Teaching Android Security Through Examples: A Publicly Available Database of Vulnerable Apps

By Daniel E. Krutz and Samuel A. Malachowsky,
RIT - Software Engineering

Security is hard, and teaching security can be even harder. Here we describe a public educational activity to assist in the instruction of both students and developers in creating secure Android apps. Our set of activities includes example vulnerable applications, information about each vulnerability, steps on how to repair the vulnerabilities, and information about how to confirm that the vulnerability has been properly repaired. Our primary goal is to make these activities available to other instructors for use in their classrooms ranging from the K-12 to university settings. A secondary goal of this project is to foster interest in security and computing. All project activities may be found on the project website [2].

The mobile revolution has allowed anyone with a basic understanding of development to upload their applications ("apps") to an app store, making them available to millions of potential users. With extreme openness comes great danger—inexperienced developers have the capability to create vulnerable apps that can negatively affect millions of users. Additionally, experienced developers can and do make mistakes due to the challenging nature of creating secure software.

Developers inadvertently create vulnerable software for a wide range of reasons— simple errors, ignorance of how to create secure apps, or a lack of understanding of the importance of secure app development. In order to help educate developers about the importance of secure app development as well as how to create secure apps, we have created a public sample set of vulnerable Android apps. Each example contains a clear demonstration of the negative ramifications of the vulnerability, steps to repair the vulnerability, and posted actions to ensure that it has been resolved.

We have two primary goals for the project. First, we believe that it's important to educate Android developers about the specific example vulnerabilities in our study. Second, we aim to demonstrate the importance of security on a more general level for the extremely diverse set of Android developers. We would like developers of all experience levels to become more interested in security through hands-on examples.

There were two primary concerns which lead us to create this repository:

- **Development Effort.** Creating activities, especially those with vulnerabilities can be a long and difficult process—something which many instructors do not have the necessary resources to build. The availability of these activities frees instructors to do what they do best, teach. Additionally, since these activities will be refined through

regular usage, instructors can be more confident that any issues have already been resolved.

- **Required Skillset.** Teaching students and developers to create secure software is a difficult process. Not all instructors teaching mobile development are expected to be adequately prepared to create a diverse, informative set of activities such as ours.

Our project has several goals.

- **Ease of use.** All activities should be usable 'out of the box' by instructors. Additionally, students and developers working through the activities on their own should be able to do so with as few roadblocks as possible due to the documentation and clear instruction sets provided with each of the activities.
- **Relevance.** We have found that creating relevant, real-world projects helps to foster student interest in the activity and in security. Each of our activities contains relevant examples from the commercial apps, including steps to recreate the vulnerability and demonstrate its relevance.
- **Meet a diverse range of skillsets.** Not all students or developers are at the same skill level, and security is important to all experience and skill levels. Additionally, anyone can learn about security and mobile development from elementary-age to retirees. Our activities are designed to assist students and developers of all experience levels and ages.

We have created ten publicly available activities that can be found on our project website [2]. This list will continue to grow, both through our own efforts as well as submissions from the external sources. Our apps were created by an experienced Android developer with over 1 million app downloads in the Google Play store, and more in other app marketplaces.

ACTIVITIES

Although our list of activities is growing, we currently have ten vulnerability examples ranging from basic examples as proper intent protection to more complicated activities such as correct use of content providers. The process outline of each activity is outlined in Figure 1.

Each of the exercises contains:

- mobile apps which contain well-defined vulnerabilities;
- documentation about the adverse effects of the vulnerabilities and how they may be exploited;
- step-by-step documentation on how to repair the vulnerabilities, along with their rationale; and
- instructions for how to verify that the vulnerability has been repaired

Activities begin with providing the user some background about the specific vulnerability being targeted—when, why, and how the vulnerability may occur. Whenever possible, students are also provided with a real-world example of occurrences of the vulnerability such as where they occurred in specific apps. Also included are some basic reasons about why the vulnerability occurs and common developer mistakes which lead to the vulnerability.

Each activity has an associated app which contains an instance of the discussed vulnerability. These apps—created specifically for these exercises—are typically very simple, having the sole intention of conveying the example vulnerability. Using the provided instruction set, students are able to recreate the vulnerability, demonstrating its possible negative ramifications. In some cases, activities also utilize free, third party tools such as Fiddler [1] to provide visibility of the vulnerabilities or to later demonstrate that the vulnerability has been repaired. Figure 2 demonstrates an example of a third-party app inappropriately accessing message data from another app, and Figure 3 is a developer-side demonstration of another vulnerability—both would be part of an included vulnerability demonstration workflow.

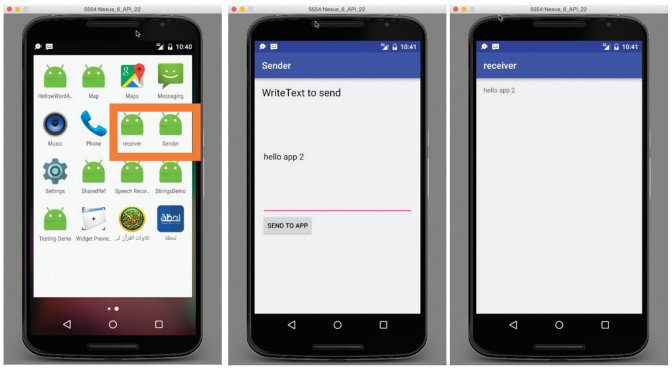

Figure 2: Example of Vulnerability Demonstration

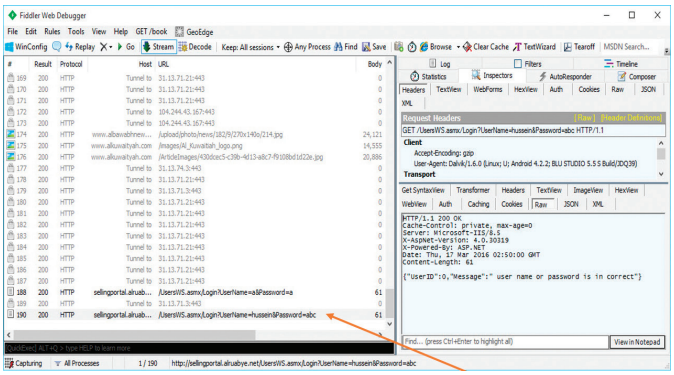

Figure 3: Example of Vulnerability Demonstration Using Fiddler

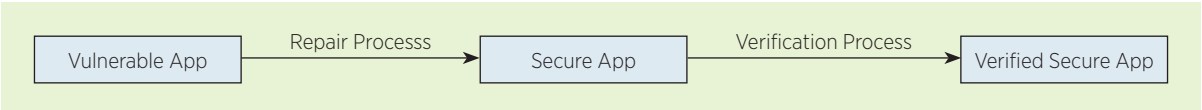

Figure 1: App Repair Process

The user is then provided with information on how to repair the vulnerability, which includes any relevant code snippets and proper information about how the introduced new (defensive) coding practices protect against the vulnerability. Our documentation also provides clear steps on how to repair the vulnerability within the provided app.

The final step has the user attempt to re-test whether the vulnerability still remains within the app (i.e., the vulnerability should no longer exist). The concluding portion of the activity provides several important benefits to the user including demonstrating the importance and proper procedures of basic security testing, providing a sense of accomplishment for the user.

Here are some of the activities, along with their anticipated experience/difficulty level.

- **Activities Access** (Beginner). Security issues arise when people try to access specific unauthorized activities. An example could be a bank app where users try to access a balance management activity without properly logging into the system.
- **Intent Protection** (Beginner). Android uses "intents" to pass data between apps, for examples between the Facebook and Facebook Messenger apps. Data passing between these apps may be easily (and improperly) read by other apps. This module explains how to protect information being sent via intents between apps.
- **XML** (Beginner). Since XML is very easy to read using reverse engineering, it is best to avoid saving important information such as Ads code or Map Code within XML files.
- **Android Javascript** (Medium). This activity demonstrates the negative implications of using JavaScript in Webview to pass data from an Android app to a server. This is considered bad practice because anyone could use malicious JavaScript code on their website to gain private user information associated with the app.
- **Broadcast** (Medium). Broadcast data sent by the app is easy to access from any other app in the system, so when Broadcasting to specific apps, the data should be encrypted. Intercepted unencrypted Broadcasts could lead to serious security and privacy issues.
- **Data Storage** (Medium). When an app does not secure storage data such as data files, shared references, and databases (i.e., SQLite), it has the potential to be read by any other app. This means that important information stored in these files (such as a database connection information) should be encrypted.
- **Data Over HTTP** (Medium). Data that moves over an unencrypted HTTP (internet) connection is vulnerable to "Man in the Middle" attacks. One example of this is credit card information, which if passed over an unsecure connection, could be intercepted midstream.
- **DOS** (Medium). Denial of Service (DoS) attacks are a common problem with Android, because a malicious party could create an overwhelming number of (HTTP) requests directed towards a specific server. The environments must be managed to make them less vulnerable to these types of attacks.
- **Ad Libraries** (Advanced). In-app advertisements ("ads") libraries are able to use all of the permissions given to the app which contains the ad library, even the people who did not give this permission to the ad library itself. This can open up various security and privacy issues within the app including ad libraries collecting sensitive user information such as locations or contact info.
- **Content Providers** (Advanced). Content providers share data between apps, and any app in the system can access the "content providers" database. Because of this, data stored here must be kept secure and encrypted so that it can only be read by an authorized app.

Although there are many more families of Android vulnerabilities, we have initially selected these for several reasons including their prevalence in existing apps, their ease of demonstration, and their potential for negative ramifications.

CONCLUSION

We have created a publicly available instruction set of vulnerable Android apps which includes ten groups of vulnerabilities. Our goal is for instructors to adopt these activities in a diverse set of courses, allowing users of varying experience levels to examine, demonstrate, and repair common Android vulnerabilities. All course materials can be found on our project website [2]. ❖

Acknowledgements
Elements of this work are sponsored by a SIGCSE Special Projects Grant.

References
1. Fiddler; http://www.telerik.com/fiddler. Accessed 2016 July 17.
2. Teaching Mobile Security; http://www.teachingmobilesecurity.com. Accessed 2016 July 17.

Other Android-related research projects by the authors:
1. Darwin Project: A tool which downloads, analyzes and reports on several key analytics about Android apps including: Reported JLint errors, adherence to coding standards, utilized permissions, over permissions and under permissions. To date, the project has analyzed over 70,000 Android apps.
2. Androsec Project: Collects and analyzes Android version control repositories from F-Droid. In addition to various security and quality results gathered from static analysis tools, we've also collected version control information such as commit messages from the GIT repositories.
3. M-Perm: A tool for detecting the permission gap in Android 6.0 and above apps.

Daniel E. Krutz
RIT - Software Engineering
1 Lomb Memorial Drive, Rochester, NY, USA
dxkvse@rit.edu

Samuel A. Malachowsky
RIT - Software Engineering
1 Lomb Memorial Drive, Rochester, NY, USA
samvse@rit.edu

DOI: 10.1145/2994152