Securing Devices

AP Cybersecurity ▪ Unit 4

AP Cybersecurity

What we will cover

1. Devices & malware types
2. Hashing & salt
3. Password attacks
4. Authentication factors
5. Protecting devices
6. Detecting attacks

1 Malware

Devices, and the malware that hunts them

A device is any computer – server, laptop, phone, or an **embedded computer** 嵌入式计算机. Everyday embedded devices are **IoT** 物联网: water pumps to washing machines.

> The main threat is **malware** 恶意软件 – and each type is defined by one trait.

Biometric login: something you are

The malware types, by how each behaves

virus 病毒	attaches to a host file or program and runs when that host executes
worm 蠕虫	spreads by itself, with no human action – which is why it spreads so fast
spyware 间谍软件	secretly tracks what you do
logic bomb 逻辑炸弹	lies dormant and triggers only when a condition is met – a date, a version, a name removed from payroll
rootkit 根工具包	buries itself in the operating system and can make itself invisible to the tools looking for it
ransomware 勒索软件	encrypts the victim's data and demands payment for the key

> Adversaries get in through **unpatched software** 未打补丁的软件, weak passwords, unprotected BIOS/UEFI settings and open ports – and **IoT** 物联网 devices, which are rarely patched at all.

The malware families worth naming

Trojan	A **trojan** arrives disguised as something wanted – an installer, a document – and runs with whatever rights the user had.
Remote access trojan	A **remote access trojan (RAT)** gives the attacker an interactive session: files, camera, keystrokes.
Ransomware	Encrypts the files and sells the key. Backups, offline and tested, are the only reliable answer.
Worm	Spreads by itself across a network with no user action – which is why patching is a defence against it and awareness is not.

An **indicator of compromise (IOC)** is the evidence left behind: a hash, a domain, a registry key. Detection is built from them.

Fileless malware, and what adversaries exploit

Most malware is a file. **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.

> It leaves no file for a scanner to find – which is exactly why signature scanning misses it.

2 Hashing

What can these hash results tell you?

A **hash function** 密码散列函数 turns input data into a fixed-length **hash** 散列值.
These are real SHA-256 results for UTF-8 text with no final newline.

Input	First 16 hex characters	Full hex length
`cat`	`77af778b51abd4a3`	64
`cat, again`	`77af778b51abd4a3`	64
`cats`	`d936608baaacc6b7`	64

Which rows illustrate repeatability? What stays fixed when the input length changes?
Do these three results prove that no two different inputs can ever share a hash?

Only prefixes are shown. A full hash comparison must use every output bit.

Same input, fixed length, different questions

Repeatable: the two `cat` inputs produce the same full hash. **Fixed length:** both `cat` and `cats` give 64 hex characters. Each hex character represents 4 bits:

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

The different `cats` result illustrates sensitivity to an input change. It does not prove that collisions are impossible.

collision resistant 抗碰撞 means finding two different inputs with the same full hash is computationally difficult, not that no such pair exists.

pre-image resistance 抗原像 means recovering a suitable input from a hash is computationally difficult in general. Weak passwords can still be guessed.

Why a collision must exist

Imagine a **toy 4-bit hash**. It is deliberately too small for security. Its output can be any one of these 16 bit patterns:

0000	0001	0010	0011
0100	0101	0110	0111
1000	1001	1010	1011
1100	1101	1110	1111

$$N = 2^n = 2^4 = 16$$

Hash **17 different inputs**: at least two must share an output because there are only 16 possible outputs.

Real cryptographic hashes have far larger output spaces. Collisions still exist; the security goal is to make finding one infeasible.

One-way does not mean unguessable

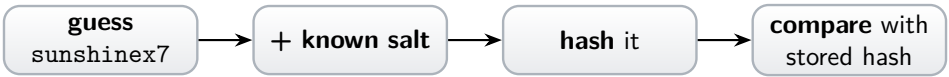
A stolen record contains salt `x7` and a hash whose prefix is `f9325402a7a1542a`. An adversary tests two guesses offline:

Guess	Input after salt	SHA-256 prefix
`rainbow`	`rainbowx7`	`2a0e6ce47f2c701d`
`sunshine`	`sunshinex7`	`f9325402a7a1542a`

Which guess should be checked against the *full* stored hash? Was a hash decrypted? Would the website's account lockout stop this computation?

SHA-256 and the short salt illustrate the idea only. Real password storage needs a purpose-built slow password hash and a suitably sized unique random salt.

Guess, hash, compare: no decryption step



`sunshinex7`: full hashes match `rainbowx7`: no match – no reverse or decryption step

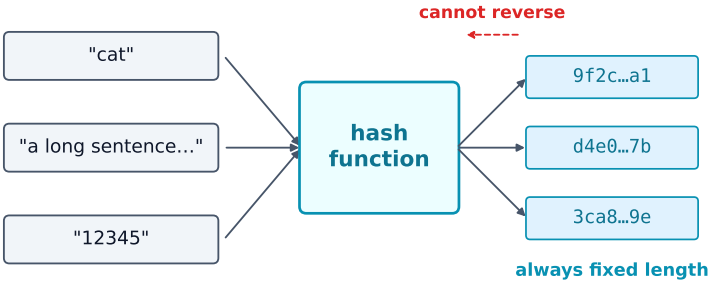
- A unique **salt** blocks reuse of one unsalted lookup table across accounts. It is stored with the hash, need not be secret, and does not make a weak password strong.
- **Account lockout** limits live login attempts but cannot stop computations on a stolen copy; those guesses create no login events (the theft or later use may still leave evidence).

Real hash functions, and how they are attacked

SHA 安全散列算法	the Secure Hash Algorithm family – SHA-256 and SHA-512 are today's standard
brute force 暴力破解	tries all candidates in a chosen search space; exhaustive search can be impractical
dictionary attack 字典攻击	tries common words and known passwords first, because most people choose guessable ones
rainbow table 彩虹表	precomputed hashes looked up instantly – which is precisely what a salt defeats
collision 碰撞	two inputs with the same hash; an efficient collision attack undermines the function; MD5 and SHA-1 are deprecated

For length L and A choices per position, the candidate count is $N = A^L$. Real attacks try likely choices first.

The cryptographic hash function, in detail



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

schematic labels, not computed digests

A **hash function** 密码散列函数 is one-way maths turning any input into a fixed-length **hash** 散列值.

Worked example – why salt matters

Two users both choose `sunshine`

❌ **No salt:** both stored hashes are identical – cracking one instantly cracks the other.

✅ **With salt 盐值:** give each a unique random salt – shortened here to `x7` and `q2`. The service hashes `sunshinex7` and `sunshineq2`, so the stored hashes look completely different.

> Different salts prevent reuse of one unsalted lookup table across accounts. Weak passwords remain guessable; salt is stored with the hash.

A secure service stores a salted password hash. Salt is not a substitute for a slow password-hashing scheme.

Password attacks – and their log signatures

Password spraying 密码喷洒
one common password against many accounts
indicator: many users failing in quick succession

Credential stuffing 撞库
stolen or default credentials (AP CED usage)
CED indicator: a burst of default-login attempts

Rainbow table 彩虹表
a precomputed table of passwords and hashes

Guessing
log: one user + many wrong passwords

Online attacks submit credentials to a live login. Offline attacks steal the hash database and crack it on the adversary's own machine – those guesses generate no live login events. Theft or later account use may still leave evidence.

Each password-policy setting slows a specific attack

complexity 复杂度	a character from each set – upper, lower, digit, special. Slows brute force and dictionary attacks
minimum length 最小长度	*N* characters. Length matters more than anything else: brute-force cost grows exponentially with it
maximum age 最长有效期	an expiry setting; not a universal recommendation for password security
password history 密码历史	stores the last 5–10 hashes and blocks reuse, stopping recycling of an old, possibly leaked, password
lockout 锁定	locks the account after 3–5 wrong tries – ends online guessing outright

> NIST SP 800-63B-4: do not require periodic password changes or character-class composition rules. Change compromised passwords; block common and compromised choices.

Authentication factors

something you **know**
a password

something you **have**
a token or phone

something you **are**
a **biometric**
生物特征

somewhere you **are**
a location factor

MFA 多因素身份验证 combines two or more categories – a fingerprint plus a password, not two passwords.

Authentication, and the devices to defend

Multifactor authentication	**Multifactor authentication (MFA)** combines factors: something you know, have, and are. A stolen password alone is then not enough.
Password manager	A **password manager** makes long unique passwords practical – which removes reuse, the single biggest cause of account takeover.
Endpoint detection	**Endpoint detection and response (EDR)** watches behaviour on the device itself and can isolate it from the network.
Internet of things	**Internet of things (IoT)** devices ship with default credentials, rarely update, and sit inside the network – the weakest link in most homes.

> Defence in depth: assume each control fails, and ask what the next one catches.

3 Protect & detect

Protecting devices

Managerial sets the rules: an **acceptable use policy** 可接受使用政策, a password policy (length, reuse), a software installation policy.

Anti-malware 反恶意软件 – matches signatures, quarantines

each **patch** 补丁 closes a known hole first

a **host-based firewall** 主机防火墙 guards one device

> Patching closes known holes before adversaries can use them – most breaches use a hole with a patch already available.

Detecting attacks – indicators of compromise

Logs reveal an **indicator of compromise** 入侵指标 – evidence an adversary got in:

- **host-based** – unexpected processes, changed settings
- **file-based** – a file whose hash matches known malware
- **behaviour-based** – many failed logins, unusual hours

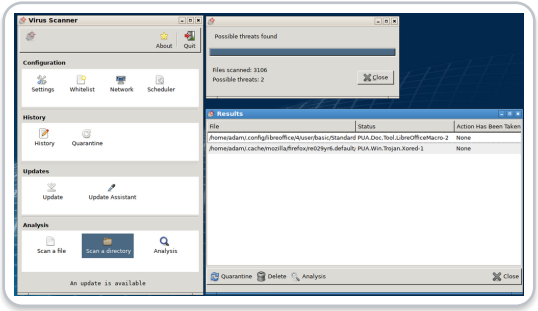
Choosing a method weighs performance (signature is lighter – better for weak devices), cost (**EDR** 端点检测与响应 is powerful but expensive), and how sensitive the device is.

Detecting what got past the controls

indicator of compromise 入侵指标	IoC: evidence a system has already been breached – an unexpected outbound connection, a new admin account, a changed system file
EDR 端点检测与响应	endpoint detection and response: continuously watches each device's behaviour, and can isolate one automatically
keylogger 键盘记录器	records keystrokes – an IoC in itself, and a way credentials leave
logging	an IoC is only visible if something recorded it, which is why log retention is a control in its own right

> Prevention fails eventually. Detection is what turns a breach lasting months into one lasting hours, and that difference is most of the cost.

How anti-malware scans and quarantines



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

Exam tips

- Know each **malware type by its defining trait**: a worm self-spreads, a virus attaches to a host and runs when the host executes, ransomware encrypts for money, a RAT gives remote control, a rootkit hides.
- A hash is **one-way and fixed-length**; a unique **salt** changes the hashed input even when passwords match. Secure storage uses a **salted password hash**; raw SHA-256 is too fast for password storage.
- Name real algorithms: **SHA-256/SHA-512** are current; **MD5** and **SHA-1** are **deprecated** because efficient collision attacks exist.

Exam tips (continued)

- Match the password attack to its log signature: cite failed-login entries, timing and users. These are indicators, not proof. AP CED includes default-credential attempts under **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 guessing creates no live login events**: it happens on the adversary's copy. The theft or later credential use may still be detected.

4 Recap

Unit 4 – key takeaways

1 One trait each

worm self-spreads; virus needs a host; rootkit hides

2 Salted hash

secure storage uses a salted password hash

3 Log signatures

cite failed attempts, users and timing; a pattern needs context

4 MFA

two different categories – not two passwords

Check yourself

1. Worm or virus: which attaches to a host program?
2. Why do offline guesses create no live login events?
3. Are a password and a PIN multifactor authentication?
4. What does a unique salt change when passwords match?

Answers **1.** virus **2.** guesses run on a stolen copy **3.** no – both are "something you know" **4.** the input to the hash function