Security Software Development Module Reflection

Wang Wang

https://github.com/William-Wang-Dev/eportfolio/tree/main/documents/SSD

Introduction

Honestly, in spite of my daily work is programming, I didn't think deeply about security in software before learning the module. I just took security issues as things related to empirical or logical scope. However, from this module, I found a big misunderstanding about security software development in my previously thought. People Gain high security not just as a particular technical skill; it is more like a systematic thinking and evaluation process with essential principles.

Security principle and case studies

I read a lot of classical papers to describe principles of security in software, and the issues listed in OWASP 10 are highly related to the correct adherence to these principles or not. And I notice some security guides share the same philosophy in the best practices in coding. For example, the economy of mechanism in protection data prefers simple and small ways as a solution; least privilege, as same as least knowledge to design interface between classes.

In studying OWASP 10 and TrueCrypt safety audit report, I feel the software desired security level actaully built upon many assumptions, one of assumption broken could make a flow can be exploited by attackers. Applying defense in depth can buy time for people to fix vulnerabilities, and it reminded me that single node failure could lead to disaster, in which I found a similar principle in system redundant design. After touching the OWASP 10 and TrueCrypt, I have subscribed security news and regularly checked the latest safety issue in the online software vulnerabilities database. As there is no perfect software, zero security flaws in the software are also impossible. No wonder people usually called mitigate security risks but not eliminate them. However, continuously improved and regular audits are still the best practices in security software development.

Take security as a requirement

When people make evaluating the quality of software, the best approach is exam it with per-defined requirements. On the other hand, nobody will consider software complete if it can't get desired functionality, even if it could have grace style or use the art of edge technologies. Security should also be part of a software requirement and involved initially. In a complete software development life cycle, there would have a dedicated activity to be related to the concern about security.

If we take security as a requirement, it means adequate details let all project stakeholders understand what security level the software should achieve, how to measure them, and what approach to mitigate misuse cases. I believe it does matter the goal you get more specific, and the failure chance is lower. It has a considerable gap between a function/class/module design with or without security consideration. Every newly added requirement will impact developers on how to organize their code; testing engineers make a testing plan and scenario. Taking security as a requirement can fit traditional waterfall development modules and also match the agile coding style. It has research shows a scrum way to develop software with detailed security requirements, on which even developers do not have much experience, the security evaluation of a final product still had a certain quality of safety (Pohl, C. and Hof, H.J., 2015). In our team assignment, in the coding stage and testing, we selected the pattern/implement style, hugely influenced by the scenario related to a security listed in the design document. Everybody always remembers what outcome we should achieve.

software design

The module is not only security but also offers knowledge about system design and software quality. I feel a design is a somewhat subjective activity, and it is related to how people look at a problem they try to resolve. However, people still have standard ways to evaluate codes, like readability, complexity, maintenance ability, extensibility, and cohesion. And all the design patterns serve these purposes. Another I want to mention is ontology. I think it is an excellent approach to system analysis. Ontology could guide me in applying a logical way to draw out a system and the relationship between entities. I have used this in my daily coding job. I have always been interested in software architecture, so I spent a lot of time reading related chapters, which I feel the most gain is how to deal with the relationship between the system modules. These relationships should not only be considered from the business logic; performance and security should be reference points. That is, a good system should be achieved a balance between business, performance and security.

Teamwork

This module involved two group assignments, designing and implementing a security repository system. Our team filled people with different backgrounds and cultures, and everyone had their unique perspectives during the discussion. I could have more coding experience than others, and I was frustrated in the first meeting when my teammates couldn't understand my point. However, I changed my communication strategy, replaced specialized conversations with more concrete examples, and it worked. And I found putting myself in someone else shoes to think about their doubts or concerns always lets me rethink my reasonable point. Having an open mind and keeping friendly were keys to completing our project successfully. When the team is stuck in some situations, proactively making a demo can inspire teammates. Since the players are at different times, resulting in not a large window of time for internal online meetings, we used a forum-like mechanism to communicate and divide the work through Google Docs. If someone has any questions, they leave a message on the doc, and those who can answer it leave a message below it, and everyone can see its detailed discussion process.

Conclusion

This module studying for me was an inspiring experience. I learned a lot of concepts and foundation knowledge in security software development and started to apply them in my daily job. Attacking the software is becoming increasingly sophisticated, and attack surfaces could be more and more broadly. However, I believe what I learned from this module can be a solid base to help me take on challenges in my future career.

Reference:

1. Pohl, C. and Hof, H.J., 2015. Secure scrum: Development of secure software with scrum. arXiv preprint arXiv:1507.02992

The final project vs the design document

In the design stage, the team had planned to use a relational database to store all data, even the content of files. However, in the actual implementation, we used the NoSQL style, that is, JSON file as a database, to store the relationship between files, users' permission, and the necessary properties of a file. Because the NoSQL is more natural and straightforward to save these, which could be a complicated format in the relationship model. Another change was adding a session module to our system, which was not mentioned in the design document. In the implementation stage, especially for the authorisation module, without a session, a valid login user is easily forged by a malicious attacker. With these two modifications, the software desired functionality still didn't change, and we would say the whole system's security got improved.