Security: Bridging the Academia-Industry Gap Using a Case Study

Kirti Garg, Vasudeva Varma
International Institute of Information Technology, Hyderabad
kirti@research.iiit.ac.in, vv@iiit.ac.in

Giridhar K N, Abhishek K. Mishra
Microsoft India Development Centre, Hyderabad
{girikn, akmishra}@microsoft.com

Abstract

Security is one of the major concerns of modern software development, but there is a wide gap between the industry practice and the academic instruction. Security issues are usually not addressed in academic setup and those attempt to make it as part of their software engineering curriculum realize quickly that it is difficult to make learning happen in a conventional lecture based approach. We explored a case study centered non-conventional approach and found it to be instrumental in bridging this gap between the industry and the academia. We give a detailed methodology for using the case study and our experience, observations and result of the approach. Whole case exercise aims at creating software professionals who realize the importance of security and are well versed with the related concepts, skills and dispositions and thus handle security related challenges in a structured manner.

1. Introduction

Though secure software are getting due consideration by many organizations, we find very few examples of Software Engineering courses [12] where concepts of secure applications are being promoted. Not many courses deal with security in a methodical manner and emphasize its importance at all phases of Software Engineering, despite the fact that security is becoming one of the prime concerns for real world software development.

We have developed a case study (Mobile Email Client) that is based upon actual challenges faced by the industry and the ways they are handled. The aim is to come up with a course content that brings the security focus to software development process.

We partnered with a major software company that helped in bringing the real life aspects into the case study. Detailed teaching notes were developed for the case study. These notes give teaching goals, external reading resources and assist the faculty in conducting the case study by suggesting the most suitable way to handle the case and a probable solution.

This document reports a summary of the case study, teaching goals, a detailed strategy for handling the case in class, assessment and other factors that may affect teaching. An analysis of the benefits, especially for cognitive skill [19], and results from a recent use of the case are also presented. Term ‘participant’ is used to refer to students who participate in case solving.

2. Case Study Approach

Case studies have been considered as prominent non-conventional teaching tool. Senge identified at least 14 types of skills that case studies inculcate into the case participants [14]. Most importantly, they infuse the critical ability to ‘analyze’, ‘apply’ and ‘evaluate’ what has been taught as a theory. Not only the solving of the case study challenges, but even the peer discussion of various solutions of the case study, facilitates learning.

Case study approaches have been successfully used for management education, nursing, law and even medical science. Raju et al tried it out with engineering education [13]. Clancy and many others used it for teaching programming [2]. Butler, Dalcher and recently Varma et al have been attempting to use case studies as an instrument for software engineering education (SEEd) [1, 5, 16]. Pfleeger et al examined the use of case studies as a empirical research method to evaluate tools and methods in Software Engineering, particularly in practice [9]. Recently case studies have been identified as collaborative learning tools that can play important role in increasing the overall effectiveness of SEEd [17].

Inspired by the successful application of the case study approach in subjects difficult to teach in a conventional manner and by the recent results obtained while conducting a Software Engineering course, we believe that it can bridge the gap between theory and application, while bringing the learning focus to a critical issue such as security at an early stage [16].
3. The Case Study

The case study we have developed can be seen as a sequence of subplots (Plot i), issues (Issues i), and challenges (Challenge i). The plot represents a set of events that form the context, i.e. of real world scenarios, way the event unfold, associated people and may discuss various issues, that further act as ‘tension events’ for the case and bring up some twist. Challenges are the decision dilemmas or challenges faced by the protagonist in the case.

For Mobile Email Client case being discussed here, the sequence and the summarized version are as follows:
Plot 1 → Plot 2 → Issue 1 → Challenge 1 → Issue 2 → Challenge 2 → Issue 3 → Challenge 3.

(Plot 1) SoftCorp is an emerging software development company and aims at leadership position in the upcoming mobile computing market, by developing cutting edge products that address the entire range of handheld user needs. To start with, they conceive an idea of a Mobile Email Client (MEC) to sort, organize and provide an easy access to the user’s emails. This product development would also help them to get familiar with the handheld programming environment. The team considered providing support for new technologies such as IMAP4, Bluetooth connectivity and support for WAP, to try and win over their competitors, who already were quite few in number. A survey was done to identify regular and other key features that could be incorporated as the Unique Selling Points (USPs) of the product.

(Plot 2) A three milestone strategy is adopted, such that the first version can be released within twelve months of the start date. Requirement engineering begins and the core features like reading and composing emails, folder hierarchy support, attachment support etc, were be completed before the first milestone (End of sixth month). Some new ideas were discovered during this phase. It was decided that a couple of developers would investigate these ideas and build prototypes to evaluate them before including in the milestone two. Milestone three was more concerned with adding new features that were still to be investigated.

Team gets started with coding, once the design phase over. Quick and enthusiastic, the team starts coding a week before the actual milestone deadline.

(Issue 1) This is when Kumar, the Chief Technical Officer of SoftCorp attends a conference on Software Security. His attention is drawn by a poster presented by a couple of graduate students that talks about security requirements of software applications for handheld devices. Getting real inquisitive, he spends a week collecting information about prominent security issues and the modus-operandi of common threats.

Wondering whether SoftCorp has taken into consideration these issues, Kumar enquires various product groups if they followed any specific procedure to address the security issues. This email made Raj, the Project Manager to think. He was intrigued as he had recently read an analysis on the exploits of the Melissa Macro Virus which ruled the roost in mid 1999 and created panic among email users as well as the product companies. Other recent readings also suggested that the handheld device users are also vulnerable to grave security related attacks.

An all team meeting is scheduled and everyone is asked to participate in a brainstorming session in order to come up with various security threats that can influence the product in short or long term. Some issues are identified, but these are very simple issues, such as authentication and authorization, for which the product design already has some provisions. Raj demands some more discussion, and asks the team to identify all possible issues along with cost effective solutions for the identified threats. One member expressed concern that since the coding phase is already in progress, the measures should be related to writing secure code, i.e. how the code can be made secure.

(Challenge 1) The case study participants, empathizing with the developers, should perform the brainstorming sessions to identify security threats for the given product (product requirements and design are provided).

(Issue 2) Meanwhile the test team moves ahead with the testing in order to save some time. They find that the product is crashing almost 60% of the time due to some bad APIs being used. This is a serious concern and Raj asks for further analysis from the test team. They identify a few bad code fragments that also relate to some of the earlier identified security issues.

(Challenge 2) Some code fragments are provided to the participants (each one plays the role of a developer) for analysis from a security related perspective. They should identify the secure code related issues and should modify the code using secure code practices that help minimize the security loop holes.

(Issue 3) Meanwhile, Kumar forwards Raj some articles on designing for security in form of threat modeling. Going through the articles, Raj thinks that it will be a useful exercise to perform threat modeling. In fact he was of strong opinion that though it will take some extra time, it should be done.

(Challenge 3) Now the team is facing another dilemma: Should they consider a redesign of the product to ensure all security guidelines have been
followed, in turn introducing a delay into the development process, which can prove to be costly? Can they handle security in some later SDLC phase such as testing? OR should they postpone the process to further releases of the product? What guidelines should they consider while redesigning and performing the threat modeling? How should they proceed if they want to handle security in future course of development?

4. Teaching Goals and Objectives

Each case has a set of major and minor teaching goals. It is not possible to have a single teaching goal, as it will not represent a real world situation where issues don’t come up isolated [16, 17]. Major teaching goals for this case are a) To make participants think about the importance of security, b) To make participants aware of the various security issues, prominently secure code, c) To make participants see the connection between software development process and security (figure 1), d) To introduce a structured approach for handling security issues.

The minor teaching goals include a) Bring focus to good practices in secure coding and b) make participants aware as how security is handled currently.

4.1 Security and Software Development

The important goal of this case study is to help the software engineers to view their project development process from security perspective (figure 1). These are some of the recommended structured approaches towards security. The case does not force the participants to follow only a fixed approach. They can follow any approach to solve the challenges. We discuss some of these issues in the following subsections.

It is recommended to incorporate security in requirements engineering [7], by considering scenarios that have a potential for security loophole. Similarly we can perform threat analysis and modeling while software design, and practice safe coding [8] techniques. Similarly, security testing should be explored further, though this case does not talk about security testing explicitly, it encourages the participants to think in that direction while solving challenge 3.

4.2 Security Requirements Engineering

Security requirements are usually depicted as constraints on the functionality of a system. Based upon the concept that ‘actors operations on assets’, Pfleeger and others describe the general categories for constraints, namely CIAA, abbreviated for Confidentiality, Integrity, Accessibility and Authentication. Confidentiality limits visibility of the asset to those actors authorized to see it. Integrity ensures that a set is not corrupt, Accessibility ensures ready accessibility of the asset to actors that need it and Authentication ensures that the provenance of the asset or actor is known. [11].

One of the most important responsibilities of a requirements engineer is to work out these security constraints by applying a qualitative reasoning method and figure out the exact behavior of the system before it is actually built. Requirements usually describe what is expected from the system not what it actually provides. But descriptions about the actual behavior of the system suggest of how the system will actually work. Many systems are built with some underlying assumptions about the deployment environment. The requirements engineer should analyze how much trust can be placed on these assumptions.

Abuse cases [10], misuse cases [15] and anti-requirements [4] can be used to generate explicit threat scenarios and the respective counter-actions. Security requirements engineering should focus the issues such as security policies at different levels of abstraction, integrating aspects such as multi-level security, discretionary access control, separation of duties, delegation, roles, groups, generalization hierarchies, and supervisory hierarchies. Apart from these, the concern can be to identifying different standards that would help distinguish between technical and non-technical security policies, and organizational procedures.

4.3 Security during Design Phase

A Simple way of manage security during the design phase is through threat modeling. It is a more economical mean to evaluate and rectify the threats to the application before coding. This not only reduces the overall risk, but also helps to understand the application better, especially for new team members.
This is because one spends time analyzing the makeup of application in a relatively structured manner. It is an easy way to find out bugs, and it has been seen that 50% of the bugs have been found through threat analysis alone [8]. Complex design bugs and multi-step security bugs (several small failures combining to form a disaster) can be easily identified using threat analysis. It is very important for the product teams to know about threat models. These models can be used in testing phases later. Threat modeling process usually consists of the following phases [8]:

1. Assemble the Threat Modeling Team.
2. Decompose the application.
3. Determine the threats to the system.
4. Rank the threats by decreasing risk.
5. Choose how to respond to the threats.
6. Choose the techniques to mitigate threats.
7. Choose the appropriate technologies for the identified techniques.

Data Flow Diagrams (DFDs) are one of the most effective and convenient means to decompose an application into independent key components.

Threat trees [8] describe the decision making process of the attacker. Thus coming up with these trees implies identification of possible failure modes for the system under consideration.

Determining threats to the system is a very intuitive process and requires some amount of skill and creativity. The idea is to use the identified components from the decomposition as threat targets for the threat model. Some of the questions that need to be asked before identifying threats could be a) Can a non-authorized user view the confidential network data? b) Can an un-trusted user modify the database? c) Can someone deny application services to valid users? d) Is there a possibility of someone taking advantage of the feature or component to raise their privileges to that of an administrator? [8].

A categorization of threats is the next step. One such useful technique is STRIDE, which is an acronym derived from the common six threat categories: Spoofing user identity, Tampering with data, Repudiation, Information disclosure, Denial of service and Elevation of privilege [8].

Once identified, the threats should be prioritized as per the risk involved. A simple way to calculate risk is by multiplying the criticality (damage potential) of the vulnerability by the likelihood of the vulnerability occurring, where one (1) is low criticality or likelihood of occurrence and ten (10) is high criticality or likelihood of occurrence [8]:

\[
Risk = \text{Criticality} \times \text{Likelihood of Occurrence}
\]

Parameters such as Damage Potential, Reproducibility, Exploitability, Affected users and Discoverability, abbreviated as DREAD [8] can also be used for this prioritization. Next steps involve the identification of mitigation steps and their application to make the application more secure.

### 4.4 Writing Secure Code

Many consider the writing of secure code as a task requiring great skill, but awareness about potential security vulnerabilities make the coding easy and effective. Some of the issues that should be considered include buffer overruns, access control list(ACL), privilege issues, cryptography related issues and database related issues [8].

A buffer overrun occurs when the program attempts to write beyond the end of a buffer. They are broadly classified as stack overruns, heap overruns, array indexing errors, Unicode and ANSI buffer size mismatches, etc. They typically create vulnerabilities that can have major consequences such as elevation of privilege by arbitrary code execution, Denial of
Service (DoS) attacks or even rooting a machine and loading worms.

Access Control Lists (ACL) and Privilege Issues are another important aspect. ACL is an access control method employed by many operating systems to determine to what degree an account is allowed to access a resource. It is really important how these ACL’s are designed because they decide who can access what at any given time.

Cryptography is a useful mechanism, but secure storage, exchange, and use of keys are issues worth considering from the perspective of security. Precaution should be taken while dealing with random number generators, secret keys and exchange protocols for the keys.

The issue with the database input mainly due to misplaced trust; i.e. trusting that the user has given a well-formed data to the application, when in fact the user has not. SQL injection is the biggest threat in the database arena. Hence a possible good practice here could be using quotes in the input query string. Use of Stored Procedures also could help in minimizing injection attacks.

5. Teaching Approach

5.1 Structure of the Case

Each section of the case study, identified by various tags, namely Plot, Issues and Challenges is serving a purpose.

Several artifacts are embedded in the case study and are given as exhibits to the case participants. These include a) a part of the document which highlights how Melissa, a mail mailing virus exploits security vulnerabilities b) a small article on threat modeling c) part of the detailed requirements document that is listing all functional and non-functional requirements. These exhibits provide useful information to the case participants in order to help them solve the case.

Like many other case studies, this case study too ends with an invitation to solve a set of challenges. The participants place themselves into the shoes of the case protagonist(s), developers in this case, and apply their skills, knowledge, tools and techniques to solve the challenge. Table 1 discusses the purpose of each section.

<table>
<thead>
<tr>
<th>Table 1: Purpose of each Case Study Section</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Section</strong></td>
</tr>
<tr>
<td>Plot 1</td>
</tr>
<tr>
<td>Plot 2</td>
</tr>
<tr>
<td>Issues 1</td>
</tr>
<tr>
<td>Challenge 1</td>
</tr>
<tr>
<td>Issues 2</td>
</tr>
<tr>
<td>Challenge 2</td>
</tr>
<tr>
<td>Issues 3</td>
</tr>
<tr>
<td>Challenge 3</td>
</tr>
</tbody>
</table>
5.2 Conducting the Case

Various approaches can be used to conduct a case study in class. The simplest is to present a case to the course participants and ask them to take up the challenges embedded in the case.

Case study presentation is a tricky job. A case study should be presented in a neutral manner, not biased by the views of presenter or the case writer. The presenter should stick to presenting the narration and should encourage the participants to read between the lines, analyze the facts, apply the concepts, and solve the challenges. During this process, the participants may come up with several alternate solutions and then evaluate these alternatives to find the most suited as per the solution requirements.

Given sufficient time, nearly a week, for coming up with the solution, discussions should take place where the participants share their solutions & reactions to the challenges, and these solutions & reactions are discussed by whole class. These discussions bring out the real and most effective learning in a case study approach. Discussions should be guided by the faculty, while not biasing the participants and encouraging creative, justified and proactive solutions. The discussions should aim to fulfill the teaching goals while enhancing the analytical thinking, evaluation and communication skills of the participants. Once a discussion is over, the participants can be given some more time to come up with a detailed solution report.

Changes to their strategy could be suggested in light of the discussion, thus creating a learning environment while focus shifts from grades to learning and improvement.

Thus case study approach usually involves a number of learning activities: reading case, solving case, discussing solutions and finally communicating the solutions.

Various changes can be done to this basic approach to satisfy the constraints such as level of course, class-size, course load etc. If it is being used as teaching instrument in a first course in Software Engineering, a lecture on security aspects can precede the case solving. If challenge 3 appears to be difficult to achieve as per the level of students, then only a fragment of the case study (up to challenge 1 or 2) can be used.

Based upon the class size, groups can be formed and multiple groups can present the solution to same problem set.

In case the class size is huge, and it is not possible to allocate sufficient time to all the participants for presenting their solutions, only a few groups can present and discuss. Rest of the participants can be asked to submit a report that records case summary, participant’s learning from the case, solutions to chosen exercises, appreciation and criticism of solutions discussed in class, alternate solutions etc. This report will serve multiple purposes, namely involving all in case study, providing all with an opportunity to air their views and assistance in grading.

5.3 Affected Skills

A number of skills are affected by each of these learning activities. Hiltz et al identified a number of parameters to capture student’s perception of learning. [3,18]. These parameters map with Bloom’s taxonomy of cognitive educational goals [19]. Table 2 gives a summarized view of the major skills [adapted from 18, 3,14] that are honed by each of these learning activities.

Table 2. Purpose of Learning Activities

<table>
<thead>
<tr>
<th>Activity</th>
<th>Purpose of the Activity (Honing skills)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case presentation by facilitator</td>
<td>Active listening, analyzing, seeing relationships between ideas, integrating factual knowledge</td>
</tr>
<tr>
<td>Working on case study challenges</td>
<td>Problem solving, applying concepts, critical thinking, analytical skills, seeing relationships between ideas, (synthesis) learning facts,</td>
</tr>
<tr>
<td>Writing reports</td>
<td>Communication skills, reinforced learning, analysis &amp; evaluation skills, leading to active participation by all, providing chance for improvement</td>
</tr>
<tr>
<td>Listening to case solutions by peers</td>
<td>Critical thinking, analysis &amp; evaluation skills, seeing relations between ideas, active listening</td>
</tr>
<tr>
<td>Receiving peer reviews</td>
<td>Evaluation, critical thinking</td>
</tr>
<tr>
<td>Critiquing other students' work</td>
<td>Critical thinking, evaluation, analytical skills, seeing relationships between ideas</td>
</tr>
</tbody>
</table>

5.4 Target Audience

This case study is suitably designed to work in academic as well as corporate environment. The audience or the participants can be graduates and undergraduates with an experience in software development project and with understanding of software engineering concepts such as various SDLC models and economic factors that play important role in software development. Background in computer programming is mandatory.
When used in the corporate environment, the participants can be the professionals at any level (developers, architects, project managers).

The zeal to listen actively, learn, and apply the concepts being taught will be an added advantage.

5.5 Resource Requirements

Case studies and the accompanying teaching notes form a self-contained unit. This unit and the case study facilitator are the only mandatory resources. The facilitator should be well versed with software security issues and should thoroughly understand the case study approach. His goal should be to facilitate learning for all, by presenting the case in an unbiased manner and then moderating the discussion. The facilitator may have to assess the students according to the case study solutions and discussions.

Experienced people from industry add a lot of value to the case study while it is being conducted in the classroom.

6. Evaluation and Assessment

A multi-layered evaluation takes place in a case study-based approach. The facilitator along with the subject matter experts (if any) assess the solutions. Participants receive peer reviews and can evaluate themselves. The written reports (detailed solution report and individual reports) can be used for more concrete assessment.

Class participation is also an excellent tool to engage and assess participants. Here the active learning skills of the participants are judged. This is reflected through the questions she asked in the class. This also includes the relevant concerns and issues raised by the student during the discussions while the solutions are being presented by various student groups and when the facilitator is presenting the case study to the participants.

7. Our Experiences

The case study was conducted as part of a first course in Software Engineering. The course was attended by both undergraduate and graduate students in Computer Science and Engineering. The case was assigned to three teams, with each team working on two out of three sets of problems. The teams were so enthusiastic about the case that one team worked on all three problem sets. They discussed about security issues at length and did a thorough study to come up with a comprehensive list of possible security threats, each supported by examples. Teams developed a big number of threat trees and Data Flow Diagrams, which was unexpected, as we wanted them to come up with the most important ones only.

Our industry contacts were present during the case study presentation as well as solution presentation and discussion by student teams. This provided a platform for students to get their queries solved at a fast pace.

A case specific feedback gave the following results:

![Figure 2: Usefulness of Case Study Approach for teaching Security](image)

The students reported that they think that the case study can be used for teaching security (Figure 2). Most of the students found the exercises to be challenging and relevant to issues they have faced during various projects (Figure 3).

![Figure 3: Usefulness of Exercises given in Case Study](image)

Thus we can say that case study approach for teaching security brought the focus to security issues and inspired students to work more on the subject.

8. Conclusion

The case study approach appears as a promising candidate that can a student’s focus to security related
issues in software development. They not only learn the concepts, and related issues, but also get a hands-on experience by analyzing, applying and evaluating what has been taught in books. It also provides a new perspective that helps the audience to understand issues related to writing secure code. The future steps include the validation of this case study in a corporate training setup. The results have reinforced our views that case studies can work as effective teaching and learning instruments for providing an integrated view of fundamentals as well as the industry practices, and hence can fill the void between the academic instruction and the industry practice.

9. References


