Authentic Learning Secure Software Development (SSD) in Computing Education

Kai Qian[1], Dan Lo[1],Reza Parizi[2]
[1] *Computer Science Department*
[2] *Software Engineering and Game Design*
Kennesaw State University
Kennesaw, GA, USA
kqian,dlo2,rparizi1@kennesaw.edu

Fan Wu
Computer Science Department
Tuskegee University
Tuskegee, AL, USA
fwu@tuskegee.edu

Emmanuel Agu
Computer Science Department
Worcester Polytechnic Institute
Worcester, MA, USA
emmanuel@wpi.edu

Bei-Tseng Chu
Department of Software and Information Systems
University of North Carolina at Charlotte
Charlotte, NC, USA
billchu@uncc.edu

***Abstract*—As mobile computing is now becoming more and more popular, the security threats to mobile applications are also growing explosively. Mobile app flaws and security defects could open doors for hackers to break into them and access sensitive information. Most vulnerabilities should be addressed in the early stage of mobile software development. However, many software development professionals lack awareness of the importance of security vulnerability and the necessary security knowledge and skills at the development stage. The combination of the prevalence of mobile devices and the rapid growth of mobile threats has resulted in a shortage of secure software development professionals. Many schools offer mobile app development courses in computing curriculum; however, secure software development is not yet well represented in most schools' computing curriculum. This paper addresses the needs of authentic and active pedagogical learning materials for SSD and challenges of building Secure Software Development (SSD) capacity through effective, engaging, and investigative approaches. In this paper, we present an innovative authentic and active SSD learning approach through a collection of transferrable learning modules with hands-on companion labs based on the Open Web Application Security Project (OWASP) recommendations. The preliminary feedback from students is positive. Students have gained hands-on real world SSD learning experiences with Android mobile platform and also greatly promoted self-efficacy and confidence in their mobile SSD learning.**

Keywords-Mobile; Security; Computer science; learning.

I. INTRODUCTION

We have seen numerous major cyber-attacks, resulting in stolen personal credit card numbers, leakages of classified information, banking and financial losses, and many other malicious damages. The insecure software leaves security holes for the attacks. The hackers have managed to make secure software computing a more difficult task. This has resulted in the need for not only the concepts of cybersecurity, but also for the SSD in the computing education [8].

More and more higher education institutions have realized the importance for computer science and software engineering students to be exposed to the mobile software design and development. Unfortunately, despite the great need for mobile professionals and existing efforts in mobile software development education, secure mobile software development is a relatively weak area and is not well represented in most schools' computing curriculum. The challenges in teaching the principles and practices of secure software development include: scarce learning materials and hands-on practice labs; dedicated staff and faculty in this field; the teaching formats and setting in a dedicated course or integration into multiple relevant courses.

Security flaws and defects could open doors for hackers to break into the app, access sensitive information, and conduct all kinds of malicious attacking. Most vulnerabilities should be addressed and fixed in the early stage of software development phase. If all the mobile apps are secure or have less security flaws and vulnerabilities, the security threat risks will be greatly reduced. All computer users, managers, and developers agree that software and systems need to be "more secure". Such efforts require support from both the education and training communities to improve software assurance, particularly in writing secure code. More and more people realized that the ability to write secure code should be as fundamental to a CS undergraduate as basic literacy from beginning, rather than adding it afterwards [9].

To achieve broader SSD education, we aim to develop Android-based portable and easy-to-adopt SSD learning modules on foundational SSD topics, which can not only be applied to mobile SSD, but also to all kinds of SSD.

The Authentic Learning approach is to create an authentic and active learning environment that encourages

The work is partially supported by the U.S. National Science Foundation under awards: 1723586, 1723578, 1723555, 1636995, 1623724, and U.S. Department of Homeland Security Scientific Leadership Award grant number: 2012-ST-062-000055.

978-1-5386-1174-6/18/$31.00 ©2018 IEEE

all students in learning concepts with practices for mobile SSD. This approach provides students with hands-on laboratory practice on real-world secure mobile app development. The Authentic Learning project consists of multiple learning modules, covering all major vulnerabilities for mobile software development. Each topic consists of a series of progressive sub-lab activities: a pre-lab, lab, and student add-on post-lab activities.

The developed modules can easily be integrated into many courses in CS and IT curricula as a learning module with lab practice or used in a dedicated mobile security course. The laboratories can be carried out anywhere and anytime by students in the classroom or out of the classroom.

The selected labs have been implemented in various computing courses, such as mobile app development, networking, database, and other security related courses such as "Mobile Security" to enhance the security learning in the discipline. The preliminary results from student performance evaluations and their feedback showed that this approach can increase their learning confidence for the state-of the art technology and promote their self-efficacy. Many students showed their creativity with their new findings and new solutions to protect mobile devices and mobile apps. Students appreciated the hands-on learning experiences. Also, all labs are affordable and can easily be adopted with the open source Android Studio IDE and Android mobile devices, such as smartphones and tablets.

The selected labs have been implemented in various computing courses, such as mobile app development, networking, database, and other security relevant courses such as "mobile security" to enhance security learning in the discipline. The preliminary results from student performance evaluations and their feedback showed that this approach can increase their learning confidence in the state-of the art technology and promote their self-efficacy. Many students showed their creativity with their new findings and new solutions to protect mobile devices and mobile apps. Students appreciated the hands-on learning experiences. Also, all labs are affordable and can easily be adopted using the open source Android Studio IDE and Android mobile devices, such as smartphones and tablets.

The rest of the paper is organized as follows: Section II describes our related work. Section III describes the pedagogical strategies for learning SSD. Section IV describes how we design the learning module. Section V presents sample SSD modules. Module implementations and students feedback are presented in Section VI, following by a conclusion of our work in section VII.

II. Related Work

Many efforts have been made to enhance the secure software development education in recent years. However, these efforts focus primarily on desktop and server environments and provide no hands-on experience with real world security vulnerability problems on mobile platforms. University of North Carolina at Charlotte (UNCC) has designed and developed an Application Security IDE (ASIDE) plug-in for Eclipse that alert programmers of potential vulnerabilities in their Java code, assist them in addressing these vulnerabilities, and improve their awareness and understanding of security vulnerabilities in software [1-3]. ASIDE only works in the Java Eclipse IDE and cannot support the Android IDE. Many computing instructors and professors have integrated mobile application developments into their curriculums. Some teaching and learning modules for secure coding practices were developed to enhance the essential skills of application developers in secure programming [4-6]. Some security topics/materials in the Information Assurance and Security (IAS)/Defensive Programming Knowledge Area (KA) mapped to the ACM/IEEE Computer Science Curricula 2013 are taught in CS0/CS1 courses [7].

Recently, NSF has supported a number of research projects for mobile security research for education [13-16] which mainly focus on the mobile threat analysis, rather than the effective learning approach for SSD.

Authentic active learning can engage and situate students in learning contexts where they encounter activities on real world problems, which students are likely to face in their real world career. The common authentic learning components are authentic contexts, authentic activities, multiple roles and perspectives, collaboration, opportunities for reflection, opportunities for articulation, coaching and scaffolding, authentic assessment. These components form the basis for teachers to plan and design learning environments [10-20].

In recent years, many institutions have offered courses in mobile application development [18]. These courses focused on mobile application APIs, but not on secure programming. They offered no insight into the security vulnerability of mobile software applications. In contrast, our Authentic Learning project is designed to broaden SSD education with a set of real world-based innovative SSD learning modules, which will promote students to analyze and penetrate the real vulnerability and find the solutions to fix the security flaws. All Authentic Learning modules are portable, modular, and integratable, and easy-to-adopt with a hands-on learning environment. Some Authentic Learning modules have been introduced in [25,26].

III. Authentic Pedagogical Strategies For Learning SSD

We developed an authentic learning environment for SSD with open source Android platform. The aim of this project is to investigate the potential student learning benefits (learning engagement and effectiveness) with authentic pedagogical activities in SSD education.

To implement the SSD authentic learning for mobile security, this work employs the following strategies

- authentic contexts reflect the knowledge in real-world mobile software vulnerability and hands-on experience.
- authentic activities are based on OWASP investigation and recommendation.
- multiple roles and perspectives are provided for the SSD knowledge of attacks and defense.
- opportunities for reflection involving concepts and practice in SSD.
- coaching students by the teacher in a dedicated web site.
- authentic assessment reflects the countermeasure solutions to software security vulnerability with hands-on experience.

The goal of our Authentic Learning project is to engaging students critical thinking in a real scenario in which they can understand better about the SSD concepts and enhance their problem-solving capability.

Each activity requires students to read a paragraph describing the study, predict the results, perform the appropriate calculations, and then evaluate the results in light of their predictions.

The Authentic Learning project focuses on student-centered learning, where students participate in group discussions on the concepts, risk analysis, and solutions. Each module designs each case from both of the attack and defense perspectives so that students can gain more insights and can design better defense solutions via the experience with actual attacks. In a group, some students play a role of attackers, and some other students play a role of defenders for a specific problem and exchange the roles later to find the best solution against the vulnerability. Authentic Learning project also emphasizes the learning environment connected the abstract security concepts to real-world mobile security cases so that students can better understand the concepts and can work more actively and effectively with facts and realistic problems. Each module infuses hands-on practices in such a way that they can be performed on mobile devices directly [21-23].

Authentic Learning project also encourages students to identify for themselves the mobile software security vulnerability issues and provides students with opportunity of reflection in action so that they can actively make connections to their existing knowledge, and to explore concepts in a new context with particular strategies for problem solving [24].

IV. LEARNING MODULE DESIGN

All the learning modules are Android Java platform based which can be easily integrated into any computing courses in computing curriculum or used for a dedicated SSD course.

The SSD learning modules are organized in the following categories as show in table I:

TABLE I. THE SSD LEARNING MODULES

M0	Getting started
M1	Data Sanitization for input validation
M2	Data Sanitization for output encoding
M3	SQL Injections
M4	Data protection
M5	Secure Inter-process communication (IPC) and Inter-Application Communication(IAC)
M6	Secure mobile database
M7	Unintended Data Leakage
M8	Access Control

Each module discusses specific mobile malware attacks and demonstrates instances of real-world Android malware, the defense strategies, and instructs on practicing defense solutions.

Real world examples of vulnerable code example is discussed to demonstrate the security weakness or unsecure coding patterns in the development of different Android app components, including Activity, Intent, Service, Content Provider, and Broadcast Receiver. The learning modules also discuss the best practices for improving the security of the Android app coding, such as reducing intent vulnerability for unintended intent receipt and intent spoofing.

The Getting Started module (M0) helps to setup the learning environment (Android Studio) and starts with a first Hello Android app for the beginners.

The Data Sanitization modules (M1 and M2) explore security flaws without input validation and output decoding and demonstrate validation strategies such as regular expression filtering for white and black list approaches.

The SQL Injection module (M3) focuses on Android SQL injection patterns and a prevention strategy with parameterized query.

The Data Protection module (M4) discusses how to utilize Android's built-in cryptography mechanisms to protect the data stored on the device (database storage, shared memory, shared preferences, internal and external storage) and data in transit in the course of network communications (e.g., SSL).

The Unintended Data leakage module (M5) demonstrates the leakage of private data from mobile devices and communications (e.g., location information, clipboard cache, logging and usage patterns). The Access Control module focuses on the Android permission model, including its signature-level permission for intra and inter application communication.

Working on the Android software with practical experiences for the secure mobile programming will increase students' learning interest because students will be able to see their applications running in real time. This provides students with a sense of accomplishment and realworld relevance, which will promote their programming confidence and self-efficacy.

Portable and modular designed SSD learning labs are easy to be adopted and integrated into lower division computing courses, such as lower division Java computer programming courses, and upper division computing courses, such as mobile app development, database, networking, information security, and others.

Programmers need not only to learn the specific ability of writing secure code, but also the knowledge of the principle behind the skills such as Input Validation and Output Encoding. We developed one learning module for each secure programming principle based on the OWASP top mobile vulnerabilities [17]. Each module consists of a series of progressive sub-labs: a pre-lab, hands-on lab activities, and a student add-on post-lab. Many secure mobile software development principles can be applied to all kinds of software development and an overview of each of them is presented in the corresponding pre-lab. Its related mobile specific secure programming skills, such as SMS, GPS data, mobile data sync, and possible flaws in the mobile programming for intent and content provider are covered in the hands-on lab activities. The learning modules are designed in a such way that they can not only be used for SSD, but also be used for general secure software development education. Selected student work is posted in the post-lab to show student's creative achievement and learning outcomes through pre-lab and post-lab study and practice.

The pre-lab presents learning objectives, concept overview on the subject, fundamental background knowledge on the subject, ethics issues, and targeting courses.

In the hands-on lab activity, students will not only learn how to avoid and fix the known flaws and mitigate such security risks (practical aspect) through hands-on practice, but also identify the root cause (theoretical aspect) of their formation. The students will get a good understanding of the mobile security risks and vulnerabilities and get the secure programming skills to avoid such vulnerabilities as well. The lab is designed to engage and motivate students in building mobile apps on their own mobile devices, and it will boost their positive emotions through mastery-building experiences.

The post-add-on lab activity provides opportunity for students to show their new findings and new creativity works based on previous existing work, and students will work on their own to analyze and assess new mobile vulnerability and risks in the subject and develop and integrate defensive tools for risk prevention and mitigation. These post-labs can help students to build self-efficacy by observing a successful task from their peers and strengthen their confidence and promote their critical thinking and creativity and assess their learning as well.

Its mobile specific secure programming skills for SMS, GPS data, mobile data sync, and possible intent and content provider flaws in the mobile programming are covered in the hands-on lab activities. The learning modules are designed in a way that they can not only be used for mobile SSD, but also be used for general secure software development education.

Our experience shows that teaching "Secure Programming" helps students not only gain significant knowledge of secure programming principles, but also to understand the risks of insecure programming and to master the skills of risk mitigation. There is no better way to learn secure design and programming concepts than actually seeing the flaws and its security consequences through real code. We are developing eight hands-on mobile SSD learning modules with the secure programming concepts and practice guidance to identify the security flaws.

Each module is organized in the following sections:

- Overview
- Learning Objectives
- Ethics
- Targeting Courses
- Activities
    - Pre-Lab Activities
    - Lab Activities
    - Post-Lab Activities
        - Review questions and answers
        - Assignments
        - Projects

Each module consists of Pre-Lab Activities (Overview, Learning Objectives, Ethics issues, Targeting Courses), Hands-on Lab Activities, and Post-Lab Activities (Student add-on labs, Review questions and answers, Assignments, and Projects. In the hands-on lab activities, students will not only learn how to avoid and fix known flaws and mitigate such security risk (practical aspect), but also identify the root cause (theoretical aspect) of their formation. The student learning outcomes expect that they will get a good understanding of the risks and vulnerabilities and form basic ideas of secure programming to reduce such vulnerabilities.

V. SAMPLE SSD LEARNING MODULE LABS

In this section, we illustrate the SSD learning modules and labs with brief descriptions on selected subjects to show how they are organized and implemented.

A. Intent Spoofing SSD Learning Module

We illustrate the SSD learning modules with the brief descriptions on intent spoofing vulnerability. Intent is a

very important component for support intra-app and inter-app communication in Android application development, but it may result in security disasters when it is misused. The intent spoofing learning module is a sub module in the inter-app vulnerability leaning module (M5) where Unintended intent receipt (interception) is another sub module.

1） Lab learning objective:

By completing this lab, students will be able to

- To understand the fundamental concepts of intent spoofing vulnerability
- To grasp the basic knowledge and skills of secure software development against intent spoofing.

2） Pre-lab: Overview

Here is the brief overview for the intent spoofing sub module.

The intent is the main communication mechanism between Android components such as activity, service, and broadcast receiver. Intent can be used to start activities and services, bind to services, and convey notify and pass data to broadcast receivers.

There two types of intent: Explicit intents have its explicit recipient and Implicit intents does not name its explicit recipient, and it will notify an appropriate component based on the specification of the intent.

Intent spoofing is an attack where a malicious application induces or injects undesired behavior to a component via implicit intent which only expects to receive intents from other components within the same app. By default, a component only receives intents from other components in the same application, but it can also accept intents from other apps if the android:exported attribute is set in the manifest XML.

In other words, if your application uses an exported component, a malicious application can send an intent to it, which is an intent spoofing attack.

Misused and overused implicit intent will result in intent spoofing, which may cause DoS and phishing attacks.

Exporting attribute allows all applications to send intents to that component, which even opens a door for explicit intent.

An Android developer should not let any component expose to other applications which makes program itself vulnerable to attacks. Typically, a broadcast receiver is vulnerable to broadcast injection, in which the receiving component thought the malicious broadcast coming from another app is what it expects within the same app. Activities and services may also be vulnerable to fake spoofing activation or binding attacks.

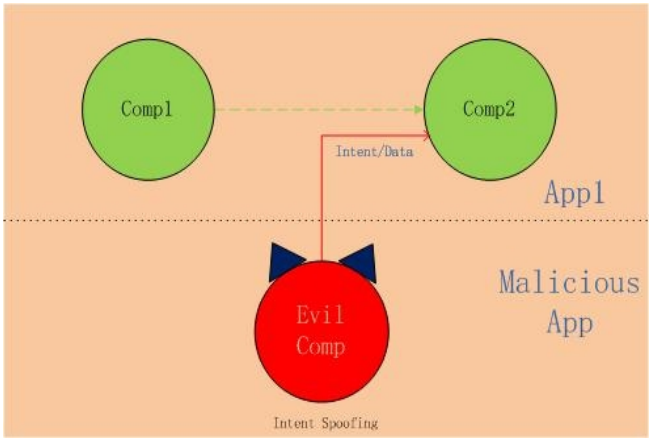

Fig.1. Intent Spoofing

The Fig. 1 above shows the concept of intent spoofing where comp1 and comp2 are two Android components (Activity, Service, BroadCastReceiver) in same application app1. The comp2 expects to get intent from Comp1 in App1 but instead it gets a malicious injection via an implicit intent sent by the malicious app due to comp2's exposure.

3） Lab activities:

Students practice the hands-on labs via exploring the intent spoofing flaws in IPC communication with broadcast receiver and think of solution to secure the program to prevent it in the future.

The victim's manifest may be defined as follows:

TABLE II. THE VICTIM'S MANIFEST FILE

```
<receiver android:name=".MyBroadCastReceiver" android:enabled="true" android:exported="true">
<intent-filter>
<action android:name="com.example.MyBroadcast"/>
</intent-filter>
</receiver>
```

An intent spoofing attacker may attack the victim with spoofing injection as follows:

TABLE III. AN INTENT SPOOFING WITH INJECTION

```
public void onClick(View v) {
    Intent intent = new Intent();
    intent.putExtra("number", 1);
intent.addFlags(Intent.FLAG_INCLUDE_STOPPED_PACKAGES);
intent.setComponent(new ComponentName("example.com.broadcastreceiver","example.com.broadcastreceiver.MyBroadCastReceiver"));
    intent.setAction("com.example.MyBroadcast");
    sendBroadcast(intent);
  }
```

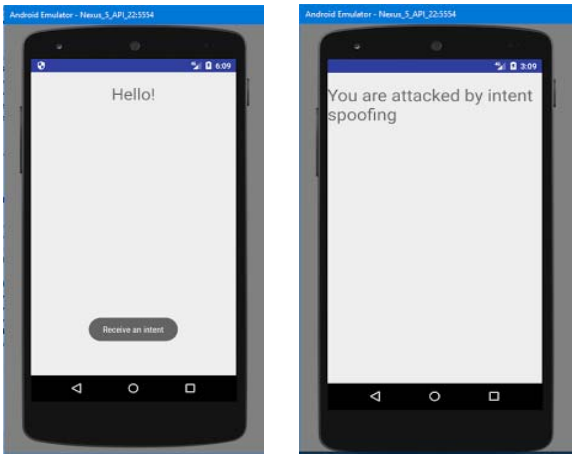

Fig. 2. BroadCastReceiver Intent Spoofing

The Fig. 2 shows that a BroadcastReceiver app has a filter that can receive intent either from external apps(in this case IntentSender is an external app) or internal components. When BroadcastReceiver receives an intent from an internal component, the normal action will show "Hello", otherwise, it shows "You are attacked by intent spoofing" spoofing injection message through intent spoofing.

B. Secure Inter-Process Communication (IPC)

1） Learning objectives

- Learn Android core programming component Activity and intent
- Learn the intent vulnerability and its countermeasures

2） Pre-lab: Overview

An Intent object is a messaging object is used to signal/activate other components, such as activities, services, and broadcast receivers. E.g., an activity B can register with intent A's explicit intent so that the bound activity B will be notified and activated when the event fires. An explicit intent tells Android system to run a specific component such as Activity B.

TABLE VI. EXAMPLE OF AN INTENT OBJECT

```
Intent intent = new Intent(ActivityA.this, ActivityB.class);
startActivity(intent);
```

An implicit intent tells Android system to select an activity (ActivityB), in this case, that can do these things that are specified with desired action and category in the intent filter in a manifest.

TABLE V. EXAMPLE OF AN IMPLICIT INTENT OBJECT

```
Intent intent = new Intent();
intent.addAction("thisAction");
intent.addCategory("thisCategory");
startActivity(intent);
```

TABLE VI. EXAMPLE OF AN ANDROID MANIFEST FILE

```
<activity android:name="ActivityB">
  <intent-filter>
    <action android:name="thisAction"/>
    <category android:name="thisCategory"/>
  </intent-filter>
</activity>
```

3） Intent Security

Unsecured code may allow malicious apps to intercept intents to steal sensitive information or attack. A malicious Activity such as phishing is launched instead of the right one.

An implicit intent is sent to the Android OS system, which will select a best matched component, such as Activity B, based on action and category requirement. Any activity can have multiple intent filters to tell the system that what kind of implicit intents they would accept described by action name, category, etc. An intent can also contain data. This will leave a hole for a malicious activity or app to compete with benign activities to hack data. Because implicit intents do not name a specific target component, instead just declare a general action to perform, which allows a component from another app to handle it.

If the hacker application has one activity with the same action name and other characters, it might get the information. Because if the intent matches an intent filter, the system starts that component and delivers the Intent object. If multiple intent filters are compatible, the system displays a dialog so the user can pick which app to use. If a hacker app becomes default that would result in security threats as shown in Fig.3. You can add as many categories as possible to the intent to prevent the intent from having unintended consequences. In addition, a malware could register a higher priority intent filter and beat others to get sensitive data sent instead.

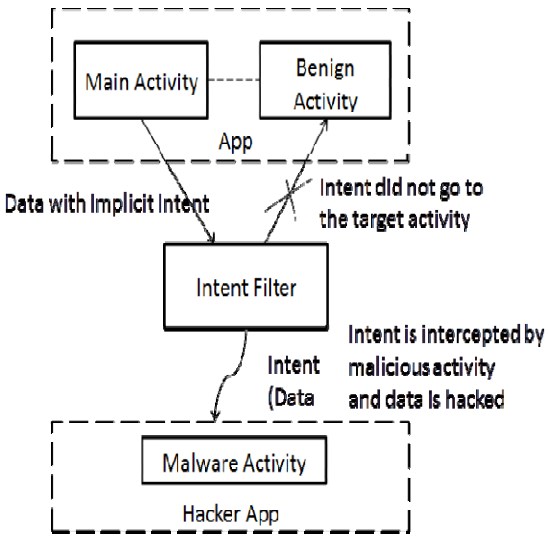

Fig. 3. Vulnerability with implicit intent

Explicit intent is always recommended. If you do need to pass sensitive data with intent, you need to validate the right recipient. Also, secure data by setting permissions for IPC between apps and apply necessary encryption. If your app does not need to talk to other apps, set the android:exported attribute to false in the component's manifest element, such as for the <service> element.

4） Lab Activities

This lab demonstrates how a hacker gets the data sent by an activity with implicit intent. E.g., the username/password is typed, and an intent object is somehow intercepted by a malware activity shown in Fig. 4.

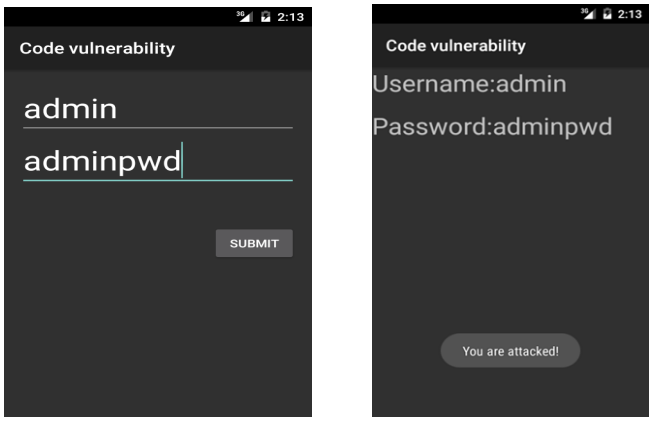

Fig. 4. A malware activity intercepted intent and extracted sensitive information

TABLE VII. SOURCE CODE OF SETACTION()

```
Intent intent = new Intent();
intent.putExtra("username", name);
intent.putExtra("password", password);
intent.setAction("edu.spsu.SendIntent.LoginActivity");
intent.addCategory(Intent.CATEGORY_DEFAULT);
startActivity(intent);
```

Instead of using setAction() to reach implicit intent, we will use setClassName() for explicit intent:

TABLE VIII. SOURCE CODE OF SETCLASSNAME()

```
Intent intent = new Intent();
intent.putExtra("username", name);
intent.putExtra("password", password);
intent.setClassName("edu.spsu.SendIntent",
                    "edu.spsu.SendIntent.LoginActivity");
startActivity(intent);
```

Since the explicit intent will go straightly to the activity we want and ignore the intent filters, the hack app will never intercept it.

VI. Implementation And Student Feedback

We have implemented selected labs in various computing courses such as Database, Mobile App Development, Wireless Networking, and other security relevant courses, such as special topics on “Mobile App & Security,” to enhance the security learning in the discipline. The preliminary results from student performance evaluations and their feedback showed that this approach increased their learning confidence in the state-of-the-art technology and promoted their self-efficacy. Many students showed their creativity with their new findings and new solutions to protect mobile devices and mobile apps. Students appreciated the hands-on learning experiences. Also, all labs can easily be adopted with the open source Android Studio IDE, Android smartphones, and tablets. In Fall 2017, we conducted a quantitative survey in four classes at Kennesaw State University (KSU) and Tuskegee University (TU), and in Spring 2018, we conducted a quantitative survey in four different classes at KSU, TU, and Worcester Polytechnic Institute (WPI). About 130 students participated in the survey, and the overall response from students was overwhelmingly positive.

The survey has the following questions as show in table IX:

TABLE IX. SURVEY QUESTIONS

Q1	I like being able to work on Android App development with this hands-on labware.
Q2	The real world mobile security threat and attacks in the labs help me understand SSD.
Q3	The hands-on labs help me gain authentic learning and working experience on SSD.
Q4	The online lab tutorials help me work on student add-on labs/assignments.
Q5	The project helps me apply learned SSD to develop secure mobile application.

In the “Introduction to Database” class in Fall 2017, 15 students participated in the survey and the survey result is shown in Fig. 5 in the scale of: [Agree], [Neutral], [Disagree]):

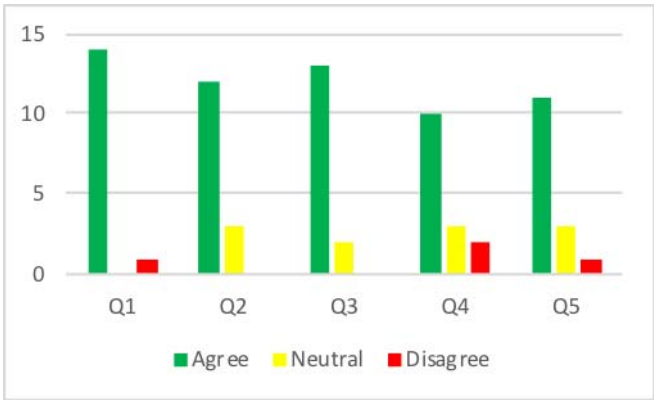

Fig. 5. Survey result of Introduction to Database (Fall 2017)

Some qualitative student responses to the survey:

The hands-on portion of this course made it easier to understand the material and put it into practice. If not for the hands-on labs and assignments where I was able to create mobile apps myself, I do not think I would have done very well in the class.

Working with Android Studio and using online tutorials really assisted in helping me understand the majority of the material, without feeling lost or confused throughout the semester.

The lab is very clear. Through the lab, I learn about the android. It gives me what I need for the future.
I enjoyed this experience as a whole and liked the development process
I enjoyed working with Android Studio as I haven't worked before
I liked what it taught me.

VII. CONCLUSION

This paper presented a hands-on SSD learning approach to immerse students with secure software development learning experience. Some of the selected learning modules are being implemented in the CS Database, CS Mobile App Development, CS Mobile Security, and IT Wireless Security classes. The preliminary feedback from students was very positive. Students have gained hands-on real world experiences on SSD and mobile security with Android mobile devices, which also greatly promoted students' self-efficacy and confidence in their mobile security learning.

ACKNOWLEDGMENT

The work is partially supported by the U.S. National Science Foundation under awards: NSF proposal 1723586, 1723578, 1723555, 1636995, 1623724， and U.S. Department of Homeland Security Scientific Leadership Award grant number: 2012-ST-062-000055. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation and Department of Homeland Security.

REFERENCE

[1] Heather Richter Lipford, Bei-Tseng Chu, Supporting Secure Programming Education in the IDE, http://grantome.com/grant/NSF/DUE-1044745

[2] Jing Xie, Heather Richter Lipford, Bei-tseng Chu, Evaluating interactive support for secure programming. Proceeding CHI '12 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 2012: 2707-2716

[3] Xiaohong Yuan, Kenneth Williams, Huiming Yu, Bei-tseng Chu, Audrey Rorrer, Li Yang, Kathy Winters, Joseph M. Kizza, Developing Faculty Expertise in Information Assurance through Case Studies and Hands-On Experiences. HICSS 2014: 4938-4945

[4] Xiaohong Yuan, Li Yang, Bilan Jones, Huiming Yu, Bei-Tseng, Secure Software Engineering Education: Knowledge Area, Curriculum and Resources, Journal of Cybersecurity Education, Research and Practice, Volume 2016 Number 1

[5] Hongmei Chi, Edward L. Jones, John Brown, Teaching Secure Coding Practices to STEM Students, Proceedings of the 2013 on InfoSecCD '13: Information Security Curriculum Development Conference

[6] Edward Agama, Hongmei Chi, A framework for teaching secure coding practices to STEM students with mobile devices, ACM SE 2014

[7] Kenneth A. Williams, Xiaohong Yuan, Huiming Yu, Kelvin Bryant, Teaching secure coding for beginning programmers, Journal of Computing Sciences in Colleges archive, Volume 29 Issue 5, May 2014 Pages 91-99

[8] Kalyan Mondal, Introducing Secure Coding Concepts in Engineering Programming, Middle Atlantic Section Proceedings ASEE, Spring 2013

[9] Matt Bishop, Deborah Frincke, Teaching Secure Programming, IEEE Security & Privacy. Volume: 3, Issue: 5, 54 – 56, 2005

[10] Diana L. Burley, Matt Bishop, Final Report, Summit on Education in Secure Software, Support for this work was provided through the National Science Foundation Directorates of Computer and Information Science, and Engineering and of Education and Human Resources under Award #1039564, 2011

[11] Kara Nance, Brian Hay, Matt Bishop, Coding Education: Are We Making Progress? Proceedings of the 16th Colloquium for Information Systems Security Education, 2012

[12] Matt Bishop, Elizabeth Hawthorne, Kara Nance, Blair Taylor, Teaching secure coding: the myths and the realities, Proceeding of the 44th ACM technical symposium on Computer science education, 2013

[13] REU Site: Research on Security of Mobile Devices and Wireless Networks, https://www.nsf.gov/awardsearch/showAward?AWD_ID=1559652

[14] EDU: Collaborative: Enhancing Pervasive and Mobile Computing Security Education with Research Integration, https://nsf.gov/awardsearch/showAward?AWD_ID=1419280&HistoricalAwards=false

[15] Collaborative Project: Capacity Building in Mobile Security Through Curriculum and Faculty Development, https://www.nsf.gov/awardsearch/showAward?AWD_ID=1241651&HistoricalAwards=false

[16] EDU: Deploying and Evaluating Secure Programming Education in the IDE, https://www.nsf.gov/awardsearch/showAward?AWD_ID=1523041&HistoricalAwards=false

[17] OWASP Mobile Security Project: https://www.owasp.org/index.php/OWASP_Mobile_Security_Project

[18] Jeremy Andrus, Jason Nieh, Teaching Operating Systems Using Android, Proceedings of the 43rd ACM Technical Symposium on Computer Science Education (SIGCSE 2012), 2012 (Best Paper Award)

[19] Jan Herrington, Jessica Mantei, Anthony Herrington, Ian Olney and Brian Ferry, New technologies, new pedagogies: Mobile technologies and new ways of teaching and learning, Proceedings ascilite Melbourne, 2008. http://www.ascilite.org.au/conferences/melbourne08/procs/herrington-a.pdf

[20] Brown, J.S., Collins, A., & Duguid, P. (1989). Situated cognition and the culture of learning. Educational Researcher, 18(1), 32-42.

[21] Anderson LW, Krathwohl DR, editors. A taxonomy for learning, teaching and assessing: a revision of Bloom's Taxonomy of educational objectives: complete edition. Longman; New York, NY: 2001.

[22] Bloom BS, editor. Taxonomy of educational objectives, handbook 1: the cognitive domain. Longman; New York, NY: 1956.

[23] Carlson A. Innovative Teaching Showcase. Center for Instructional Innovation & Assessment, Western Washington University; Bellingham, WA: 2001. Authentic learning: what does it really mean? pp. 2001–02, pandora.cii.wwu.edu/showcase2001/authentic_learning.htm, Accessed 11/2017

[24] Jordon K. March, Kyle C. Jensen, Nathan T. Porter, and Donald P.

Breakwell, Authentic Active Learning Activities Demonstrating the Use of Serial Dilutions and Plate Counts, J Microbiol Biol Educ. 2011; 12(2): 152–156.

[25] Kai Qian, Hossain Shahriar, Fan Wu, Lixin Tao, Prabir Bhattacharya. Labware for Secure Mobile Software Development (SMSD) Education, Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education - ITiCSE '17, 2017.

[26] Kai Qian, Dan Lo, Hossain Shahriar, Lei Li, Fan Wu, Prabir Bhattacharya. Learning Database Security with Hands-on Mobile Labs, Proceedings of the 2017 IEEE Frontiers in Education Conference (FIE), 2017.