

Exercise walk-through

5.1

Application and Data Vulnerabilities and Attacks

AP Cybersecurity

The application attacks the exam names

buffer over-flow

sending more data than the **buffer** holds, so it spills into nearby memory and may run the adversary's code

SQL injection SQL

input crafted so the database reads it as commands rather than data

cross-site scripting

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.

The three regulated data types, and their laws

PII

personally identifiable information – name, address, SSN, biometrics, date of birth. Governed by the Privacy Act and state breach laws

PHI

protected health information – health, treatment and healthcare-payment records. Governed by HIPAA (1996)

PCI

payment card information – card number, expiry, CVV, cardholder name. Governed by PCI-DSS, an industry standard rather than a law

the obligation

an organisation that collects regulated data must **label** it and hold policies keeping its storage, transmission and handling **compliant**

The higher the sensitivity, the stricter the handling. **Compliance** is not the same as security – a compliant system can still be insecure, and vice versa.

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 naming only the first is incomplete.

Exercises 2.1-2.2

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]

Exercises 2.1-2.2

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

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

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

Exercise 2.3 (a)

2.3 Name the attack shown by each piece of evidence.

(a) A URL containing `../../../../etc/passwd`.

[1]

Exercise 2.3 (a)

2.3 Name the attack shown by each piece of evidence.

(a) A URL containing `../../../../etc/passwd`.

[1]

(a)

- directory traversal

Exercise 2.3 (b)

2.3 Name the attack shown by each piece of evidence.

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

[1]

Exercise 2.3 (b)

2.3 Name the attack shown by each piece of evidence.

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

[1]

(b)

- SQL injection

Exercise 2.3 (c)

2.3 Name the attack shown by each piece of evidence.

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

[1]

Exercise 2.3 (c)

2.3 Name the attack shown by each piece of evidence.

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

[1]

(c)

- cross-site scripting

Exercise 2.3 (d)

2.3 Name the attack shown by each piece of evidence.

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

[1]

Exercise 2.3 (d)

2.3 Name the attack shown by each piece of evidence.

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

[1]

(d)

- buffer overflow

Exercises 2.4-2.5

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]

Exercises 2.4-2.5

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

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

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

Exercise 2.6

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

[3]

Exercise 2.6

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

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

Exercise 2.7 (a)

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

(a) A buffer overflow.

[1]

Exercise 2.7 (a)

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

(a) A buffer overflow.

[1]

(a)

- it checks the **length** of the input and rejects anything longer than the buffer can hold

Exercise 2.7 (b)

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

(b) An SQL injection.

[1]

Exercise 2.7 (b)

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

(b) An SQL injection.

[1]

(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

Exercise 2.8 (a)

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]

Exercise 2.8 (a)

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]

(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

Exercise 2.8 (b)

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

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

[2]

Exercise 2.8 (b)

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

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

(b)

- confidentiality; **low** - a short key is a real weakness, but a lunch menu is not sensitive, so the impact is minimal

Exercise 2.8 (c)

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

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

Exercise 2.8 (c)

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

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

(c)

- confidentiality; **moderate** - the data is sensitive but is encrypted, just not strongly enough

Exercises 3.1-3.2

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

Exercises 3.1-3.2

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

3.1 C

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

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

3.2 B

- Validation is what stops input being treated as instructions

Exercises 3.3-3.4

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

Exercises 3.3-3.4

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

3.3 B

- A is SQL injection, C is directory traversal, D is XSS

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

3.4 B

- A is high; C and D are low

Exercise 3.5 (a)

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]

Exercise 3.5 (a)

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]

(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

Exercise 3.5 (b)

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.

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

Exercise 3.5 (b)

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.

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

(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

Exercise 3.5 (c)

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.

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

[3]

Exercise 3.5 (c)

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.

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

(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

Exercise 3.5 (d)

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.

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

Exercise 3.5 (d)

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.

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

(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

Exercise 3.5 (e)

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.

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

Exercise 3.5 (e)

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.

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

(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