An Engineering of Secure Mobile Applications Course

Daniel E. Krutz and Samuel A. Malachowsky
Software Engineering Department
Rochester Institute of Technology
1 Lomb Memorial Drive
Rochester, NY 14623
{dxkvse, samvse}@rit.edu

ABSTRACT

Mobile technology has not only transformed our computing experience, but our everyday lives. The smartphone & tablet allow users to locate their favorite restaurants, view real time news updates, and stay in constant communication with friends . Unfortunately, with this benefits come profound dangers. Due to their mobile nature , malicious applications may gain a wide variety of information which was not typically available on traditional desktop computers such as the user's location, their contact lists and the ability to charge premium services to the user by making voice calls or sending large amount of text messages which could drive up the customer's bill.

Numerous universities have created courses and even entire programs in educating students in programming of mobile devices, software engineering or developing secure software. Unfortunately, there are no known courses which combine these foundations of creating robust, secure mobile application which are maintainable from a security and overall software engineering perspective and are created on time and on budget.

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

[really make sure that the abstract is solid]

Categories and Subject Descriptors

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

General Terms

Design , Reliability, Verification

SIGCSE'15, March 4–7, 2014, Kansas City, MO, USA.

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 *apps* has come under increased scrutiny due to malicious software and vulnerabilities have caused detrimental effects for users such as compromising their personal information and hindering their mobile experience. Additionally, since mobile software is typically created for profit , software developers need to balance the basic fundamentals of software engineering to create the project on time, on budget, and with high quality.

Students today are typically taught how to program mobile applications, basic software engineering principles or proper techniques for creating secure software. Unfortunately, we are not aware of any course that combines these three fundamental principles of software development to prepare students for creating secure software using properly software engineering principles. *[Dan says: I feel like this sentence can be cleaned up]*

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 will be to instruct students in building a robust mobile application using the Software Engineering mindset needed to deliver it on time, on budget and high quality; but with a continuing focus on ensuring its security not only for its initial release, but for future releases as well.

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

[find research that backs up the notion of building software that is maintainable from a security perspective] [cite]

Mobile applications create a unique opportunity for software developers in that they allow access to information and features which not possible in conventional, desktop applications. Additionally, mobile applications allow the relatively

mechanism for updating 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.

However, these benefits also create a unique set of challenges for mobile developers. The access to information and mobile features also creates an attack mechanism and honey pot for malicious users who can use this data to their advantage in a variety of ways including data theft. Developers are also tasked rolling out new software versions in an extremely expedited fashion**[Add more to this]**

The Engineering of Secure Mobile Applications course would help to prepare to 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 will be altered depending on the depth the instructor wishes to explore these issues in their class .

This course is innovative because no other known programs integrate these three fundamental topics to allow students to see the big picture**[reword]** of creating secure mobile using the proper software engineering mindset which are required to create mobile applications which are secure, but also profitable as well . *[Dan says: This needs to be cleaned up]*

[really drive this part home]

In the following paper, we describe

The rest of the paper is organized as follows. Section **??**

Typical users today are abandoning traditional desktop and laptop computers for portable devices such as smart phones and tablets. Their computing needs are primarily social networking, web searching, and email. Some users extend this to applications such as word processing, spreadsheets, and financial recording. In almost all cases, the smart phone and/or tablet can fulfill their requirements.

Application developers, trained traditionally, develop for devices that can store programs and data as well as execute them. Portable devices, on the other hand, are limited in what they can store and execute. Applications must be divided between the portable device and the cloud with portions moving from one to the other to execute properly. Threads must be carefully controlled. This requires specialized training in order to create applications that both efficient and are not easy targets for malware.

While users are very tool savvy, they are technologically illiterate. They do not understand (and have no reason to understand) what happens âĂŸunder the hoodâĂŹ. This makes the user experience extremely important. The focus on making a user interface intuitive to a typical user could well cause security to be compromised.

Portable device applications are maintained via frequent downloads from a service provider rather than occasional batch downloads. This requires applications that are designed for this type of maintenance in order to avoid creating opportunities for security breaches.

The combination of distributed applications, unsophisticated users, and frequent maintenence can create opportunities for malicious hackers.

TYPES OF MALWARE THAT ATTACK PORTABLE DEVICES

Malware that attacks portable devices can be broken into several categories (Symantec uses these categories; `http://www.symantec.com/content/en/us/enterprise/other_resources/b-intelligence_report_05-2014.en-us.pdf`):

Track user Spy on the individual using the device. Intercepting SMS messages, phone calls, photos, etc. Steal information Collection of both device-specific and user-specific data Traditional threats Traditional malware functions such as back doors Reconfigure device Elevate privileges or modify the operating system Send content Sending SMS messages that ultimately result in charges to the user or using the device to send copious amounts of spam Adware annoyance Display unwanted and unwarranted ads or generally disrupting the users ability to employ the device

Of these categories tracking user, sealing information, and traditional threats were the most commonly encountered.

Of these six categories, tracking the user, collecting device and user specific information, and sending content are much more specific to portable devices than traditional devices.

Another way malicious hackers can compromise portable devices is via Wi-Fi eavesdropping. Since this is perfectly legal, it is imperative that the device employ methodology to prevent eavesdropping from putting the device or the user at risk.

2. COURSE NEED

This course will focus on helping the student understand how malware is inserted into a device as well as ways to secure the data within the device both while inside the device and while it is being transmitted. Further, it will provide the necessary information for the student to repair defects in the application via normal software maintenance.

3. 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 server to reinforce concepts. The primary components of the course will include lectures, in class activities, homework assignments, vulnerability of the day discussions and a significant course project.

3.1 Student Background

Optimally, students should have some background in Software Engineering, security or mobile software development. 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.

3.2 Recommended Text Books

Text book(s) will be selected based upon their A) Relevance B) 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. Due to the fast past nature of security and mobile software development, it is not possible for us to recommend any specific textbooks since they will undoubtedly quickly become out of date and irrelevant for course use.

At this time, it is impractical to find a textbook for all of the topics in this course. Parts of this course are covered in several available Software Engineering texts. However, to complete course coverage articles and reports from groups such as IEEE, ACM, Symantec, McAfee, and many more can be used.

3.3 HomeWork

Students will be asked to deliver short homework assignments and in class activities. These will serve to reinforce the lecture topics and introduce students to concepts in a more hands on manner.

[Add to this, or move it or something]

3.4 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 depending on 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 and institution, we encourage the selection of a specific platform to use throughout the term for all activity and project work. We describe general guidelines for each week below:

1) Mobile Development, Security & Software Engineering Review
Reacquaint students with the three core foundation topics of the course. The depth which these areas are explored will be largely dependent 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.

2) General Security Concepts in Mobile Applications
The instructor may choose to discuss security issue 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 & storage, types of attacks on mobile applications,

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 [3] and Androguard [1].

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 cyle 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 how it fits into the mobile engineering process. Potential topics include principle of least privilege, protection against injection attacks, *defense in depth* [5], and secure data storage.

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

8) Creating Maintainable Secure Software
Creating maintainable software from a functional perspective is important. Bug fixes, and feature additions should be implementable as cheaply and easily as possible. Mobile applications should be able to be updated quickly and as easily as possible from a security perspective as well. Possible updates may include those to SSL libraries, with a recent example being updates required due to the heart bleed vulnerability [7]. A special focus will be on how the mobile application should be maintainable from a security perspective at every phase of the development cycle. **[find citation]**

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 remediation plan, good coding practices, 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. *[Dan says: really clean this up]* **[State the steps that this includes?]**

10) Vulnerability Resources
Ensuring the security of mobile applications is never a completed process. Developers need to 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 [12] and the Contagio Mobile mini dump [4].

11) Code Inspections
Introduce code inspections in mobile development and describe its importance and how it may be integrated into the mobile engineering process. An example activity may be to provide students with a portion of 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.

12)
Blah x

13) Deployment & Distribution: Patching Security Managers
Blah x

14)
Blah x

15) Future Trends in Mobile Security
While it is impossible to predict the future, where is mobile security heading in the future? 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? A possible activity may be to break the course into separate groups, with some groups describing what mobile computing will look like in five years, with another group formulating an attack against these new devices.

[Make sure all of these described topics match]

3.5 Course Project

We recommend that an instructor use two team based projects in their course. The first project (A) should last approximately five weeks and has students repair vulnerabilities in an existing mobile application. The second project (B) should last approximately ten weeks and has students create a mobile application from scratch. We encourage instructors to alter these projects and timelines based upon their experiences and how they see fit for their classroom. The course project 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.

3.6 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 [15,18]. However, this may also be completed individually by students, or by using smaller teams. Each team will examine an open source mobile application with a publicly accessible version control system for vulnerabilities. Based on the preference of the instructor, they may provide students with predetermined open source applications to be analyzed, or may allow the students to choose the applications with the approval of the instructor.

Students will be asked to first identify the vulnerabilities, provide justification why it was a vulnerability, 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 should last approximately 5 weeks of a 15 week term, with its primary deliverable being the final report. Students shall be evaluated on the thoroughness of the report, the relevance and quality of the vulnerability tests carried out to find the defect and the quality of the repair of discovered vulnerabilities. Projects will also be assessed on the overall software process used in the repair including documentation and the testing of the application for functional correctness. **[add more?]**
[Add to this]

3.7 Project B: Developing 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 the mobile application which is both secure and follows 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 first define a software project for teams to create and will also serve the role of the project customer. While the precise nature of the project will be based on instructor preference, they 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. Example projects include a secure file transmission app, **[add more]**. Based on the discretion of the instructor, they may choose to have all teams work on the same project, 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 as a demonstration of the importance of a good system design and in creating software that was not only secure for each release, but for future releases as well. Additionally, for 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. Some encouraged tools may be Androrisk [1], Drozer [6], and Android Security Evaluation Framework (ASEF) [3].

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 **[cite]** and will enable students to experience the creation maintainable and extensibility of mobile software from a functional and security perspective.

The following will be significant deliverables required by each Team:

[change weeks?]
Initial Design
Students will submit their design and plans to the instructor for evaluation. A brief, semi formal class wide 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

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	x
4	Planning: Risk assessment & test planning	x
5	Design: Threat modeling	Project A Final Delieverable
6	Implementation: defensive coding practices	x
7	Defensive coding practices	x
8	Creating Maintainable Secure Mobile Applications	x
9	Repairing Vulnerabilities Using a Proper Software Engineering Process	x
10	Vulnerability Resources	x Project B Initial Deliverable
11	Code Inspections	x
12	x	x
13	Deployment & Distribution: patching, security managers	Project B Cross Team Testing
14	x	x
15	Future Trends in Mobile Security	Project B Final Deliverable

a security perspective.

Week 4: Release 1
Teams will submit a limited, but functional version of the mobile application with the desired functionality as stated by the customer/instructor. As with all releases, students will be expected to demonstrate the functionality and security of their application, along with ensuring that all proper software engineering processes have been followed.

Week 5: 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 testing on the target application using a mixture of existing tools, techniques and informal processes with the goal of finding vulnerabilities or weak areas of the application from a security perspective. The main benefits of this activity are that teams witness how other teams code and protect their applications and teams will code more defensively knowing that their applications will be scrutinized from a security perspective. Finally, through their attempts to exploitation 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 6: Release 2
Similar requirements to Release 1, but with more expected functionality being delivered.

Week 8: Release 3
Similar requirements to Release 1, but with more expected functionality being delivered.

Week 10: 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 entire class demonstrating their application, explaining how they ensured it is secure and development measures to ensure its security.

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, 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 the teams 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. Projects will be evaluated upon the how well they followed proper software engineering practices, the overall robustness of the application, their security test plan, and against the scores of several existing security testing tools.

[Make sure these releases match up with the large table]

3.8 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. Students will spend approximately 5 minutes reading about this example, and roughly the next 5 minutes discussing the example. 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 there 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 [16,17]. 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 for numerous 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, easier manner using proper software engineering techniques. An additional discussion may revolve around different testing tools and techniques [14] to ensure that the vulnerability had in fact been repaired.

4. TEACHING SUGGESTIONS

[update this section quite a bit] Mobile software development and security are extremely fast moving topics. What was important several months ago may no longer be relevant today. Conversely, security principles such as the principle of least privilege which were important aspects of software security are still relevant today. The instructor will need to balance the instruction of fundamental aspects of security, along with cutting edge and relevant topics. Focusing too much on fundamental aspects of security will not only lose student interest, but will not get them ready for the real world. Conversely, not giving the students a good, fundamental background understanding of these principles will only ready students for the ?now?, which will become quickly outdated and irrelevant before the students even graduate. A good balance of theoretical lectures and current hands on activities and readings should be used in this course.

5. 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. First of all, this course is very specialized which many institutions will be unable to offer due to unit's resource constraints. Many institutions do not even offer software engineering, mobile software development or security courses, so this next generation course being proposed may be far too specialized for many institutions. Additionally, finding students with the necessary prerequisite skillset to take the course may be difficult for many institutions as well.

A significant component of the course is its Vulnerability of the day activity. The vulnerability of the day activities will need to be significantly updated for each course iteration due to the need of keeping these 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 many 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 focus of the course on the correct path.

[Keep working on this section]

6. RELATED COURSES & PROGRAMS

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 [9] through the use of the iOS [10], Android [2] and Microsoft platform. [8]. Other universities off courses in mobile security [11] or software engineering [13].

[Dan says: Add to this, or is this suffice?]

7. FUTURE WORK

Dan

8. SUMMARY

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

9. REFERENCES

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

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

[3] Asef - android security evaluation framework. https://code.google.com/p/asef/.

[4] Contagio mobile malware minidump. http://contagiominidump.blogspot.com.

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

[6] Drozer. https://www.mwrinfosecurity.com/products/drozer/.

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

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

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

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

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

[12] National vulnerability database. http://nvd.nist.gov.

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

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

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

[16] 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.

[17] 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.

[18] 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.

[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.