Securing Applications and Data

AP Cybersecurity ▪ Unit 5

AP Cybersecurity

What we will cover

1. Injection attacks
2. Data states & access control
3. Linux permissions
4. Symmetric cryptography
5. Asymmetric cryptography
6. Protecting & detecting

The biggest danger is unchecked user input

When a program does not check what a user types, an adversary can slip in commands – an **injection attack** 注入攻击. **Data validation** 数据验证 is the defence.

SQL injection 数据库注入
SQL commands
into an input field
log: `OR 1=1` and `--`

Cross-site scripting 跨
script that runs in
other user's brows
log: `<script>` tag

Buffer overflow 缓冲区溢出
more data than the
buffer 缓冲区 holds
log: unusually long input

Directory traversal 目
`../` to reach files
outside the folde
log: `../` sequence

Crypto enigma supports the idea on this slide

Rate data risk by sensitivity: unencrypted

Data states, and the laws that follow them

At rest 静态数据
– stored on a drive

In transit 传输中数据 –
moving between devices

In use 使用中数据
– being processed

Enigma machine supports the idea on this slide

Four access-control models – know the decider

RBAC 基于角色的访问控制
– your role decides

RuBAC 基于规则的访问控制
– a condition decides

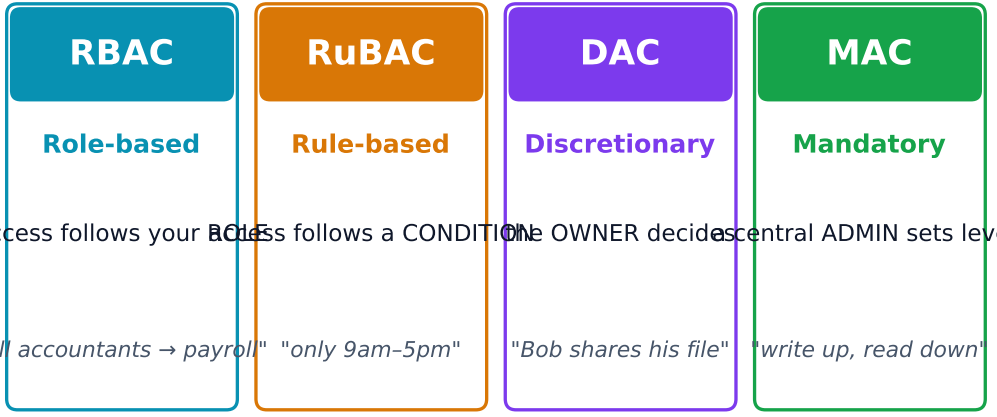
DAC 自主访问控制 –
the file's owner decides

MAC 强制访问控制 –
a central admin decides

Padlock crypto supports the idea on this slide

The **Bell-LaPadula** model summarises MAC as "write up, read down".

Four access-control models – know the decider



Linux permissions – add 4 + 2 + 1

Each file has read (4), write (2) and execute (1) for three groups: the owner, the group, and others.

> `chmod 750` = owner `rwx` (7), group `r-x` (5), others nothing (0).

read/write/execute for owner, group, others

Worked example – setting a permission

Only the principal may read *and* edit; her staff group may read; no one else may touch it

owner read+write $= 4 + 2 = 6$
group read $= 4$
others nothing $= 0$

```
chmod 640 file
```

The listing then shows `-rw-r-----`.

To also let the owner run it as a program, add execute: $4 + 2 + 1 = 7$, giving `chmod 740`.

Practise converting both ways – number to letters, and letters to number.

Symmetric encryption – one shared key

Cryptography 密码学 combines **plaintext** 明文 with a **key** 密钥 to make **ciphertext** 密文.

> The **keyspace** 密钥空间 is the number of possible keys. An *n*-bit key has a keyspace of 2^n – bigger means longer to guess.

> **Symmetric** 对称加密 uses the same key both ways. The standard is **AES** 高级加密标准, a **block cipher** 分组密码 on 128-bit blocks.

It secures Wi-Fi, browsing and stored files. The challenge: both sides need the same secret key – so how do you share it safely?

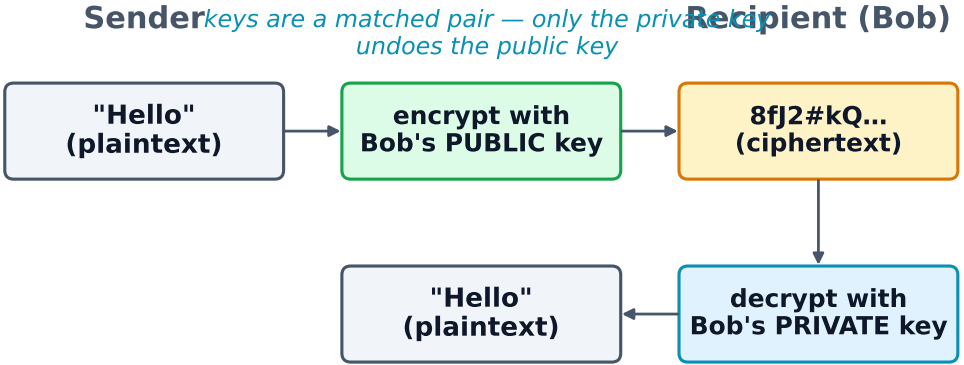
Asymmetric encryption – the key pair

A **key pair** 密钥对: a **public key** 公钥 anyone may see, and a **private key** 私钥 kept secret. They are mathematical inverses – what one locks, only the other unlocks.

> Encrypt with the recipient's public key; only their private key decrypts – so no secret was ever shared in advance.

RSA and **ECC** 椭圆曲线密码学 are the common algorithms. Compare key lengths only within the same algorithm – RSA 4096 is not comparable to AES 256.

Asymmetric encryption – the key pair



Protecting applications

Secure by design 安全设计
security in every phase of development, not an afterthought

Secure by default 默认安全
it ships with security already enabled – safe out of the box

> **Input sanitization** 输入清理 is the single best answer for preventing injection attacks.

Control characters 控制字符 – the single quote, double quote and semicolon – can manipulate a system, so a good program removes or rejects them before processing. One defence covers SQL injection, XSS and traversal alike.

Detecting attacks on data

Systems perform **accounting** 审计记录 – logging who accessed what, and when.

> A **honeypot** 蜜罐 is a fake file that looks valuable. Nobody has a real reason to open it – so any access is a clear sign of an attack.

> Re-hash and compare: if the digest changed, the file was altered.

Weigh cost against sensitivity: honeypots are cheap; a **DLP** 数据泄露防护 service is powerful but pricey.

Unit 5 – key takeaways

1 Read the log

OR 1=1 = SQLi; <script> = XSS; ../ = traversal

2 The decider

RBAC role, RuBAC condition, DAC owner, MAC admin

3 chmod

add 4+2+1 per group – owner, group, others

4 Two cryptos

symmetric shares one key; asymmetric solves sharing

Check yourself

1. A log shows `../../etc/passwd`. Which attack?
2. Whose key do you encrypt with to send me a secret?
3. What is `chmod 640` in letters?
4. Why is any access to a honeypot suspicious?

Answers **1.** directory traversal **2.** my public key **3.** `-rw-r-----` **4.** nobody has a real reason to open it