

Exercise walk-through

5.5

Protecting Applications

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 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.

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 2.1-2.3

2.1 Explain what 'secure by default' means, including one limit to that claim. [2]

2.2 Give a suitable control for each destination: a value in an SQL query; a plain-text comment on a web page; a user-requested file path. [3]

2.3 A developer deletes all apostrophes from names. Explain one usability problem and why this is not a complete SQL-injection defense. [2]

Exercises 2.1-2.3

2.1 Explain what 'secure by default' means, including one limit to that claim. [2]

2.1 Protective settings are enabled from the start. This does not guarantee no vulnerabilities or remove the need for updates.

2.2 Give a suitable control for each destination: a value in an SQL query; a plain-text comment on a web page; a user-requested file path. [3]

2.2 SQL: a fixed query with separately bound values. Browser: safe text rendering or context-specific output encoding. File path: resolve the path and enforce an allowed directory boundary.

2.3 A developer deletes all apostrophes from names. Explain one usability problem and why this is not a complete SQL-injection defense. [2]

2.3 It damages a legitimate name such as O'Neill. An attack on an unsafe numeric SQL expression can need no apostrophe; binding values separately is required instead of relying on that blacklist.

Exercises 3.1-3.2

3.1 Which control keeps an entered name as data rather than SQL syntax?

[1]

☐

A deleting only semicolons

B binding the value separately in a fixed query

C encoding the name as HTML

D accepting every value shorter than 20 characters

3.2 'Secure by design' means security is

[1]

☐

A added only after testing finishes

B considered throughout development

C delegated entirely to each user

D guaranteed by deleting punctuation

Exercises 3.1-3.2

3.1 Which control keeps an entered name as data rather than SQL syntax? [1]

☐

A deleting only semicolons

B binding the value separately in a fixed query

C encoding the name as HTML

D accepting every value shorter than 20 characters

3.1 B. Binding keeps the value separate from query syntax; HTML encoding addresses a different destination.

3.2 'Secure by design' means security is [1]

☐

A added only after testing finishes

B considered throughout development

C delegated entirely to each user

D guaranteed by deleting punctuation

3.2 B. Security is part of design, implementation and review rather than a final addition.

Exercise 3.3 (a)

3.3 A form joins an age input directly into `SELECT name FROM pupils WHERE age =` followed by the entered text. An adversary enters `0 OR 1=1`.

(a) State the attack risk.

[1]

Exercise 3.3 (a)

3.3 A form joins an age input directly into `SELECT name FROM pupils WHERE age =` followed by the entered text. An adversary enters `0 OR 1=1`.

- (a) State the attack risk. [1]
- (a) SQL injection: the input may alter the query condition and expose unintended rows.

Exercise 3.3 (b)

3.3 A form joins an age input directly into `SELECT name FROM pupils WHERE age =` followed by the entered text. An adversary enters `0 OR 1=1`.

(b) Explain how to keep that input from changing the SQL structure while still checking that a permitted age was entered. [2]

Exercise 3.3 (b)

3.3 A form joins an age input directly into `SELECT name FROM pupils WHERE age =` followed by the entered text. An adversary enters `0 OR 1=1`.

(b) Explain how to keep that input from changing the SQL structure while still checking that a permitted age was entered. [2]

(b) Use a fixed query with a bound value instead of joining input into SQL. Validate that the value is a number in the permitted age range, rejecting this expression as invalid input.