An Engineering of Secure Mobile Applications Course

xxxxxxxxxxxxxxxx
xxxxxxxxxx
xxxxxxxxxx
xxxxxxxxxx
xxxxxxxxxx, xx, xxx
xxxxxx@xxxxx.xxx

xxxxxxxxxxxxxxxx
xxxxxxxxxx
xxxxxxxxxx
xxxxxxxxxx
xxxxxxxxxx, xx, xxx
xxxxxx@xxxxx.xxx

ABSTRACT

Mobile technology has not only transformed our computing experience, but our everyday lives as well. Unfortunately the benefits of mobile computing come with profound dangers. Due to their mobile nature, malicious applications may access a wide variety of information typically unavailable on traditional computers. Some of which include the user's location, contact lists, and the ability to access premium services such as voice calls, text messages, and wireless data, affecting the customer's bill.

Many universities have created courses or entire programs designed to educate students in programming mobile devices, creating applications using proper software engineering principles, or with a focus on creating secure software, but as separate disciplines. Unfortunately there are no known courses which combine these three foundations: creating robust, secure mobile applications while keeping appropriate attention on the software engineering process and its benefits.

In the following paper, we propose a new course entitled *Engineering of Secure Mobile Applications*; we describe the need for the course and clearly define it so that other instructors may incorporate it into their curriculum.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computers and Information Science Education- Computer science education; Curriculum

General Terms

Design, Security, Verification

Keywords

Software Engineering Education, Mobile Software Development, Mobile Security

1. INTRODUCTION

Mobile computing is ubiquitous in today's world, allowing users to do everything from update their Facebook status to purchase stocks. Recently the security of these mobile applications (apps) has come under increased scrutiny, often due to malicious software and vulnerabilities. Users have suffered from instances of compromised personal information, devices that no longer operate as designed, and increased fees and phone bills. Additionally, since mobile software is typically created for profit, software developers need to balance the very fundamentals of software engineering: schedule, cost, and quality.

Students today are typically taught how to program mobile applications, basic software engineering principles, or proper techniques for creating secure software. We are not aware of any course that combines these three fundamental principles of software development to prepare students to address the need for security in the mobile application engineering process.

In order to address these issues, we are proposing an undergraduate *Engineering of Secure Mobile Applications* course which will combine three existing disciplines: computer science, software security and software engineering. The primary objective of the course is to instruct students in building a robust mobile application using the software engineering mindset and a continuing focus on ensuring its security not only for its initial release, but for future maintenance and upgrades as well.

Good software engineering practices focus on producing a high quality product that is on time and in budget, using a defined process. Various *umbrella*, or helper components that are typically a part of modern projects using a proper software engineering process; these include the use of a version control system, testing, teamwork, and proper requirements and design planning [17].

Mobile applications create a unique opportunity for software developers in that they allow access to information and features typically not possible in conventional, desktop applications. Additionally, mobile applications allow the developer to easily update software for feature enhancements, bug fixes, or support for hardware devices. Finally, small development teams or even individuals have the capability to create, release and support apps used by millions of mobile users.

These benefits also create a unique set of challenges for mobile developers. The access to information and mobile features creates an attack mechanism for malicious users, who can use this data to their advantage in a variety of ways including data theft. Developers are also tasked with rolling out new software versions in an extremely expedited fashion, which can expose the temptation to ship or abbreviate vital security planning or testing.

The Engineering of Secure Mobile Applications course would help to prepare students in overcoming these challenges. Students would optimally have a reasonably proficient background in software engineering, mobile development, and security. However, the amount of recommended proficiency could be altered depending on the depth the instructor wishes to explore these areas in their class.

While this is the first known proposed course in Engineering Secure Mobile applications, there are a substantial amount of curriculum this class is based upon. Numerous institutions across the country have courses which teach mobile application development [5] through the use of the iOS [6], Android [1] and Microsoft platform [4]. Other universities offer courses in mobile security [7] or software engineering [8].

This course is innovative because no other known programs integrate these three fundamental topics to allow students to understand and experience the benefits and challenges of these concepts to create secure and profitable mobile software.

The rest of the paper is organized as follows. Section 2 describes information about the course including weekly topics, two projects, and the vulnerability of the day activity. Section 3 provides teaching suggestions for instructors, and section 4 discusses some challenges which may be encountered when teaching this course. Finally, section 5 provides a summary of our work.

2. ABOUT THE COURSE

The course should be a mixture of lectures, in class activities, discussions, and a team based project. The lectures will serve to introduce core concepts while the project, activities, and discussions will serve to reinforce concepts. Other components of the course will include homework assignments, vulnerability of the day discussions, and a significant course project.

2.1 Student Background

Optimally, students should have some background in software engineering, security, or mobile software development. A recommended prerequisite is *Computer Science 1* or an equivalent introductory programming course. However, the depth of the students' knowledge and experience required as course prerequisites is dependent upon the instructor's desire to deeply explore these issues.

2.2 Recommended Text Book(s)

Textbook(s) will be selected based upon their relevance and understandability. Due to the fast moving nature of mobile development and malware, examples will be taken from the web and external readings, while the textbook will be expected to provide a firm theoretical foundation. At the time of publication, example textbooks include *Mobile Design and Development* by Fling [11] and *Mobile Design Pattern Gallery* by Neil [15].

2.3 Weekly Topics

In Table 1, we describe an example set of weekly classroom topics and project deliverables. In this example, we assume a 15 week term, and that many of the students already have a reasonably proficient background in mobile development, software engineering, and secure software development. We do however, provide class time to reacquaint students with these topics as they are foundations of the course. The instructor is encouraged to alter this schedule to match the skill set of their students and the length of their course term. We would also like to note that in this schedule, we are keeping the course technology agnostic and are not choosing a mobile platform such as iOS, Android, or Windows to use in the classroom. Based on the situation, and preferences of the instructor, we encourage the selection of a specific platform for use throughout the term. General guidelines for each week are described below:

1) Mobile Development, Security & Software Engineering Review
Reacquaint students with these three core foundation topics of the course. The depth which these areas are explored will largely depend upon the skill sets of the students and the depth which the instructor hopes to explore these topics in their class. Outside readings and small exercises are encouraged to reacquaint students with these topics. Hands on activities are also encouraged.

2) General Security Concepts in Mobile Applications
The instructor may choose to discuss security issues with a focus on a desired technology selected for the class, or make it generic for all mobile computing. Possible topics include secure data transmission and storage, types of attacks on mobile applications, and the ways that security breeches affect users.

3) Vulnerability Assessment Tools & Techniques
Discuss various manual and automated techniques of assessing the vulnerability level of mobile applications and how they may be integrated into the mobile engineering process. Example tools include ASEF[1] and Androguard[2].

4) Planning: Risk Assessment & Security Test Planning
Create test plans and perform risk assessments of mobile applications and potential vulnerabilities of the applications. An emphasis will be on creating a mobile security test plan, demonstrating its importance, and how it should be integrated into the software development life-cycle in mobile applications.

5) Design: Threat Modeling
Introduce concept of threat modeling in mobile applications and how they may be integrated into the software engineering process. Potential topics include software, attacker, and asset based threat modeling.

6) Discussion: Defensive Coding Practices in Mobile Development
Discuss defensive coding in mobile applications and demonstrate its importance the software and mobile engineering process. Potential topics include principle of least privilege,

[1]https://code.google.com/p/asef/
[2]https://code.google.com/p/androguard/

Table 1: Weekly Topics

Week	Classroom Topics	Project
1	Mobile Development, Security and Software Engineering Review	Team Formation
2	General Security Concepts in Mobile Applications	Project A Application Selection
3	Vulnerability Assessment Tools & Techniques	
4	Planning: Risk Assessment & Security Test Planning	
5	Design: Threat Modeling	Project A Final Deliverable
6	Discussion: Defensive Coding Practices in Mobile Development	Project B Start
7	Implementation: Defensive Coding Practices	Project B Initial Design
8	Creating Maintainable Secure Mobile Software	
9	Repairing Vulnerabilities Using a Proper Software Engineering Process	Project B Release 1
10	Vulnerability Resources	Project B Cross-Team Testing
11	OS Vulnerabilities	Project B Release 2
12	Fuzz Testing Introduction & Tool	
13	Fuzz Testing Tool Development	Project B Release 3
14	Deployment & Distribution: Patching, Security Managers	
15	Future Trends in Mobile Security	Project B Final Release

protection against injection attacks, *defense in depth* [2], and secure data storage.

7) Implementation: Defensive Coding Practices
Implementing the discussed defensive coding practices will serve to reinforce this critical topic. Potential activities include a student *show & tell* where they discuss and demonstrate how they performed defensive coding in their project and receive feedback from their classmates, or by having students perform cross team testing on code written by other teams.

Code inspections are a helpful way to ensure proper defensive coding practices have been followed. The concept of code inspections in mobile development is introduced here, including its importance and how it may be integrated into the mobile engineering process. An example activity is to provide students with some mobile code with known vulnerabilities and have the students perform a code inspection to detect the vulnerability. A subsequent discussion would consider the benefits and drawbacks of code inspections in the mobile development process from a security perspective.

8) Creating Maintainable Secure Mobile Software
Creating maintainable software from a functional perspective is important. Bug fixes and feature additions should be implementable as cheaply and easily as possible from both a feature and a security perspective. Possible security updates may include those to SSL libraries, with a recent example being updates required due to the heart bleed vulnerability [3]. A special focus will be on how the mobile application should be maintainable from a security perspective at every phase of the development cycle.

9) Repairing Vulnerabilities Using a Proper Software Engineering Process
Potential vulnerabilities and defects should be planned for when developing any type of application. A proper mitigation strategy, sufficient documentation, and adherence to coding standards are typically used to help alleviate the costs of these changes. This discussion will focus on how vulnerabilities may be planned for and repaired with as little negative ramifications on the mobile development process as possible.

10) Vulnerability Resources
Ensuring the security of mobile applications is never a completed process. Developers must be up to date on the latest vulnerabilities and methods of protecting their applications. An emphasis will be placed on discussing resources for assisting this process. Some potential resources include The National Vulnerability Database[3] and the Contagio Mobile mini dump[4].

11) OS Vulnerabilities
Mobile devices and operating systems often suffer from security vulnerabilities and may expose mobile applications to exploits and malware attacks. This discussion will focus on methods of protecting mobile applications from these exploits including the importance of updating mobile applications to work with the security patches released by the mobile operating system vendor and to ensure that mobile applications are built with adequate security procedures to protect against common operating system vulnerabilities. An example discussion might be whether to support older vulnerable mobile OS's or to require an upgrade to continue use of the application.

12-13) Customized Fuzz Tester
A fuzz testing tool is a type of exploratory testing tool used to find weaknesses in applications by scanning its attack surface. In this project, students will create a fuzzer from scratch using the programming language of their choice, with possible recommended languages being Ruby, Python or Java.

Creating a fuzz testing tool will help students to understand how they work along with the types of attacks they are capable of finding; thus allowing them to keep this in mind when creating their software leading to better defensive coding strategies. Fuzz testing applications will be evaluated on its ability to find vulnerabilities in its target, along with its ability to report its findings in a robust, useful format.

[3]http://nvd.nist.gov
[4]http://contagiominidump.blogspot.com

14) Deployment & Distribution: Patching Security Managers
A discussion of distribution and patching strategies for mobile applications. Potential topics include processes for implementing vulnerability fixes in existing mobile applications along with case studies about problematic deployment strategies and what can be learned from these mistakes.

15) Future Trends in Mobile Security
While it is impossible to predict the future, where is mobile security heading? What new types of attacks may come to fruition? How will new development in both mobile hardware and software make users more vulnerable to attack? These questions and relevant recent developments in mobile security will be discussed. A possible activity may be to break the students into separate groups, with some groups describing what mobile computing may look like in five years, with another group formulating an attack against this predicted mobile environment.

2.4 Course Project

We recommend that the instructor use two team based projects in their course. The first project (A) should last approximately five weeks and will have students find and repair vulnerabilities in an existing mobile application. The second project (B) should last approximately ten weeks and will have students create a mobile application from scratch. We encourage instructors to alter these projects and timelines based upon their experiences and how they see most appropriate for their classroom. The course projects should reinforce classroom concepts, provide real world experiences to students, and demonstrate the relevance and importance of creating secure mobile applications using proper software engineering techniques.

2.5 Project A: Repairing Vulnerabilities

The first project will last approximately five weeks and will entail students repairing vulnerabilities in existing open source mobile applications. We recommend forming teams of 4-6 students since this is often the size of groups in industry and has been found to be conducive to student learning in previous projects [12, 16]. This may also be completed individually by students, or by using smaller teams if appropriate. Each team will examine an open source mobile application with a publicly accessible version control system for vulnerabilities. Instructors may provide students with predetermined open source applications, or may allow the students to choose the application themselves(instructor approval is likely needed). Potential sources of open source mobile applications include F-Droid for Android[5], Code4App[6] for iOS, and SourceForge[7] for Windows mobile applications. Malicious Android app examples can be attained from Contagio Mobile Minidump.

Students will be asked to identify the vulnerabilities, provide justification how they identified the vulnerability, why the vulnerability was introduced (to the best of their ability using the version control system), how they made the project more secure, and how they repaired the vulnerability. At the conclusion of the project component, students will be encouraged to submit their results to the authors of the original open source project. The primary learning objective of the project will be to instruct students how to identify, understand, and repair vulnerabilities in mobile software. This will also serve as a supplemental introduction to mobile application development.

The project component should last approximately 5 weeks of a 15 week term, with its primary deliverable being the final report. Students should be evaluated on the thoroughness of the report, the relevance and quality of the vulnerability tests carried out, and the quality of the repair of discovered vulnerabilities. Projects should also be assessed on the overall software process used in the repair including documentation and the testing of the application for functional correctness.

2.6 Project B: Developing an Application

The second project will last approximately 10 weeks of a 15 week term and entails students creating a small mobile application from scratch. Teams should be comprised of 3-5 students and should be expected to create a mobile application which is secure while following proper software engineering processes. Examples of good software engineering practices include proper use of version control, development and maintenance of appropriate documentation, following proper and defined software development process, and formulation of functional testing processes.

The instructor will define the software product for teams to create and will also serve the role of customer. While the type of software product is the preference of the instructor, it should focus on software which, by it its nature, is more susceptible to vulnerabilities. Examples include applications which transmit data, store encrypted information, or have a wide range of access to Internet resources. The final software product could be a secure file transmission, communication, or financial app. The instructor may choose to have all teams work on the same project description or may provide each team with a different project assignment.

Teams will be expected to deliver defined sets of functionality for each release in order to mimic the frequent release structure of a real world project. This would also serve to demonstrate the importance of good system design as well as creating software that was not only secure for the current release, but for future releases as well. Additionally, with each release teams will be expected to use a myriad of existing security tools and techniques to demonstrate that their application has been designed and implemented in a secure manner. Example tools could include Androrisk [8], Drozer[9], and the Android Security Evaluation Framework (ASEF).

The application will be developed iteratively, with portions being submitted at each major release. The final submission will be a culmination of all releases to form a completed application. This will mimic the development process of many mobile applications [10,18] and will enable students to experience the creation maintainable and extensibility of mobile software from a functional and security perspective.

The following will be the significant deliverables for each team, based upon a 15 week semester and Project A (above) completing at week 5:

[5]http://f-droid.org
[6]http://code4app.net
[7]http://sourceforge.net/directory/os%3Awinmobile/

[8]https://androguard.googlecode.com/hg/androrisk.py
[9]https://www.mwrinfosecurity.com/products/drozer/

Week 6: Project Start
The project will begin with a short introduction of the project goals and requirements. While the project requirements will be provided to the students in as complete a manner as possible, there will undoubtedly be numerous requirements questions. This is quite beneficial to the learning process, as the requirements elicitation phase is an important aspect of software engineering projects [17].

Week 7: Initial Design
Students will submit their design and plans to the instructor for evaluation. A brief, semi-formal class presentation of these plans is encouraged for teams to receive feedback from their classmates on their design. Students will be evaluated upon their overall system design, but largely through a security perspective.

Week 9: Release 1
Teams will submit a limited but functional version of the mobile application with the first release subset of functionality specified by the customer/instructor. As with all releases, students will be expected to demonstrate both the functionality and security of their application, and that all proper software engineering processes have been followed.

Week 10: Cross-Team Testing
Teams will submit their application to another team for cross-team testing. Each team will be expected to conduct a robust set of security vulnerability tests on the target application using a mixture of existing tools, techniques, and informal processes with the goal of finding vulnerabilities or insecure areas of the application. The main benefits of this activity are that teams witness how other teams code and protect their applications and that teams will code more defensively knowing that their applications will be scrutinized by their peers. Finally, through their attempts to exploit their classmates applications, students will begin to experience how hackers or malicious users may attack applications. Teams will be evaluated on the thoroughness of their cross-team testing report and the robustness of attacks (successful or not) on their target application.

Week 11: Release 2
Similar requirements to Release 1, but with more expected functionality being delivered. Students will be expected to formulate and execute a plan to fix vulnerabilities found in cross-team testing.

Week 13: Release 3
Similar requirements to Release 1, but with more expected functionality being delivered. Functionality in this release should include elements of higher risk/vulnerability from a security perspective.

Week 15: Final Release
The final release will be a culmination of the team's work and will result in a final, fully functional application which has been demonstrated to be secure by each software development team. Teams will be expected to conduct a final presentation to the class demonstrating their application, highlighting development measures designed to ensure its security, and demonstrating how those measure can repel specific vulnerabilities or attacks.

Teams will be evaluated by how well they followed proper software engineering development techniques, developed and adhered to a proper security test plan, any applicable documentation, the quality of their final presentation, and overall application functionality. During the project evaluation, the instructor is encouraged to check for vulnerabilities using existing tools, by analyzing the team's source code, or through the use of manual evaluation methods. Teams will also be evaluated on how well they reacted to issues and problems discovered by the instructor and other teams. The instructor should also be sure to manually examine the team's source code for maintainability from both a functional and security perspective. The primary learning objective of this activity will be instruct students in designing, creating and maintaining secure mobile applications with the proper software engineering mindset.

2.7 Vulnerability of the Day

The vulnerability of the day (VoTD) activity will be carried out in the majority of classroom sessions and serves to acclimate students with relevant, real world mobile vulnerabilities, demonstrate the importance of creating secure mobile applications and to keep the student interest in the course high. The activity demonstrates the importance of creating secure software and the negative implications of vulnerabilities from a technical, ethnical, and business perspective.

Instructors will first identify interesting, relevant and informative real world examples of security vulnerabilities in mobile applications. Next, students will spend approximately 5 minutes reading about this example, and roughly 5 minutes in discussion. Possible topics of discussion include how the the vulnerability was fixed, why it occurred, what its implications were, how would the students have fixed it, and what recommendations the students have for ensuring the problem does not occur again. While a benefit of this activity is that students will gain experience from real world examples, the primary objective of this activity should be to foster critical thinking, not necessarily in formulating a perfect solution. A similar activity has been successfully used in previous computing courses [13, 14]. Due to the fast paced nature of mobile computing and security, instructors will likely need to update a large number of the VoTD examples for each course term. However, historically significant or relevant examples can be used across terms.

An example VoTD activity is the repair process of the Heartbleed bug in mobile applications [19]. As part of this VoTD discussion, students may discuss how repairing this issue could properly conducted in a cheap, efficient manner using proper software engineering techniques. An additional discussion may revolve around different testing tools and techniques [9] to ensure that the vulnerability had in fact been repaired.

3. TEACHING SUGGESTIONS

Mobile software development and security are extremely fast moving topics. What was important several months ago may no longer be relevant today. Conversely, foundations of security, such as the principle of least privilege which have been important aspects of software security for quite some time will remain applicable for the foreseeable future. The instructor will need to balance the instruction of fundamental aspects of security with cutting edge and relevant topics.

Focusing too much on fundamental aspects of security risks losing student interest, and may not get them ready for today's world. Conversely, not giving the students a good, fundamental understanding of these principles will only familiarize students with topics which are likely to become quickly outdated and irrelevant before graduation. A good balance of theoretical lectures and current hands-on activities and readings should be used in this course.

4. COURSE CHALLENGES

While we believe that this course is a powerful and integral part of any mobile computing or computing curriculum, it is not without its challenges. This course is very specialized in a way that many institutions will be unable to offer due to resource constraints. Many institutions do not even offer software engineering, mobile software development, or security courses, so this next generation course may be far too specialized. Additionally, finding students with the necessary prerequisite skill-set to take the course may prove difficult as well.

A significant component of the course is its vulnerability of the day activity. This activity will need to be significantly updated for each course iteration because the need to keep examples current and the fast paced nature in which mobile security changes. Examples which were relevant one term ago may no longer be applicable and will need to discarded. This heavy workload may be too much for some instructors. One alternative may be to cut down on the number of these vulnerability of the day activities.

The course being proposed is not merely a software engineering, mobile development, or security course, but is a hybrid of these fields. The instructor will need to make sure that they are properly balancing specializations to keep the course correctly focused.

5. SUMMARY

We have proposed a course *Engineering Secure Mobile Applications* which we hope institutions will be able to add to their curriculum providing example projects, textbooks, and a weekly outline. While there are numerous challenges to the course including technology and its specialization, we believe that it fills a hole in many mobile, software engineering and security programs.

6. REFERENCES

[1] Android app development. http://ccpe.kennesaw.edu/computers/syllabus/fvcm1041.pdf.

[2] Defense in depth. `http://www.nsa.gov/ia/_files/support/defenseindepth.pdf`.

[3] Heartbleed in openssl: Take action now! http://www.symantec.com/connect/blogs/heartbleed-openssl-take-action-now.

[4] Mobile app design. https://sites.google.com/site/specialprojectsmobile/syllabus.

[5] Mobile application development. `http://www.csit.parkland.edu/~dbock/Class/csc212/Syllabus.html`.

[6] Mobile application development. `http://cidse.engineering.asu.edu/wp-content/uploads/2013/01/CSE494-S12Mobile_Application-Syllabus.pdf`.

[7] Mobile security. http://wnss.sv.cmu.edu/courses/14829/f10/syllabus.php.

[8] Software engineering at rit. http://www.se.rit.edu.

[9] Heartbleed detector: Check if your android os is vulnerable with our app. https://blog.lookout.com/blog/2014/04/09/heartbleed-detector/, 2014.

[10] P. Abrahamsson, A. Hanhineva, H. Hulkko, T. Ihme, J. Jäälinoja, M. Korkala, J. Koskela, P. Kyllönen, and O. Salo. Mobile-d: An agile approach for mobile application development. In *Companion to the 19th Annual ACM SIGPLAN Conference on Object-oriented Programming Systems, Languages, and Applications*, OOPSLA '04, pages 174–175, New York, NY, USA, 2004. ACM.

[11] B. Fling. *Mobile Design and Development: Practical Concepts and Techniques for Creating Mobile Sites and Web Apps - Animal Guide.* O'Reilly Media, Inc., 1st edition, 2009.

[12] J. Guo. Group projects in software engineering education. *J. Comput. Sci. Coll.*, 24(4):196–202, Apr. 2009.

[13] D. Krutz and M. Lutz. Bug of the day: Reinforcing the importance of testing. In *Frontiers in Education Conference, 2013 IEEE*, pages 1795–1799, Oct 2013.

[14] A. Meneely and S. Lucidi. Vulnerability of the day: Concrete demonstrations for software engineering undergraduates. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 1154–1157, Piscataway, NJ, USA, 2013. IEEE Press.

[15] T. Neil. *Mobile Design Pattern Gallery, Color Edition.* Oreilly and Associate Series. O'Reilly Media, Incorporated, 2012.

[16] D. Petkovic, G. Thompson, and R. Todtenhoefer. Teaching practical software engineering and global software engineering: Evaluation and comparison. In *Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, ITICSE '06, pages 294–298, New York, NY, USA, 2006. ACM.

[17] R. Pressman. *Introduction to Software Engineering.* 7 edition, 2009.

[18] C. Scharff and R. Verma. Scrum to support mobile application development projects in a just-in-time learning context. In *Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering*, CHASE '10, pages 25–31, New York, NY, USA, 2010. ACM.

[19] Y. Zhang, H. Xue, and T. Wei. If an android has a heart, does it bleed? http://www.fireeye.com/blog/technical/2014/04/if-an-android-has-a-heart-does-it-bleed.html, April 2014.