WORKSHEET

F11.8 Writing secure programs

Cyber security · GCSE · OCR J277 1.4.2, AQA 8525 3.6.3, Edexcel 1CP2 5.3.2 · about 15 min

BugBotLab

Name _____ Class _____ Date _____

What this lesson is about

Validating input, access levels in code, and testing for security.

- Never trust input
- Access levels in code
- More defensive habits
- Testing for security

Questions

5 marks in all

1. How does a program reduce the risk of SQL injection? [1 mark]

- ☐ **A** Validating input and keeping it separate from commands
- ☐ **B** Running faster
- ☐ **C** Using a bigger database
- ☐ **D** Encrypting the screen

2. What does 'fail safely' mean? [1 mark]

- ☐ **A** If something goes wrong, stop in a safe state and reveal nothing
- ☐ **B** Ignore all errors
- ☐ **C** Crash loudly with details
- ☐ **D** Keep running whatever happens

3. What is the principle of least access? [1 mark]

- ☐ **A** Give each user or program only the access its job needs
- ☐ **B** Give everyone admin access
- ☐ **C** Remove all access
- ☐ **D** Share one account

4. Why should you test with invalid and boundary input? [1 mark]

- ☐ **A** Attacks often use unexpected input
- ☐ **B** It makes the code shorter
- ☐ **C** Valid input never fails
- ☐ **D** It speeds up the program

5. What does this program print? [1 mark]

```
def ok(t):
    return t.isdigit() and 0 <= int(t) <= 100
print(ok('50'), ok('999'), ok('go'))
```

The task: an access checker

Write `allowed(role, action)` using the `can_do` table, returning True or False. For each `(user, role, action)` in `requests`, print `<user> (<role>) <action>: allowed` or `: denied`. Validate first: if the role is not in `can_do`, print `<user> (<role>) <action>: unknown role` instead. At the end print `denied or blocked: <n>`.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

can_do = {"student": ["view"], "teacher": ["view", "edit"], "admin": ["view", "edit",
"delete"]}
# user, role, action
requests = [
    ("Sam", "student", "view"),
    ("Sam", "student", "edit"),
    ("Mr Lee", "teacher", "edit"),
    ("root", "admin", "delete"),
    ("ghost", "hacker", "delete"),
]
```

Plan your program here, then type it in and press Run.

Do it on the robot

www.bugbotlab.com/learn/f11-8-writing-secure-programs/

The simulator checks it and tells you when it passes. Nothing to install, no account.

Challenges

1. Add a `governor` role that can only `view`. Check a governor cannot delete.
2. Extend `safe_speed` from the first cell to also reject an empty string and text with spaces.
3. Explain how validating input stops SQL injection.