Modules for Teaching Secure in Android Application Development

Christopher Doss, Ph.D.
cdoss@ncat.edu

Xiaohong Yuan, Ph.D.
xhyuan@ncat.edu

Varshar Chennakeshva
vkchenna@aggies.ncat.edu

Aakiel Abernathy
aabernat@aggies.ncat.edu

Kenneth Ford
kmford@aggies.ncat.edu

North Carolina A & T State University
Greensboro, NC 27411

Abstract - The rise of mobile computing devices has resulted in an increased emphasis on mobile application development courses within computing curricula. While there are many available teaching modules for app development, there is a dearth of materials that incorporate secure software development for mobile devices. The goal of this project is to develop teaching modules that highlight the need for security in app development. These modules are based on CERT Java secure coding rules applicable to developing Android applications published by the Software Engineering Institute at Carnegie Melon University. This paper describes the modules we developed/are developing. These modules are further development and extension of a set of hands-on labs developed and assessed in two courses in the Spring 2017 semester. The assessment results is also discussed. These modules can be adopted by instructors of Android application development.

Keywords

software security, secure coding, Android application development, teaching module

1 INTRODUCTION

The proliferation of mobile devices, especially those based on the Android operating system, has led to the development of courses targeting mobile application development. These courses differ from traditional computer programming courses as they must incorporate the features and limitations of mobile devices, including various sensors, limited battery life, accelerated operating system updates, and operations being restricted to background threads. However, the increased importance of mobile devices, especially their use for personal data, means there are potential security risks [1-3]. Thus, it is imperative that students understand how to mitigate security risks as they learn how to develop these applications. Unfortunately, there is a lack of readily available teaching modules that address security for mobile application development courses.

We are developing nine teaching modules with hands-on labs to teach students the common mistakes made by Android application developers, how these mistakes lead to security vulnerabilities and how these vulnerabilities can be mitigated by following Android secure coding rules. Each teaching module includes learning objectives, presentations, instructor notes, online resources, hands-on lab menu, quizzes, discussion questions, test questions, and example solutions to questions. The estimated completion time of each teaching module is between 1 and 4 hours. These modules are further development and extension of a set of hands-on labs for teaching Android secure coding. These hands-on labs were used in two classes in the Spring 2018 semester.

This paper is organized as follows. Section 2 discusses the nine modules. Section 3 presents a sample of one of the modules. Section 4 shows the assessment results of the hands-on labs based on which the modules are developed. We conclude in Section 5.

2 MODULES FOR TEACHING ANDROID SECURE CODING

2.1 Module 1: CERT Android Secure Coding Rules

This module provides an overview of the CERT Android Secure Coding Rules [4]. Hands-on lab: Students use SACH [5] to scan a vulnerable app for non-compliance of CERT rules. They will then correct issues and rerun to ensure issues have been addressed.

2.2 Module 2: Sensitive Content Provider

A content provider allows an application to share its data with other applications. While this allows for an enhanced overall experience, issues can occur when sensitive information is shared. When a content provider is declared public, sensitive information may be leaked to malicious apps. This module discusses the vulnerability of leaking sensitive information through a public content provider. Hands-on lab: students will create a malicious app to access sensitive information stored in SQLite database through a public content provider. Students will then make the content provider private, and verify that the malicious app can no longer access the sensitive information.

2.3 Module 3: Sensitive Activity

Intent filters allow an activity to be exported to other apps. This may enable unintended uses and abuses from malicious apps. This module discusses the vulnerability of not restricting access to sensitive activities. Hands-on lab: Students will use the Android Asset Packaging Tool (aapt) to identify activities with an intent filter, and develop a malicious app that accesses the activity to extract sensitive information. They will then correct this vulnerability and verify removal of the threat.

2.4 Module 4: Eavesdropping on Broadcast Message

Android apps use broadcast to send messages or intents to multiple applications. Broadcasts using implicit intent allow passive eavesdropping, denial of service attacks, or service hijacking. This module discusses different types of broadcasts, and how to prevent the vulnerability of broadcasting sensitive information using an implicit intent. Hands-on Lab: Students will implement a malicious app to eavesdrop a message broadcasted by a vulnerable app. The will then correct the vulnerable app and verify it is no longer susceptible to eavesdropping.

2.5 Module 5: Debuggable Apps

This module discusses how sensitive information can be leaked if an app is released as debuggable, and how to prevent the vulnerability. Hands-on Lab: Students will use Drozer [6], an open source analysis tool, to identify debuggable apps. They will then use the Android SDK tool adb (Android Debug Bridge) [7] to find the databases of a debuggable app. Next, students will insert entries into the database of the debuggable app. Students will then make the app non-debuggable, and verify that the app can no longer be exploitable.

2.6 Module 6: Logging Sensitive Information

Logging is useful to debug applications. However, developers may log sensitive information and forget to remove those logs before releasing the application. This module discusses the vulnerability of logging sensitive information. Hands-on lab: Students will use adb to retrieve log files, and examine them for sensitive information. Students will also correct the code, and verify that the app is no longer logging sensitive information.

2.7 Module 7: Storing Sensitive Information on External Storage

Files on external storage of a mobile device can be modified or read by other apps installed on the device for Android versions which allow read/write. The external storage can also be removed from the device and mounted elsewhere, and thus can be accessed by anyone who has the external storage. This module discusses the vulnerability of storing sensitive information on external storage. Hands-on Lab: Students will write a malicious app to read the content of a file stored on an external storage by a vulnerable app. Students will then modify the vulnerable app source code to store the file in the internal storage device, and verify the malicious app can no longer access the file.

2.8 Module 8: Storing Sensitive Information in Shared Preferences

Data can be stored in shared preference in the form of key-value pairs. Shared preference stores data in an xml file under the application directory. Shared preference files can be uploaded using the adb tool. This module discusses the vulnerability of storing sensitive information in shared preferences, and how to prevent the vulnerability by encrypting the data before storing it in shared preference. Hands-on Lab: Students will first add unencrypted user id and password to shared preference, then show how a malicious user can back up and view this data. Students will then encrypt the data and store it in shared preference.

2.9 Module 9: Allowing Database Backup

If the attribute android:allowBackup is set to true for an app, then all runtime app information such as shared preferences and databases can be backed up by anyone who has access to the device. This module discusses the vulnerability of allowing backup of SQLite database storing sensitive information. Hands-on Lab: Students will first backup the database of an app storing sensitive information to a PC using adb tool, and view the database content. Students will then set android:allowBackup to false, and verify that database can no longer be backed up to the PC. Students will also encrypt the sensitive information in the database.

3 SAMPLE CONTENT OF A MODULE

The following content is from *Module 3: Sensitive Activity*.

SENSITIVE ACTIVITY

Module Description: An activity may be exported to other apps if an intent filter is declared for the activity. This may lead to other apps, including malicious apps to activate the activity for unintended use. This micro-module discusses the vulnerability of not restricting access to sensitive activities.

This module focuses on basic knowledge of java programming and the use of exploiting various intent filters. It focuses on the basic constructs and useful commands such as widgets, activity calling, and extracting valuable information to execute certain tasks during the lab assignment. In this module, Student's will implement a malicious app that will have the ability to access sensitive information of another application.

Prerequisite Knowledge: Basic knowledge of programming (variables, widgets, implicit/explicit intent filters); knowledge of a scripting inside of a command prompt. Students should be able to read a manual page with guidance. Instructors should conduct a pre-assessment to confirm that students have the above background knowledge.

Length of Completion: 1-4 contact hours.

Level of Instruction: Undergraduate level

Learning Setting: This module is intended for in-class/online instruction.

Lab Environment: Android Studio 3.0.1 and the emulator

Activity/Lab Tasks: Students will use the Android Asset Packaging Tool (aapt) to extract androidmanifest.xml file from an apk file, and identify the activities in the app that have intent filters declared. Students will then write a malicious app to activate the activity to access sensitive information. The students will then make the activity not exported, and then verify the malicious app can no longer activate the activity to access sensitive information.

Lab Files that are Needed:

1. Vulnerable.apk (an unsecure application used to store user information). The user can add, update, and delete information in the SQLite database)

2. ContentProviderDB source code (the source code for the Vulnerable.apk). Can be downloaded at:

   http://ccd.ncat.edu/mobile/SWdevelopment/ContentProviderDB(Assignment).zip

Module Learning Outcomes:

1. Explain activity and sensitive activity

2. Explain the difference of implicit intent filter and explicit intent filter

3. Write a malicious app to activate the activity to access sensitive information

Instructional Files and Online Resources that are Needed:

Lesson: Sensitive Activity

- SensitiveActivity_Presentation.pptx
- SensitiveActivity_Lab.docx
- SensitiveActivity_LabFiles
- SensitiveActivity_LabSolutions.docx
- SensitiveActivity_AssessmentGuide.docx

4 ASSESSMENT RESULTS OF ANDROID SECURE CODING HANDS-ON LABS

Seven of the Android secure coding labs, not in the module format, were taught in a graduate level course "COMP 727 Secure Software Engineering" in Spring 2017. These labs include Modules 1 – 7 from Section 2 above. The class has 21 students. Seventeen (17) students submitted the lab assignment. The average score of the assignment is 86. After the students submit the lab assignments, the students were asked to participate in a survey. Twelve students participated in the survey.

The survey asked students to rate their level of knowledge in the learning objectives of the labs using a scale of 1 (very low) to 5 (excellent). The survey also asked students questions on the effectiveness of the course module. Table 1 shows the average student rankings on the learning objectives. Table 2 shows student responses to survey questions.

Table 1
Student self-ranking of learning objectives

Learning objective	**Average ranking**
Identify and correct security vulnerabilities in Android program according to CERT	3.08
Discuss methods to prevent security vulnerabilities in Android program	3.25
Demonstrate how some of the vulnerabilities can be exploited	3.25

Table 2
Student responses to survey questions

Question	**Response: Percentage**
How effective was the Android Secure Coding course module material?	Moderately effective 33.33% Very effective: 25.00% Extremely effective: 8.33%
How organized was the Android Secure Coding course module material?	Neither agree nor disagree: 8.33% Somewhat agree: 33.33% Strongly agree: 8.33%
How motivated were you to learn about Android Secure Coding material?	Neither agree nor disagree: 8.33% Somewhat agree: 33.33% Strongly agree: 33.33%
The hands-on lab exercises helped you better understand the material compared to having only power point presentation describing the concepts.	Neither agree nor disagree: 0.00% Somewhat agree: 25.00% Strongly agree: 41.67%
The hands-on lab exercise with SACH helped you understand security vulnerabilities in Android programs.	Neither agree nor disagree: 8.33% Somewhat agree: 41.67% Strongly agree: 33.33%
The hands-on lab exercises helped you better understand how some of the vulnerabilities in Android programs can be exploited.	Neither agree nor disagree: 8.33% Somewhat agree: 50.00% Strongly agree: 25.00%

Table 2
Student responses to survey questions

Question	Response: Percentage
You enjoyed doing the lab exercises	Neither agree nor disagree: 8.33% Somewhat agree: 25.00% Strongly agree: 16.67%
The learning objectives (listed in questions 1 above) were met.	Neither agree nor disagree: 16.67% Somewhat agree: 41.67% Strongly agree: 25.00%
The lab instructions were clear.	Neither agree nor disagree: 16.67% Somewhat agree: 0.00% Strongly agree: 8.33%
The level of difficulty in the lab exercises is appropriate.	Neither agree nor disagree: 25.00% Somewhat agree: 25.00% Strongly agree: 8.33%
The lab exercise helped you to develop problem solving and critical thinking skills, and ability to learn independently.	Neither agree nor disagree: 0.00% Somewhat agree: 50.00% Strongly agree: 16.67%
Approximately, how many hours did you spend on this hands-on assignment?	< 5 hours: 8.33% 6-10 hours: 33.33% 11-20 hours: 33.33% 21-30 8.33% >30 hours: 16.67%

Table 2
Student responses to survey questions

Question	Response: Percentage
The time I spent on the lab exercises was worthwhile.	Neither agree nor disagree: 16.67% Somewhat agree: 16.67% Strongly agree: 25.00%

The students commented that, through this module they learned how to use Android studio for android development, how to find vulnerabilities in an Android app, different types of vulnerabilities, and some good practices for coding and debugging an app. The major concern students had is that the lab instructions were not clear. Most of the students didn't have Android programming experience, which made this course module difficult for them.

The above hands-on labs were also taught in the class COMP365 Programming Methodologies in the Spring 2017 semester. There were 27 students enrolled in the course. However, only 10 submitted the assignment. The average score of the students' assignment was 76 out of 100. Students found the hands-on labs difficult because they lack background in Android programming, and that the lab instructions were not clear.

The initial assessment shows the hands-on labs were useful in introducing secure coding practice to Android developers. However, the lab instruction needs to be improved in terms of clarity. Therefore we are re-developing and re-organizing the lab menus into modules so it is easy to and use and can be widely adopted by instructors of computer/Android programming. Background information on Android programming was added to the modules to address the issue that some students lack knowledge and experience with Android programming.

5 CONCLUSIONS

We are developing nine modules that will enable instructors to incorporate security into their mobile application development courses. These modules include teaching materials, labs, quiz and test questions, and solutions. Our initial assessment indicates that the students were able to understand the vulnerabilities inherent in Android applications and how to address them. The modules will be incorporated into the undergraduate/graduate "ECEN 485/685 Application Development for Android Devices" course and the "COMP727 Secure Software Engineering" course in the Spring 2018 semester. These modules will be assessed again in these courses.

ACKNOWLEDGEMENT

This work is partially supported by the National Security Agency under the grant H98230-17-1-0400. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Security Agency.

REFERENCES

[1] Miller, K.W., Voas, J. and Hurlburt, G.F. 2012. "BYOD: security and privacy considerations", IT Professional, Vol. 14 No. 5, pp. 53-55.

[2] Yun, H., Kettinger, W. and Lee, C. 2012. "A new open door: the smartphone's impact on work-to-life conflict, stress, and resistance", International Journal of Electronic Commerce, Vol. 16 No. 4, pp. 121-152.

[3] OWASP. Projects/OWASP Mobile Security Project - Top Ten Mobile Risks. Retrieved January 7, 2017 from https://www.owasp.org/index.php/Projects/OWASP_Mobile_Security_Project_-_Top_Ten_Mobile_Risks.

[4] Seacord Robert. The CERT Oracle Secure Coding for Java: Android (DRD). Retrieved July 30, 2017 From https://www.securecoding.cert.org/confluence/pages/viewpage.action?pageId=1115.

[5] Aakiel Abernathy, Edward Hill, Xiaohong Yuan, Kevin Bryant, Jinsheng Xu, Kenneth Williams, SACH: A Tool for Assisting Secure Android Application Development, IEEE SoutheastCon 2017.

[6] MWR LABS. Drozer: Comprehensive security and attack framework for Android. Retrieved July 30, 2017, from https://labs.mwrinfosecurity.com/tools/drozer/

[7] Android Studio. Android Debug Bridge. Retrieved July 30, 2017, from https://developer.android.com/studio/command-line/adb.html