Robust software: audit trails and code reviews

3.2.3 · Computers · revision notes

Robust software keeps working correctly even when it gets unexpected input, meets an error or is attacked. You need to know why robustness matters and two ways of finding vulnerabilities in software: code reviews and audit trails. Paper 1 asks 'Explain why…' and 'Describe how…' questions, sometimes with a short piece of code or an audit log to examine.

Key facts

- Robust software: keeps working correctly with unexpected input, errors or attacks
- Code review: other programmers systematically check the code for errors and vulnerabilities
- Audit trail of code changes → trace the change that caused a fault, and roll it back
- Common weaknesses: unvalidated input, hard-coded passwords, unlimited log-in attempts, unencrypted data
- Vulnerability: a weakness that could be exploited or could cause failure
- Audit trail: a time-ordered record of who did what, and when
- Audit trail of user activity → spot unusual behaviour that suggests an attack

Notes

Robust software

- **Robust** software keeps working correctly when things go wrong: unexpected or invalid input, missing files, heavy use or a deliberate attack. It does not crash or behave unpredictably.
- A **vulnerability** is a weakness in software that an attacker could exploit, or that could make it fail.
- Why it matters: crashes can lose work and data; vulnerabilities let attackers steal data or money; failures in systems such as medical equipment or cars can hurt people; organisations can face fines, legal action and loss of trust.
- It is much cheaper to find and fix problems before the software is released than afterwards.

Code reviews

- In a **code review**, programmers who did not write the code examine it systematically, line by line, before it is released.
- They look for weaknesses such as inputs that are not validated, passwords written into the code (hard-coded), sensitive data that is not encrypted, unlimited log-in attempts, logic errors and code that does not follow the team's standards.
- A fresh pair of eyes spots mistakes the original programmer missed. Problems found are recorded and fixed before release.
- Automated tools can also scan code for known risky patterns. Reviews take time, and reviewers can still miss problems.

Audit trails

- An **audit trail** is a record of events, in time order, showing **who** did **what** and **when**.
- During development, it records every change made to the code. If a bug or vulnerability appears, the team can trace the change that caused it and undo it (roll back to an earlier version). Programmers are accountable for their changes.
- In a working system, it logs activity such as log-ins, failed log-in attempts and data viewed or changed. Unusual patterns, such as a log-in at 3 am or thousands of records downloaded, can reveal someone exploiting a vulnerability.
- An audit trail must be protected so that ordinary users cannot change or delete it.

How to answer each type of question

Explain why robust software is important 2 to 4 marks **Grade 5**

1. Say what could go wrong if the software crashed or behaved unexpectedly.
2. Say what an attacker could do through a vulnerability.
3. Link every point to the people in the scenario.

Example. A hospital uses software to work out doses of medicine for patients.
Explain why it is important that this software is robust. (3)

Model answer. Any three from: if the software crashed, staff might not be able to work out doses when they are needed (1); unexpected or invalid input could produce a wrong dose, which could harm a patient (1); a vulnerability could let an attacker see or change patients' records (1); the hospital could face legal action and lose the trust of patients (1).

Describe how code reviews and audit trails identify vulnerabilities 2 to 4 marks **Grade 6**

1. Code review: say who does it (other programmers), how (line by line) and what they look for.
2. Audit trail: say what it records (who, what, when) and how that helps trace or detect a problem.
3. Finish with what happens next: the problem is fixed or rolled back.

Example. (a) Describe how a code review can help to make software robust. (2)
(b) A development team keeps an audit trail of all changes to its program code. After an update, a vulnerability is found. Explain how the audit trail helps the team. (2)

Model answer.

(a) Programmers who did not write the code examine it carefully, line by line (1). They look for weaknesses, such as inputs that are not validated, so these can be fixed before the software is released (1).
(b) The audit trail shows who changed the code, what they changed and when (1), so the team can find the change that introduced the vulnerability and fix it or roll back to the earlier version (1).

Identify vulnerabilities in code and suggest fixes

4 marks | Grade 7

1. Check every input: is it validated before it is used?
2. Look for passwords or PINs written into the code.
3. Look for log-in loops with no limit on attempts.
4. Give the line number, the problem and a matching fix.

Example. This Python code is about to be checked in a code review.

```
01 pin = '4071'
02 tries = 0
03 guess = input('Enter PIN: ')
04 while guess != pin:
05     tries = tries + 1
06     guess = input('Wrong. Enter PIN: ')
07 print('Unlocked')
```

Identify **two** vulnerabilities in the code. For each one, describe how it could be fixed. (4)

Model answer.

Line 01: the PIN is written into the code (hard-coded), so anyone who can read the code knows it (1). Fix: store the PIN outside the code, e.g. in an encrypted file that users cannot read (1).
Lines 04 to 06: there is no limit on the number of attempts, so someone could keep guessing until they find the PIN (1). Fix: use tries to stop the loop and lock the device after, for example, 3 wrong attempts (1).

Don’t lose marks

- Saying a code review is when programmers check their own code, or confusing it with testing (running the program).
- Saying an audit trail stops attacks. It records events, which helps you detect and trace problems.
- Writing 'make it more secure' without saying how.
- Saying robust software has no bugs at all. It copes with problems without failing.
- Naming a problem in code without saying which line it is on, or without giving a matching fix.

More tips

Memory tricks

- Audit trail = who, what, when.
- Code review = fresh eyes on someone else's code, before release.
- Robust = it does not break when you give it something unexpected.
- For any code snippet, check: inputs validated? passwords in the code? limit on attempts?

Exam technique

- In code questions, quote the line number and pair each problem with a fix.
- Link 'why robust' answers to the scenario: patients, customers' money, personal data.
- For audit trails, 'who, what and when' is what examiners look for.
- Keep code reviews and testing separate: a review reads the code; testing runs it.

What each grade needs

What you need to be able to do, from the first marks up to the top grade.

Grade 3 — **Define robust software**
Software that keeps working correctly, without crashing, even with unexpected input or when attacked.

Grade 4 — **State what an audit trail records**
Who did what, and when: for example who changed the code, or who logged in and what they accessed.

Grade 5 — **Explain why robust software is important**
Crashes and vulnerabilities can lose data or money, harm people, and damage an organisation's reputation.

Grade 5 — **Describe how a code review finds vulnerabilities**
Other programmers check the code line by line for errors and weaknesses before it is released.

Grade 6 — **Explain how audit trails help find problems**
They show which change caused a fault, or reveal unusual activity that suggests an attack.

Grade 7 — **Spot vulnerabilities in a piece of code**
Look for unvalidated input, passwords written into the code and unlimited log-in attempts, and suggest a fix for each.

Grade 8 — **Evaluate ways of making software robust**
Weigh code reviews, audit trails and testing against their cost and the fact that none of them can find every problem.

Free notes, Full packs, practice papers and mock exams at chhetriacademy.com. Written for this site; not taken from past papers. Stuck? 1-to-1 tutoring with a free 30-minute trial lesson.