered defenses matter.
- Track disclosures through the **CVE** list (Common Vulnerabilities and Exposures). Recent examples include browser memory-corruption flaws, a bypass of Secure Boot, and sandbox escapes on phones.

Compare: web application injection flaws

Attack	What the attacker supplies	Where it runs	Typical goal
SQL injection	SQL fragments in an input field	The database	Read, change or delete data
XSS	Script (JavaScript) in a link or stored post	The victim's browser	Steal cookies and session IDs
Buffer overflow	Oversized input	Application memory	Crash or take control of the program
Directory traversal	`../` sequences in a request	The web server's file system	Read files outside the web root

Remember for the exam

- **Injection family:** SQLi targets the database, XSS targets other users' browsers. `OR 1=1` is the SQLi giveaway.
- **Reflected XSS** needs a victim to click a crafted link; **stored XSS** hits everyone who views the page.
- **TOCTOU** is the named race condition: the state changes between the check and the use.
- **EOL is not EOSL.** Support ends at end of service life, so that is when to replace.
- **VM escape** moves between guests on one hypervisor; **resource reuse** leaks data through shared memory.
- **Jailbreak/root** changes the OS; **sideloading** installs apps from outside the store.
- **Zero-day** means no patch exists yet. Compensating controls such as firewall rules and segmentation are the answer.
- **Supply chain** problems come through vendors, MSPs, hardware and software updates, so verify signatures and audit suppliers.

Check yourself

1. A search box returns every customer record after someone types `' OR '1'='1`. Which vulnerability? *(SQL injection: unchecked input altered the database query.)*
2. A comment containing a script runs in the browser of everyone who opens the thread. Which attack? *(Persistent (stored) cross-site scripting.)*
3. A thermostat vendor announces it will never patch a newly found flaw in a model it stopped supporting. Which date has passed, and what do you do? *(End of service life; replace it, or isolate it with firewall rules until you can.)*
4. An analyst sees an attacker move from a compromised guest VM to other VMs on the same host. What is this? *(VM escape.)*
5. A signed, vendor-delivered update contains a backdoor. Which category? *(Supply chain attack through a malicious update.)*

Do next

Without notes, list each vulnerability above and write one sentence on how you would detect it and one on how you would fix it. Then continue to 2.4, which covers the malware, network attacks and log evidence that show these weaknesses being exploited.

2.4 Indicators of malicious activity

Exam domain: 2. Threats, Vulnerabilities, and Mitigations · **Videos:** An Overview of Malware (6 min) · Viruses and Worms (6 min) · Spyware and Bloatware (4 min) · Other Malware Types (8 min) · Physical Attacks (4 min) · Denial of Service (6 min) · DNS Attacks (9 min) · Wireless Attacks (8 min) · On-path Attacks (6 min) · Replay Attacks (6 min) · Malicious Code (4 min) · Application Attacks (12 min) · Cryptographic Attacks (10 min) · Password Attacks (7 min) · Indicators of Compromise (11 min)

Key ideas

An **indicator of malicious activity** is something you can observe that suggests an attack is happening or has happened. This objective covers the attack types themselves and then the evidence they leave.

Malware overview

- **Malware** is any software that does harm: stealing keystrokes, showing profit-making ads, encrypting files, or recruiting your machine into a botnet.
- Types usually work together. A worm might exploit a known flaw, drop a backdoor, and the backdoor then lets the attacker install more tools.
- Malware gets in through a clicked email attachment or link, a pop-up, or a **drive-by download**, where simply visiting a page triggers a download.
- **Ransomware** encrypts your data and demands payment (usually cryptocurrency) for the key. The operating system keeps working so you can read the demand. Defenses: **offline backups** (online ones can be encrypted too), prompt patching of the OS and applications, and **anti-malware with current signatures**.

Viruses and worms

- A **virus** needs a person or action to start it, such as opening a file or clicking a link, and then it replicates. Some cause outages; others stay hidden.
- Virus forms: **program** viruses (executables), **boot sector** viruses (run at start-up), **script** viruses (in browser or OS scripts), and **macro** viruses (in office documents).
- A **fileless virus** never writes malicious files to disk, so antivirus that watches storage misses it. It runs in memory, often by abusing PowerShell, and may add a registry autostart entry so it survives a reboot.
- A **worm** spreads on its own across networks with no user action, at network speed. Firewalls, host firewalls and intrusion prevention systems help stop it, provided they know its signature.
- **WannaCry** is the standard example. It spread using the EternalBlue exploit against SMBv1, installed a backdoor, then loaded ransomware.

Spyware and bloatware

- **Spyware** watches what you do: browsing habits, personal data, ad injection, even affiliate fraud. It arrives via peer-to-peer software, fake security tools and malicious links, and may include a

keylogger. It can be hard to uninstall. Use anti-malware, install only trusted software, and keep a clean backup.

- **Bloatware** is software pre-installed by the manufacturer, often because the vendor is paid to include it. It uses storage, may run at start-up, and each program is another attack surface that may contain flaws. Remove it through the OS's list of installed apps or its own uninstaller; a third-party uninstaller tool is a last resort.

Other malware types

- **Keylogger:** records everything you type, and sometimes the clipboard, screenshots, chat messages and searches, then sends a file to the attacker. Typing is not encrypted, so VPNs and disk encryption don't help. A **RAT** (remote access Trojan) often bundles keylogging with remote control.
- **Logic bomb:** dormant code that triggers on a condition, such as a date, time or a particular user logging in. Because it is custom code, it has no signature. Detect it by **monitoring critical files for changes**, using change control, and applying **least privilege**. In 2013 a time-triggered attack in South Korea wiped systems at banks and broadcasters, including ATMs.
- **Rootkit:** hides deep in the operating system, often in the kernel, so it doesn't show up as a process and antivirus can't see it. Standalone removal tools exist for known ones. **Secure Boot** in UEFI checks the OS kernel's signature before starting, which blocks a tampered kernel.

Physical attacks

- Whoever can touch a machine can often get around its software defenses.
- **Brute force** works on doors and windows too: just breaking in.
- **RFID cloning** copies an access badge or fob in seconds with a cheap device, sometimes just by standing near the owner. **Multifactor authentication** (a PIN or biometric as well as the badge) limits the damage.
- **Environmental attacks** target the surroundings: cutting power, shutting off HVAC so systems overheat, or tampering with fire suppression, each of which takes the site offline.

Denial of service

- A **denial of service (DoS)** forces a service to fail by overload, by abusing a flaw, or simply by cutting power. Motives include competitive advantage and a smokescreen for another attack.
- Self-inflicted outages count too: a switching loop without spanning tree, a huge download that eats the bandwidth, a burst pipe over the server room.
- A **distributed DoS (DDoS)** uses many machines, typically a **botnet** of infected computers controlled through one command. It is an **asymmetric** threat: a small attacker can overwhelm a large target.
- **Reflection and amplification:** the attacker sends a small request with a **spoofed source address** to a public service, and the large reply goes to the victim. DNS, NTP and ICMP can all amplify. In one DNS example, a short query to an **open DNS resolver** returned a response dozens of times larger.

DNS attacks

- **DNS poisoning** makes a name resolve to the wrong IP address, sending users to an attacker's server. Routes include: editing the **hosts file** on a client (needs local admin rights), intercepting queries and answering with a fake reply (needs an on-path position), and changing records on the **DNS server** itself.
- **Domain hijacking** takes over the registration of a domain, for example by brute-forcing, phishing or reusing leaked credentials for the registrar account. The attacker then controls where the domain points. In 2016 attackers redirected dozens of a Brazilian bank's domains for hours.
- **URL hijacking** (also **typosquatting** or **brandjacking**) registers domains that resemble a real one: a misspelling, an added or missing letter, or the same name with another top-level domain. The site may show ads, harvest logins, host malware or be sold back to the real owner.
- Don't follow links blindly; look closely at the domain.

Wireless attacks

- **Deauthentication attack:** a DoS technique: the attacker sends forged **management frames** that disconnect a device from the access point. Older 802.11 management frames weren't encrypted, so anyone nearby can forge them. The attacker needs the device's MAC address and the access point's identifier, which are visible in captures (tools such as airodump-ng and aireplay-ng are used). Newer standards encrypt many management frames (protected management frames), though some, such as beacons and probes, must stay in the clear.
- **RF jamming:** transmitting interference to raise noise and cut the signal-to-noise ratio, so every device in range loses service. The signal can be constant, random, made of legitimate-looking frames, or **reactive**, switching on only when someone transmits. Microwave ovens and fluorescent lights can cause accidental interference on 2.4 GHz.
- Both attacks need the attacker to be nearby. A **fox hunt** locates a jammer using a directional antenna, and an attenuator to narrow the direction as the signal gets strong.

On-path attacks

- An **on-path attack** (formerly called man-in-the-middle) puts the attacker between two devices, where they can read and sometimes modify the traffic. Neither side notices.
- **ARP poisoning (spoofing)** works on a local subnet. ARP maps IP addresses to MAC addresses and has no authentication, so the attacker sends false replies claiming "I am the router's IP, here is my MAC." Victims update their ARP caches, and traffic flows through the attacker.
- **On-path browser attack** (man-in-the-browser): malware or a Trojan on the victim's own machine acts as a proxy. It sees data before encryption and after decryption, so HTTPS doesn't help. It can wait for a banking login and then start hidden sessions.

Replay attacks

- A **replay attack** captures valid data and sends it again later to impersonate the victim. The capture may come from a network tap, ARP poisoning (an on-path attack) or malware. Gathering the data may involve an on-path position, but sending it back later is a separate step.

- **Pass the hash:** the attacker captures a username and password hash during authentication and replays it to the server, which accepts it as valid. Defenses: encrypt traffic, and use **salted** hashes with a server that won't accept the same value twice.
- **Cookies and session hijacking (sidejacking):** cookies hold session state and aren't malicious themselves, but a stolen **session ID** lets an attacker use your logged-in session with no password. IDs can be taken from captured headers, from XSS, or by tampering with them. Tools like Wireshark, Kismet and Scapy are used. Defenses: **end-to-end encryption (HTTPS)**, HTTPS-only browser settings, and a VPN when end-to-end isn't possible.

Malicious code

- **Malicious code** is a catch-all: executables, scripts, macros, Trojans. It matters when social engineering and default credentials won't work, because it takes more technical skill.
- Defense is layered: anti-malware, firewalls, continuous patching and user training.
- Examples: WannaCry's SMBv1 code execution, a breach of an airline's payment pages through a small amount of injected JavaScript (cross-site scripting), and a national health database accessed by SQL injection.

Application attacks

- **Injection:** untrusted input is treated as code. SQL is the most common; others are HTML, XML and LDAP. It works when the app doesn't validate input.
- **Buffer overflow:** writing past the end of a variable's memory. It often just crashes the app, but a repeatable one can be powerful.
- **Replay:** reusing captured credentials or session IDs (see above).
- **Privilege escalation:** gaining rights beyond those assigned. **Vertical** escalation moves up to administrator or system level. **Horizontal** escalation moves sideways to another user's access at the same level. Mitigate by patching, updating anti-malware signatures, using **data execution prevention (DEP)** to stop code running in data areas of memory, and **address space layout randomization (ASLR)** to move things around in memory on each run.
- **Cross-site request forgery (CSRF or XSRF):** tricks a logged-in user's browser into sending an unwanted request to a site that trusts it, such as a funds transfer, usually via a crafted link. It abuses the site's trust in the browser. **Anti-forgery tokens** in each request defeat it. Compare XSS, which abuses the user's trust in a site.
- **Directory traversal:** a web server misconfiguration or flaw lets requests read or write files outside the web root. In logs, look for `../` sequences in the URL.

Cryptographic attacks

- Algorithms are public and heavily tested, so attackers target the **implementation** or weak algorithms rather than the key.
- **Birthday attack:** a hash collision attack. By the birthday paradox, among only 23 people the chance that two share a birthday is about 50%, so collisions are easier to find than brute force suggests. In cryptography, two different inputs give the same hash. **MD5** is known to have collisions; researchers

once forged a certificate that looked CA-signed. Defense: use a **longer hash output** and modern algorithms.
- **Downgrade attack:** forces two parties to use weaker encryption or none. **SSL stripping** is an on-path variant: the attacker proxies the connection, keeps the victim on plain HTTP while talking HTTPS to the server, and reads everything in between.

Password attacks

- Passwords should be stored as **hashes**: fixed-length output that can't be reversed, where different inputs give very different hashes. Storing plaintext is a serious flaw.
- **Spraying:** try a few very common passwords (such as `123456` or `password`) against many accounts, staying under the lockout limit so nothing alarms.
- **Brute force:** try every combination until the hash matches. Online it's slow and triggers lockouts, so attackers steal the **password file** and work **offline** with unlimited attempts, limited only by time and computing power. Long passwords and strong hash algorithms slow this.

Indicators of compromise (IOC)

- An **IOC** is evidence that you have probably been breached. Watch for:
  - **Account lockouts** you didn't cause, or an account disabled without anyone's request (an attacker may want the help desk to reset it).
  - **Concurrent session usage:** one user logged in from two locations at once.
  - **Blocked content:** security sites and updates unreachable because malware stops the system patching itself.
  - **Impossible travel:** logins from distant places within minutes.
  - **Resource consumption:** unusual data transfers, traffic spikes at 3 a.m., or odd traffic in firewall logs, which can mean exfiltration.
  - **Resource inaccessibility:** a crashed server, a disrupted network, files encrypted by ransomware, locked accounts.
  - **Out-of-cycle logging:** activity at unexpected times, such as installs outside the patch schedule.
  - **Missing logs:** attackers delete logs, so alert on gaps.
  - **Published or documented data:** your private data shows up online, often after ransomware extortion.

Remember for the exam

- **Virus** needs user action; **worm** spreads alone; **fileless** lives in memory; **rootkit** hides in the kernel (Secure Boot helps); **logic bomb** waits for a trigger and has no signature.
- **Offline backups** are the key ransomware defense.
- **Spraying** = few passwords, many accounts. **Brute force** = many passwords, usually offline against a hash.
- **XSS** abuses a user's trust in a site; **CSRF** abuses a site's trust in a user's browser.
- **Vertical** privilege escalation goes up; **horizontal** goes across to another user.

- **Typosquatting** = URL hijacking = brandjacking; **domain hijacking** = taking the registration.
- **Deauthentication** exploits unprotected management frames; **jamming** exploits RF interference.
- **ARP poisoning** is local-subnet on-path; **SSL stripping** is an on-path **downgrade** attack.
- **Impossible travel** and **concurrent sessions** are IOCs found in authentication logs.

Check yourself

1. Files across a share are encrypted and a note demands payment. Which malware, and what lets you recover without paying? *(Ransomware; restore from an offline backup.)*
2. A PC shows nothing suspicious in the process list but has been compromised at the kernel level. What is this? *(A rootkit.)*
3. An attacker tries "Summer2025!" once against 5,000 accounts. Which attack? *(Password spraying.)*
4. Staff are knocked off the Wi-Fi repeatedly with no warning, while the access point is up. Which attack? *(Wireless deauthentication, a DoS.)*
5. A user logs in from Omaha, and 20 minutes later the same account logs in from Sydney. What is this? *(Impossible travel, an indicator of compromise.)*
6. A user is logged in to their bank when a crafted link transfers money without their knowledge. Which attack? *(Cross-site request forgery.)*

Do next

Make a two-column table: attack on the left, the log entry or symptom that reveals it on the right. Then move on to 2.5, which looks at the mitigation techniques that stop these attacks.

2.5 Mitigation techniques

Exam domain: 2. Threats, Vulnerabilities, and Mitigations · **Videos:** Segmentation and Access Control (6 min) · Mitigation Techniques (7 min) · Hardening Techniques (12 min)

Key ideas

Segmentation and access control

- **Segmentation** splits a network into smaller pieces so an incident stays contained. It can be **physical** (separate hardware), **logical** (VLANs on a switch) or **virtual** (cloud and virtual machine networks).
- Reasons to segment: **performance** (a dedicated subnet for a high-bandwidth app), **security** (users reach an application server, never the database behind it) and **compliance** (PCI DSS may require card data to sit apart from the rest of the network).
- An **access control list (ACL)** allows or denies traffic or access based on attributes such as source and destination address, port, user or time of day. One rule might let a single host reach `192.168.1.0/24` only on TCP 80, 443 and 8088. Take care that a new ACL doesn't lock you out of managing the device.
- File and folder permissions on an operating system are also ACLs.
- **Application allow list:** nothing runs unless approved. Strict, and the strongest against unknown malware. **Application deny list:** everything runs except what's named. Easier to live with, but it only stops known-bad software. Antivirus is the classic deny list.
- Windows can identify an application by its **hash** (any change breaks the match), by its **digital-signature certificate**, by the **path** it runs from, or by the **network zone** (private or public).

Mitigation techniques

- **Patching** closes known vulnerabilities before they are used. Large organizations test patches first, then push them out. An emergency patch can break the normal schedule when attackers are already exploiting a flaw.
- **Encryption** limits what a thief can read. File-level encryption (Windows **EFS**) protects chosen files; **full disk encryption** (BitLocker on Windows, FileVault on macOS) covers the whole volume, including the OS. Applications can also encrypt their own data.
- **Monitoring** needs logs from firewalls, IPS devices, operating systems and sensors. A **SIEM** gathers them in one place for reports and alerting.
- **Least privilege** gives each person only what the job needs. Ideally nobody works as an administrator day to day; they elevate briefly and drop back. A compromised account then exposes less.
- **Configuration enforcement** checks a device at login with a **posture assessment**: OS version, patches, antivirus or EDR signatures, local firewall status, even a device certificate. A failing device is quarantined or put on a restricted VLAN until fixed.
- **Decommissioning:** before retiring equipment, wipe or remove the storage. A drive that will be reused internally can be erased and reformatted; a drive that held sensitive data and isn't needed again is better physically destroyed.

Hardening techniques

- **Operating system hardening:** apply updates, set a password policy (minimum length and complexity), limit admin accounts, restrict which IP addresses may reach the system, and run endpoint protection.
- **Encrypt data** at rest (EFS, BitLocker, FileVault) and in transit (VPN, HTTPS).
- **Endpoint hardening** has to suit each platform: desktops, laptops, tablets and phones run different systems and apps. No single setting does it, so layers combine into **defense in depth**.
- **EDR (endpoint detection and response)** goes beyond signatures. It uses **behavioral analysis**, machine learning and **process monitoring**, performs **root-cause analysis**, and can act alone by isolating the host, quarantining the threat or rolling back changes. It reports to a central console, and APIs make the automation possible.
- **Host-based firewall:** software on the endpoint that allows or blocks traffic in both directions per process. It sees traffic before encryption or after decryption, can flag unknown processes, and is managed centrally.
- **Host-based IPS (HIPS):** often built into EDR or anti-malware. It uses signatures and heuristics and can see OS-level events such as a buffer overflow, a registry change or edits to core system folders, then alert and block.
- **Open ports and services:** every listening port is a possible way in, so close what isn't needed, using the host or a firewall (ideally a next-generation firewall that sees the service, not just the port number). Installers sometimes open ports silently, and some vendors wrongly tell you to open every port, 0 through 65535. **Nmap** scans a system to show what is open.
- **Default credentials:** change the factory username and password on routers, switches, firewalls, access points and application consoles. Defaults are public. Add MFA or centralized authentication where possible.
- **Remove unnecessary software:** every installed app has its own bugs and update process. Uninstalling what you don't use shrinks the patching burden and the attack surface.

Remember for the exam

- **Allow list = default deny** (only approved runs). **Deny list = default allow** (only listed items blocked).
- **EFS = file or folder; BitLocker/FileVault = whole disk.**
- **EDR** differs from classic antivirus by behavior, automation and response, not just signatures.
- **Least privilege** plus **posture assessment** plus **patching** are the standard mitigations; know which problem each one answers.
- Segmentation limits the **blast radius**; ACLs are the common way to enforce it.
- Hardening checklist: patch, change defaults, close ports, remove unused software, encrypt, host firewall and HIPS.

Check yourself

1. A laptop fails the antivirus-version check when it connects and lands on a restricted VLAN. Which mitigation is this? *(Configuration enforcement through a posture assessment.)*

2. Only approved programs may run on a kiosk. Allow list or deny list? *(Allow list.)*
3. An EDR agent spots odd behavior from a new process with no known signature and isolates the host. Which EDR capability? *(Behavioral analysis with an automated response.)*
4. A new printer is installed with its factory admin password. Which hardening step was missed? *(Changing default credentials.)*
5. Old retired drives held customer data and will not be reused. Best option? *(Physically destroy them.)*

Do next

From memory, list five hardening steps for a new server and name the threat each blocks. Then move to 3.1, where these controls are placed into whole architectures such as cloud, virtual and segmented networks.

3.1 Architecture models

Exam domain: 3. Security Architecture · **Videos:** Cloud Infrastructures (10 min) · Network Infrastructure Concepts (7 min) · Other Infrastructure Concepts (14 min) · Infrastructure Considerations (14 min)

Key ideas

Cloud infrastructures

- **Shared responsibility:** the provider and the customer split the security work, and the split depends on the service model. Providers publish a **responsibility matrix**, and a contract can change it. With SaaS and PaaS the provider handles the operating system; with IaaS (and on-premises) you do. The customer is always responsible for their own accounts and identities.
- **Hybrid cloud** combines more than one cloud deployment, usually public and private, often across several providers. It adds flexibility but also complexity: providers don't talk to each other, so authentication, server settings and firewall rules must be matched by hand, and logs come in different formats. Data moving between providers crosses the public internet, so it needs protection in transit.
- **Third-party vendors** (for example a firewall product placed in front of your cloud app) need a **vendor risk management** policy, a place in your incident response plan, and ongoing monitoring of their security and availability.
- **Infrastructure as code (IaC)** describes servers, web tiers and databases in code. You can build, change and rebuild an identical environment on any provider from the same definition.
- **Serverless architecture:** the application is split into small **functions** that run on demand and disappear when idle. You don't manage servers or care about the OS underneath, you pay for use, and most of the security work sits with the cloud provider and the code itself.
- **Microservices vs monolithic:** a monolithic app is one big program holding the interface, login and business logic, installed and updated as a unit. A microservice design splits those parts into separate services, reached through an **API gateway**. Busy parts scale on their own, one failed service doesn't take down the rest, and each service gets security that fits its job (authentication versus database access).

Network infrastructure concepts

- **Physical isolation / air gap:** devices on separate switches, with no link between them, can't be reached from each other if one is compromised. Examples: web servers and database servers on separate racks, or a managed service provider giving each customer its own switch. It doesn't scale (100 customers, 100 switches).
- **Logical segmentation with VLANs:** ports on one switch are placed in separate VLANs that can't talk directly, giving a similar effect with fewer devices.
- **Software-defined networking (SDN)** separates a network device into three planes:
  - **Data plane (infrastructure layer):** forwards traffic. NAT, encryption and trunking happen here.
  - **Control plane:** decides where traffic goes, with routing tables, dynamic routing updates and address tables.
  - **Management plane:** where administrators configure the device, through SSH, SNMP or APIs.

- Because each plane becomes software, a cloud design can add a load balancer, or firewalls between the internet and the web tier and between the web tier and database, with a click.

Other infrastructure concepts

- **On-premises vs cloud security:** on-premises gives full control and local tools, but you pay for staff, hardware, space and cooling, and new equipment takes time to buy and install. Cloud security is centralized, with no hardware to run, but you depend on the provider. Attackers go after either one.
- **Decentralized environments** (several sites, clouds and operating systems) are hard to secure, so teams build a **centralized view**: one console for alerts, logs, updates and monitoring. It improves visibility but becomes a **single point of failure** and needs more storage and CPU as the organization grows.
- **Virtualization:** a **hypervisor** on the hardware runs several virtual machines, each with its own guest OS. Identical guest systems repeat the same OS overhead.
- **Containers:** applications and their dependencies are packaged without an OS and share the **host operating system**, managed by container software such as Docker. They are lighter and quicker to swap, and remain isolated from each other, but apps must suit the host OS.
- **IoT (Internet of Things):** thermostats, lighting, doorbells, watches and sensors. Convenient, but built by companies that aren't security specialists, and one weak device can give an attacker a foothold in the network.
- **SCADA / ICS (industrial control systems):** connect factory, power and refinery equipment to a central control room. They must be tightly segmented from the outside, because an attack can cause real physical damage.
- **RTOS (real-time operating system):** a **deterministic** OS that guarantees tasks finish within fixed time limits, as in car braking, military and manufacturing equipment. Ordinary desktop and mobile systems are non-deterministic. Real-time systems tend to be closed and self-contained.
- **Embedded systems:** hardware and software built together for one purpose, such as traffic lights, smartwatches and medical monitors. No app store and little direct OS access, which also makes them hard to update.
- **High availability (HA)** keeps a service running through a failure, using automatic **failover**. It is a step beyond simple **redundancy**, where a spare exists but may need manual setup. Pairs can be **active/standby** or **active/active**. Each extra layer (second firewall, second network, second power feed) costs more, so it's a business decision how far to go.

Infrastructure considerations

- **Availability:** systems should be up, but only for the right people. Uptime is quoted as a percentage over time (99.999% is "five nines").
- **Resilience and recovery time: MTTR (mean time to repair)** measures how long it takes to restore something, and depends on finding the root cause, then replacing hardware, patching software or bringing in redundant parts.
- **Cost:** includes purchase and installation, maintenance and support, and replacement. Finance adds **depreciation** (a capital expense), operating costs and tax effects.

- **Responsiveness:** how quickly an application answers. It varies by function, because some transactions need more steps.
- **Scalability and elasticity:** apps are sized for the current load and grow when demand rises (end of quarter, peak hours), often automatically. **Elasticity** is how fast the footprint expands and shrinks. Security monitoring must extend to every added instance.
- **Ease of deployment:** one application is many parts (web server, database, cache, firewall). Plan hardware, budget, **change control**, people and location. In the cloud, **orchestration** builds the whole thing automatically.
- **Risk transference:** moving financial risk to someone else, for example **cybersecurity insurance** covering ransomware losses, downtime and legal costs. It doesn't stop the attack.
- **Ease of recovery:** plan for the fastest repair. Restoring a system image in about ten minutes beats reinstalling from media over an hour.
- **Patch availability:** patch, test, then deploy quickly. Some embedded devices (HVAC controls, time clocks) never get patches, so protect them with extra layers such as a firewall in front.
- **Power:** electricity underpins everything, on-premises or cloud. Have an electrician assess current and future needs, and plan a **UPS** or **generator** for outages.
- **Compute:** the processing resources for an application. In the cloud it's a compute engine that can scale from one processor to many.

Remember for the exam

- **Responsibility matrix:** IaaS puts the OS on you; SaaS and PaaS put it on the provider; **accounts and identities are always the customer's**.
- **SDN planes:** data = forward, control = decide, management = configure.
- **VM = guest OS per VM on a hypervisor; container = shared host OS.**
- **Serverless** means functions on demand, not "no servers exist".
- **Redundant** is not the same as **highly available**: HA fails over automatically.
- **SCADA/ICS and RTOS:** segment tightly. **Embedded** devices are hard to patch, so compensate with network controls.
- **Cyber insurance is risk transference**, not mitigation.
- A **centralized console** helps visibility but is a single point of failure.

Check yourself

1. A company runs apps at two cloud providers and finds firewall rules differ between them. Which hybrid-cloud problem is this? *(Settings must be matched manually because providers don't integrate.)*
2. An admin connects over SSH to change a switch setting. Which SDN plane? *(Management plane.)*
3. Which uses less overhead for ten identical Linux apps, VMs or containers, and why? *(Containers: they share one host OS, so no guest OS per app.)*
4. A building's HVAC controller can't be patched. What should you do? *(Add compensating protection, such as placing a firewall in front and segmenting it.)*
5. A firm buys a policy to cover ransomware downtime. Which concept? *(Risk transference.)*

Do next

Sketch a three-tier cloud application (load balancer, web, database) and mark where each plane, firewall and failover pair would sit. Then move to 3.2, which shows how to secure those networks with firewalls, IPS, proxies and secure protocols.

3.2 Securing enterprise infrastructure

Exam domain: 3. Security Architecture · **Videos:** Secure Infrastructures (6 min) · Intrusion Prevention (5 min) · Network Appliances (12 min) · Port Security (4 min) · Firewall Types (8 min) · Secure Communication (10 min)

Key ideas

Secure infrastructures

- Networks differ by industry, but the building blocks repeat: **firewalls** to separate segments, plus **honeypots**, **jump servers**, **sensors** and **load balancers** that add security or support the design.
- A **security zone** groups devices by purpose or access level rather than by subnet. The simplest design is **trusted** (inside) and **untrusted** (outside); finer designs add zones such as internet, servers, databases or a **screened subnet** (formerly called a DMZ) for publicly reachable services. Zone-based firewall rules ("untrusted to screened: allowed", "untrusted to trusted: blocked") are easier to read and maintain in a large rule base.
- The **attack surface** is every way in: application code, open ports, authentication, and human error such as one misconfigured firewall rule. Shrink it by auditing code, closing ports and watching traffic in real time.
- **Connectivity is part of the surface.** Protect cabling and wall jacks physically and logically, and use **application-level encryption** so a tap on the wire yields nothing readable. For remote sites and remote staff, use **IPsec site-to-site tunnels** and a **VPN concentrator**.

Intrusion prevention

- An **IPS** inspects traffic in real time and blocks exploits: known OS or application vulnerabilities, buffer overflows, SQL injection. An **IDS** only detects and alerts; it can't block.
- **Fail-open vs fail-closed** matters for any inline device. If it crashes, fail-open keeps traffic flowing with no inspection; fail-closed cuts the link. Which to choose depends on whether uptime or inspection matters more; check how each product behaves.
- **Active (inline) monitoring:** the IPS sits in the path (for example between firewall and core switch) and drops bad traffic before it arrives. Risks: an outage breaks connectivity, and an over-aggressive rule can block legitimate traffic.
- **Passive monitoring:** a copy of the traffic goes to the IPS, so it cannot cause downtime but also cannot stop anything in real time. It behaves like an IDS. The copy comes from a **port mirror / SPAN (Switch Port Analyzer)** or a physical **network tap**.

Network appliances

- A **jump server** is a hardened host reachable from outside that lets authorized admins reach internal devices in two steps (connect to the jump server, then SSH on to the target). If it is compromised, the inside is exposed, so harden it heavily.
- A **proxy** makes requests on behalf of a client. Uses: **caching**, **URL filtering**, and **content scanning** of responses. Types:

    - **Explicit** (clients are configured with its address) vs **transparent** (users don't know it's there).
    - **NAT** is the simplest proxy-like function; an **application proxy** understands protocols such as HTTP, HTTPS or FTP.
    - **Forward proxy:** internal users reaching the internet. **Reverse proxy:** outside users reaching an internal service; it can drop malicious input and cache responses to lighten the web server.
    - **Open proxy:** public and unmanaged. Users employ it to bypass controls, and the unknown owner could inject ads or malware, so organizations block it.
- A **load balancer** spreads traffic across a farm of servers and gives **fault tolerance**: when one server fails, the others take over quickly. Features include **TCP offloading**, **SSL/TLS offloading** (decrypt at the balancer, re-encrypt on the way back), caching, prioritization and **content switching** (send certain requests to specific servers). Configurations: **active/active** (all servers used) and **active/passive** (standby servers promoted on failure).
- **Sensors and collectors:** switches, routers, firewalls, IPS devices, authentication logs and web server logs feed statistics to a central **collector**. A **SIEM** collects from many device types and also correlates, compares and reports on events (failed logins, port scans).

Port security

- **Port security** controls who can use a switch port or wireless connection. You meet it when a network asks for credentials before it grants access.
- **EAP (Extensible Authentication Protocol)** is the framework for the login exchange. **802.1X** (port-based network access control, NAC) applies it: a device plugged into a switch gets no network access until it authenticates. It works with back ends such as RADIUS, LDAP, TACACS+ and Kerberos.
- Three roles: the **supplicant** (client), the **authenticator** (switch or access point) and the **authentication server** (the credential database). The authenticator blocks everything while it relays EAP requests and responses between the other two. When the server accepts the credentials, the authenticator opens the port.

Firewall types

- A firewall controls traffic between two points. It filters by rule, can restrict websites (corporate policy, parental controls) and can host extra controls such as antivirus scanning.
- **Network firewall:** a dedicated appliance. Traditional ones decide by layer 4 (TCP/UDP port). Many also act as layer 3 routers, with NAT and routing, and often terminate VPNs.
- **UTM (unified threat management):** an older all-in-one box with firewall, URL filtering and content inspection, malware and spam filtering, IDS/IPS, VPN concentrator and bandwidth shaping. Turning many features on slows it down, and many only look at layer 4.
- **NGFW (next-generation firewall):** works at layer 7 with deep packet inspection, so it recognizes the **application** and the user, not just the port. For example, it can allow one application while blocking another that uses the same port, or let people view a site but not post. It often includes IPS signatures and **URL category** filtering (block "gambling" or a named site).
- **WAF (web application firewall):** inspects HTTP/HTTPS input to a web app and blocks attacks such as SQL injection and cross-site scripting. It is commonly used alongside an NGFW, and PCI DSS calls

for it for card-handling apps.

Firewall	Looks at	Typical job
Traditional network	Layer 3/4 (IP, port)	Allow or block by port, routing, NAT
UTM	Often layer 4	Many security features in one box
NGFW	Layer 7 (application, user)	App control, IPS, URL categories
WAF	Web requests (HTTP/S)	Stop SQL injection, XSS against one web app

Secure communication

- **VPN:** an encrypted tunnel across a public network, ending at a **VPN concentrator** (often built into a firewall; can be software). Client software, often built into the OS, authenticates to it, and the concentrator decrypts traffic into the corporate network.
- **IPsec tunnel:** the original IP header and data are encrypted, then wrapped with an IPsec header and trailer plus a new IP header addressed to the far concentrator. The far end strips and decrypts.
- **SSL/TLS VPN:** runs over TCP 443, so it passes most firewalls. Used for **remote access** from a single device, via a client or even a browser, with ordinary credentials (certificates optional). It can be **always-on**.
- **Site-to-site VPN:** commonly IPsec between firewalls or concentrators at each location, invisible to the users behind them.
- **SD-WAN:** software-defined wide area networking built for apps that live in several clouds. Sites reach cloud services directly instead of hauling everything through a central data center.
- **SASE (Secure Access Service Edge):** the cloud-delivered "next generation of VPN". Security services sit in the cloud next to the services, and clients on every device (office, home, mobile) connect through them.
- Organizations often combine them: SSL VPN for users, IPsec for sites, SD-WAN for cloud connectivity and SASE for security over it.

Remember for the exam

- **IPS = inline, blocks; IDS = passive, alerts.** An IPS fed by a mirror or tap acts like an IDS.
- **Fail-open** keeps the network up; **fail-closed** keeps it secure but cuts the link.
- **802.1X roles:** supplicant, authenticator, authentication server. **EAP** is the framework.
- **NGFW = layer 7 and applications; WAF = web attacks (SQLi, XSS); UTM = all-in-one.**
- **Forward proxy = outbound, internal users; reverse proxy = inbound, protects servers.**
- **IPsec = site-to-site; SSL/TLS (TCP 443) = remote access.**
- Load balancer: **active/active** vs **active/passive**; it also offloads TLS.
- A **jump server** is a high-value, heavily hardened target.

Check yourself

1. A security device must stop exploits before they reach servers, even if it adds a possible point of failure. Which placement? *(IPS inline, active monitoring.)*
2. A laptop plugs into a switch port and gets nothing until it sends credentials. Which standard and which device plays the authenticator? *(802.1X with EAP; the switch is the authenticator.)*
3. A web shop must defend its checkout app against SQL injection to meet PCI DSS. Which firewall type? *(WAF.)*
4. Branch offices should connect to HQ automatically with no client software. Which VPN? *(IPsec site-to-site between firewalls.)*
5. Admins outside the network need a controlled way to manage internal servers. What do you deploy? *(A hardened jump server.)*

Do next

Draw a network with an internet link, firewall, screened subnet and internal zone, then place an IPS, load balancer, reverse proxy and VPN concentrator, and say whether each is inline. Then move to 3.3, which turns to the data itself: how to classify and protect it.

3.3 Protecting data

Exam domain: 3. Security Architecture · **Videos:** Data Types and Classifications (6 min) · States of Data (6 min) · Protecting Data (14 min)

Key ideas

Data types and classifications

- **Regulated data** is controlled by an outside party. Examples: payment card data under PCI DSS, and data covered by government laws that also set how long it must be kept.
- **Trade secrets** are processes and know-how known only to the organization, so they need strong protection.
- **Intellectual property** is often visible to others, so it is protected mainly by law (copyright, trademark, patents).
- **Legal information** is partly public (court records, names of judges and attorneys) but can contain private details such as PII, which is held in separate systems or formats.
- **Financial information** is sensitive for companies and individuals alike (internal results, account numbers, payments).
- **Human-readable vs non-human-readable data:** a spreadsheet is readable; encoded data such as a barcode or QR code isn't. Some formats mix both, such as a barcode with its digits printed below.
- **Proprietary data** is information an organization created or gathered that is unique to it.
- **PII (personally identifiable information)** ties data to a person: name, date of birth, mother's maiden name, biometrics, address.
- **PHI (protected health information)** is health status, health records and payments for care, tied to an individual.
- **Classification levels** set who can see what and how. Different data gets different sensitivity: a license plate lookup is not a medical history. Typical labels:
  - **Public / unclassified:** anyone may see it.
  - **Sensitive:** could include intellectual property, PII or PHI.
  - **Confidential:** more sensitive; needs additional access.
  - **Private, restricted or classified:** needs extra rights, possibly a non-disclosure agreement, sometimes a restricted network area.
  - **Critical:** must stay available, so plan for uptime.

States of data

- **Data at rest** is stored on a drive, SSD or flash media, whether or not it is encrypted. Protect it with full disk encryption, database or column encryption, file or folder encryption, and permissions assigned to users and groups.
- **Data in transit (in motion)** crosses a network. Without encryption anyone tapping the link can read it. Protect it with a firewall or IPS policy, **TLS** for web traffic and a **VPN** such as IPsec site-to-site.

- **Data in use** sits in RAM or the CPU while being processed, usually decrypted, which makes it an attractive target. In the 2013 Target breach, attackers put malware on point-of-sale terminals and read card data from memory, where it was decrypted for processing: about 40 million card numbers, plus personal details of some 70 million customers.
- **Data sovereignty:** data stored in a country falls under that country's laws and court orders. Some rules require data to stay within a region. **GDPR** (EU) governs personal data of EU residents and has requirements about where that data may go. Multinational organizations must check each location's rules.
- **Geolocation** finds where a user or device is, through GPS, 802.11 (Wi-Fi) information or mobile carrier data. Access can then depend on location, as with region-locked streaming, or full access inside the office and less outside.

Protecting data

- **Geographic restrictions** limit access by place. An IP subnet works on a known internal network but not for mobile devices, which need geolocation (GPS, or matching visible Wi-Fi SSIDs against a database). **Geofencing** applies policy to a defined area, for example sensitive data available only inside company facilities.
- Data lives on devices, on networks and in memory, so protection must cover every location: encryption, network policy and permissions.
- **Encryption** turns **plaintext** into **ciphertext**; the right algorithm and **key** reverse it. A good result looks nothing like the original, a property called **confusion**.
- **Hashing** turns data into a fixed-length **digest** (a "fingerprint") that cannot be reversed. Uses: storing passwords, checking a downloaded file's **integrity** against the published hash, and **digital signatures** (hash plus public key cryptography, giving authentication and integrity). A **collision** is two different inputs with the same hash; algorithms prone to it are retired. SHA-256 produces 256 bits (64 hex characters), and changing one character gives a completely different output.
- **Obfuscation** makes something hard for humans to understand while it works the same. Developers use it on code; attackers use it to hide malicious scripts.
- **Masking** hides part of the data, such as a receipt showing only the last four card digits. Techniques include asterisks, shuffling and encryption.
- **Tokenization** replaces a sensitive value with a different, unrelated token. No encryption or hashing is involved, and the value cannot be derived from the token. In mobile payments, your phone holds one-time tokens issued by a token service; the store passes one on for validation, and a captured token can't be replayed.
- **Segmentation:** don't keep all data in one big database. Split it so a breach exposes less, and give each store security that matches its sensitivity.
- **Permission restrictions:** safe authentication (password policy, multiple factors) plus group and file permissions limiting what each account can reach.

Technique	Reversible?	Typical use
Encryption	Yes, with the key	Protect stored or transmitted data

Technique	Reversible?	Typical use
Hashing	No	Passwords, integrity checks, signatures
Masking	No (original kept elsewhere)	Display partial card or ID numbers
Tokenization	Only by the token service	Payments, replacing SSNs or card numbers

Remember for the exam

- **At rest, in transit, in use:** know the protection for each. **Data in use** is the hardest to encrypt, because it's decrypted in memory.
- **PII vs PHI:** PHI is health-related personal data (HIPAA territory).
- **Hash = integrity, not confidentiality;** encryption = confidentiality.
- **Tokenization replaces the value; masking hides part of it; obfuscation hides meaning.**
- **Data sovereignty** is about which country's laws apply; **geofencing** is about where the user is.
- **Classification** drives handling: public, sensitive, confidential, restricted, critical.
- A **collision** weakens a hash algorithm.

Check yourself

1. A phone payment sends a one-time value instead of the card number. Which technique? *(Tokenization.)*
2. A receipt prints only the last four digits of a card. Which technique? *(Data masking.)*
3. Malware scrapes card numbers from a point-of-sale terminal's RAM. Which data state was attacked? *(Data in use.)*
4. A Linux download page lists a SHA-256 value. What does comparing it prove? *(File integrity: it wasn't changed or corrupted.)*
5. An EU company must keep customer data within the EU. Which concept? *(Data sovereignty: the data stays under EU law.)*

Do next

Take five kinds of data from an organization you know, assign each a classification and a state, and name one control for it. Then move to 3.4, which covers keeping that data and the systems around it available: backups, redundancy and recovery.

3.4 Resilience and recovery

Exam domain: 3. Security Architecture · **Videos:** Resiliency (10 min) · Capacity Planning (4 min) · Recovery Testing (5 min) · Backups (12 min) · Power Resiliency (4 min)

Key ideas

Resiliency

- **Redundancy vs high availability:** having a spare component in a box is redundancy, but it takes time to rack, configure and power on. **High availability (HA)** runs the second system alongside the first, already on, ready to take the load. It costs more: duplicate hardware, better power and higher-grade parts.
- **Server clustering:** several servers work as one logical server. Users see a single system, nodes can be added or removed live, nodes usually run identical operating systems, and they use **shared storage** so each node sees current data.
- **Load balancing:** a central balancer distributes requests among servers that don't know about each other. They can run different operating systems. The balancer adds capacity when you add servers and removes a failed server from rotation automatically. Difference to remember: cluster nodes are aware of one another; load-balanced servers are not.
- **Site resiliency:** a recovery site holds synchronized data so operations can move there during a disaster, for hours or weeks, and return afterward.
  - **Hot site:** an exact replica of the data center, with matching hardware, current applications and continuously synchronized data. Fastest, most expensive.
  - **Cold site:** an empty building with power and space. You bring the equipment, data and staff. Cheapest, slowest.
  - **Warm site:** in between. Some equipment and some data are present, and you bring the rest.
  - **Geographic dispersion:** the recovery site should be far enough away that one regional event (hurricane, flood) can't hit both. Distance makes moving equipment and staff harder.
- **Platform diversity:** use different operating systems for different roles (Linux and Windows in the data center, macOS and Windows on clients), because a vulnerability usually affects one platform, not all.
- **Multi-cloud systems:** use more than one cloud provider so an outage or security problem at one doesn't stop the service. Services and data can sit in several locations.
- **Continuity of operations planning (COOP):** when technology isn't available at all, fall back to non-technical methods, such as paper receipts, manual sign-off or phoning in card approvals. Design these procedures before a disaster.

Capacity planning

- Balance **supply and demand**. Too little means slowdowns and outages; too much wastes money. Plan across **people**, **technology** and **infrastructure**.
- **People:** hiring and training are slow, and reducing staff is costly, so staffing is harder to scale than machines (for example, a call center).

- **Technology:** choose things that scale. Web servers behind a load balancer let you add or remove servers invisibly. Databases can be split across several SQL servers. Cloud services can be added on demand, at a price that grows with use.
- **Infrastructure:** a physical build means buying, shipping, racking and testing hardware; the cloud means a few clicks. Scale up for peaks and **right-size** afterward.

Recovery testing

- A **disaster recovery plan** must be tested regularly, with a defined scope, a specific scenario and a set time frame, without hurting production. Afterward, evaluate the results and update the plan.
- **Tabletop exercise:** a group talks through the recovery steps without moving anything. Cheap, and good at exposing missing steps and cross-team gaps.
- **Failover test:** checks that redundant components take over, ideally automatically and unnoticed by users. Think redundant internet links, routers, firewalls, switches and server links, plus load balancers.
- **Simulations:** exercise the people and tools, for example a **phishing simulation** that shows whether filters catch the message and who clicks, so further training can be assigned. Other simulations cover password resets or data exfiltration alerts.
- **Parallel processing:** spreading work across multiple CPUs, cores or machines is faster and tolerates the loss of one processor.

Backups

- **Planning questions:** how much data, which backup type, which media (tape, disk, cloud), where it is stored, what software, and what schedule (full each week with daily changes, for instance).
- **On-site backup:** local, fast to restore, cheaper, no WAN link. **Off-site backup:** shipped media or a network copy, which survives a site disaster. Many organizations use both: on-site for fast restores, off-site for long-term storage.
- **Frequency and retention:** daily, weekly or monthly sets (about 30 daily, 4 weekly or 12 monthly copies over time). Rarely changing servers need fewer backups.
- **Encrypt backups.** They hold sensitive data, and media can be stolen in transit (tapes taken from an employee's car). Keep the recovery keys safe, since you need them to restore. Cloud backups should effectively always be encrypted.
- **Snapshots:** point-in-time copies of a virtual machine or cloud instance, made in a click and used before risky changes or daily. They store only the changed data after the first one, like an incremental backup, so a daily snapshot of a 100 GB VM adds roughly the day's changes.
- **Test restores** regularly, with different methods and different sources. A restore isn't finished until the application can use the data or the system boots from a full backup.
- **Replication:** copies data to one or more places in near real time, so a current copy always exists elsewhere. It suits hot-site recovery.
- **Journaling:** writes changes to a journal before the final write. If power fails mid-write, the system replays the journal at startup and repairs the damage; once both writes succeed, the journal entry is cleared.

Backup type	What it saves	Restore
Full	Everything	One set
Incremental	Changes since the last backup of any kind	Full plus every incremental
Differential	Changes since the last full backup	Full plus the latest differential

(The video doesn't cover these three types; they're useful background because practice questions often use them.)

Power resiliency

- Most data centers take power from a utility they don't control, so plan for outages from minutes to days.
- **UPS (uninterruptible power supply):** battery backup for short outages. It covers **blackouts**, **brownouts** (low voltage) and **surges** (excess voltage).
  - **Offline / standby:** runs on mains and switches to battery on failure; the cheaper type.
  - **Line-interactive:** can adjust voltage upward during brownouts.
  - **Online / double-conversion:** always runs from the battery path, with no switching.
  - Choose capacity for the runtime you need, and prefer units that signal servers to **shut down gracefully** as the battery drains.
- **Generator:** long-term power while fuel lasts, for a whole building or marked outlets. It takes about a minute to start, so organizations use **both**: the UPS bridges the gap until the generator takes over.

Remember for the exam

- **Hot > warm > cold** in speed, readiness and cost. Hot = exact replica, constantly synchronized.
- **Cluster:** nodes know each other and share storage. **Load balancer:** servers independent, balancer in control.
- **Tabletop exercise** is a discussion, not a technical recovery. **Failover test** is technical.
- **Snapshot** = VM/cloud point-in-time copy; **replication** = near real-time copy to another place; **journaling** = protects against corruption on power loss.
- **Encrypt backups,** especially off-site and cloud, and keep the keys.
- **UPS** for short outages and clean transitions; **generator** for long ones.
- **Platform diversity and multi-cloud** reduce shared-vulnerability and single-provider risk.
- **COOP** is the non-technical fallback.

Check yourself

1. A business wants a site with matching hardware and live-synchronized data that it can switch to within minutes. Which site? *(Hot site.)*
2. A team walks through the disaster plan in a meeting room with no systems touched. What is it? *(Tabletop exercise.)*
3. Power drops mid-write and the database is left half-updated. Which feature repairs it at startup? *(Journaling.)*

4. A data center needs power that bridges the minute before the generator starts. What provides it? *(A UPS.)*
5. Why store the recovery keys for encrypted backups separately and safely? *(Without them the backups can't be restored, and if stolen with the media they defeat the encryption.)*

Do next

Write a one-page recovery plan for a small shop: pick its recovery site type, backup schedule and locations, and the power setup, then list how you would test each part. Then move to 4.1, the start of Domain 4, where you harden the systems themselves.

4.1 Securing computing resources

Exam domain: 4. Security Operations · **Videos:** Secure Baselines (4 min) · Hardening Targets (10 min) · Securing Wireless and Mobile (9 min) · Wireless Security Settings (11 min) · Application Security (8 min)

Key ideas

Secure baselines

- A **security baseline** is the agreed set of security settings for one deployment: firewall rules, patch level, OS configuration and application settings. Every new instance should be built with the same baseline, and you then keep checking that it still matches.
- **Where baselines come from:** the application developer (file permissions, app settings), the OS vendor (Microsoft publishes Windows and Windows Server baselines and a Security Compliance Toolkit to apply them), and the makers of purpose-built appliances. Treat these as a starting point and adjust them to your organization.
- **Deploying at scale** needs automation: Active Directory group policy for Windows, an MDM for phones and tablets, or a central console. A baseline rarely changes, but it must be revised after a new vulnerability, an application update or a new OS. When two vendors' baselines conflict, you choose the better one for that instance.
- **Test before deployment, audit after.** Auditing is how you find systems that have drifted from the baseline, so you can fix them.

Hardening targets

- **Hardening** means tightening a default install, because defaults are rarely secure. Vendors and third parties publish hardening guides for specific systems.
- **Mobile devices:** install updates, and keep business data in a separate segment from personal data so a compromise of one does not expose the other. An MDM monitors and pushes updates.
- **Workstations:** patch the OS, applications and firmware. Vendors often bundle patches into a monthly release, which lets you test them together. Remove software you don't use, since every program is a possible entry point.
- **Network infrastructure (switches, routers, firewalls):** these run embedded operating systems. Change default credentials, use local or central authentication, and watch for vendor patches. They are rare, so each one matters.
- **Cloud infrastructure:** secure the cloud management workstation (it has broad access), apply least privilege to services, apps and network settings, run EDR, and back up, ideally to a second provider.
- **Servers:** apply OS patches and service packs, enforce minimum password length and complexity, give accounts least privilege, disable unused accounts, restrict which hosts may reach the server, and run EDR or anti-malware.
- **SCADA/ICS:** systems that manage, monitor and control industrial equipment such as power generation or manufacturing. They are usually isolated on their own network behind an **air gap**, with no internet access.

- **Embedded systems** (smart TVs, watches, appliances) have a locked-down OS, are hard to update, and get patches rarely. Install any patch promptly and consider a segmented network with a firewall in front.
- **Real-time operating system (RTOS):** deterministic, so every task must finish within a guaranteed time. It is used in industrial, military and automotive equipment. Isolate it, run only the services it needs, and use a separate firewall or host-based protection if it must talk to the network.
- **IoT** devices (lighting, heating and cooling) come from makers who aren't security specialists. Prioritize their patches and put them on their own segment so a compromised device can only reach other IoT devices.

Wireless and mobile deployment

- A **site survey** shows which access points are already in the area (yours and others'), their channels and the frequency use, so you can choose channels and placement. Repeat it periodically because neighboring networks change.
- A **heat map** is a floor-plan view of signal strength, so you can see weak rooms. **Wireless survey tools** list SSIDs, BSSIDs, channels and bands; NetSpot is one example. A **spectrum analyzer** shows every signal on a frequency, including non-Wi-Fi devices causing interference.
- **MDM (mobile device management)** lets an administrator push policies, require apps, enforce screen locks and PINs, switch features off in certain places (for example the camera on company premises), and keep business and personal data in separate partitions.
- **Deployment models:**
  - **BYOD** (bring your own device): the employee owns the phone, and it must still meet the standards needed for MDM enrollment. Plan for privacy of personal data and for wiping or re-enrolling when the person trades the phone in.
  - **COPE** (corporate owned, personally enabled): the company buys and assigns the device, and personal use is allowed.
  - **CYOD** (choose your own device): the company owns it, but the employee picks from an approved list.
  - On corporate and BYOD devices, partitioning lets the admin delete the business partition without touching the rest.
- **Cellular risks:** phones are portable, hold lots of data and are easy to conceal. Traffic can be monitored and location tracked, so patching is critical.
- **Wi-Fi risks:** unencrypted public networks (use a VPN), eavesdropping, **on-path** attacks, and radio interference used as a denial of service.
- **Bluetooth** is a personal area network (PAN). Pairing exists so a stranger's device can't reach your data, so never auto-connect to unknown devices.

Wireless security settings

- Wireless needs three things: **encryption** (so sniffed frames are unreadable), **integrity** (a message integrity check, MIC) and **authentication** of users.

- **WPA2** uses a four-way handshake at connection time. An attacker can capture the hash from it and brute-force the pre-shared key offline, and GPUs or cloud cracking make that fast.
- **WPA3** fixes this. It uses **GCMP** (Galois/Counter Mode Protocol), a stronger cipher mode that provides both confidentiality and integrity (via a Galois message authentication code). It replaces the handshake with **SAE (simultaneous authentication of equals)**, a Diffie-Hellman-based exchange also called the **dragonfly handshake**. Both sides prove knowledge of the password, session keys are derived locally and never sent, and every user gets a different session key, so users on the same network can't read each other's traffic.
- **Two ways to authenticate:**
  - **Pre-shared key (PSK):** one shared password. Fine at home, poor at work. WPA3-Personal.
  - **802.1X:** each person logs in with their own credentials, checked by a central server. WPA3-Enterprise.
  - An **open** network (shown as "none") has no authentication or encryption.
- **AAA** is the framework behind a central authentication server: **authentication** (who you are), **authorization** (what you may use) and **accounting** (a record of your session, such as login times and data volume). Common AAA servers use **RADIUS**, LDAP or TACACS+. RADIUS is very widely supported and checks logins for VPNs, switches, routers, servers and Wi-Fi. Disabling one account in the central server removes access everywhere.
- **802.1X** is port-based network access control (NAC), for wired or wireless. It involves three roles: the **supplicant** (the client), the **authenticator** (the device you connect to, such as an access point or switch) and the **authentication server**. **EAP (Extensible Authentication Protocol)** is the framework that carries the credential exchange; vendors can extend it. The authenticator blocks the supplicant until the server approves.

Application security

- **Input validation:** the application checks every input against what it expects (length, character set, format, such as a postal code) and asks the user to correct bad data. Do it for every field.
- **Fuzzing (dynamic analysis):** automated tools throw random or malformed input at an application to see if it crashes, errors or misbehaves, revealing where validation is missing. Also called fault injection or robustness testing. The first fuzzer came from a 1988 university class; CERT's Basic Fuzzing Framework is one you can try.
- **Secure cookies:** a cookie is just a data file, not code, but it can reveal information. The **Secure** attribute makes the browser send it only over HTTPS. Because cookies are readable in the browser, never store sensitive data in them.
- **Static code analysis (SAST):** a tool reads source code looking for flaws such as buffer overflows (for example, using `gets` instead of `fgets`) and injection. It misses problems that only appear when the code runs, such as weak cryptographic implementation, and it reports false positives that a developer must review.
- **Code signing:** the developer signs the software with a key that a certificate authority has vouched for. The OS checks the signature at install time to confirm the code is unchanged and really from that developer, and warns you if not.

- **Sandboxing:** the application can reach only the data it needs. Developers also build in isolated sandboxes, away from production. VMs and mobile app sandboxes work the same way: a phone browser can't read your camera roll.
- **Application monitoring:** built-in logging lets developers spot SQL injection attempts, exploitation of old flaws and unusual activity like odd file transfers or a jump in client access.
- **Installing packages:** get them from the vendor, not a third party that could add malware. If unsure, test in a lab first.

Remember for the exam

- **WPA2 vs WPA3:** WPA2's four-way handshake exposes the hash to offline brute force; WPA3 uses **SAE** (no hash sent), mutual authentication and **GCMP**.
- **PSK vs 802.1X:** Personal = shared password; Enterprise = individual credentials through RADIUS/LDAP/TACACS+ and EAP.
- **Supplicant, authenticator, authentication server:** the client, the AP or switch, and the AAA server.
- **BYOD, COPE, CYOD:** who owns the device and who chooses it. BYOD = employee owns; COPE and CYOD = company owns.
- **Segmentation and air gaps** protect SCADA/ICS, embedded, RTOS and IoT devices that can't be patched quickly.
- **SAST** analyzes code without running it; **fuzzing** runs the app with random input; **code signing** proves origin and integrity.
- **Baseline cycle:** establish, deploy, maintain, test and audit for drift.

Check yourself

1. Which WPA2 weakness does WPA3 remove, and how? *(Capturing the four-way handshake lets attackers brute-force the PSK offline; WPA3's SAE never sends a hash across the network.)*
2. A company buys phones and lets staff choose from three models, also allowing personal use. Which model? *(CYOD, also personally enabled; the key point is that it's corporate owned.)*
3. A factory control network must stay running and can't be patched often. What should you do? *(Isolate it, ideally with an air gap or tight segmentation, with no internet access.)*
4. A scanner flags a flaw in code, but you aren't sure it's real. What is this called, and what do you do? *(A possible false positive; a developer reviews the finding.)*
5. Which control shows software is unchanged since the vendor released it? *(Code signing.)*

Do next

Write the steps of an 802.1X login (supplicant, authenticator, authentication server) from memory, then list one hardening step for each device type above. Next, 4.2 covers tracking and retiring the assets you have just learned to secure.

4.2 Asset management

Exam domain: 4. Security Operations · **Video:** Asset Management (9 min)

Key ideas

- **Acquisition and procurement** is the formal path for getting hardware, software and services. A user states a need, IT and purchasing check it against the budget, managers approve, and the buyer negotiates price, license terms and contract details with the supplier. The supplier delivers and then sends an invoice, which may be due immediately or after a set period such as 30 or 60 days.
- **Asset tracking system:** once goods arrive, they are recorded centrally and followed for their whole life. Each record names an **owner** (a laptop is assigned to a person, so you know whom to visit to service it), a **type** (laptop, desktop, phone) and a **class**: hardware or software.
- **Hardware vs software for accounting:** hardware is usually a capital expense that depreciates over time, while software (especially subscriptions) is often treated as an operating expense, so the two can be taxed differently.
- **Inventory and help desk:** the system lets you list what you own and where it is, and lets a technician open a ticket already filled with the user and the device's make and model.
- **Enumeration:** a device is a single asset but also a set of parts (CPU, memory, drives, peripherals). The system can show either level.
- **Asset tags** (a number or barcode attached to the device) tie the physical item to its record and help identify it if lost or stolen.
- **Media sanitization:** before a device with storage is reused or retired, the data on it must be handled to match what happens next.
  - **Reuse inside the organization:** securely erase the data with a utility so it can't be recovered, then the drive can go back into service.
  - **Disposal or recycling:** remove everything, or destroy the drive.
- **Destruction:** shredding or pulverizing, drilling or hammering through the platters, **degaussing** (a strong magnetic field wipes the data and ruins a hard drive), and incineration. Large volumes go to a third-party destruction company, which provides a **certificate of destruction** as proof.
- **Data retention:** rules or regulations may require keeping certain data (emails or financial records for a number of years), including originals, copies and backups. Keep written policies and procedures that cover which data, how long and how it's stored. Retention also supports good practice: restoring accidental deletions and recovering at another site after a disaster. Different data needs different retention periods.

Remember for the exam

- **Asset tracking covers the whole life cycle:** acquisition, ownership, inventory, tagging, then disposal.
- **Sanitize to match the plan:** reuse means secure erase; permanent disposal means destruction.
- **Degaussing** works on magnetic media (hard drives) and makes them unusable.
- **Certificate of destruction** is the proof you get from a third-party destruction vendor.
- **Retention** is driven by legal and regulatory requirements; deletion that breaks them is a problem too.

- **Ownership** in the tracking system tells you who is responsible for each asset.

Check yourself

1. A department is donating old laptops internally to another team. Which sanitization fits? *(Secure erase, so the drives can be reused.)*
2. What proves to auditors that a vendor destroyed 2,000 drives? *(A certificate of destruction.)*
3. Why does the tracking system record whether an item is hardware or software? *(Hardware is usually a depreciable capital expense and software often an operating expense, so the accounting and tax treatment can differ.)*
4. A regulation requires keeping financial records for several years. Which policy area applies? *(Data retention.)*

Do next

List the steps an asset takes from purchase request to destruction, noting what record or proof exists at each step. Next, 4.3 covers finding and fixing the weaknesses in those assets.

4.3 Vulnerability management

Exam domain: 4. Security Operations · **Videos:** Vulnerability Scanning (8 min) · Threat Intelligence (5 min) · Penetration Testing (6 min) · Analyzing Vulnerabilities (10 min) · Vulnerability Remediation (9 min)

Key ideas

Vulnerability scanning

- A **vulnerability scan** checks whether a system could be attacked; it doesn't exploit anything (that would be a penetration test). A **port scan** is the simplest example: it shows open and closed ports, and an open port is a possible route, not proof of a weakness.
- Scan from outside to see what an outside attacker sees, and from inside to cover insiders. Scanners are good at discovering every host in a subnet, including forgotten ones, such as an unsupported Unix system nobody remembered.
- Reports are long and sorted by **severity**: critical, high, medium, low, informational. Each finding needs checking before you act. Confirmed high-severity findings go through **change control** and get patched. An example finding is a weak random number generator in OpenSSL that makes SSH host keys weak.
- **Static application security testing (SAST)** reviews source code for flaws such as buffer overflows and injection. It can miss issues in how authentication or cryptography is used, and its findings also need review for false positives.
- **Dynamic analysis (fuzzing)** feeds random input into a running application and watches for crashes, errors or exceptions. It is automated because thousands of attempts may be needed.
- **Package monitoring:** verify downloaded packages come from the vendor, and test unknown ones in a lab before production.

Threat intelligence

- **Threat intelligence** is knowledge of current threats and the actors behind them, used to decide which tools, training and defenses you need. Researchers and IT teams both use it.
- **OSINT (open-source intelligence)** is public information: forums, social media, researcher posts, government reports and hearings, and public company filings.
- **Proprietary or third-party intelligence** is sold by companies that see many customers at once, so they can spot attack trends early and warn others.
- **Information sharing:** groups like the **Cyber Threat Alliance (CTA)** collect submissions, put them in a standard format, validate and score them, and distribute them to members.
- The **dark web** is an overlay network reachable only with special software. Watching its forums and markets shows attacker tools and techniques, and whether your organization's name or stolen data is being traded.

Penetration testing

- A **penetration test** simulates an attack and actually exploits vulnerabilities to see whether access can be gained. Regular tests are good practice and sometimes required.
- **Rules of engagement** are written before testing: when tests may happen (for example, only after business hours), what types are allowed (physical, internal, external), which IP ranges and systems are **in scope** and which are **out of scope**, emergency contacts, and how to handle any sensitive data found. They protect critical systems because exploits can crash a service.
- **Typical techniques:** password brute force, social engineering, injection and buffer overflows (used, for example, to gain privilege escalation).
- **After the initial foothold: lateral movement** (moving from device to device), **persistence** (a way back in that doesn't rely on the original flaw, such as a new account, changed passwords or a backdoor) and **pivoting** (using the compromised system as a relay or proxy to reach internal systems that firewalls hide from outside).
- **Disclosure and bug bounties:** a researcher reports a flaw to the vendor, who fixes and tests it, and only then is the flaw made public, often in the CVE list. That may take weeks or months. A **bug bounty** is a reward paid for finding and reporting a vulnerability.

Analyzing vulnerabilities

- **False positive:** the scanner reports a flaw that doesn't exist. Low or informational findings are still real; they are not false positives.
- **False negative:** a real flaw is not reported. This is usually worse, because you don't know to fix it. Update scanner signatures before each scan to reduce both.
- **CVE and NVD:** the **CVE** list gives each public vulnerability a standard identifier. The **National Vulnerability Database** (nvd.nist.gov) adds scores and details. Cross-reference scanner findings there and with the vendor's own advisories.
- **CVSS (Common Vulnerability Scoring System):** scores a vulnerability from 0 to 10, with 10 the most severe. Scores vary by version, such as 2.0 and 3.x.
- **Scanner scope:** scanners cover desktop and mobile applications, web applications and network devices (firewalls, switches, routers).
- **Exposure factor:** the percentage of an asset's value or function that would be lost if the flaw were exploited. A flaw that could knock a service out half the time is a 50% exposure factor; one that could fully disable a public, unpatched service is 100%.
- **Prioritization considers the environment:** an internet-facing cloud system outranks an isolated lab machine. Also weigh the number and type of users, internal versus external exposure, whether the system is critical or revenue-generating, and how easy the flaw is to exploit. Impact depends on the organization: a hospital or a power utility suffers more from the same outage than most firms.
- **Risk tolerance:** how much unpatched risk the organization accepts. Testing a patch leaves systems exposed, so a widespread, easily exploited flaw means low tolerance and faster, shorter testing.

Remediation

- **Patching** is the main fix. Patches come on a schedule (weekly or monthly) or as an unscheduled emergency release for a **zero-day** being exploited. The cycle of testing and deploying never ends.

- **Cybersecurity insurance** transfers part of the financial risk (losses, recovery, lawsuits). It doesn't cover everything, for example intentional acts or funds transfers.
- **Segmentation:** separate devices onto their own network or VLAN to limit what an attacker can reach. **Air gap** means physically separate networks with no link. A VLAN separates traffic logically on one switch, and only a router can connect VLANs. A next-generation firewall can watch the flows between segments by application.
- **Compensating controls** when you can't patch: disable the vulnerable service, revoke access to the application, block it at the edge firewall, use router ACLs, or add a host-based firewall. Patching is still best.
- **Exceptions and exemptions:** a change-control or security committee may decide a system will go unpatched, for example because the flaw needs local access to a server in a secured data center. The decision is made by a group, weighing risk against availability, and documented.
- **Validation:** rescan after patching to confirm the flaw is gone and to catch systems you missed. Audit a sample of systems, and use reporting (counts of identified flaws, patched versus unpatched, new threats, failed patches, exceptions and exemptions) to track the work, which is essential at scale.

Remember for the exam

- **Vulnerability scan vs penetration test:** a scan identifies; a pen test exploits.
- **False positive vs false negative:** a false negative (a missed flaw) is the more dangerous one.
- **CVSS runs 0 to 10;** CVE is the identifier, NVD adds the score.
- **Rules of engagement** define scope, timing, contacts and data handling; test only what is in scope.
- **Lateral movement, persistence, pivoting:** moving across systems, keeping access, and using a foothold as a relay.
- **Exposure factor** is a percentage; **risk tolerance** drives patch speed.
- **When you can't patch:** segmentation, disabling the service, firewall rules or ACLs (compensating controls), or a formally approved exception.
- **After patching, rescan** to validate.

Check yourself

1. A scan misses an unpatched flaw that an attacker later exploits. Which error is this? *(False negative.)*
2. Which document tells a pen tester which IP ranges they may attack and during what hours? *(The rules of engagement.)*
3. A vendor won't ship a fix for a legacy server. Name two options. *(Segment it or move it to an air gap, block access at a firewall or ACL, disable the vulnerable service, or get a formal exemption.)*
4. A tester used a compromised workstation to reach a database server that the internet cannot see. What is this called? *(Pivoting, a form of lateral movement.)*
5. What do you do right after a major patch rollout? *(Rescan to confirm the vulnerability is gone and find systems missed.)*

Do next

Take any CVE from nvd.nist.gov and write a two-line priority decision using its CVSS score, where the system lives and how easy the exploit is. Next, 4.4 covers the monitoring and alerting that keeps this process running.

4.4 Alerting and monitoring

Exam domain: 4. Security Operations · **Videos:** Security Monitoring (10 min) · Security Tools (14 min)

Key ideas

Security monitoring

- **What to watch:** logins and where they come from (many sign-ins from a country where you have no staff is a red flag), services and their activity, backup completion, installed software versions (to judge patching), application availability, traffic volume (a sudden surge above normal may be data **exfiltration**), VPN users split into employees, vendors and guests, and firewall and IPS event spikes. Roll results into **dashboards** for a quick view of your posture.
- **SIEM (security information and event management):** collects logs from firewalls, switches, servers, routers and more into one database with one reporting engine. That lets you **correlate** different sources, for example a VPN login followed by what that user then reached, and measure normal data volumes to alert on excess. Because it stores history, it also supports **forensics**: looking back at what happened before an incident.
- **Continuous scanning:** laptops and phones move around, so systems constantly gather operating system and version details, drivers, installed apps and log anomalies into a database.
- **Actionable reports:** reports that tell you what to do next, such as which devices are out of compliance and what must be updated. With the data stored, you can answer "who is affected?" the moment a new vulnerability is announced, and run **ad hoc "what if" reports**, such as how many systems will be exposed when an OS reaches end of life in six months.
- **Retention:** breaches often go unnoticed for months (one 2022 industry report put the average near nine months to identify and contain), so keep logs and backups for a long time. Some laws mandate it.
- **Alerting:** don't wait for someone to read a report. Send an immediate **SMS or email**, or raise an alarm in the security operations center (SOC), for events like a rise in failed logins (brute force or password spraying) or a large outbound data transfer.
- **Response:** a common first reaction is to **quarantine** the affected system so the attacker can't spread.
- **Tuning:** alerts must be accurate. A **false positive** raises an alarm for harmless activity; a **false negative** misses a real event. Tune thresholds over time to balance them.

Security tools

- **SCAP (Security Content Automation Protocol):** a NIST standard that gives vulnerabilities and settings a common language, so an NGFW, an intrusion prevention system and a scanning tool all describe one flaw the same way. That lets tools work together: scanner finds, management system patches, with no human step.
- **Benchmarks:** published best-practice configurations for operating systems, apps and cloud services. The **Center for Internet Security (CIS)** hosts a large library, for example a mobile benchmark that disables screenshots and screen recording and forces encrypted backups.
- **Compliance checks:** run when a system connects or logs in. An **agent-based** check is installed ahead of time, runs continuously and needs ongoing updates. An **agentless** check runs on demand

(often at VPN login), executes in memory and removes itself, so there is nothing to maintain, but it only runs when triggered.

- **Antivirus and anti-malware:** find Trojans, worms, macro viruses, spyware, ransomware and fileless malware. Today the two terms mean essentially the same product.
- **DLP (data loss prevention):** watches traffic in real time and blocks sensitive data, such as Social Security numbers, medical records or card numbers, from leaving. It can be an appliance, endpoint software or a cloud service.
- **SNMP (Simple Network Management Protocol):** collects device metrics. Each device keeps a **MIB (Management Information Base)** of values, each named by an **OID (object identifier)**, a string of numbers. A management station **polls** devices over **UDP 161** (for example, bytes on an interface) on a schedule and graphs the changes. An **SNMP trap** is the reverse: a device-side rule fires and the agent sends an alert to the management station over **UDP 162** (for example, after CRC errors increase by five). The station can then send emails, texts or start automation.
- **NetFlow:** monitors traffic flows and application use rather than device counters. A **probe** (built into a switch or router, or external, fed by a SPAN port or a physical tap) summarizes flows and sends them to a **collector**, which reports the top conversations and top endpoints.
- **Vulnerability scanner:** minimally invasive; it never exploits. It also does port scans and finds active hosts in an IP range. Scan from inside and outside, run scans regularly, and verify results because they aren't all accurate.

Remember for the exam

- **SIEM** = central log collection, correlation, reporting and forensics.
- **SNMP polling uses UDP 161; traps use UDP 162.** Traps are device-initiated alerts; polling is station-initiated.
- **MIB and OID:** the MIB is the database; OIDs identify each value in it.
- **NetFlow** shows traffic flow and application statistics; SNMP shows device-level metrics.
- **SCAP** = one common language for vulnerabilities (NIST). **CIS** = benchmarks.
- **Agent vs agentless:** agents are persistent and need maintenance; agentless is on demand and leaves nothing behind.
- **False positive vs false negative:** an alert for nothing versus a missed real event.
- **Quarantine** is the standard first response to a triggered alert.

Check yourself

1. An NGFW, an IPS and a scanning tool each label one flaw differently. What standard fixes this? *(SCAP.)*
2. Which SNMP feature notifies you without waiting for a poll, and on which port? *(A trap, UDP 162.)*
3. You want to see which two hosts exchange the most traffic. Which tool? *(NetFlow, via its collector.)*
4. A check runs at VPN login, then deletes itself. Agent-based or agentless? *(Agentless, a dissolvable on-demand check.)*
5. A new vulnerability is announced and you need to know your exposure today. What makes that possible? *(Scan data stored in a central database, queried with a report.)*

Do next

Sketch where a SIEM, NetFlow collector and SNMP management station sit on a small network and list what each one receives. Next, 4.5 turns to the controls these tools monitor: firewalls, filters, protocols and endpoint security.

4.5 Enhancing enterprise security

Exam domain: 4. Security Operations · **Videos:** Firewalls (12 min) · Web Filtering (10 min) · Operating System Security (3 min) · Secure Protocols (5 min) · Email Security (7 min) · Monitoring Data (7 min) · Endpoint Security (9 min)

Key ideas

Firewalls

- A **network firewall** sits inline and allows or blocks traffic. Many also act as VPN endpoints (site-to-site or remote access) and as layer 3 routers at the ingress/egress point, where they can do NAT and dynamic routing.
- **Traditional firewall:** decides by port and protocol only. **Next-generation firewall (NGFW):** identifies the actual application by inspecting traffic, so it can allow or block "SSH" or "DNS" no matter which port carries it. Also called an application layer gateway, stateful multi-layer inspection or deep packet inspection device. NGFW rules can also match user, URL category and more.
- **Port-based equivalents** you must know: web `TCP 80` (HTTP) and `TCP 443` (HTTPS), SSH `TCP 22`, Remote Desktop `TCP 3389`, DNS `UDP 53`, NTP `UDP 123`. ICMP (ping) has no port numbers.
- **Rule base (ACL):** a list of rules, each with source, destination, port, protocol, application, time and action. The firewall checks them top to bottom and stops at the first match, so put **specific rules first** and broad ones lower. An **implicit deny** at the end drops anything no rule allowed.
- **Screened subnet:** a segment off the firewall for services the internet must reach, such as web or mail servers, which keeps internet traffic away from the internal network holding sensitive data.
- **IPS (intrusion prevention system):** inspects traffic in real time and blocks attacks, and is often built into an NGFW. It matches **signatures** (such as one for the Conficker worm) and can also detect **anomalies** or generic attack patterns, like SQL injection, without a specific signature. Rules can be grouped (for instance all injection rules set to block) and tuned to cut false positives.

Web filtering

- A **content filter** (URL filter, website category filter) controls which sites and content users reach, and is also used for parental controls at home. Some block known-bad sites carrying malware, and some inspect traffic in both directions.
- **URL scanning:** maintain an **allow list** and **block list**, or filter by **category** (gambling, adult, education, travel and dozens more). Each category can be set to allow, allow with a log or alert, or block.
- **Reputation:** sites are rated, for example trustworthy, low risk, medium risk, suspicious and high risk. Ratings are mostly automated, and you can override one manually. Block the high-risk ones.
- **Where filtering runs:**
  - Inside an **NGFW**, which assumes users are on the firewall's network.
  - **Agent-based** filtering on the device, which works anywhere (remote staff, travelers) but the agents need frequent updates of the category list, managed from a central console.
  - A **proxy** between users and the internet.

    - **DNS filtering**.
- **Proxy:** requests go to the proxy, which fetches the page, checks the response and passes it on. It can also **cache** pages, enforce **access control** by username or IP, and scan for malware. An **explicit proxy** must be configured in the application; a **transparent proxy** needs no client setup. A **forward proxy** (internal proxy) serves internal users going out to the internet.
- **DNS filtering:** the DNS service refuses to return an IP address (or returns a harmless one) for domains on threat-intelligence lists. It also stops malware from finding its **command-and-control** server, since the lookup fails, and it protects more than web browsing.

Operating system security

- **Active Directory (AD)** is the central database of users, computers, groups, file shares and printers. It handles authentication and permissions, assigned per user or per group.
- **Group policy** applies configuration and security settings to users and computers in AD through the Group Policy Management Editor: logon scripts, network settings and security requirements.
- **Linux** uses **discretionary access control (DAC)** by default, where owners set permissions. **SELinux (Security-Enhanced Linux)** adds **mandatory access control (MAC)**, where an administrator assigns rights centrally, supporting least privilege so that malware or a breach is limited in scope. It is open source.

Secure protocols

- Some protocols send data unencrypted: **Telnet, FTP, SMTP, IMAP, POP3 and HTTP**. A packet capture of such traffic shows readable content, not just headers. Use an encrypted replacement, or avoid the application.

Insecure	Port	Secure replacement	Port
Telnet	TCP 23	SSH	TCP 22
HTTP	TCP 80	HTTPS	TCP 443
FTP	TCP 21	SFTP (runs over SSH)	TCP 22
IMAP	TCP 143	IMAPS	TCP 993
POP3	TCP 110	POP3S	TCP 995

- **Don't assume from the port.** Port 443 usually means HTTPS but confirm the server's settings, and use a packet capture to verify traffic is really encrypted.
- **Encrypt everything at the network level** if an app can't: use **WPA3** on Wi-Fi (an open access point carries cleartext), or a **VPN**, which builds an encrypted tunnel to a VPN concentrator that then forwards the plain traffic inside the network. VPNs need client software and a concentrator or a commercial service.

Email security

- Email protocols have little built-in checking, so **spoofed** senders are common. Defenses are added through DNS records and enforced by a **mail gateway**, the gatekeeper that receives mail from other servers and decides whether to deliver, discard or send it to spam. On premises it belongs in a screened subnet; it can also be a cloud service.
- **SPF (Sender Policy Framework):** a DNS **TXT** record listing the mail servers allowed to send for your domain. The receiving gateway checks that the sending server is on the list.
- **DKIM (DomainKeys Identified Mail):** your mail server adds a **digital signature** to outgoing messages (visible in the headers, not something the user adds). The public key is published in a DNS **TXT** record, and the receiver uses it to verify the signature.
- **DMARC (Domain-based Message Authentication, Reporting and Conformance):** builds on SPF and DKIM. A DNS TXT record tells receivers what to do with mail that fails: accept it (monitor only), **quarantine** it (spam folder) or **reject** it. It also names where to send aggregate **reports**, so the domain owner can see how much legitimate mail passes and how much spoofing is attempted.

Monitoring data

- **File integrity monitoring (FIM):** alerts when files that should never change (executables, libraries) are modified. In Windows, **SFC (System File Checker)** scans protected OS files on demand and replaces bad ones. On Linux, **Tripwire** monitors changes, including in real time. A **host-based IPS** can also do FIM along with blocking attacks.
- **DLP** blocks sensitive data (Social Security numbers, medical data) in real time, and is classified by where the data is:
  - **Data in use** (in a system's active memory): **endpoint DLP**, software on the computer.
  - **Data in motion** (crossing the network): a network DLP appliance or an NGFW feature.
  - **Data at rest** (in storage): software on the server or OS that watches stored files.
- **Endpoint DLP** can control USB storage. A 2008 US Defense Department incident, where a USB drive spread the agent.btz worm, led to a ban on USB media until new rules were issued in 2010.
- **Cloud DLP** watches traffic to and from cloud apps and blocks sensitive uploads (and malware).
- **Email DLP** checks outbound mail for sensitive data (W-2s, fake wire transfers, hidden spreadsheet cells holding employee records) and inbound mail for suspicious keywords and impostors, and quarantines it.

Endpoint security

- **Defense in depth:** layer different protections across desktops, laptops, phones and tablets. The **edge** firewall is one layer, but its rules are fairly static.
- **Access control rules** limit who or what can reach data by user or group, location, the data's location and the application, and administrators can change them as the security posture shifts.
- **Posture assessment:** check a device when it connects or logs in remotely. Is it a trusted device (a company certificate)? Is antivirus installed, running and current? Are apps updated? Is the disk encrypted? Failing devices get limited access.
- **Agents for the check:**
  - **Persistent agent:** permanently installed, can run any time, and needs updates.

    - **Dissolvable agent:** runs at login, then removes itself.
    - **Agentless NAC:** tied to Active Directory, so it only operates at login and logoff.
- **Failed assessment:** quarantine the device, perhaps on a separate **VLAN**, with instructions to fix it, then reassess.
- **EDR (endpoint detection and response):** goes beyond signatures to behavioral analysis and machine learning, watches processes, and provides **root-cause analysis**. Response can be automatic: isolate the system, quarantine the malware and roll back to a known-good state.
- **XDR (extended detection and response):** correlates data from many systems and data types (including network traffic), so it catches more, reduces false positives and speeds investigation. **User behavior analytics** builds a baseline of normal users, devices, traffic and data access, then flags abnormal activity using rules, pattern matching or statistics.

Remember for the exam

- **Ports:** SSH 22, Telnet 23, DNS 53 (UDP), HTTP 80, NTP 123 (UDP), HTTPS 443, RDP 3389, IMAPS 993.
- **Traditional firewall = ports; NGFW = applications.** Top-down rules, first match wins, implicit deny at the end.
- **Secure replacements:** Telnet to SSH, HTTP to HTTPS, FTP to SFTP, IMAP to IMAPS.
- **SPF** lists allowed senders, **DKIM** signs messages, **DMARC** sets the failure policy (none/accept, quarantine, reject) and reporting. All three are DNS TXT records.
- **DLP states:** in use (endpoint), in motion (network), at rest (storage).
- **Proxy types:** explicit versus transparent; forward means internal users out to the internet.
- **Persistent vs dissolvable vs agentless:** all posture-assessment options; agentless NAC runs only at AD login and logoff.
- **EDR vs XDR:** EDR watches one endpoint with behavior and root cause; XDR correlates across many sources.

Check yourself

1. A firewall rule base allows everything on port 443 but nothing else, and has no other rules. What happens to an SSH attempt? *(It is dropped by the implicit deny.)*
2. Which DNS record tells receivers to reject mail that fails SPF and DKIM? *(The DMARC record, with a reject policy.)*
3. Staff work from home and travel. Which web filtering style still works? *(Agent-based filtering on the device, or DNS filtering.)*
4. An employee tries to copy files to a USB stick and DLP blocks it. Which DLP state is this? *(Data in use, via endpoint DLP.)*
5. A laptop connects to the VPN without antivirus running. What is the check and the best response? *(A posture assessment; quarantine it, perhaps in a remediation VLAN, until fixed.)*
6. Which protocol replaces Telnet, and on which port? *(SSH, TCP 22.)*

Do next

Write the SPF, DKIM and DMARC steps a receiving gateway performs for one incoming message, and recite the port table above without looking. Next, 4.6 moves from technical controls to identity and access management.

4.6 Identity and access management

Exam domain: 4. Security Operations · **Videos:** Identity and Access Management (13 min) · Access Controls (8 min) · Multifactor Authentication (4 min) · Password Security (6 min)

Key ideas

Identity and access management

- **Identity and access management (IAM)** covers the whole life of a person's access: it starts when an account is created and ends when it is disabled. Along the way it handles changes such as promotions or transfers, and it logs and monitors what the account does. That logging is often a regulatory requirement.
- **Provisioning and deprovisioning** are the create and remove steps, usually tied to onboarding and offboarding. The goal is **least privilege** from day one: nobody gets admin rights just because they joined.
- **Identity proofing** checks that a person is who they claim to be before an account exists. **Resolution** is the step that ties the claimed identity to a real person. **Attestation** confirms it with evidence, such as a passport or driver's license, an in-person meeting, or automated checks against records like credit history.
- **Single sign-on (SSO)** lets a user log in once and reach many resources without new prompts. The access normally lasts for a set period, such as a day, and then the user must authenticate again. The authentication and authorization systems must both support it.
- **LDAP (Lightweight Directory Access Protocol)** is a protocol for reading and updating a network directory. It is a lighter version of the older DAP and is based on the **X.500** standard. Entries are named with **distinguished names**, written as attribute=value pairs such as `CN=` (common name). Objects sit in a tree called the **directory information tree**, with containers for country, organization and organizational unit, and the individual objects as leaves.
- **SAML (Security Assertion Markup Language)** lets a resource trust a separate identity provider instead of keeping its own user database. The flow: the user requests a resource, is redirected to the identity provider, logs in there, receives a signed **token** (assertion), and presents it to the resource. SAML was not designed with mobile devices in mind.
- **OAuth** is an **authorization** framework, not an authentication protocol. It lets one service act on your behalf at another without sharing your password, for example a third-party tool getting permission to read and edit files in your cloud drive. It is often paired with **OpenID Connect**, which adds the authentication part.
- **Federation** lets users sign in with an account held by another organization (a social or work identity) instead of a local database. The two sides must set up a trust relationship first.
- **Interoperability:** pick technologies that fit what you already run. A VPN that can query LDAP fits an existing Active Directory; an OAuth-based app needs an authentication source behind it, often reached through an **API**.

Access control models

- **Least privilege** applies to every model: give only what the job needs. If malware runs under a limited account, the damage is limited too.
- **Mandatory access control (MAC):** every resource carries a label such as confidential, secret or top secret, and the administrator decides which clearance each user holds. Users cannot change it.
- **Discretionary access control (DAC):** the owner of the data decides who may read or change it. Flexible, but weaker, because security depends on every user making good choices.
- **Role-based access control (RBAC):** permissions belong to job roles. An administrator builds groups (manager, director, project lead), assigns permissions to each group, and adds users to the groups. In Windows these are groups.
- **Rule-based access control:** system rules set by an administrator are attached to objects and checked on every request. Example: a lab's data is available only between 9:00 and 17:00. Users cannot create or change rules.
- **Attribute-based access control (ABAC):** decisions combine many attributes, such as source IP address, time, requested action (read or write), and the user's relationship to the data. It supports the most detailed policies.
- **Time-of-day restrictions** can be layered on any model, for instance blocking a training-room network overnight. Global organizations need to include the user's time zone.

Multifactor authentication

- **Authentication factors** are the kinds of proof a login can ask for. Using two or more different kinds is **multifactor authentication (MFA)**.
- **Something you know:** password, PIN, unlock pattern.
- **Something you have:** smart card (usually paired with a PIN), USB security key holding a certificate, hardware token showing changing numbers that the server also generates, software token app, or a phone receiving an SMS code.
- **Something you are:** biometrics such as fingerprint or voiceprint. The system stores a mathematical template, not a picture. Biometrics are hard to change if compromised and can sometimes be bypassed, so pair them with another factor.
- **Somewhere you are:** location, based on IP address, GPS or both. A login from a different country minutes after one at home can be refused. IP-based location is only approximate.

Password security

- **Entropy** is how unpredictable a password is. Longer passwords mixing upper and lower case, numbers and symbols, or a long phrase, resist **brute force** and **password spraying**. Eight characters is a floor, and requirements keep rising.
- **Password age** and **expiration** force changes after a set time (30, 60 or 90 days; shorter for critical systems). **Password history** blocks reuse of old passwords.
- Use a **different password for every account**, so one leak does not open the others.
- A **password manager** stores everything in an encrypted database protected by strong credentials and MFA. It can generate random passwords and report weak or compromised ones. Built-in and enterprise versions exist.

- **Passwordless authentication** replaces the password with something like face unlock or a device-bound PIN, often after an initial password-based setup.
- **Just-in-time (JIT) permissions** give a technician temporary admin rights for a task. The person requests access from a central vault, which applies policy and issues **ephemeral** credentials built from the vault's primary ones. The primary credentials are never revealed, and the temporary ones are deleted after the session. A breached everyday account therefore has no admin power.

Remember for the exam

- **OAuth authorizes; SAML and LDAP authenticate/look up identities; OpenID Connect adds authentication to OAuth.** Match the verb in the question.
- **MAC = labels set by the administrator; DAC = owner decides; RBAC = roles/groups; rule-based = administrator rules (time, browser); ABAC = many attributes.** "Owner controls access" is always DAC.
- **Rule-based vs role-based:** the first checks rules on each request, the second looks at job function.
- **Factors must be different kinds.** A password plus a PIN is two "something you know" and is not MFA.
- **SMS codes and push approvals are "something you have"** (the phone), though weaker than a hardware key.
- **Resolution/attestation** belong to identity proofing at account creation, not to everyday login.
- **JIT permissions** solve standing admin rights: temporary, vaulted, ephemeral credentials.
- Longer beats complex for **entropy**; unique passwords plus a manager beat reuse.

Check yourself

1. A file's creator shares it with two colleagues as read-only. Which access control model? *(Discretionary access control: the owner decides.)*
2. A site grants access to a database only from 8:00 to 18:00 on weekdays, set by an administrator. Which model? *(Rule-based access control, using a time-of-day restriction.)*
3. A bank login asks for a password and a code from an authenticator app. Which factors? *(Something you know and something you have.)*
4. A user wants a design app to edit her cloud drive files without handing it her password. Which framework? *(OAuth, an authorization framework.)*
5. A help-desk technician needs admin rights for 30 minutes without ever seeing the real admin password. What is this? *(Just-in-time permissions with ephemeral credentials from a vault.)*

Do next

Without notes, list the five access control models with a one-line example each, then list the four authentication factors with two examples each. Next, 4.7 looks at scripting and automation, which is how many of these account tasks (provisioning, disabling) are done at scale.

4.7 Automation and orchestration

Exam domain: 4. Security Operations · **Video:** Scripting and Automation (8 min)

Key ideas

- **Scripting and automation** replace repeated manual steps with code that runs on its own. A tested script is fast and does not mistype, so it can fix problems at 3 a.m. or before anyone notices them, and staff can spend time on better work.
- **Enforcing baselines:** a script can watch for a new security patch and push it to every system that needs it, so no machine drifts from the standard.
- **Standard configurations:** scripts build the same router or firewall configuration, with all required security settings, every time a new device or remote site is deployed. This removes human variation.
- **Scaling cloud infrastructure:** when an application scales up, the script has to build the matching firewalls and security controls at the same time as the new servers and databases, so security grows with capacity.
- **Continuous monitoring and fast reaction:** a script can check a condition around the clock and respond, for example clearing temporary files when disk space runs low.
- **Onboarding and offboarding:** one script can create the account, add the user to the right groups, and give access to the home directory, a printer and email. The same approach removes access at departure.
- **Guardrails** are automated checks on what people enter. If a technician selects far too large a folder to delete, the guardrail blocks the action before critical files are lost.
- **Security group management:** automation can add and remove people from groups, and can alert immediately when someone is added to a sensitive group such as administrators.
- **Ticket handling:** an incoming email can become a help desk ticket and be routed to the right person. If monitoring finds a problem, the script may fix it, or escalate to the on-call technician when it cannot.
- **Enabling and disabling services** for a fixed period, then switching them off again with no manual step.
- **Continuous integration and deployment:** developers use scripts to build and push code updates automatically.
- **API integration:** scripts call the **APIs** of cloud services, firewalls and other devices to control them directly instead of using a console.

Drawbacks to weigh

- **Complexity:** scripts touch many systems, so they need thorough testing.
- **Cost:** someone must write them, which takes time and money.
- **Single point of failure:** if the script breaks, everything depending on it is affected.
- **Technical debt:** automating around a symptom can hide a deeper problem and let it grow instead of being fixed.
- **Ongoing support:** operating systems and scripting languages change, so scripts need maintenance and updates.

Remember for the exam

- Know the benefits as a list: **efficiency/time saved, enforcing baselines, standard infrastructure configurations, scaling in a secure manner, employee retention, reaction time, workforce multiplier.**
- Know the drawbacks: **complexity, cost, single point of failure, technical debt, ongoing supportability.**
- **Guardrails** validate input and block harmful actions; they are not the same as monitoring.
- Automation can **escalate** to a person when it cannot resolve an issue.
- User provisioning, ticket creation, resource provisioning, service enablement and CI/CD are the standard use cases listed for this objective.
- **Orchestration** coordinates many automated tasks across systems into one workflow; a single script is automation.

Check yourself

1. A script blocks any delete command that would remove a system directory. What is this? *(A guardrail.)*
2. A company's custom patch script has not been updated in years and nobody understands it. Which drawback? *(Ongoing supportability, with technical debt and single point of failure risk.)*
3. An email to the help desk automatically becomes a ticket assigned to the right technician. Which use case? *(Ticket creation and escalation automation.)*
4. Why script firewall deployment for new sites? *(Every site gets the same correct configuration with all security controls, with fewer human errors.)*

Do next

Pick one repetitive task from your own IT work (such as account setup) and write the steps a script would perform, plus one guardrail. Next, 4.8 covers incident response, where automation helps with fast containment.

4.8 Incident response

Exam domain: 4. Security Operations · **Videos:** Incident Response (9 min) · Incident Planning (7 min) · Digital Forensics (10 min)

Key ideas

Incident response process

- A **security incident** is any event that threatens systems or data: malware from a clicked attachment, a botnet-driven DDoS that floods the WAN link, data exfiltration followed by an extortion demand, or software that opens a way in from outside. Expect all of them.
- **NIST SP 800-61 (Computer Security Incident Handling Guide)** is the reference for the lifecycle. For the exam, learn the phases in this order:
  1. **Preparation**
  2. **Detection**
  3. **Analysis**
  4. **Containment**
  5. **Eradication**
  6. **Recovery**
  7. **Post-incident activity** (lessons learned)
- **Preparation** happens before anything goes wrong:
  - A current **contact list** and agreed communication methods.
  - An **incident go bag** with laptops, removable media, forensic software and a camera or imaging gear.
  - Reference material: system documentation, network diagrams, **security baselines**, and **file hashes** of critical files.
  - Mitigation tools: known-good OS and application images.
  - **Policies and procedures** everyone follows. Training happens now, because there is no time for it during an incident.
- **Detection** is hard because internet-facing systems are attacked constantly, so real intrusions hide among automated noise. Sources include:
  - Logs, for example web server logs showing a vulnerability scan.
  - A calendar of vendor patch releases, to watch for attacks on unpatched systems.
  - Direct contact from attackers.
  - Alerts: an IPS reporting a buffer overflow attempt, antivirus reporting malware, changes to security configuration, and unusual spikes in outbound traffic that suggest data theft.
- **Containment** means stopping the attack quickly rather than waiting to watch it. A **sandbox** is an isolated system for running suspect code to see what it changes. Malware may detect the sandbox or virtual machine and alter its behavior or delete itself.

- **Eradication and recovery:** remove malicious software or reimage the system, disable accounts the attacker used or created, fix the vulnerability that allowed entry, and restore from known-good backups or original installation media. Then harden the system.
- **Post-incident activity:** hold a **lessons-learned** meeting soon after, while memories are fresh. Ask what happened, what the timeline was, how well the plan worked, whether indicators were missed, and what monitoring should change. Feed the results back into the plan.

Incident planning and testing

- Test plans before a real incident, on **test systems**, with limited time since participants have other duties. Review afterward and change what failed.
- **Tabletop exercise:** people discuss a scenario around a table and walk through procedures. Cheap, quick (a few hours), and good for finding gaps. A full-scale disaster test costs far more.
- **Simulation:** a staged attack with real effects, such as a **phishing simulation**, a call to the help desk asking for a password reset, or a test data exfiltration to see if monitoring notices. Phishing reports show who clicked, and a third-party sender also tests mail filters and anti-phishing protections.
- **Root cause analysis:** work out how the attacker first got in, using logs and artifacts they left. There can be several root causes, including human mistakes, and the fix addresses each.
- **Threat hunting:** proactively look for weaknesses and attackers before an attack lands, by watching new vulnerability announcements, checking patch levels, and reviewing firewall rules. Automated monitoring tools can block known attack types when no one is watching.

Digital forensics

- **Digital forensics** collects and analyzes data to understand an event, improve defenses, and possibly serve as legal evidence years later. Follow best practices (see **RFC 3227**, guidelines for evidence collection and archiving) and keep detailed notes.
- **Legal hold:** a legal request, usually from a lawyer, to preserve specified data. It goes to the **data custodian**, who decides where to collect it. The data is kept as **ESI (electronically stored information)** in a separate repository. Collection may involve converting proprietary formats, for example email.
- **Integrity and chain of custody:** data must stay unmodified. **Chain of custody** records who handled evidence and when. In the physical world it is a sealed, signed evidence bag; digitally, **hashes** and **digital signatures** prove the data has not changed and the log shows who accessed it. Every piece of collected data needs its own chain.
- **Acquisition** is usually the first step. Sources include disk, memory, firmware, file system, other devices in a multi-system attack, servers, network devices, firewall logs, and VM **snapshots**. Less obvious sources: log files, the recycle bin, temporary storage, browser bookmarks and saved logins.
- **Live acquisition** captures data from a running system. This matters when encryption locks data once the machine is powered off, and for memory contents. Mobile devices can be remotely wiped, so work from copies.
- **Preservation:** work from **copies** of the original media, which also gives you a backup.

- **Reporting:** a summary of the event, detailed steps from source to acquired data (so a third party can verify the integrity checks), a factual analysis of the data, and conclusions about what happened.
- **E-discovery:** finding, gathering, reviewing and handing over electronic records, usually for a legal case. The job is to deliver the data; analyzing it is a separate step. Example: imaging a drive and handing it to forensic specialists, who then look for deleted data.
- **Order of volatility** (standard forensics knowledge): collect the most short-lived data first: CPU registers and cache, then RAM, then swap/temporary files, disk, remote logs, and finally backups and archives.

Remember for the exam

- The phases in order: **preparation, detection, analysis, containment, eradication, recovery, lessons learned.** Containment comes before eradication, and recovery comes after it.
- **Tabletop = discussion only; simulation = actual test actions.** Both are tested on non-production systems or controlled scope.
- **Chain of custody** proves who held evidence and that it did not change; hashing is the digital seal.
- **Legal hold** is a preservation order; **e-discovery** is collection and production without analysis.
- **Preservation** means working on copies and keeping the original untouched.
- **Root cause analysis** finds how the attacker got in; **lessons learned** improves the process; **threat hunting** is proactive.
- Acquire the most **volatile** data (memory) before disk.

Check yourself

1. Malware is found on a server. The team disconnects it from the network before cleaning. Which phase? *(Containment, followed by eradication.)*
2. Managers meet to talk through a ransomware scenario with no systems touched. What is this? *(A tabletop exercise.)*
3. A lawyer asks the company to keep all email about a project. What is the request called and who receives it? *(A legal hold, sent to the data custodian.)*
4. How can you prove a disk image has not changed since collection? *(Compare its hash with the original hash and keep a chain of custody log.)*
5. Which phase produces updated detection rules after the team learns they missed a warning sign? *(Post-incident activity / lessons learned.)*

Do next

Write the seven phases from memory and one concrete action for each. Next, 4.9 covers the log data you rely on during detection and analysis.

4.9 Data sources for investigations

Exam domain: 4. Security Operations · **Video:** Log Data (14 min)

Key ideas

- **Log data** is the main raw material of an investigation. Logs record which traffic was allowed or blocked, exploit attempts, blocked URL categories, and **DNS sinkhole** hits that can reveal a compromised internal host. Correlating logs from different devices lets you follow one event end to end.
- **Firewall logs:** each entry shows time and date, source and destination IP addresses, ports, and the **disposition** (allowed or blocked). A **next-generation firewall (NGFW)** adds the application in use, URL categories, and anomalies in the traffic. Some logs also show the MAC address that received the traffic.
- **Application logs:** the Windows Event Viewer's application section and, on Linux and macOS, files under `/var/log`. These are typically forwarded to a **SIEM (security information and event management)** system, where they can be filtered and viewed in different ways.
- **Endpoint logs:** laptops, desktops, phones and tablets record logon and logoff events, system events, running processes, device management actions such as password changes and lockouts, and directory-service activity. Sending them to the SIEM lets you compare endpoint activity with network activity.
- **OS-specific security logs:** operating systems log authentication, brute-force attempts, and changes to critical files. A service being disabled when no administrator did it can raise an alert. Because the volume is huge, forward to the SIEM only what supports security decisions.
- **IDS/IPS logs:** show known vulnerabilities and attack types. Standalone IDS/IPS boxes are rare now since NGFWs include the function. An example entry (from the open source Snort) has a timestamp, alert class (possible denial of service, a SYN flood), a priority, and source and destination IP addresses and ports.
- **Network logs:** switches, routers, wireless access points and VPN concentrators log routing table changes, failed authentication to the device, and attacks. For instance, a router log can note a TCP SYN attack on an interface and an automatic 60-second block.
- **Metadata:** data about data, found in documents and traffic.
  - **Email headers** show the sending servers, addresses and results of checks such as **SPF**, which helps trace origin.
  - **Photos** can record the device and **GPS coordinates**.
  - **Web browsers** expose OS, browser type and IP address.
  - **Office documents** can hold the author's name, contact details and title.
- **Vulnerability scan logs:** results show devices missing a firewall, antivirus or anti-spyware, misconfigurations such as shares open without a password or enabled guest accounts, and systems with known vulnerabilities needing patches (once scanner signatures are current). Summaries can flag unsupported operating systems or NFS shares readable by anyone.
- **Automated reports:** SIEMs can produce scheduled reports, built in or via third-party tools. A report no one reads has no value, and broad reports against terabytes of data are slow, so make them specific.

- **Dashboards:** at-a-glance, near real-time views for the security operations center showing current system status, active firewall rules, warnings, and users and devices. They avoid long-term data because it is slow to compile. They can be customized or predefined.
- **Packet captures:** record traffic at the packet level using a tool like **Wireshark**, or capture features on switches, routers and firewalls, on wired and wireless networks. The summary pane lists each packet; the detail pane breaks one packet into Ethernet, IPv4, TCP and HTTP layers (for example, an HTTP GET request).

Remember for the exam

- **SIEM** = central collection, correlation, reports and dashboards for logs from many sources.
- Firewall log fields: **time, source/destination IP, port, application (NGFW), disposition**.
- **NGFW logs add application and URL information**, which a traditional port-based firewall log lacks.
- **Metadata** examples: email headers, photo GPS, document author. Email headers help trace phishing.
- **Vulnerability scan** logs reveal missing protections and misconfigurations, not live attacks.
- **Dashboards** are real-time summaries; **reports** are scheduled and detailed.
- **Packet capture** gives the most detail (every packet) but the largest volume; Wireshark is the standard tool.

Check yourself

1. Which log shows the application in use and whether a flow was blocked? *(NGFW firewall log.)*
2. You need to trace which servers handled a suspicious email. Where do you look? *(The email's headers, metadata.)*
3. Why send only selected logs to the SIEM? *(Volume is huge; only log data useful for security decisions should be collected.)*
4. A SOC wall screen shows live status, not last quarter's trends. What is it? *(A dashboard.)*
5. Which source shows exactly which HTTP request a client sent, byte by byte? *(Packet capture, such as Wireshark.)*

Do next

Take a sample firewall log line and write down its source, destination, port, application and disposition, then name the SIEM feature that would alert on it. Next, 5.1 begins Domain 5, where policies and standards define what these logs are checked against.

5.1 Security governance

Exam domain: 5. Security Program Management and Oversight · **Videos:** Security Policies (11 min) · Security Standards (6 min) · Security Procedures (7 min) · Security Considerations (5 min) · Data Roles and Responsibilities (2 min)

Key ideas

Governance works in layers. **Policies** say what must happen and why, **standards** set the specific measurable requirements, and **procedures** give the steps to follow. Keep that order in mind through everything below.

Security policies

- **Information security policies** are the master set of written rules that protect confidentiality, integrity and availability (CIA). They cover broad goals (data storage, event handling) and detailed rules (Wi-Fi use, remote access). They answer questions like what to do about a virus on a workstation. They also name **roles and responsibilities**. Writing a policy does nothing on its own; the organization must enforce it.
- **Acceptable use policy (AUP):** what people may do with company computers, phones and other technology. It informs staff and gives the organization legal backing, for example when someone is dismissed for violating it.
- **Business continuity plan (BCP):** how to keep operating when normal technology is unavailable. Example: a store whose card network is down processes payments manually and phones in approvals. It must be planned and tested in advance.
- **Disaster recovery plan (DRP):** the broader plan for widespread or long outages from natural disasters (tornado, flood), system failures or human actions. It covers an alternate recovery site, data recovery, application restoration, and having staff available at the recovery location.
- **Incident response policies:** define who does what for malware, DDoS or botnet attacks, and theft of confidential data. **Incident response roles:** the trained **incident response team**, **IT security management** (obtains resources and people), **compliance officers** (where mandates apply), **technical staff**, and **users**, who report what they saw. NIST SP 800-61 describes the lifecycle: preparation, detection and analysis, containment/eradication/recovery, post-incident activity.
- **Software development lifecycle (SDLC):** the process from idea to deployed application: requirements, build, test, deploy, maintain. **Waterfall** is linear. **Agile** repeats short cycles of design, develop, test, deploy and review before launch.
- **Change management:** a formal process for changes to applications, firewall rules, routers and switches so changes do not harm the business. It documents timing, duration, installation steps and, most importantly, a **fallback procedure**.

Security standards

- Organizations write their own standards or adopt published ones from **ISO** (International Organization for Standardization) and **NIST**.

- **Password standard:** complexity, how resets are verified, change frequency, secure storage, approved password managers, and which authentication method is used. A standard can forbid local accounts on network devices and require central authentication such as LDAP to Active Directory.
- **Access control standard:** what access, and when. It may mandate MAC and ban DAC, require manager sign-off or training before access, and define how access is removed (security issue, account or contract expiry, departure).
- **Physical security standard:** ID badges and electronic locks, different rules for employees, contractors and guests, biometric locks, monitoring, motion detection, mandatory visitor escorts, and offboarding steps.
- **Encryption standard:** approved algorithms and how they are implemented, such as storing passwords as a **salted hash** with a named algorithm. Requirements differ by data state: **at rest**, **in transit** and **in use**.

Security procedures

- **Change management steps:** define the **scope** (one server or many, a whole OS or one file), assess the **risk**, create a plan, get approval, have a **change control board** review and schedule the change and confirm a **backout plan**, then document what changed.
- **Onboarding:** new hire or transfer receives the handbook and signs the AUP, gets an account with correct rights, and receives devices.
- **Offboarding:** decides what happens to hardware and its data. **Disable** the account rather than deleting it, because deletion can lose encryption keys or data you may need later.
- **Playbooks:** step-by-step checklists for particular events, such as investigating a data breach or recovering from ransomware. A **SOAR** platform (security orchestration, automation and response) connects many tools and automates playbook steps.
- **Keep procedures current:** review playbooks, tighten change control, add playbooks for new technology, and account for emerging threats.
- **Governance structures:** a **board** (of directors) sets broad objectives, and **committees** of subject experts work out how to meet them and report back for approval. **Government** (public sector) bodies deal with legal, administrative and political concerns, and meetings are typically open to the public. **Centralized** governance has one group deciding for the whole organization; **decentralized** pushes decisions to those doing the work.

Security considerations

- **Regulatory:** rules on what data is kept and for how long, including logs. **SOX (Sarbanes-Oxley, 2002)** concerns the financial data of public companies. **HIPAA** protects health information, including how it is transferred and disclosed.
- **Legal:** IT staff need processes to report illegal activity and to respond to a **legal hold**. Many places require **breach disclosure** within a set time, and the rules vary by location.
- **Data sovereignty and cloud:** some countries require data about their citizens to stay within their borders, which conflicts with deploying cloud systems anywhere.

- **Industry:** power utilities may be **air-gapped** from other networks. Healthcare needs wide availability to staff plus strong encryption.
- **Geographic scope:** local or regional (city/state records), national (defense, many agencies, stronger encryption), and global (different data protection laws in each country).

Data roles and responsibilities

- **Data owner:** senior person accountable for a body of data, such as the VP of sales for customer data or the treasurer for financial data.
- **Data controller:** decides how and why data is used. **Data processor:** handles it on the controller's instructions. Example: the payroll department (controller) and an outside payroll company (processor).
- **Data custodian / data steward:** manages day-to-day security, accuracy and privacy of the data, assists with legal and regulatory compliance, applies **sensitivity labels**, and ties them to access control.
- A **data protection officer (DPO)** oversees privacy compliance, which is common under GDPR.

Remember for the exam

- **Policy = what/why; standard = specific requirement; procedure = steps.** Playbooks are procedures.
- **BCP** keeps operations going; **DRP** restores IT after a major disaster.
- **Owner = accountable; custodian/steward = hands-on protection; controller decides use; processor acts on instructions.**
- **Offboarding:** disable first, delete later.
- **Change management** always includes approval, a backout plan and documentation.
- **SOX = financial reporting of public companies; HIPAA = healthcare data.**
- **Data sovereignty:** the law of the country where data resides or its citizens' country can apply.

Check yourself

1. A contractor leaves. IT deletes the account and later finds files encrypted with its keys. What should have been done? *(Disable the account instead of deleting it.)*
2. A payroll vendor processes salaries following your HR team's instructions. Who is controller and who is processor? *(HR is the controller; the vendor is the processor.)*
3. A requirement says all passwords are stored as salted hashes using a named algorithm. Policy, standard or procedure? *(Standard.)*
4. Which group approves and schedules firewall rule changes, and confirms a rollback plan? *(The change control board.)*
5. A hospital must keep patient data inside the country. What concern? *(Data sovereignty/regulation, plus HIPAA-style health data protection.)*

Do next

Write one line each for a policy, a standard under it, and a procedure under that, for a topic like passwords. Next, 5.2 turns to risk management and the formulas used to put numbers on it.

5.2 Risk management

Exam domain: 5. Security Program Management and Oversight · **Videos:** Risk Management (4 min) · Risk Analysis (9 min) · Risk Management Strategies (3 min) · Business Impact Analysis (3 min)

Key ideas

Risk management and assessment types

- **Risk management** finds where an organization is exposed, from inside or outside, so risks can be handled before they grow. Risk grows with the organization.
- **Risk assessment types:**
  - **One-time (ad hoc):** done for a single purpose, such as evaluating an acquisition, new equipment or software, or a specific attack type the CEO heard about. A short-lived committee does it, reports, and disbands.
  - **Recurring:** done on a schedule (quarterly, every six months, yearly), often by an internal team keeping continuing documentation.
  - **Continuous:** built into a process, like the risk step of every change in change control.
  - **Mandated:** required by a standard. Organizations holding card data must assess risk yearly under **PCI DSS**.

Risk analysis

- **Qualitative analysis** rates factors in broad terms such as low/medium/high, often on a traffic-light grid. Each factor (legacy Windows clients, untrained staff, devices without antivirus) gets ratings for impact, likelihood, cost of controls and overall risk. It shows where to focus.
- **Quantitative analysis** uses numbers:
  - **Asset value (AV):** what the asset is worth to the organization. It can include lost sales and fines, not just replacement cost.
  - **Exposure factor (EF):** the share of the asset's value lost in one event, from 0 to 1. Losing a quarter is 0.25; losing it all is 1.0.
  - **Single loss expectancy (SLE)** = AV × EF. The expected loss from one event.
  - **Annualized rate of occurrence (ARO):** how many times the event is expected per year. A hurricane's ARO is lower in Montana than in Florida.
  - **Annualized loss expectancy (ALE)** = SLE × ARO. The expected yearly loss.
  - **Worked example:** a stolen laptop is worth $1,000 and the whole value is lost (EF 1.0), so SLE = $1,000 × 1.0 = $1,000. If seven laptops are stolen per year, ALE = $1,000 × 7 = $7,000. The formula ignores the data on the laptops, which may be worth more, so qualitative review still matters.
  - Compare the ALE with the cost of a control to judge whether the control is worth buying.
- **Impact areas: life** (always first), **property**, **safety** and **finance**.
- **Likelihood** is a qualitative description (rare, possible, almost certain). **Probability** is a number based on past or expected frequency. The two are often used loosely and can convert into each other.

- **Risk appetite:** how much risk the organization is willing to take on. The **risk appetite posture** describes the attitude: **conservative**, **neutral** or **expansionary**.
- **Risk tolerance:** how far above the appetite the organization will actually allow before acting. Example: a highway limit is the appetite; police not stopping you until you are well over it shows a higher tolerance.
- **Risk register:** a project's list of risks. Each entry has a **key risk indicator (KRI)** (for example, an undefined project scope or schedule), an **owner** responsible for it, and a **risk threshold**. It balances the cost of fixing a risk against the cost of leaving it.
- **Risk reporting:** a regularly updated summary of tracked risks, including critical and emerging ones, how they are handled, and what management needs for business decisions.

Risk management strategies

- **Transfer:** shift the risk to someone else, such as buying **cybersecurity insurance**.
- **Accept:** take the risk knowingly. This is common. Approval can come through an **exemption** (a device is excused from a policy, such as an unpatchable factory machine kept off the network) or an **exception** (the rule is temporarily relaxed, such as delaying the three-day patch deadline until an application is fixed).
- **Avoid:** remove the activity so the risk no longer exists.
- **Mitigate:** reduce the likelihood or impact, for example with a next-generation firewall for internet exposure.

Business impact analysis (BIA)

- **Recovery time objective (RTO):** the maximum acceptable time to get a system or process back up after an outage. If the business needs both the database and web server running to count as operating, the time to restore both is the RTO.
- **Recovery point objective (RPO):** the maximum acceptable amount of data loss, measured as time back from the outage. It decides how current the restored data must be, and so how often to back up. An RPO of one hour means backups or replication at least hourly.
- **Mean time to repair (MTTR):** average time to fix and restore a failed item, including diagnosis, obtaining and installing replacement hardware, and configuration. Spare equipment on site or a two-hour replacement contract shortens it.
- **Mean time between failures (MTBF):** the expected time between breakdowns of repairable equipment, from the manufacturer or history. Rough calculation: total uptime ÷ number of failures. It helps judge how risky a device is.

Remember for the exam

- **SLE = AV × EF. ALE = SLE × ARO.** Do not mix them up; EF is a fraction between 0 and 1.
- If a question gives ALE and ARO, **SLE = ALE ÷ ARO**.
- **Qualitative = categories/ratings; quantitative = money and numbers.**
- **Appetite = how much risk you want; tolerance = how much deviation you will put up with.**

- **RTO = how long to recover; RPO = how much data you can lose (determines backup frequency).**
- **MTTR = repair time; MTBF = time between failures** (higher MTBF is more reliable).
- Four strategies: **transfer, accept, avoid, mitigate.** Insurance is transfer. Exemption/exception are forms of accept.
- Life safety outranks other impacts.

Check yourself

1. A server is worth $50,000. A flood would destroy 40% of it, and floods are expected once every two years. What are SLE and ALE? *(SLE = $50,000 × 0.4 = $20,000; ARO = 0.5, so ALE = $10,000.)*
2. The business can lose at most 15 minutes of transactions. Which metric? *(RPO.)*
3. A company buys cyber insurance for ransomware losses. Which strategy? *(Transfer.)*
4. A vendor's lab device cannot be patched, so management signs off and isolates it. Which strategy? *(Accept, through an exemption.)*
5. What does MTTR measure? *(The average time to repair and restore a failed system.)*

Do next

Work three ALE problems of your own, changing the AV, EF and ARO each time, and define RTO, RPO, MTTR and MTBF out loud. Next, 5.3 covers the risks that come from vendors and suppliers.

5.3 Third-party risk

Exam domain: 5. Security Program Management and Oversight · **Videos:** Third-party Risk Assessment (12 min) · Agreement Types (5 min)

Key ideas

Third-party risk assessment

- **Third-party (vendor) risk:** suppliers such as payroll services, email marketing platforms, travel providers or raw-material sellers often hold some of your data. Assess what they do with it and how they protect it, and write those expectations into the **contract**, including penalties for breach.
- **Penetration testing** actively tries to exploit weaknesses, unlike a vulnerability scan that only finds them. A contract can require both sides to run tests at set intervals, often by an independent testing firm that reports results.
- **Rules of engagement** set the boundaries of a test:
  - The type of test (on-site physical breach, internal, or external over the internet).
  - Timing: a date, working hours only, or after hours only.
  - Target **IP address ranges**.
  - **Emergency contacts** in case something goes wrong.
  - How to handle **sensitive data** discovered during the test.
  - What is **in scope** and **out of scope**.
- **Right to audit:** a contract clause allowing regular audits of the vendor's security, especially when it stores your data or carries your traffic (an ISP, for example). Audits are usually done by an independent third party and may be required by compliance rules. They look at access management, offboarding, password handling and storage, and VPN access controls. Repeat them over the life of the relationship.
- **Supply chain analysis:** examine every step from raw material or code to finished product or service: how it is delivered, how teams coordinate, how security is handled between technical teams, and how business processes change. The **SolarWinds** compromise shows why: attackers inserted malware into the product's code, and the vendor shipped it in a legitimately signed update. At least 18,000 of roughly 300,000 customers installed it.
- **Independent assessments:** an outside expert brings experience from many organizations and sees what insiders miss.
- **Due diligence:** investigating a company before doing business, by verifying its claims (finances, customer numbers), running background checks, and interviewing people.
- **Conflict of interest:** something that could compromise judgment, such as the vendor also serving your top competitor, employing an executive's relative, or offering gifts to win the contract.
- **Vendor monitoring:** keep checking after signing. Review financial health and IT security, watch news and social media, and use both quantitative and qualitative measures. A named internal group owns each relationship.
- **Questionnaires:** a simple way to learn about a vendor: its due diligence and conflict-of-interest practices, disaster recovery plans, and how data is stored and protected. Answers feed the ongoing

risk analysis.

Agreement types

Agreement	Purpose	Binding?
SLA	Minimum service levels	Yes (contract)
MOU	Broad shared goals	No, informal
MOA	More detail on the relationship	Usually not enforceable
MSA	Framework terms for future work	Yes
SOW	Specific job details	Yes, under an MSA
NDA	Confidentiality	Yes
BPA	Formal business partnership	Yes

- **Service level agreement (SLA):** minimum service terms, for example at most four hours of unscheduled downtime, technician dispatch, and spare equipment kept on site.
- **Memorandum of understanding (MOU):** an informal, unsigned outline of goals. It may include confidentiality statements.
- **Memorandum of agreement (MOA):** one step above an MOU, with more detail on the relationship. It is not a contract and generally is not legally enforceable, though parts can be binding.
- **Master service agreement (MSA):** a legal contract for an ongoing relationship covering general terms such as billing. Later work is added under it.
- **Statement of work (SOW)** or work order: details of one job: scope, location, deliverables schedule, and expectations. It resolves disputes about what was or was not covered.
- **Non-disclosure agreement (NDA):** a signed contract keeping information confidential. **Unilateral** (one-way) binds one party. **Bilateral** (mutual) binds two. **Multilateral** binds more than two.
- **Business partners agreement (BPA):** covers financial terms, ownership stakes, who makes decisions, and what happens in a financial problem or disaster.

Remember for the exam

- **Right to audit** and **rules of engagement** are the two phrases to link with vendors and pen tests.
- **MSA + SOW:** MSA sets the general framework; the SOW defines a specific project.
- **MOU is informal and non-binding; MOA is more detailed but generally not a contract.**
- **SLA** is about measurable service levels (uptime, response time).
- **NDA types:** unilateral, bilateral (mutual), multilateral.
- **Due diligence** happens before the deal; **monitoring** continues after it.
- **Supply chain** attacks hit you through a trusted supplier; SolarWinds is the classic case.

Check yourself

1. You want guaranteed maximum downtime from an ISP. Which agreement? *(SLA.)*

2. Two companies exploring a partnership want a first, informal statement of shared goals. Which document? *(MOU.)*
3. A consultant is hired for a defined project under an existing contract. Which document details the work? *(Statement of work, under the MSA.)*
4. Your pen tester asks which networks are off limits and whom to call if a server crashes. Where is this written? *(The rules of engagement.)*
5. A vendor you are about to sign employs your CFO's brother. What is this? *(A conflict of interest.)*

Do next

Match each of the seven agreement types to a one-line situation without looking. Next, 5.4 covers the compliance and privacy laws that these contracts often have to satisfy.

5.4 Security compliance

Exam domain: 5. Security Program Management and Oversight · **Videos:** Compliance (8 min) · Privacy (5 min)

Key ideas

Compliance

- **Compliance** means meeting a set of requirements that come from laws and regulations or from agreements with other parties. They depend on your industry and location and may be national or international.
- **Consequences of non-compliance:**
  - **Fines** and **sanctions**; in serious cases personal penalties, including loss of a job or prison.
  - **Reputational damage**, for example when a legally required breach disclosure hurts public trust or share price. Uber hid a 2016 breach for over a year and paid hackers to stay quiet; it was later fined $148 million, and its former security chief was convicted. Announcing the breach on time would have been compliant.
  - **Loss of license** to operate or sell, and sometimes restrictions on who may buy from you.
  - **Contractual impact:** a breach of a compliance clause is a breach of contract, which two private parties can often settle without going to court.
- **Chief compliance officer (CCO)** (also called compliance officer): the person responsible for making sure the organization follows applicable rules and for reporting its compliance status.
- **External requirements and reporting:** third parties may require regular compliance reports, and a missed or inaccurate report can bring penalties.
- **Examples of regulatory compliance:**
  - **SOX (Sarbanes-Oxley Act of 2002):** financial reporting by public companies.
  - **HIPAA:** privacy of health information in the United States. Penalties rise with intent: wrongful disclosure carries a fine and possible prison time, more if done under false pretenses, and the most if done for commercial gain, personal gain or malicious harm.
  - **GLBA (Gramm-Leach-Bliley Act, 1999):** financial institutions must protect customer data and explain their privacy practices, which is why customers receive privacy notices.
- **Due care and due diligence:** both show good-faith effort. As the lesson frames it: **due care** applies to your own internal practices; **due diligence** applies to investigating and monitoring third parties.
- **Attestation and acknowledgment:** a responsible executive formally states, in writing, that compliance is in good standing and accepts responsibility for that claim.
- **Compliance monitoring and automation:** track status continuously with internal tools, gather information from people and third parties, and use automated compliance platforms to collect evidence and compile reports.

Privacy

- **Privacy laws** exist at several levels: local and state (property, vehicle and professional licensing records), national (HIPAA in the US), and international (**GDPR**).
- **GDPR (General Data Protection Regulation):** European Union regulation protecting people living in the EU. Covered personal data includes name, address, photo, email, bank information, social media posts, genetic data and location data.
- **Data subject:** the identifiable person the data describes. GDPR puts control in the data subject's hands. They can request removal of their data, known as the **right to be forgotten** (right to erasure).
- GDPR and similar laws define privacy from the **data subject's** point of view, rather than placing duties only on the company holding the data.
- **Data roles** (revisited): the **data owner** is the senior person accountable for a data set (VP of sales for customer data, treasurer for finances). The **data controller** decides how data is used and the **data processor** actually processes it, internally or at a third party. A payroll department (controller) sending data to a payroll company (processor) is a standard example, often protected with an NDA.
- **Data inventory:** a list of the data an organization collects and stores, including owner, update frequency and format. It underpins privacy work by showing what exists and how it is used (collaboration, security, quality checks).
- **Sharing with third parties:** before you share data, know what it is and confirm the sharing is allowed by law and regulation.

Remember for the exam

- **GDPR = EU, data subject rights, right to be forgotten. HIPAA = US health data. SOX = public company financial reporting. GLBA = financial institutions' customer data.**
- Non-compliance costs: **fines, sanctions, reputational damage, license loss, contract termination.**
- **Controller decides; processor processes.** The owner is accountable; the custodian protects it day to day.
- **Attestation** is a signed assurance of compliance; **acknowledgment** confirms understanding of the requirements.
- Breaches usually have **disclosure** duties, and hiding one makes it worse.
- A **data inventory** comes before privacy decisions.

Check yourself

1. A European customer asks a site to delete all her personal data. Which law and right? *(GDPR, the right to be forgotten/erasure.)*
2. A hospital mishandles patient records. Which regulation applies? *(HIPAA.)*
3. A company pays an outside firm to run payroll under its instructions. Who is the processor? *(The outside payroll firm; the company's payroll department is the controller.)*
4. A company hides a breach for a year to avoid bad press. Name two likely consequences. *(Large fines and reputational damage.)*
5. An executive signs a statement that compliance requirements are being met. What is this? *(Attestation.)*

Do next

Make a one-line card for each regulation named here (SOX, HIPAA, GLBA, GDPR) and test yourself on who it protects. Next, 5.5 covers audits and penetration tests, which check whether compliance actually holds.

5.5 Audits and assessments

Exam domain: 5. Security Program Management and Oversight · **Videos:** Audits and Assessments (3 min) · Penetration Tests (5 min)

Key ideas

Audits

- A **security audit** reviews the environment against what it should be: infrastructure, software, network devices, and the policies and procedures themselves. A good audit finds weaknesses before an attacker does.
- **Attestation** is the formal statement that the audit's findings are true. The audit gathers the evidence; the attestation vouches for it.
- **Internal audits** are run by the organization's own staff, usually to confirm compliance tasks are actually being done.
  - An **audit committee** oversees risk and decides when internal audits start and stop.
  - Audits often begin with **self-assessments**: each team checks its own processes against the requirements, and the committee combines the results into an overall picture.
- **External audits** are done by an independent third party. Some regulations require one, and the regulation usually sets the scope and how often it happens. Expect auditors on site, reviewing records and asking for evidence. The output shows where you stand today and what to improve.

Penetration tests

- A **penetration test** tries to exploit weaknesses, not just list them.
- **Physical penetration testing** tries to get into the building: doors, windows, elevators, anything that bypasses physical security. It matters because anyone who can touch a server can boot it from other media or tamper with its files, which is why servers live in locked data centers.
- **Offensive, defensive and integrated:**
  - The **red team** attacks: finds vulnerabilities and exploits them.
  - The **blue team** defends: spots attacks as they happen and blocks them.
  - **Integrated** testing joins the two, so every hole the red team finds goes straight to the blue team to fix and detect next time.
- **How much the tester is told:**
  - **Known environment:** full details of the systems in scope.
  - **Partially known environment:** some details, often to steer testers toward particular systems.
  - **Unknown environment:** nothing; testers discover everything themselves (a "blind" test).
- **Reconnaissance** comes first in every case: learn what security tools, servers and applications exist, map the networks, IP ranges and remote sites, then focus on the key systems.
  - **Passive reconnaissance** never touches the target's network: social media, the company website, forums and Reddit posts, social engineering, dumpster diving, or talking to the company's partners.

- **Active reconnaissance** queries the target directly: ping and port scans, DNS queries to its servers, OS fingerprinting, service and version scans. It's visible, and it leaves traces in firewall and other logs.

	Passive recon	**Active recon**
Touches target network	No	Yes
Detectable	Hard to detect	Shows up in logs
Examples	Social media, website, dumpster diving	Ping/port scan, DNS query, OS fingerprinting

Remember for the exam

- **Attestation** is the vouching for audit results; an **audit committee** runs internal audits.
- Some regulations **require an external** (third-party) audit; internal audits and self-assessments don't replace it.
- **Red = attack, blue = defend**; integrated teams feed findings straight into fixes.
- **Known / partially known / unknown environment** replaced the older white-, gray- and black-box terms.
- **Passive** recon uses outside sources and can't be seen by the target; **active** recon sends packets to the target and gets logged.

Check yourself

1. A tester reads the company's job postings and LinkedIn profiles to learn which firewall it runs. Passive or active? *(Passive: nothing touched the company's network.)*
2. Testers are given network diagrams and a list of in-scope servers. Which environment? *(Known environment.)*
3. A regulator requires an independent review every year. Internal or external audit? *(External, by a third party.)*
4. One group attacks systems while another watches the alerts and blocks them, and they share findings. What is this? *(Integrated red and blue team testing.)*

Do next

List five pieces of information a tester could learn about your own school or employer without touching its network, then three they could only get by scanning. Then move on to 5.6, the people side of security.

5.6 Security awareness

Exam domain: 5. Security Program Management and Oversight · **Videos:** Security Awareness (7 min) · User Training (5 min)

Key ideas

Phishing campaigns

- To learn how many staff would click a malicious link, run your own **phishing campaign**: send realistic test emails, built in house or bought as a service, and record opens, clicks and replies in a central console.
- Anyone who clicks gets an automatic notice and is sent to **follow-up training**, online or in person.
- **What users should check:** spelling and grammar in the message and the link, the real domain behind the link, inconsistencies in how the email is put together, unexpected attachments, and any request for credentials or personal details.
- Real phishing that reaches inboxes also **tests the email filter**: ideally it never gets that far.
- Users should know not to open links or attachments from unexpected email, and there should be a simple, well-known way to **report** a suspected phish to the security team.

Anomalous behavior recognition

- Watch for behavior that's out of the ordinary on users' machines, in three groups:
  - **Risky:** editing the hosts file, replacing a core OS file, uploading sensitive files.
  - **Unexpected:** a login from another country, a sudden jump in data transfers.
  - **Unintentional:** honest mistakes, such as typing the wrong domain, losing a USB drive, or misconfiguring a security setting.

Reporting and monitoring

- None of this is visible unless it's **monitored automatically**, with alerts and daily reports: phishing click rates, password manager adoption, MFA use and similar metrics.
- **First offense:** training, so the user understands the issue. **Repeat offense:** extended training, and possibly tighter security settings for that user.

Development and execution

- A **security awareness team** creates the material (emails, posters, classroom and online training), tailors it to job roles and to any compliance requirements, and tracks metrics over time to show managers and stakeholders whether the program is working.

User training

- Train people **before they first connect** to the network, and tailor it by role: accounting and shipping face different risks. Include **third parties** that connect, such as contractors, partners and suppliers,

and keep records of who has been trained.

- **Policies and handbooks:** document the security policies and make them easy to find, on the intranet and in the employee handbook.
- **Situational awareness:** stay alert during normal work for suspicious links, attachments, odd URLs and text messages, and for physical tricks such as an official-looking envelope containing a USB drive.
- **Insider threats** are hard to spot, so layer controls: multiple approvals for changes to critical processes, active **file monitoring** that alerts on changes, and processes that are hard to get around.
- **Password management:** set length and complexity requirements, and enforce them technically where possible (for example, Group Policy in Windows).
- **Removable media and cables:** an unknown USB drive can carry malware, and a stranger's charging cable can be malicious.
- **Social engineering:** teach the common techniques so users can recognize them and report them.
- **Operational security (OPSEC):** see your own work as an attacker would, and know which data is sensitive enough to need extra care.
- **Hybrid and remote work:** no family or friends on work devices, stronger endpoint security outside the office, and tighter VPN security.

Remember for the exam

- A **phishing simulation** measures click rates, tests the email filter, and routes clickers to training.
- Anomalous behavior comes in three kinds: **risky, unexpected, unintentional**. Know an example of each.
- **Monitoring and reporting** must be automated; metrics show whether training works.
- Train **before network access**, tailor by role, include **third parties**, and record completion.
- **Insider threat** controls: multiple approvals, file monitoring, hard-to-bypass processes.
- **OPSEC** = thinking like an attacker about the information you handle.

Check yourself

1. A user plugs in a USB drive that came in an official-looking envelope. Which training topic would have helped? *(Removable media, and situational awareness about physical attacks.)*
2. Monitoring shows a user's account suddenly sending far more data than usual. Which behavior category? *(Unexpected.)*
3. An employee mistypes a domain and lands on a lookalike site. Which category? *(Unintentional.)*
4. The same user clicks a test phish three months in a row. What's the next step? *(Extended training, and possibly stricter security settings for that user.)*
5. Why should contractors get security training too? *(They connect to the same systems and can introduce the same risks.)*

Do next

Write the checklist you would give a new employee for spotting a phishing email, from memory. That's the end of the course: take a full practice exam, then rewatch the objectives behind your wrong answers.
