# CS732 / SE750 Quiz – S1 2022 – Part B

## Instructions

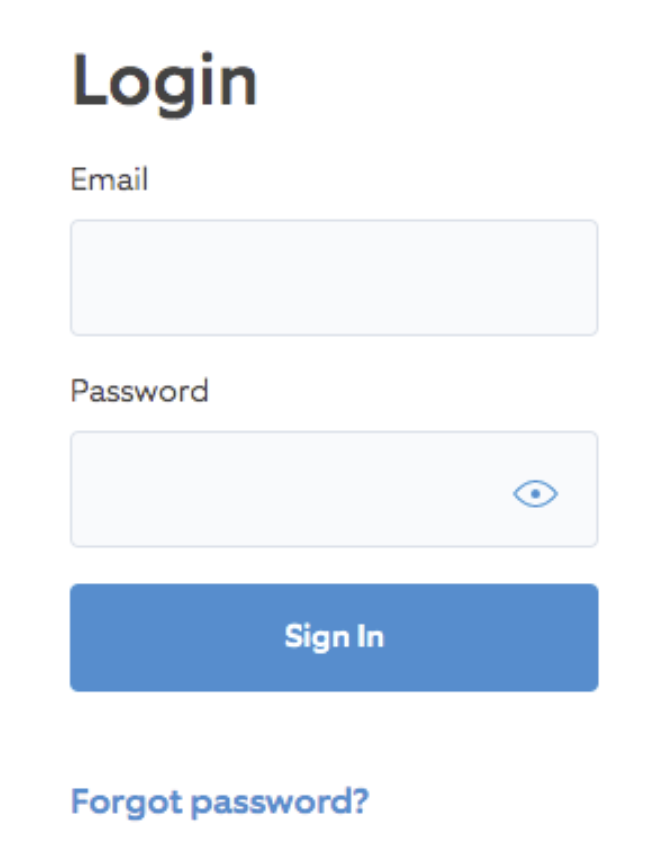
To complete this part of the quiz, write your answers in the boxes provided. You may expand each box as needed. This part of the quiz is marked out of a total of **30 points** and is worth **50%** of the total grade for the quiz. The number of points for each question is given in [square brackets] at the end of each question.

## Questions

1. Why do security issues remain unnoticed in software systems? [2]

|  |
| --- |
| This is because the security requirements vary from one software system to another and these requirements are not easily measured directly. The security requirements of software systems are in turn not physically observable by all users, so this leads to some software system security issues going unnoticed. In other words, criminals can also profit from software systems by engaging in threatening behaviour. They may attack through unknown vulnerabilities which in many cases are also unknowable and undetectable. |

1. You are hired to security test the login mechanism of a web application shown below. Explain three concrete scenarios that you would include in your testing. Each scenario should include the vulnerability, risk, and a potential countermeasure in the software. [12]



|  |
| --- |
| CASE1:  Verify that there is a rule that limits the number of login attempts. Keep trying by entering the correct email and the wrong password in an attempt to guess the password for this email. If such a rule does not exist, criminals can exploit this vulnerability to perform unlimited password cracking.  CASE2:  Verify that there are no anti-automation rules. Verify that the system does not have anti-automation rules by using some automation tools and software to automate logins to the system. This can be done by logging in with many correct emails and passwords using such automated tools. If the system does not have anti-automation rules, this means that criminals can gain massive access to user information by logging into an unlimited number of accounts.  CASE3:  Verify that a content security policy is in place. A malicious script can be injected into the input box of an email or password to verify that data can be obtained through an XSS attack or by bypassing the security policy. If there is no content security policy in place, it means that criminals can gain access to user data through an XSS attack. |

1. Why do we need different tools for security analysis? Explain three main reasons. [6]

|  |
| --- |
| Software security is a very broad technology and it will include many tools that correspond to the corresponding technology in order to provide adequate security for software. For example, tools that specialise in detecting vulnerabilities, as they require a good knowledge of the code, libraries and frameworks used in the software system in order to be able to detect the security of the system. IDE plugin checks, which check the version of dependency packages of third parties used in the software, can help us to reduce the threat posed by security risks of third-party libraries referenced in the software. Static testing is also a corresponding security policy tool that scans the code written by developers to detect security risks in the code, such as password plaintexts. |

1. When should we start threat modeling, and when can we stop it? Justify your answer. [4]

|  |
| --- |
| Threat modeling is a way of thinking about an application and its environment through a secure view. It is about making people aware of why things can go wrong and guiding further design, development and testing activities to ensure that the wrong things don't happen. Therefore, we should start threat modeling early on, and it should also be a continuous activity that never stops. It should exist as long as the software is evolving. Even in some cases, new threats may appear in a unique threat model even if the software does not change. Threat modeling therefore never stops. |

1. Would you recommend adopting black-box fuzzing or white-box fuzzing to assess a software program? Justify your answer. [4]

|  |
| --- |
| If you want to evaluate a software program well it is best to use a mix of black box, white box and even gray box testing methods. Although black-box testing can only explore shallow problems with the system under test, it is low cost. White-box testing is input testing through program analysis and constraints, and although it can be good for program verification, it is expensive. Gray-box testing requires some knowledge of the program context to validate well and is relatively costly. Therefore, I suggest that you need to choose a suitable testing scheme according to the software under test, and it is better to use a mixture of these testing methods. |

1. Why could a software company with many security experts and a good arsenal of tools still develop vulnerable software? [2]

|  |
| --- |
| Because there is no absolute security in software. From the perspective of security detection, many software vulnerabilities are not discovered before the software is released, but gradually as the technology evolves. On the other hand, even the same algorithm but there are differences in different development languages, for example, some floating-point scenarios, even the same algorithm in different development languages there are differences. So even software companies with many security experts and a good library of tools can still develop software with vulnerabilities. |