УДК 004.056.57

Горбунова Т. В.

ВДОСКОНАЛЕННЯ АЛГОРИТМУ АНАЛІЗУ ВРАЗЛИВОСТЕЙ ANDROID ДОДАТКІВ НА ОСНОВІ РЕВЕРСНОЇ ІНЖЕНЕРІЇ

Вінницький національний технічний університет

Анотація

Метою даної кваліфікаційної роботи є аналіз сучасних методів та інструментів проведення тестування безпеки Android-застосунків, аналіз ефективності цих методів та інструментів. При виявлені значних недоліків створення рішення для їх усунення, або створення рішення для покращення існуючих методик. Наукова новизна одержаних результатів полягає у використанні найбільш вдалих рішень та засобів для аналізу архітектури застосунка на предмет вразливих ділянок, та розгляд графу викликів не як кінцевого продукту аналізу вихідного коду, а розробка методів автоматичного аналізу цього графу.

Ключові слова: інформаційна безпека, інформаційні технології, безпека Android-застосунків, реверс-інжиниринг, захист інформації.

Abstract

The purpose of this qualification work is to analyze current methods and tools for testing Android-applications security, analysis of the effectiveness of these methods and tools. If significant drawbacks are found, creating a solution to fix them, or creating a solution to improve the existing techniques. The scientific novelty of the results is to use the most successful solutions and tools to analyze the architecture of the application for weak areas, and to estimate the call graph, not as the final product of the source code analysis, but to develop methods for automatic analysis of this graph.

Keywords: information security, information technology, android application security, reverse engineering, information security.

Вступ

На сучасному етапі динамічного розвитку й застосування інформаційних технологій виникає проблема підтримки належного рівня захищеності інформаційних середовищ від різноманітних видів атак та шкідливого програмного забезпечення. Для настільних застосунків є давно зарекомендований і надзвичайно зручний інструмент IDA Pro. Ця робота направлена на аналіз сучасних прийомів зворотної розробки Android застосунків та створення більш швидких і зручних методів, інструментів, технік направлених саме на ефективне тестування на проникнення.

У даній роботі розглядаються методи та інструменти тестування на захищеність Android-застосунків, з ціллю підвищення якості та ефективності цих сутностей. І в результаті досягнення високої захищеності мобільних застосунків.

Основна частина

Операційна система Android – прямий нащадок ОС Linux. І як нащадок ввібрала в себе багато механізмів безпеки від цієї ОС. Так як ОС розроблялася, як система реального часу, багато механізмів, разом і з тим і механізмів безпеки, змінилося і з’явилися нові, що змінює підходи до тестування на проникнення їх застосунків. І хоча компанія Google доповнює і ускладнює ці механізми, при розробці великих проектів залишаються архітектурні прогалини, що дозволяє компрометацію цих застосунків.

Було виділено основні відмінності в ОС, що вплинули на безпеку застосунків:

1. Створення ізольованих пісочниць та розвинутої системи міжпроцесної взаємодії призвело до дроблення програм на сервіси. Це збільшило поверхню атак на застосунки.
2. Використання мови програмування JAVA та формат виконуваних файлів заснований на її байткоді що виконується емуляторами Dalvik/ART зробило процес реверсу застосунків доволі тривіальним, а вихідний код після декомпіляції майже ідентичний вихідному.
3. Розробка механізму зберігання ключової інформації keystore ускладнила криптографічну компрометацію.
4. Відсутність ролі супер користувача знизила небезпеку підвищення привілеїв.

На сьогоднішній день ринок застосунків-аналізаторів, що стануть корисними в проведенні тестування типу чорного ящика, представлені на ринку доволі різноманітний. Серед інструментів-аналізаторів для тестування типу білого ящика таких набагато менше. Узагальнюючи, їх доречно розділити за типами, яких є два. Це автоматичні та мануальні.

Перший тип стане в нагоді для швидкого аналізу вихідного коду. Такі не аналізують архітектурних помилок, та мають високий ризик хибних спрацювань. Саме тому інженеру варто буде виконати інспекцію коду шляхом простої вичитки. Проте, безсумнівно, автоматичний тип аналізатору впорається з пошуком неправильного та недоречного послуговування криптографічних засобів та функцій, та попередить користувача про підозрілу ліцензію застосунку чи його права.

Користуючись ручним типом аналізатора варто розуміти, що програмне забезпечення обмежений отриманням коду застосунку та визначенню перемінних, а також повязування бібліотек застосунку з точкою їх входження в вихідний код додатку. Основну ж роботу все ще повинен робити розробник - інспекція, вичитка та декомпіляція.

З проведеного в роботі дослідження існуючих застосунків та аналізу їх вихідного результату стало зрозумілим, що ринок не може представити утиліту для повного і абсолютного полегшення роботи з кодом, без втручання спеціаліста. І хоч представлені та розглянуті інструменти і спрощують завдання, та не звільняють від роботи повністю - набір можливостей сучасних інструментів все ще обмежений та потрібно володіти усіма навичками та техніками спеціаліста з розробки застосунків, щоб мати змогу їх аналізувати влосноруч.

Після теоретичного огляду основ тестування безпеки, специфічно для додатків мобільної операційної системи, а саме тестування на проникнення, було окреслено усі труднощі та потенційні ризики статичного дослідження коду. Задля збільшення ефективності роботи без втрат якості та мінімізації похибок було прийнято рішення обрати інструмент з найбільш стабільними показниками з розглянутих у другому розділі. Серед сформованих вимог до утиліти основною було полегшення розуміння архітектури додатку.

Для вибору такого, найкращим рішенням було протестувати в дії кожен з них та зупинитися на тому, що найкраще пасує для вирішення поставленого завдання. Нижче буде наведено результати роботи випробуваних застосунків та проаналізовано отримані результати.

Є два типи запитів у статичному огляді apk-програми на основі Java-коду чи Smali.

ART - це розподілений час виконання, керований програмами та деякими налаштуваннями системи Android. ART та його провісник Далвік спочатку були розроблені відповідно для схеми Android. ART, як програма виконання програми, виконує налаштування Executable Dalvik та позначення байтів Dex.

ART та Dalvik підлягають виконанню коду Dex, тому розроблені програми для Dalvik повинні працювати під час роботи з ART.

Рішення впало на Smali, тому що нам не потрібно будувати декомпілятор, головним чином тому, що раніше в бізнесі є чудові інструменти декомпіляції. Людині складніше працювати з кодом Смалі, але програмніше більш очевидно. Пізніше, побачивши зменшену документацію по байт-кодам, було встановлено, що речі можна шукати в байт-коді, використовуючи регулярні вирази, оскільки вони мають чітку структуру.

Для підключення до хмарного сервісу бази даних пропонуться використати драйвер БД py2neo. Для повторного запуску був обраний застосунок з більшою кількістю класів, для перевірки всієї системи.

Часто спеціаліст з аналізу коду виявляє недоліки в кодовій базі, застосовуючи автоматизовані інструменти або шляхом ручного дослідження. Без точного знання конструкції не завжди зрозуміло, чи точно обробляються дані, які дані є і звідки вони беруться. У цьому випадку варто скористатися базою даних графів і знайти в точках входу функцію, що ініціалізується.

Результати та висновки

У мобільних додатках тестування безпеки є дуже складним і трудомістким завданням, оскільки потрібно перевірити різні аспекти та різні рівні, такі як неправильне використання SDK, контроль потоку, зв’язок, конфігурація інтерфейсу користувача, безпека локальної бази даних, правильна робота з криптографією тощо. Оскільки на ринку не існує інструменту, що може вирішити дану проблему швидко та якісно, було прийнято рішення розробити такий. В роботі описано та обгрунтовано основні дилеми, такі як побудова вимог, вибір мови програмування та власне, аналіз вихідного коду застосунку.

Інструмент має відомий недолік, проте враховуючи, що розв'язання даної проблеми не має новизни та не є актуальним, проте є корисним в утиліті, як в тиражованому продукті – це аналіз обфускованих додатків.

В якості вирішення наявної проблеми, можна запропонувати використання нових та пропрієтарних методів розробки програмного забезпечення.

Розроблена утиліта вирішує поставлену задачу, задовольняє приймальні критерії і повністю забезпечує тестування одного з аспектів виявлення вразливостей - а саме, інспекція коду на рахунок архітектурних недопрацювань застосунку.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Tariq Al-Rayes H. Studying Main Differences between Android & Linux Operating Systems / Hadeel Tariq Al-Rayes. // Vol:12 No:05. – 2012. – С. 46–49.
2. Android Platform Architecture [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://developer.android.com/guide/platfor
3. Rouse M. Linux operating system [Електронний ресурс] / M. Rouse, S. Bigelow // What's the best Linux OS for your enterprise data center? – 2017. – Режим доступу до ресурсу: https://searchdatacenter.techtarget.com/definition/Linux-operating-system.
4. About the Android Open Source Project [Електронний ресурс] // Android Source. – 2016. – Режим доступу до ресурсу: https://source.android.com/.
5. Ranganath S. Android Kernel and Linux Kernel / Ranganath. – 2015. – №3. – С. 17.
6. Maker F. A Survey on Android vs. Linux / F. Maker, Y. Chan. // Department of Electrical and Computer Engineering, University of California, Davis. – 2016. – С. 5–8.
7. Online APK Downloader [Електронний ресурс] – Режим доступу до ресурсу: https://apk-dl.com.
8. Rouse M. APK file (Android Package Kit file format) [Електронний ресурс] / Margaret Rouse // TechTarget Network WhatIs. – 2018. – Режим доступу до ресурсу: https://whatis.techtarget.com/definition/APK-file-Android-Package-Kit-file-format
9. APK Signature Scheme v2 [Електронний ресурс] // Android Developers Documentation. – 2019. – Режим доступу до ресурсу: https://source.android.com/security/apksigning/v2.
10. How to get the certificate signing information from an Android .APK [Електронний ресурс] // TheEther. – 2015. – Режим доступу до ресурсу: https://theether.net/kb/100207?id=100207.
11. Bettercap. // Kali Linux Tools Listing. – 2019. – С. 3–4.
12. McCamon M. Open Web Application Security Project / M. McCamon, E. Berman. // MediaWiki. – 2019.
13. ApkTool latetest detailed documentation [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://ibotpeaches.github.io/Apktool/.
14. A tool for reverse engineering Android apk files [Електронний ресурс] – Режим доступу до ресурсу: https://ibotpeaches.github.io/Apktool/documentation/.
15. J.M. Porup. Is source code inspection a security risk? / J.M. Porup. // COMPUTERWORLD. – 2017. – С. 76–78.
16. Kelembet E. Mobile Application Security Testing: Major Threats and the Tools Needed to Overcome Them / Ekaterina Kelembet. // Madappgang. – 2019.

Тамара Валеріївна Горбунова – студентка групи УБ-18м, факультет менеджменту та інформаційної безпеки, Вінницький національний технічний університет, м. Вінниця, e-mail: tamaragorbunova97@gmail.com

Науковий керівник: ***Сачанюк-Кавецька Наталія Василівна*** - кандидат технічних наук, доцент кафедри вищої математики, Вінницький національний технічний університет, м. Вінниця.

Tamara Valeriivna Horbunova - student of UB-18m group, faculty of management and information security, Vinnytsia national technical university, Vinnytsia, e-mail: tamaragorbunova97@gmail.com

Supervisor: ***Sachaniuk-Kavets`ka Natalia Vasylivna*** - candidate of technichal Sciences, associate Professor of higher mathematics, Vinnytsia national technical University, Vinnytsia.