AP CYBERSECURITY • EXERCISE SHEET

5.1 Application and Data Vulnerabilities and Attacks

Name: _____ Class: _____ Date: _____ **Total: 45 marks**

KEY WORDS buffer 缓冲区 • buffer overflow 缓冲区溢出 • sql injection 数据库注入 • directory traversal 目录遍历 • administrative 管理性

Objective

Build the skills to answer exam questions on application and file vulnerabilities: **SQL injection** 数据库注入, **XSS** 跨站脚本, **buffer overflow** 缓冲区溢出, **directory traversal** 目录遍历, and rating data risk.

You must be able to:

- explain how unencrypted files and weak access control expose data
- distinguish a standard user from an administrative user, and why that matters
- identify an application attack from log or URL evidence
- explain how unchecked user input enables injection attacks
- name **data validation** 数据验证 as the core defence and say what it does
- rate a data risk high, moderate or low against confidentiality, integrity and availability

1 Worked examples

Study these first. Each one shows the method for a task used later.

■ Before any clever attack: files and permissions

Two vulnerabilities need no exploit at all.

- **Unencrypted files.** Anyone with access to the device or the drive can read them. Encryption is what makes physical access insufficient.
- **Weakly configured access control.** When many users can view - or edit - files they do not need, an adversary who compromises **any one** of those accounts inherits all of it. **Administrative users** can change system settings and typically reach any file or application, so a compromised admin account is far more damaging than a compromised standard one.

■ The four application attacks

Applications are **executable data**. Some run locally, others - like web applications - run on a server the user reaches through a browser. Most take **open-ended input fields**, and that is where three of these four begin.

Attack	What the adversary sends	What it achieves
SQL injection	SQL commands and control characters into an input field	the database returns or changes data it should not - a breach of **confidentiality** or **integrity**
Cross-site scripting (XSS)	malicious JavaScript injected into a website, or embedded in a link	the **victim's browser executes** it, because the browser trusts the site
Buffer overflow	more data than the fixed-size **buffer** was allotted	the system crashes, or executes code outside the program's security policy
Directory traversal	a modified URL or GET request	reaches files elsewhere on the server's file system - usernames, passwords, configuration

Recognising them from evidence. A login field containing `' OR '1'='1` is **SQL injection**. A URL containing `../../etc/passwd` is **directory traversal**. A comment field containing `<script>` that later runs for other visitors is **XSS**. An input far longer than the field expects, followed by a crash, is a **buffer overflow**.

■ Why unchecked input is the common cause

The application treats what the user typed as **instructions** rather than as **data**. In SQL injection the typed characters become part of the query; in XSS they become part of the page the next visitor loads; in a buffer overflow they run past the memory the program reserved and into memory it did not.

Data validation is therefore the core defence: the developer checks the input before it is used - the right type, the right length, the right characters - and rejects or escapes anything else. Validation for length is exactly what prevents the overflow; validation for control characters is what prevents the injection.

■ Rating a data risk

Say **which** of confidentiality, integrity or availability is compromised, then rate it:

Rating	When
High	highly sensitive data - especially data governed by law or regulation - through a highly likely exploit
Moderate	sensitive data whose encryption is not strong enough, or whose access controls are not strict enough
Low	less sensitive data with short keys or loose access controls

The pattern to notice: the rating rises with **both** the sensitivity of the data **and** the likelihood of the exploit. Neither alone decides it.

2 Practice

2.1 Explain why storing files unencrypted is a vulnerability even when the building is locked. [2]

2.2 State **two** things an administrative user can do that a standard user cannot, and explain why this matters to an adversary. [3]

2.3 Name the attack shown by each piece of evidence.

(a) A URL containing `../../../etc/passwd`. [1]

(b) A login field submitted as `' OR '1'='1`. [1]

(c) A product review containing `<script>` that runs for later visitors. [1]

(d) 5000 characters typed into a 32-character field, followed by a crash. [1]

2.4 Explain how an SQL-injection attack breaches confidentiality. [2]

2.5 Explain why a cross-site scripting attack succeeds even though the malicious code came from another user. [2]

2.6 State what a **buffer** is and explain what happens in a buffer overflow. [3]

2.7 Explain how **data validation** prevents each of these.

(a) A buffer overflow. [1]

(b) An SQL injection. [1]

2.8 For each case, name the CIA principle compromised and rate the risk.

(a) A payroll database governed by data-protection law is reachable by any employee account. [2]

(b) An internal lunch-menu file is encrypted with a short key. [2]

(c) A customer address list is encrypted, but with an algorithm the policy no longer approves. [2]

3 Exam-style questions

3.1 A URL modified to reach files elsewhere on the server describes [1]

- **A** SQL injection
- **B** cross-site scripting
- **C** directory traversal
- **D** buffer overflow

3.2 The core defence against injection attacks is [1]

- **A** stronger passwords
- **B** data validation of user input
- **C** encrypting the database at rest
- **D** a network firewall

3.3 A buffer overflow can allow an adversary to [1]

- **A** read the database without authenticating
- **B** execute code outside the program's security policy
- **C** modify a URL to reach another directory
- **D** run script in another user's browser

3.4 Which risk is best rated **moderate** rather than high? [1]

- **A** regulated medical data reachable through a widely known exploit
- **B** sensitive data protected by encryption that is not strong enough
- **C** a public product catalogue with loose access controls
- **D** a test file containing no real data

3.5 A small online bookshop runs a web application. The product search box passes what the user types straight into a database query. Customer reviews are displayed exactly as submitted. Order confirmation files are stored unencrypted in a directory served by the same web server, and every member of staff has an administrative account "so nobody is ever blocked". A log shows a request for `/orders/../../config/db.conf`, and later a review containing a `<script>` tag.

(a) Name the attack in the logged request and explain what the adversary was trying to obtain. [3]

(b) Name the attack the review represents and explain who is harmed and how. [3]

(c) Explain how the search box could be exploited, naming the attack and the CIA principles at risk. [3]

(d) Explain why giving every member of staff an administrative account multiplies the damage of any one of these attacks. [3]

(e) Recommend **four** changes, each addressing a different vulnerability above, and rate the risk the bookshop faces before the changes, justifying the rating. [5]

Solutions

2.1 an adversary who gains access to the device or the drive can read any unencrypted file directly; physical security can fail - a stolen laptop, a disposed disk, a remote compromise - and encryption is what makes that access useless.

2.2 any two of: change system settings; access any file or application on the system; install software; create or alter accounts. It matters because compromising one administrative account gives the adversary all of that at once, whereas a standard account is limited to what that user could already reach.

2.3 (a) directory traversal. (b) SQL injection. (c) cross-site scripting. (d) buffer overflow.

2.4 SQL commands and control characters typed into an input field become part of the query the application runs, so the database returns records the user was never authorised to see - unauthorised access to sensitive data, which is a breach of confidentiality.

2.5 the malicious JavaScript is stored in or reflected by the website, and the victim's browser executes it because it arrived from a site the browser trusts, not from the attacker.

2.6 a buffer is a designated section of memory with a **fixed size** where user input is written; if more data is supplied than was allotted, it runs past that section into adjacent memory, which can crash the system or cause it to execute code outside the program's security policy.

2.7 (a) it checks the **length** of the input and rejects anything longer than the buffer can hold. (b) it checks the characters, rejecting or escaping the control characters and SQL keywords so the input is treated as data rather than as commands.

2.8 (a) confidentiality, and integrity if it can be edited; **high** - the data is governed by law and the weak access control makes exploitation highly likely. (b) confidentiality; **low** - a short key is a real weakness, but a lunch menu is not sensitive, so the impact is minimal. (c) confidentiality; **moderate** - the data is sensitive but is encrypted, just not strongly enough.

3.1 C. Modifying a URL or GET request to reach the server's file system is directory traversal.

3.2 B. Validation is what stops input being treated as instructions.

3.3 B. A is SQL injection, C is directory traversal, D is XSS.

3.4 B. A is high; C and D are low.

3.5 (a) **directory traversal**. The `../` sequences climb out of the orders directory to reach a database configuration file, which typically contains the credentials for the database itself - so one request could yield full access to all customer data.

(b) **cross-site scripting**. The harm falls on **other customers**, not on the bookshop's server: the script is stored in the review and executes in each later visitor's browser, where it can steal session cookies, capture what they type, or redirect them.

(c) **SQL injection**. Input passed straight into a query means typed control characters become part of the query, risking **confidentiality** if it returns other customers' orders

and **integrity** if it alters or deletes records; availability is at risk too if the injected command drops a table.

(d) every compromised staff account is an **administrative** account, so an adversary who phishes any one employee can change system settings, reach every file and install software - the attacks above stop being limited to what one user could see and become full control of the shop's data, which is a direct failure of least privilege.

(e) Four changes, each on a different vulnerability: validate and parameterise the search input so it can never be executed as SQL; escape or sanitise review content before display, so a script tag is shown as text; encrypt the order files **and** move them out of the web-served directory; and give staff standard accounts, reserving administrative rights for the few who need them. Rating: **high** - the data is customer orders and payment-related records, so it is sensitive and regulated, and the exploits are already being attempted in the logs, so likelihood is not theoretical but demonstrated.