Software and Security Engineering - Supervisions

Ross Anderson, Alastair Beresford and Andy Rice

There are three recommended supervisions for this course. Questions are marked up as:

- Bookwork (B): require review of lectured material and recollection of facts
- Shallow (S): direct application of facts in a formulaic manner
- Deeper (D): apply material in a new context or bring several ideas together
- Open (O): open-ended questions with no known right answer

Those questions marked with an asterisk (*) are optional.

Supervision 1

1.1 (B) Write down your own one-sentence description of each of these technical terms: *system, hazard, risk, error, failure, threat, accident, safety case, security policy, trust, reliability, subject, person, principal, secrecy, privacy, confidentiality, anonymity*, *integrity, authenticity, uncertainty,* and *safety.*

1.2 (D) You are provided with a computer system which supports multi-level security with four levels (open, confidential, secret and top secret) and permits information to flow upwards only. In addition, it permits compartmentalization of any layer into one of five compartments (A, B, C, D and E) between which lateral flows are prohibited. Subjects are authenticated and the system supports role-based access control, allowing a subject to act in one or more roles.

Describe how you could use this system to support:

- An electricity utility with four power stations
- Medical records in a hospital with four departments

Comment on the suitability of your solutions in the provision of integrity, confidentiality, the different risks to different stakeholders and safety overall.
In the case of medical records, what changes are required to accommodate researchers who want access to anonymised records, or clerks who want access to financial data on bed occupancy and the financial cost of care for each patient?

1.3 (D) Write a security policy for https://www.timetable.cam.ac.uk.

1.4 (D) List all the ways you can think of to access a friend's GMail account without their permission. *Caution: this is a theoretical exercise in order to help you understand the landscape of attack and defense; under no circumstances should you attempt to carry out any of the actions you list.*

1.5 (B) Spend 10 minutes researching prospect theory and then write a short description.

1.6 (S) Describe how a social psychology technique could be used to improve the likelihood of each of following types of attack:

- Tricking a user into visiting a specific website on their laptop
- Persuading a user to pay after their computer has been infected with ransomware
- Disregarding advice on health and safety
- Encouraging a user to install a dodgy app on their Android smartphone

You should be able to use several different social psychology techniques for each scenario.

1.7 (O) Think of another real-world security protocol that predates cryptography, similar to the example of ordering wine in a restaurant described in lectures. Discuss any interesting properties of integrity, authentication, non-repudiability, confidentiality, anonymity or deniability.

1.8 Create an attack tree for a stealing a car. You can represent this as a figure, or by an indexed list (e.g. 1, 1.1, 1.2.1, 1.2.2, 1.3, 2.1, …)

1.9* (O) Read about, and then write a basic description of, how PGP and Signal work. Which piece of software offers better security to a non-expert user who wants to leak confidential information to a newspaper? What basic advice about using such mechanisms should the newspaper offer on its "How to share information with us in confidence" web page?

1.10* (O). Write a password policy for users of the Raven authentication system here at Cambridge.

1.11* (O) Read about how bitcoin works and write down a brief description. Suggest how either the protocol, or the laws and conventions surrounding it, might be modified to make it harder to use bitcoins for money laundering and ransomware.

1.12* (O) When making an instant credit push payment (eg a 'faster payment' from a UK bank account or a phone payment via Pingit) , there is a real chance of putting in a wrong destination identifier (account number or phone number) and sending the funds to the wrong payee, and depending on the payee the money might not be recoverable. The solution adopted in quite a few countries overseas is to provide some detail about the payee before the payment is confirmed. For example, a payer who inputs an account number may get a phone number and associated name before being asked to enter her PIN to confirm. UK banks are resisting this saying it would be a privacy problem.

- Is the ability to couple an account number to a person really a serious privacy loss and why?
- What fraud could be perpetrated as a result – specifically is this a serious attack vector that is opened up against individuals and could it be used at scale?
- What mitigation countermeasures are possible on the payee side and on the PSP providing the payment service?

Supervision 2

2.1 (B) Draw flow chart diagrams to explain the waterfall, spiral and agile models of software development.

2.2 (S) For each of the following Java code snippets, identify the type of the error (e.g. arithmetic, logic, syntactic, overflow), describe why the code fails, and how you could fix it. Try these examples in a simple Java program to help you understand what has gone wrong as well checking your fix.

- Calculate whether the integer i is even or odd:
  ```
  i % 2 == 1
  ```
- Calculate the change from buying an ice cream for £1.10 with a £2 coin:
  ```
  2.00 - 1.10
  ```
- Calculate number of microseconds in a millisecond:
  ```
  long MICROS_PER_DAY = 24 * 60 * 60 * 1000 * 1000;
  long MILLIS_PER_DAY = 24 * 60 * 60 * 1000;
  MICROS_PER_DAY / MILLIS_PER_DAY;
  ```
- Adding two numbers together (and the result is not 66666; cut and paste to check):
  ```
  12345 + 54321
  ```

2.3 (D) Design a system which fixes the Clallam Bay analogue code injection vulnerability

2.4 (D) Imagine you were a product manager for a software development process in 1975, 1995 and then 2015. Make a list of the techniques and approaches available to you in each era in each phase of the software lifecycle. What differences are there in software development at each of the three timepoints?

2.5 (D) The difficulties of designing an infusion pump was discussed in lectures and in the lecture notes. Produce your own user-interface design for an infusion pump for use in less developed countries. Draw a picture of the user interface and write a short description of what should happen when you press each of the buttons. You may find it helpful to read Harold Thimbleby's paper: https://ieeexplore.ieee.org/abstract/document/6680455/ and also available here http://www.harold.thimbleby.net/cv/files/IHCIkeynote2013.pdf

2.6 (O) Spend around one hour writing a one-page summary of the London Ambulance Disaster in your own words after reading more about the failure. You should make sure you document the salient facts as well as the underlying reasons for failure. You may find the following reference helpful: http://www0.cs.ucl.ac.uk/staff/A.Finkelstein/las.html

2.7 (O) Spend around one hour writing a one-page summary of the Therac-25 issue. You should make sure you document the salient facts as well as the underlying reasons for failure. You may find the following references helpful:

- https://en.wikipedia.org/wiki/Therac-25
- Nancy Leveson, "Safeware: System Safety and Computers", 1995. [pp 515-553]

2.8* (O) Spend around one hour researching and writing a one-page summary of Britain's smart metering project. Now that Britain is leaving the EU and is no longer bound to install smart meters in 80% of homes by 2020, should the project be discontinued? The material references from the following webpage may be helpful:
https://www.lightbluetouchpaper.org/2012/09/17/the-perils-of-smart-metering/

2.9 Complete this exam question: 2009 P3 Q8

2.10 Complete this exam question: 2012 P8 Q12

Supervision 3

3.1 (B) Why don't software engineers only use integration tests? Provide examples of circumstances in which other testing strategies are important.

3.2 (S) Debunk each of the following software myths with reference to one concrete case study or example which demonstrates the opposite can sometimes be true:

- Computers are cheaper than analogue devices
- Software is easy to change
- Reuse of software increases safety
- Formal verification removes all errors
- More testing makes software more reliable
- Automation reduces risk

3.3 (O) Write a one-page (500-word) essay on *one* of the following three topics:

- A friend of yours at another university is thinking of dropping out of his degree and doing a startup. His idea is to make vote tampering impossible by putting elections on the blockchain. Read up on what goes wrong with real elections and write to him explaining why his plan isn't likely to solve the world's political problems. (Hint: look at registration abuses, districting abuses, harassment of opposition parties and all the other ways in which elections are rigged. What proportion of the documented abuses involve secret

and direct tampering with the vote addition mechanism? Would blockchains realistically stop such attacks better than traditional computer security and audit mechanisms? Would a move to electronic voting, regardless of the counting mechanism, make the other abuses easier or harder? What about the usability issues – are electronic mechanisms in general, and blockchain mechanisms in particular, easier to understand by voters, and by opposition parties? Who is the real 'customer' for an election anyway – the ruler, the opposition or the voters?)

- Same, except that the idea was solving the world's poverty problems by putting payments to and within less developed countries on the blockchain.
- Same, except that the idea was to solve the fake news problem by putting news on the blockchain.

3.4 Complete this exam question: 2007 P5 Q1

3.5 Complete this exam question: 2017 P3' Q8

3.6 A programmer creates an e-commerce library, however the project has some bugs and other shortcomings. You can download a copy here:

https://www.cl.cam.ac.uk/teaching/current/SWSecEng/ecommerce.zip

3.6.1 Unzip the archive and make sure you can compile it. The project is build with a tool called maven and therefore you can compile and run the tests on the command line as follows:

```
bash$ mvn clean compile package test
```

Note: You will need Maven and the Java developer tools installed.

Alternatively, you can import the project in to IntelliJ using "Import project from existing sources" and select the Maven option. Once imported you can run the tests by right-clicking on source file containing them and choosing Run or Debug. The source code should compile and the tests should pass. You can check this in the dialog which should appear at the bottom of the IntelliJ pane.

Write down a brief summary of the approach you took.

3.6.2 Write brief notes on how the library is supposed to be used and why it is designed this way. Make sure you address the following points:

- What is the benefit of using `PaymentDetails` rather than a `String`?
- What is the purpose of the `Order` class, why did the developer not just take payments from the `ShoppingCart` directly.
- What shortcomings have you identified?

3.6.3 Look at the unit test class: `TestShoppingClass`. What is the purpose of the setup method and why is this work not done in the constructor?

3.6.4 Critique the two test methods. What things are poorly written about them? Rewrite and improve the first test to fix the things which you think are poor.

3.6.5 Improving the second test requires changing the implementation of `ShoppingCart` to use dependency injection. Describe the concept of dependency injection and its benefit. Rewrite `ShoppingCart` and the second test.

3.6.6 You receive a bug report that when two items with the same name are added to the shopping cart, and then one is removed, both items disappear. You determine that the correct behaviour should be than when a second item is added with the same name it should instead update the price of the original item to reflect the cost of two items rather than adding a second item. Write a test to reproduce the bug; your test for the bug should fail. Fix the bug; your test should now pass. Why would it not be sensible to write a test that reproduces the reported bug?