

A. C. Patil College of Engineering, Kharghar, Navi Mumbai University of Mumbai 2022-2023

Department of Computer Science and Engineering (Internet of Things and Cyber Security Including Block Chain Technology)

Date of Performance: -01-2022 Date of Submission:

**Experiment No. 15**

**Title:-** Android Security Case Studies

**Theory:-**

* **INTRODUCTION**

Over the last years, static code analysis for security has made great progress. Commercial available tools are employed by software vendors to detect implementation-level security bugs, such as buffer overflows and injections vulnerabilities. Certainly, employing these tools is only the first step towards secure software as it is restricted to common bug classes.

More seriously are design-level flaws, since literature states that the later a change must be applied to the design of an application, the more costs will arise Methods such as Microsoft’s threat modeling or the architectural risk analysis proposed by McGraw should help to discover security problems already during the software-design phase. In academia, more formal approaches to dealing with software security have been established, notably, language-based security , model-driven security, and stepwise refinement . Although these approaches are promising, they can just be applied when software is developed from scratch, which is rarely the case

* **MOTIVATION FOR THE CASE STUDY**

It is expected that Android will become one of the major mobile phone platforms in the future [17] and is used for other devices as well. As it gains a lot of market share and is open source, it is an interesting target for security analyses. For this reason, we started a security assessment project. During the analysis, we faced several challenges, mostly related to the lacking documentation of Android’s security concepts and the complexity of the code.

We started our review of the Android platform with the assumption that not only the Linux kernel is security-critical, but also the Android middleware (the Android framework classes). For example, the permission enforcement and the reference monitor, which mediates the access to Android components, is implemented within the Java-based middleware, although the kernel is accessed to retrieve data for security decisions. We aimed to explore and understand the implementation of Android’s security mechanisms.

* **THE ANDROID CONCEPTS**

We first describe Android’s main concepts, before presenting the challenges related to analyzing the platform with respect to security. Note that there does not exist a comprehensive document on Android’s security concepts. The information is scattered throughout the Android developer’s website.

Android Components: An Android application consists of different parts, called components, having, according to its task, one of four basic component types. Activities are the presentation layer of an application, allowing a user to interact with the application. Services represent background processes without a user interface. Content providers are data stores that allow developers to share databases across application boundaries. Finally, broadcast receivers are components that receive and react to broadcast messages, for example, the Android OS itself sends such a broadcast message if the battery is low. Each component of an application runs as a separate task, making an Android device to a large distributed system, even if all processes are running on the same device.

* **SOFTWARE-SECURITY COMPREHENSION**

In the preceding section, we showed that programcomprehension and reverse-engineering techniques can be used in the area of software security. Now, we discuss research topics that need to be investigated more deeply, to develop useful techniques and tools for a security evaluator. For our more general discussion, we also consider experience gained in a research project called ASKS, which is currently being carried out with enterprises that made available their business applications, which are implemented using the Java platform, Enterprise Edition technology , for a security analysis. One conclusion that we drew from our security review is that it is necessary to create more formal architectural security views (see also Mancoridis’ statement .

These views need to be language- and platform-independent in order to be a common language to communicate with security experts who are not necessarily experts for the programming language. With the help of these views, it is easier to understand the security architecture of an application or even of a distributed system. In the following, we discuss some further ideas of how these views can be created and what security aspects may be of interest for such views.



* **ANDROID STUDY**

We conducted our security assessment along with a security expert, the third author, to understand the implementation of some aspects of the Android security mechanism with the help of the Bauhaus tool-suite.

Issues: Starting from the documentation, we planned to focus on several aspects related to the implementation of permission checking. The process of permission enforcement consists of different steps, which we aimed to understand. After reading the security documentation, available at the Android website, we picked the Bluetooth API for further investigation. Other functionality of the platform could have been analyzed in a similar way. The Bluetooth API documentation states that an application needs at least the BLUETOOTH permission to use the Bluetooth device. If a program wants to administer the Bluetooth device, it needs the BLUETOOTH\_ADMIN permission in addition. It is explicitly noted that one needs the former permission, to use the latter .



During the security assessment we wanted to answer several questions the security expert had. We will discuss these in the following: Question 1 Where are permissions enforced within the Bluetooth API (enforcement points)? Question 2 Which permissions are enforced within the Bluetooth API (access control policy)? Question 3 Can we check whether an application needs the BLUETOOTH permission, in order to use the BLUETOOTH\_ADMIN permission?

**REFERENCES**

[1] R. Pressman, Software Engineering – A Practioner’s Approach, 4th ed. McGraw-Hill, 1997.

[2] F. Swiderski and W. Snyder, Threat Modeling. Microsoft Press, 2004.

[3] G. McGraw, Software Security: Building Security In. Addison-Wesley, 2006.

[4] A. Sabelfeld and A. C. Myers, “Language-based information-flow security,” IEEE J. Sel. Areas Commun., vol. 21, no. 1, jan. 2003.

[5] M. Clavel, V. Silva, C. Braga, and M. Egea, “Model-driven security in practice: An industrial experience,” in Proc. of the 4th European Conf. on Model Driven Architecture: Foundations and Applications. Berlin, Heidelberg: Springer, 2008.

[6] H. Mantel, “Preserving information flow properties under refinement,” in IEEE Symposium on Security and Privacy, 2001

**Conclusion:-**

Thus we have studied about android and there various security and the deep study and history of the android security .