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

1 Injection

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 跨站脚本
script that runs in another user's browser
log: `<script>` tags

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

Directory traversal 目录遍历
`../` to reach files
outside the folder
log: `../` sequences

Rate data risk by sensitivity: unencrypted military plans are high; customer data with a weak key moderate.

The application attacks the exam names

buffer overflow 缓冲区溢出	sending more data than the **buffer** 缓冲区 holds, so it spills into nearby memory and may run the adversary's code
SQL injection SQL 注入	input crafted so the database reads it as commands rather than data
cross-site scripting 跨站脚本	XSS: injected script runs in another user's browser, under that site's trust
unencrypted storage	if files are stored unencrypted, anyone who reaches them can read them – no exploit needed
the common root	all of them are unchecked input or unprotected data; validating input and encrypting at rest closes most of it

> Say what the attack *does to the machine*, not just its name. "Overflows into adjacent memory and executes" is the answer; "too much data" is not.

2 Access control

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



The Enigma machine: ciphers have long protected data in transit

The three regulated data types, and their laws

PII 个人身份信息	personally identifiable information – name, address, SSN, biometrics, date of birth. Governed by the Privacy Act and state breach laws
PHI 受保护健康信息	protected health information – health, treatment and healthcare-payment records. Governed by HIPAA (1996)
PCI 支付卡信息	payment card information – card number, expiry, CVV, cardholder name. Governed by PCI-DSS, an industry standard rather than a law
the obligation	whoever collects it must label it and keep its storage, transmission and handling **compliant** 合规
DLP 数据泄露防护	data loss prevention inspects data leaving the organisation and blocks what matches these patterns

> Higher sensitivity, stricter handling. Compliance is not security: a compliant system can still be insecure.

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

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

The four access-control models

Discretionary (DAC)	**Discretionary (DAC)**: the owner of a resource decides who may use it – flexible, and only as careful as the owner is.
Mandatory (MAC)	**Mandatory (MAC)**: the system enforces labels (Secret, Top Secret) and no user can override them. Military-grade, and rigid.
Role-based (RBAC)	**Role-based (RBAC)**: permissions attach to a *role*, and people are put into roles. The model most organisations actually run.
Rule-based (RuBAC)	**Rule-based (RuBAC)**: conditions decide – time of day, source address, device posture.

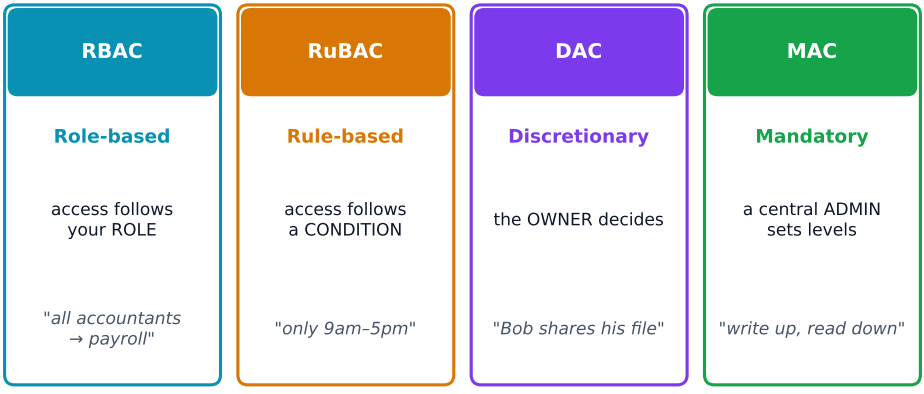
All four implement the **principle of least privilege**: each account gets exactly the access its job needs, and no more.

The principle every access-control model serves

least privilege 最小权限原则	give each entity exactly the access it needs, and no more
why	it bounds the damage of any compromised account – an attacker inherits only what that account could do
separation of duties 职责分离	no single person can complete a sensitive action alone
need to know 知情必要	access to data granted only for a current, specific task
the practical test	could this account, if stolen tonight, reach anything it has not used in a year?

> Least privilege is the one idea running through all four models. Quote it by name and then apply it to the scenario.

Four access-control models – know the decider, in detail



AP Cybersecurity · U5 · four access-control models by their 'decider'

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

Linux permissions – add 4 + 2 + 1

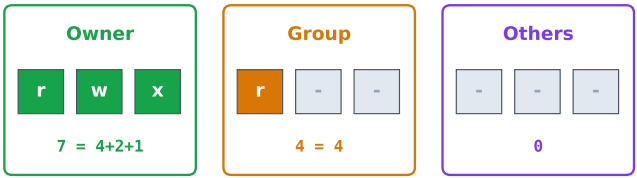
Each file has read (4), write (2) and execute (1) for three groups: the owner, the group, and others.

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

Linux permissions – add 4 + 2 + 1, in detail

Example: chmod 740 report.txt

read = 4 write = 2 execute = 1



AP Cybersecurity · U5 · Linux file permissions (rwx for owner/group/other)

read/write/execute for owner, group, others

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

3 Cryptography

Symmetric encryption – one shared key

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

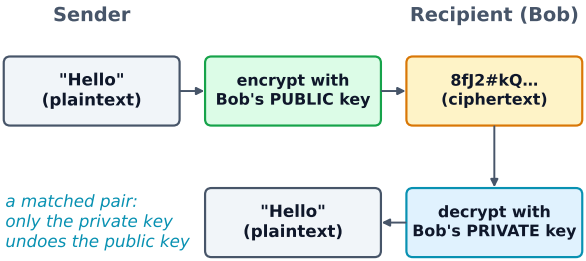
> The **keyspace** 密钥空间 is the number of possible keys. An *n*-bit key has a keyspace of 2^n – bigger means longer to guess.

> **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 is shared in advance. **RSA** and **ECC** 椭圆曲线密码学 are the common algorithms; compare key lengths only within one algorithm (RSA 4096 is not comparable to AES 256).

4 Protect & detect

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

> Match the defense to the destination: SQL, browser output or file path.

Special characters 特殊字符 can be legitimate data. Bind SQL values; safely render or encode browser output; confine resolved file paths. No universal punctuation filter protects all three destinations.

Two application weaknesses, and two defences

Cross-site scripting	**Cross-site scripting (XSS)**: an attacker's script is stored or reflected by a site and then runs in another user's browser, with that user's session.
The fix	Validate input, and escape output for the context it lands in. Never trust data because it came back from your own page.
Password strength	Length beats complexity, but **special characters** still widen the search space – and a manager makes long random passwords usable.
ECC	**Elliptic curve cryptography (ECC)** gives the same strength as RSA with far shorter keys, which is why phones and smart cards use it.
Administrative controls	**Administrative** controls – policy, training, background checks – sit beside the technical and physical ones.

> Every control belongs to one of three families: technical, physical, administrative. A plan

Secure by design: three commitments

1. own the outcome	take ownership of customers' security outcomes rather than shifting blame onto users for misconfiguring a product
2. radical transparency	publish vulnerabilities and their fixes, so the whole industry learns from each one
3. lead from the top	build the organisational structure that makes secure design a leadership priority, not an engineering afterthought
in practice	secure defaults, memory-safe languages, and features that are safe *without* the customer configuring anything

> "Secure by design" means the safe path is the default path. If a product is only secure when correctly configured, it is not secure by design.

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.

Exam tips

- Match each application attack to its **evidence in a log**: `OR 1=1` / `--` = **SQL injection**; `<script>` = **XSS**; `../` = **directory traversal**; very long input = **buffer overflow**.
- Learn the four access-control models by their **decider**: RBAC = your role, RuBAC = a condition, DAC = the file's owner, MAC = a central admin. **Least privilege** underlies them all.
- Read Linux permissions by adding **4+2+1** per group - `chmod 750` = owner rwx (7), group r-x (5), others none (0). Practice converting both ways.
- **Symmetric** = one shared key (fast, AES); **asymmetric** = a public/private key pair (solves key sharing, RSA/ECC). Encrypt with the **recipient's public key**.
- **SQL**: bind values. **XSS**: context-specific output. **Paths**: confine resolved paths. A **honeypot** detects access.

5 Recap

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