# Securing Devices

AP Cybersecurity

## Device Vulnerabilities and Attacks

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

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

- **Virus** 病毒 - must be activated by a user opening a file.
- **Worm** 蠕虫 - spreads by itself, with no human action.
- **Trojan** 木马 - hides inside software that looks safe; a **remote access trojan (RAT)** 远程访问木马 gives the adversary remote control.
- **Ransomware** 勒索软件 - encrypts your files and demands payment for the key.
- **Spyware** 间谍软件 - secretly tracks what you do.
- **Keylogger** 键盘记录器 - records every keystroke to steal passwords.
- **Logic bomb** 逻辑炸弹 - triggers only when a condition is met (a date, a version).
- **Rootkit** - deeply hides in the operating system and can even make itself invisible.

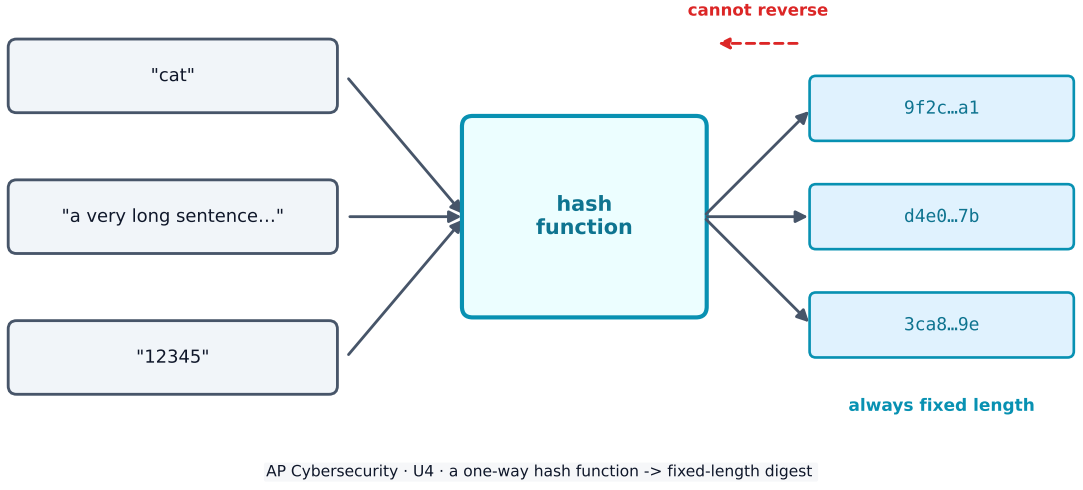
Most malware is a file, but **fileless malware** 无文件恶意软件 is different: it lives only in **RAM** 内存 and abuses legitimate programs already on the device, leaving no file for a scanner to find.

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

# Authentication


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

To store passwords safely, systems use a **cryptographic hash function** 密码散列函数 - a one-way maths algorithm that turns any input into a fixed-length string called a **hash** 散列值 (or digest). Hashes have three vital properties: they are **collision resistant** 抗碰撞 (hard to find two inputs with the same output), have **pre-image resistance** 抗原像 (you cannot work backwards to the input), and are **repeatable** (the same input always gives the same hash).




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

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

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

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

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

- **brute force** 暴力破解 - an automated tool tries *every* possible password in turn; guaranteed to work eventually, but slow, and it grows explosively with password length.
- a **dictionary attack** 字典攻击 - the tool tries a list of common words and known passwords first, because most people pick guessable ones.
- **password spraying** 密码喷洒 - one common password against many accounts (this dodges lockout, which counts failures per account).
- **credential stuffing** 撞库 - reusing stolen or default credentials, exploiting that people reuse passwords across sites.
- a **rainbow table** 彩虹表 - a precomputed table of passwords and their hashes, sorted by hash, so a captured hash can be looked up instead of recomputed.

## Password policy settings

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

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

One subtlety worth a mark: some national standards now advise **against** forced expiry, because regular changes push users into predictable patterns like `PasswordFall2028`. A **password manager** 密码管理器 solves the real problem - it generates and stores a long, unique password per site, so none is ever reused or guessable.

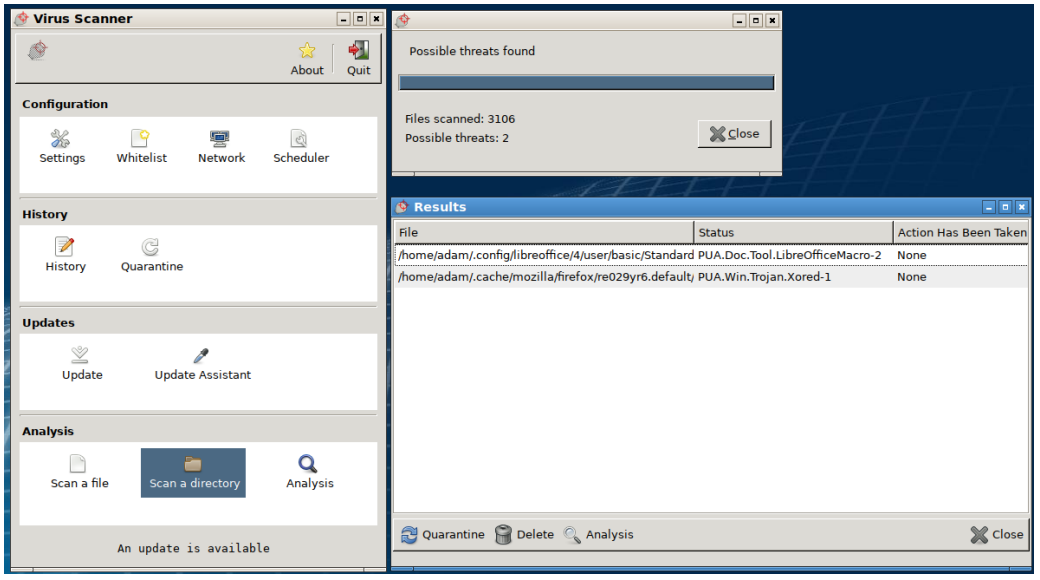
**Authentication factors** prove who you are, and fall into categories: something you **know** (a password), something you **have** (a token or phone), something you **are** (a **biometric** 生物特征 like a fingerprint or retina scan), and somewhere you **are** (a location factor). Using two or more is **multifactor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

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

# Protecting Devices

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

Technical controls do the work. **Anti-malware software** 反恶意软件 keeps a database of malware **signatures** and quarantines any file that matches. Keeping the operating system and applications **updated** - installing each **patch** 补丁 - closes known holes before adversaries can use them. A **host-based firewall** 主机防火墙 controls traffic in and out of one single device, blocking ports and services it does not need.




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

# Detecting Attacks on Devices

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

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

# Exam tips

- Know each **malware type by its defining trait**: a worm self-spreads, a virus needs a user, ransomware encrypts for money, a RAT gives remote control, a rootkit hides.
- A hash is **one-way and fixed-length**; **salt** makes identical passwords hash differently. Never say a service "stores the password" - it stores the **salted hash**.

- Name real algorithms: **SHA-256/SHA-512** are current; **MD5** and **SHA-1** are **deprecated** because efficient collision attacks exist.
- Match the password attack to its log signature: one user + many wrong passwords = guessing; many users + one IP = **spraying**; default credentials = **stuffing**.
- Sort authentication factors into **know / have / are / where**, and remember **MFA** combines two or more - a fingerprint plus a password, not two passwords.
- **Offline password attacks cannot be detected** because the cracking happens on the adversary's machine - a favourite exam "gotcha".