ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЗАХИСТУ ІНФОРМАЦІЇ

КАФЕДРА ІНФОРМАЦІЙНОЇ ТА КІБЕРНЕТИЧНОЇ БЕЗПЕКИ

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«ТЕХНОЛОГІЯ ВИЯВЛЕННЯ ВРАЗЛИВОСТЕЙ ДОДАТКІВ ДЛЯ ОС ANDROID НА БАЗІ JADX»

зі спеціальності *125 Кібербезпека*
(код, найменування спеціальності)

освітньо-професійної програми *Інформаційна та кібернетична безпека*
(назва програми)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Іванов-Кожевніков Артем Антонович
(підпис)

Виконав: здобувач вищої освіти групи БСДМ-62
Іванов-Кожевніков Артем Антонович
(прізвище, ім'я)

Керівник Д.т.н. Гайдур Галина
(науковий ступінь, вчене звання, прізвище, ім'я)

Рецензент _____
(науковий ступінь, вчене звання, прізвище, ім'я)

Київ 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЗАХИСТУ ІНФОРМАЦІЇ

Кафедра Інформаційної та кібернетичної безпеки
Ступінь вищої освіти Магістр
Спеціальність 125 Кібербезпека
Освітньо-професійна програма Інформаційна та кібернетична безпека

ЗАТВЕРДЖУЮ
Завідувач кафедри ІКБ
_____ Галина ГАЙДУР
"___" жовтня 2023 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Іванову-Кожевнікову Артему Антоновичу
(прізвище, ім'я)

1. Тема кваліфікаційної роботи: «Технологія виявлення вразливостей додатків для ОС Android на базі JADX»

керівник кваліфікаційної роботи Гайдур Галина Іванівна д.т.н.
(прізвище, ім'я, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «19» жовтня 2023 року № 145.

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 15.12.2023 р.

3. Вихідні дані до кваліфікаційної роботи
мобільні додатки;
науково-технічна література;
наукова та технічна література, експлуатаційна документація, нормативні
Мобільний пристрій.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Теоретичні аспекти безпеки додатків для ОС Android та виявлення вразливостей на базі JADX
2. Практичний аналіз вразливостей додатку 'MASTG-Hacking-Playground' на базі JADX
3. Рекомендації та висновки в контексті виявлення вразливостей додатку

'MASTG-Hacking-Playground' на базі JADX

5. Перелік графічного матеріалу
Презентація PowerPoint.

6. Дата видачі завдання 19.10.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ зп	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Уточнення постановки завдання.	19.10.2023 р.	
2.	Аналіз науково-технічної літератури.	22.10.2023 р.	
3.	Обґрунтування вибору рішення.	27.10.2023 р.	
4.	Збір даних.	03.11.2023 р.	
5.	Аналіз особливостей стану захищеності мобільних пристроїв на базі ОС Android	15.11.2023 р.	
6.	Дослідження методів та засобів забезпечення захисту інформації від сучасних загроз	26.11.2023 р.	
7.	Розробка рекомендації щодо побудови та оптимізації захисту мобільних додатків на базі ОС Android	15.12.2023 р.	

Здобувач вищої освіти _____ Артем ІВАНОВ-КОЖЕВНІКОВ
(підпис) *(ім'я, прізвище)*

Керівник кваліфікаційної роботи _____ Галина ГАЙДУР
(підпис) *(ім'я, прізвище)*

ВІДГУК РЕЦЕНЗЕНТА
на кваліфікаційну роботу

здобувача ІВАНОВА-КОЖЕВНІКОВА Артема
на тему: «Технологія виявлення вразливостей додатків для ОС Android на базі JADX»

Актуальність: У сучасному світі кожна особа користується мобільними пристроями для полегшення різних аспектів її життя, витрачаючи більшу частину часу на використання мобільних додатків. Це охоплює оплату проїзду, банківські транзакції, покупки продуктів, придбання квитків в кіно тощо. Для забезпечення захисту коштів та конфіденційної інформації користувачів важливо встановити відповідний стандарт безпеки. Часто популярні додатки надалі стають об'єктами атак хакерів, що може призводити до витоків даних. Таким чином, для всіх додатків необхідно проводити регулярні аудити забезпечення безпеки, щоб гарантувати повний захист інформації системи.

Позитивні сторони:

1. Виконання завдання відповідає його змісту, високий рівень виконання роботи.
2. Матеріали про технології захисту та аудит безпеки мобільних додатків на платформі Android викладено чітко.
3. Текст відповідає встановленим нормам і правилам. Заключні висновки сформульовані чітко та інформативно. Дослідження науково-технічної літератури свідчить про вміння користуватись відповідними матеріалами для бакалаврської роботи.

Недоліки:

1. Зазначено лише частково сучасні аспекти загроз.
2. Аспекти аналізу вразливостей мобільних додатків не отримали достатньо уваги.

Відзначені зауваження не впливають на загальну позитивну оцінку кваліфікаційної роботи.

Висновок: Враховуючи недоліки, кваліфікаційна робота заслуговує оцінку **“добре”**, а здобувач(ка) **ІВАНОВ-КОЖЕВНІКОВ Артем** – присвоєння кваліфікації магістр з кібербезпеки за спеціалізацією інформаційна та кібернетична безпека.

Рецензент:

(науковий ступінь, вчене звання)	*(підпис)*	*(ім’я, прізвище)*

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЗАХИСТУ ІНФОРМАЦІЇ

ПОДАННЯ
ГОЛОВІ ДЕРЖАВНОЇ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ
ЩОДО ЗАХИСТУ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Направляється здобувач ІВАНОВ-КОЖЕВНІКОВ Артем до захисту кваліфікаційної роботи
(прізвище, ім'я)

спеціальності 125 Кібербезпека
освітньо-професійної програми Інформаційна та кібернетична безпека
(шифр і назва спеціальності)
на тему: «Технологія виявлення вразливостей додатків для ОС Android на базі JADX».

Кваліфікаційна робота і рецензія додаються.

Директор інституту _____ Віталій САВЧЕНКО
(підпис) *(ім'я, прізвище)*

Висновок керівника кваліфікаційної роботи

Студент **Іванов-Кожевніков А.А**. обрав тему роботи, метою дослідження якої є дослідити та з'ясувати ефективні методи щодо захисту мобільних додатків на базі ОС Android. Перелік використаних джерел свідчить про вміння бакалавром розбиратись в наукових питаннях та застосовувати їх при дослідженнях. Під час виконання бакалаврської роботи **Іванов-Кожевніков А.А**. показав гарну теоретичну та практичну підготовку, вміння самостійно вирішувати питання і робити висновки. Роботу виконував сумлінно, акуратно та вчасно за планом.

Все це дозволяє оцінити виконану бакалаврську роботу студента **Іванов-Кожевніков А.А**. на оцінку **«добре»** та присвоїти йому кваліфікацію 125. Фахівець з організації інформаційної безпеки.

Керівник кваліфікаційної роботи _____ ГАЙДУР Галина
(підпис) *(ім'я, прізвище)*
"____" _____ 2023 року

Висновок кафедри про кваліфікаційну роботу

Кваліфікаційна робота розглянута. Здобувач **ІВАНОВ-КОЖЕВНІКОВ Артем** допускається до захисту даної кваліфікаційної роботи в Державній екзаменаційній комісії.

Завідувач кафедри Інформаційної та кібернетичної безпеки
(назва)

_____ Галина ГАЙДУР
(підпис) *(ім'я, прізвище)*

РЕФЕРАТ

Текстова частина бакалаврської роботи: 62 сторінока, 18 рисунків, 20 джерел.

Мета роботи - Оцінити ефективність та потенціал технології виявлення вразливостей для забезпечення безпеки додатків на платформі Android. Визначити наявні вразливості в додатку "MASTG-Hacking-Playground" та розробити рекомендації щодо їх усунення та підвищення рівня безпеки.

Об'єкт дослідження - Пошук та аналіз вразливостей в коді додатків для ОС Android.

Предмет дослідження - Технологія виявлення вразливостей у мобільних додатка для ОС Android на базі інструменту JADX

Методи дослідження -

- Статичний та динамічний аналіз: Застосування інструментів статичного та динамічного аналізу для виявлення вразливостей у вихідному коді та під час виконання додатку "MASTG-Hacking-Playground".
- Використання інструментів декомпіляції: Використання JADX для розкриття вихідного коду Android-додатка для подальшого аналізу та ідентифікації потенційних загроз.
- Експериментальне тестування: Практичне тестування виявлених вразливостей для підтвердження їхнього існування та визначення можливого практичного значення.

У цій дипломній роботі проведено комплексне дослідження проблем безпеки мобільних додатків на платформі Android, зосереджуючись на використанні технології виявлення вразливостей за допомогою інструменту декомпіляції та аналізу вихідного коду, такого як JADX. Основний об'єкт дослідження - додаток "MASTG-Hacking-Playground". Результатом дослідження стали рекомендації щодо усунення

виявлених вразливостей, що спрямовані на підвищення загального рівня безпеки мобільних додатків. Отримані дані мають важливе практичне значення для розробників, тестувальників та всіх зацікавлених сторін, які працюють у сфері мобільних технологій та кібербезпеки.

Галузь використання – кібербезпека

ANDROID OS, OWASP MSTG, CWE, ПЕРЕДАЧА ДАНИХ, ADB, APK, JAVA, JADX, JADX-GUI, MASTG

ABSTRACT

Text part of a bachelor's thesis: 62 pages, 18 figures, , 20 sources.

Purpose - To evaluate the effectiveness and potential of vulnerability detection technology to ensure the security of Android applications. Identify existing vulnerabilities in the MASTG-Hacking-Playground application and develop recommendations for their elimination and security enhancement.

Object of research - Mobile applications, in particular "MASTG-Hacking-Playground", developed for the Android operating system.

Subject of research - Vulnerability detection technology based on the JADX tool for testing the security of mobile applications, with a focus on the MASTG-Hacking-Playground application.

Research methods.

- Static and dynamic analysis: The use of static and dynamic analysis tools to identify vulnerabilities in the source code and runtime of the "MASTG-Hacking-Playground" application.

- Using decompilation tools: Use of JADX to decompile the source code of the Android application for further analysis and identification of potential threats.

- Experimental testing: Hands-on testing of the identified vulnerabilities to confirm their existence and determine their possible practical significance.

In this thesis, a comprehensive study of the security problems of mobile applications on the Android platform was conducted, focusing on the use of vulnerability detection technology using a source code decompilation and analysis tool such as JADX. The main object of the study is the MASTG-Hacking-Playground application. The research resulted in recommendations for eliminating the identified vulnerabilities, aimed

at improving the overall security of mobile applications. The findings are of great practical importance for developers, testers and all stakeholders working in the field of mobile technology and cybersecurity.

Application area - cybersecurity

ANDROID OS, OWASP MSTG, CWE, DATA TRANSFER, ADB, APK, JAVA, JADX, JADX-GUI, MASTG

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

APK	–	Android application package
GUI	–	Graphical User Interface
JVM	–	Java Virtual Machine
JIT	–	Just in Time копіляція
ADB	–	Android debug bridge
IPv4	–	Internet protocol версії 4
IPv6	–	Internet protocol версії 6
CVE	–	Common Vulnerabilities and Exposures
CWE	–	Common Weakness Enumeration
ZIP	–	Формат архівації данних
MFA	–	Багатофакторна аутентіфікація
OTP	–	Одноразовий пароль
SSL	–	Secure Sockets Layer
TLS	–	transport layer security
JADX	–	Java Decompilator
HTTP/S	–	HyperText Transfer Protocol/Secure
JAR	–	Java Archive

ВСТУП

Актуальність дослідження – В сучасному цифровому середовищі, де використання мобільних додатків стає необхідністю, зростає і важливість забезпечення їхньої безпеки. Операційна система Android, що використовується мільярдами користувачів, стає об'єктом зростаючого інтересу для кіберзлочинців. Проте, актуальність проблеми безпеки мобільних додатків на базі Android підкреслюється в контексті використання технології виявлення вразливостей, зокрема на основі інструменту декомпіляції та аналізу вихідного коду, такого як JADX. Забезпечення високого рівня безпеки стає невід'ємною складовою в розробці та використанні мобільних додатків, і технологія на базі JADX виступає ефективним інструментом у виявленні та подальшому усуненні потенційних загроз для користувачів та їхніх особистих даних.

Об’єкт дослідження - Мобільні додатки, зокрема "MASTG-Hacking-Playground", розроблені для операційної системи Android.

Предмет дослідження - Технологія виявлення вразливостей на базі інструменту JADX при тестуванні безпеки мобільних додатків, з фокусом на додатку "MASTG-Hacking-Playground".

Мета роботи - Оцінити ефективність та потенціал технології виявлення вразливостей для забезпечення безпеки додатків на платформі Android. Визначити наявні вразливості в додатку "MASTG-Hacking-Playground" та розробити рекомендації щодо їх усунення та підвищення рівня безпеки.

Наукові завдання:

- Провести аналіз теоретичних аспектів архітектури Android-додатків та механізмів їхньої безпеки.
- Вивчити та порівняти існуючі інструменти тестування безпеки для мобільних додатків, зокрема ті, що базуються на декомпіляції вихідного коду.

- Провести детальний аналіз додатку "MASTG-Hacking-Playground" за допомогою технології виявлення вразливостей на базі JADX.
- Визначити та класифікувати виявлені вразливості.
- Розробити рекомендації щодо усунення виявлених вразливостей та підвищення рівня безпеки додатку.
- Підбити підсумки проведеного дослідження та зробити висновки щодо ефективності технології виявлення вразливостей для мобільних додатків на базі Android.

Методи дослідження -

- Статичний та динамічний аналіз: Застосування інструментів статичного та динамічного аналізу для виявлення вразливостей у вихідному коді та під час виконання додатку "MASTG-Hacking-Playground".
- Використання інструментів декомпіляції: Використання JADX для розкриття вихідного коду Android-додатка для подальшого аналізу та ідентифікації потенційних загроз.
- Експериментальне тестування: Практичне тестування виявлених вразливостей для підтвердження їхнього існування та визначення можливого практичного значення.

Практичне значення одержаних результатів

- Підвищення безпеки додатку: Розробка та реалізація рекомендацій для усунення виявлених вразливостей в додатку "MASTG-Hacking-Playground", що призведе до покращення загального рівня безпеки мобільних додатків на платформі Android.Створення бази знань: Формування бази знань та методології для виявлення та усунення вразливостей в Android-додатках, які можуть бути використані розробниками, тестувальниками та іншими зацікавленими сторонами.

- Удосконалення методології тестування: Внесення вкладу у розвиток ефективних методів та підходів до тестування безпеки мобільних додатків на базі Android для попередження можливих загроз та атак.
- Захист особистої інформації користувачів: Забезпечення безпеки та конфіденційності особистих даних користувачів, які використовують додаток "MASTG-Hacking-Playground" та аналогічні мобільні додатки.
- Подальше дослідження: Надання підстав для подальших досліджень у сфері виявлення та усунення вразливостей в мобільних додатках на базі Android з використанням технології на базі JADX.

1. ТЕОРЕТИЧНІ АСПЕКТИ БЕЗПЕКИ ДОДАТКІВ ДЛЯ ОС ANDROID ТА ВИЯВЛЕННЯ ВРАЗЛИВОСТЕЙ НА БАЗІ JADX

1.1 Класифікація вразливостей для подальшого аналізу додатку «MASTG-Hacking-Playground”

Багато тестувальників мобільних додатків мають досвід тестування на проникнення в мережі та веб-додатки, що є цінним досвідом для тестування мобільних додатків. Майже кожен мобільний додаток взаємодіє з внутрішніми службами, і ці служби схильні до тих самих типів атак, що й веб-додатки на настільних комп'ютерах. Мобільні додатки відрізняються меншою поверхнею для атак, а отже, більшим захистом від ін'єкцій та подібних атак. Натомість, ми повинні визначати пріоритети захисту даних на пристрої та в мережі, щоб підвищити мобільну безпеку.

Зберігання даних і конфіденційність (MASVS-STORAGE)

Захист конфіденційних даних, таких як облікові дані користувача та приватна інформація, має вирішальне значення для мобільної безпеки. Якщо додаток неналежним чином використовує API операційної системи, такі як локальне сховище або міжпроцесний зв'язок (IPC), програма може розкрити конфіденційні дані іншим програмам, що працюють на тому ж пристрої. Також може статися ненавмисний витік даних до хмарного сховища, резервних копій, або до кешу клавіатури. Крім того, мобільні пристрої легше загубити або вкрасти порівняно з іншими типами пристроїв, тому існує велика ймовірність того, що зловмисник може отримати фізичний доступ до пристрою, а отже, легше відновити дані. Розробляючи мобільні додатки, ми повинні бути особливо обережними при зберіганні даних користувачів. Наприклад, ми можемо використовувати відповідні ключові API для зберігання даних і скористатися перевагами апаратних засобів захисту, якщо вони доступні. Фрагментація - це проблема, з якою ми маємо справу, особ-

ливо на пристроях Android. Не кожен пристрій Android пропонує апаратну підтримку безпечне сховище, а багато пристроїв працюють на застарілих версіях Android. Щоб додаток підтримувався на цих застарілих пристроях застарілих пристроях, його потрібно створювати за допомогою старої версії API Android, в якій можуть бути відсутні важливі функції безпеки. Для забезпечення максимальної безпеки найкращим вибором є створення додатків з актуальною версією API, навіть якщо це виключає деяких користувачів.[8]

Криптографія (MASVS-CRYPTO)

Криптографія в контексті мобільного застосування відіграє ключову роль у забезпеченні конфіденційності, цілісності та автентичності даних. Вона використовує різні математичні та обчислювальні методи для захисту інформації від несанкціонованого доступу та модифікацій. Однією з основних задач криптографії в мобільних додатках є забезпечення безпеки під час передачі даних через мережу. Використання шифрування забезпечує захист від прослуховування та зміни інформації під час передачі, забезпечуючи конфіденційність та цілісність. Криптографічні алгоритми також використовуються для зберігання конфіденційної інформації на мобільних пристроях. Вони забезпечують шифрування файлів, баз даних та інших ресурсів, що зберігаються на пристрої, запобігаючи несанкціонованому доступу до даних. Важливою частиною використання криптографії є правильне управління ключами. Захист ключів від несанкціонованого доступу та їх безпекне зберігання є критичними для ефективної роботи криптографічних заходів. При розробці мобільних додатків важливо враховувати сучасні криптографічні стандарти та рекомендації забезпечення безпеки. Крім того, системи повинні регулярно аудитуватися та тестуватися на виявлення можливих уразливостей та атак, спрямованих на обхід криптографічного захисту.[8]

Автентифікація та авторизація (MASVS-AUTH)

У більшості випадків відправка користувачів для входу у віддалений сервіс є невід'ємною частиною загальної архітектури мобільного додатку. Незважаючи на те, що хоча більша частина логіки автентифікації та авторизації відбувається на кінцевій точці, існують також деякі проблеми з реалізацією на стороні мобільних додатків. На відміну від веб-додатків, мобільні додатки часто зберігають довготривалі токени сеансів, які розблоковуються за допомогою функцій автентифікації між користувачем і пристроєм, таких як сканування відбитків пальців. Це дозволяє пришвидшити вхід та покращити користувацький досвід (ніхто не любить вводити дані користувацький досвід (ніхто не любить вводити складні паролі), це також створює додаткову складність і можливість для помилок.

Архітектура мобільних додатків також все частіше включає фреймворки авторизації (такі як OAuth2), які делегують авторизацію окремим службам або зовнішнім сервісам. авторизацію окремим сервісам або передають процес автентифікації постачальнику послуг автентифікації на аутсорсинг. Використання OAuth2 дозволяє логіку автентифікації на стороні клієнта передати іншим програмам на тому ж пристрої (наприклад, системному браузеру). Тестувальники безпеки повинні знати переваги та недоліки різних можливих фреймворків та архітектур авторизації.[8]

Мережевий зв'язок (MASVS-NETWORK)

Мережевий зв'язок є ключовим аспектом безпеки мобільних додатків, оскільки взаємодія з серверами та іншими джерелами даних відбувається через мережу. Розділ Mobile Application Security Verification Standard (MASVS) – Network визначає стандарти та вимоги до забезпечення безпеки в області мережевого зв'язку мобільних додатків. Основна мета MASVS-NETWORK – це визначення найкращих практик та стандартів для забезпечення конфіденційності та цілісності даних, що передаються через мережу. Стандарт визначає рекомендації щодо захисту мобільних додатків від різноманітних загроз, таких як перехоплення

трафіку, атаки типу "Man-in-the-Middle" (MITM) та інші атаки на рівні мережі. Серед ключових вимог MASVS-NETWORK можна виділити використання шифрування для захисту передаваних даних, перевірку сертифікатів для уникнення MITM-атак, використання безпечних протоколів та механізмів аутентифікації. Також важливою є валідація вхідних даних, щоб запобігти атакам на рівні мережі через некоректно оброблені вхідні параметри. Використання стандартів та рекомендацій MASVS-NETWORK дозволяє забезпечити надійний та безпечний обмін даними між мобільним додатком і віддаленим сервером, зменшуючи ризик потенційних атак та підвищуючи загальний рівень безпеки мобільного застосунку.[7]

Взаємодія з мобільною платформою (MASVS-PLATFORM)

Архітектура мобільних операційних систем суттєво відрізняється від класичної архітектури настільних комп'ютерів. Наприклад, всі мобільні операційні системи реалізують системи дозволів для додатків, які регулюють доступ до певних API. Вони також пропонують більш (Android) або менш багаті (iOS) засоби міжпроцесної взаємодії (IPC), які дозволяють програмам обмінюватися сигналами і даними. Ці специфічні для кожної платформи функції мають свої підводні камені. Наприклад, при неправильному використанні IPC API, конфіденційні дані або функціонал можуть бути ненавмисно доступні іншим програмам, що працюють на пристрої. Взаємодія з мобільною платформою є важливим елементом розробки безпечних та ефективних мобільних додатків. Розділ Mobile Application Security Verification Standard (MASVS) – Platform створений для визначення стандартів та рекомендацій забезпечення безпеки на рівні платформи для мобільних застосунків. Основна мета MASVS-PLATFORM – це забезпечення найвищого рівня безпеки та довіри шляхом встановлення стандартів для розробників мобільних додатків. Цей розділ визначає вимоги до безпеки платформи, включаючи операційні системи iOS та Android. Ключові аспекти MASVS-PLATFORM включають контроль доступу до платформених ресурсів, ефективне управління сесіями, валідацію та фільтрацію введених даних, захист від атак ін'єкцій, а також забезпечення безпеки

взаємодії з вбудованими платформеними компонентами. Врахування та дотримання стандартів MASVS-PLATFORM дозволяє розробникам створювати надійні та безпечні мобільні додатки, що ефективно взаємодіють з операційними системами платформи, зменшуючи ризики вразливостей та атак з боку шкідливих суб'єктів.

Якість коду та захист від експлойтів (MASVS-CODE)

Традиційні проблеми з ін'єкціями та управлінням пам'яттю не часто зустрічаються в мобільних додатках через меншу поверхню атаки. Мобільні додатки здебільшого взаємодіють з довіреним бекенд-сервісом та інтерфейсом користувача, тому навіть якщо в додатку існує багато вразливостей, пов'язаних з переповненням буфера в додатку, ці вразливості зазвичай не відкривають жодних корисних векторів атаки. Те ж саме стосується і браузерних експлойтів, таких як міжсайтовий скриптинг (XSS дозволяє зловмисникам впроваджувати скрипти у вебсторінки), які дуже поширені у веб-додатках. Однак завжди є винятки. У деяких випадках XSS теоретично можлива на мобільних пристроях, але це дуже рідкісний випадок XSS-проблеми, якими може скористатися людина, трапляються дуже рідко. Цей захист від ін'єкцій та проблем з керуванням пам'яттю не означає, що розробники додатків можуть безкарно написання недбалого коду. Дотримання найкращих практик безпеки призводить до створення захищених (безпечних) збірок релізів, які є стійкими до втручання. Безкоштовні функції безпеки, що пропонуються компіляторами та мобільними SDK, допомагають підвищити безпеку та пом'якшити наслідки атак.

Захист від несанкціонованого втручання та зворотного впливу (MASVS-RESILIENCE)

Багато експертів з безпеки категорично відкидають захист на стороні клієнта. Однак, засоби захисту програмного забезпечення широко використовуються у світі мобільних додатків, тому тестувальникам безпеки потрібні способи роботи

з цими засобами захисту. Я вважаю, що засоби захисту на стороні клієнта, якщо вони застосовуються з чіткою метою та реалістичними очікуваннями, а не використовуються для заміни засобів контролю безпеки, то в такому випадку це гарна ідея для профілактики. Захист від несанкціонованого втручання та зворотного впливу є критично важливим аспектом в розробці безпечних мобільних додатків і визначається в рамках розділу Mobile Application Security Verification Standard (MASVS) – Resilience. Цей розділ створений для забезпечення міцності та стійкості мобільних додатків до небажаних впливів та атак, а також для мінімізації можливих наслідків і втрат в разі втручання. Ключові аспекти MASVS-RESILIENCE включають в себе розробку заходів забезпечення високого рівня стійкості до атак, таких як виявлення та обмеження дій несанкціонованих користувачів, захист збережених даних від зловмисного доступу, та забезпечення безпеки важливих операцій, таких як аутентифікація та авторизація. Розділ MASVS-RESILIENCE надає рекомендації щодо використання сучасних методів шифрування, підпису та аутентифікації для захисту даних та забезпечення безпеки обчислювальних операцій. Додатково, він наголошує на важливості перевірки правильності введених даних та униканні проблем з введенням через відповідні механізми валідації. Захист від несанкціонованого втручання та зворотного впливу в мобільних додатках – це необхідна умова для забезпечення високого рівня безпеки, конфіденційності та інтегритету важливих інформаційних ресурсів та функціоналу.

1.2 Аналіз Статистики Вразливостей у Додатках для Android" за OWASP MSTG

В сучасному цифровому ландшафті мобільні додатки для Android стали невід'ємною частиною нашого повсякденного життя. Зростання популярності цих додатків відзначено різким поглибленням досліджень щодо їхньої кібербезпеки та виявленням вразливостей. Одним із важливих інструментів для такого аналізу є OWASP Mobile Security Testing Guide (MSTG).

OWASP MSTG — це важливий засіб, який надає стандарти та методології для тестування безпеки мобільних додатків. Аналіз статистики вразливостей у додатках для Android за допомогою MSTG дозволяє виявляти потенційні проблеми безпеки та вживати заходи для їх виправлення. Однією з ключових областей вивчення є аутентифікація та авторизація. MSTG дозволяє провести аналіз захисту облікових записів користувачів, перевірити правильність механізмів авторизації та виявити можливі слабкі місця. Ще однією важливою категорією є безпека мережі. Дослідження, проведене згідно з OWASP MSTG, дозволяє визначити, чи використовуються ефективні заходи для захисту передачі даних через мережу та чи існують потенційні загрози безпеці. Аналіз вразливостей за допомогою OWASP MSTG враховує також криптографічні аспекти, перевірку дозволів та безпеку введення даних. Всі ці елементи дозволяють підняти безпековий рівень мобільного додатка та зменшити ризик його використання в зловмисних цілях. Загальна мета аналізу статистики вразливостей у додатках для Android за OWASP MSTG полягає в удосконаленні процесів розробки та впровадження безпеки, щоб забезпечити найвищий рівень захисту для користувачів мобільних платформ."

Організація Open Web Application Security Project (OWASP) пропонує інструменти та рекомендації для виявлення та усунення потенційних загроз безпеці в мобільних додатках. Переважна частина вразливостей може бути класифікована за допомогою OWASP Mobile Security Testing Guide (MSTG), який став необхідним для розробників та тестувальників. Однією з ключових областей аналізу є аутентифікація та авторизація. Вразливості в цих механізмах можуть викликати серйозні проблеми з безпекою, що стає об'єктом уваги в рамках MSTG. Аналіз облікових записів користувачів, механізмів авторизації та контролю доступу визначає рівень захищеності додатка від несанкціонованого доступу. Безпека мережі – ще один ключовий аспект. Передача даних через мережу може бути уразливою, і MSTG дозволяє виявити та усунути можливі проблеми, пов'язані з шифруванням та захистом інформації в процесі передачі. Аналіз криптографічних

практик, перевірка прав доступу та оцінка безпеки введення даних – всі ці етапи допомагають створити додаток, який не лише функціональний, а й захищений від можливих загроз. Загальний аналіз статистики вразливостей у додатках для Android є ключовим етапом в розробці безпечних та надійних мобільних додатків. Методика MSTG визначає стандарти безпеки, що дозволяє розробникам ефективно протистояти сучасним кіберзагрозам і забезпечити високий рівень захисту для кінцевих користувачів." OWASP Mobile Security Testing Guide (MSTG) визначає та описує різноманітні вразливості, які можуть впливати на безпеку мобільних додатків. Ось короткий огляд кожного пункту вразливостей з OWASP Mobile Security Testing Guide:

Insecure Data Storage (Небезпечне зберігання даних):

Опис: Некоректне зберігання конфіденційних даних, таких як паролі або особиста інформація, на мобільних пристроях.

Заходи безпеки: Використання безпечних механізмів зберігання даних та шифрування для захисту інформації.

Insecure Data Transmission (Небезпечна передача даних):

Опис: Вразливість, пов'язана з небезпечною передачею даних через мережу, що може призвести до перехоплення конфіденційної інформації.

Заходи безпеки: Використання шифрування для захисту даних під час їх передачі.

Insecure Authentication (Небезпечна аутентифікація):

Опис: Проблеми з механізмами аутентифікації, такі як слабкі паролі або відсутність додаткових заходів безпеки.

Заходи безпеки: Використання сильних методів аутентифікації та впровадження двофакторної аутентифікації.

Insecure Direct Object References (Небезпечні прямі посилання на об'єкти):

Опис: Проблеми з управлінням доступом, які можуть дозволити несанкціонованим користувачам отримувати доступ до конфіденційних об'єктів.

Заходи безпеки: Контроль доступу та використання ідентифікаторів сесій для обмеження доступу.

Unintended Data Leakage (Непередбачені витоки даних):

Опис: Виток конфіденційної інформації через недостатні заходи безпеки.

Заходи безпеки: Впровадження механізмів захисту даних та обмеження доступу до конфіденційної інформації.

Broken Cryptography (Пошкодження криптографії):

Опис: Недоліки в реалізації криптографічних алгоритмів, які можуть призвести до порушення безпеки.

Заходи безпеки: Використання сильних алгоритмів шифрування та правильна їхньої реалізації.

Security Decisions Via Untrusted Inputs (Прийняття рішень щодо безпеки через ненадійні введені дані):

Опис: Вразливості, пов'язані з використанням ненадійних даних для прийняття рішень щодо безпеки.

Заходи безпеки: Перевірка та обробка введених даних для попередження атак на прийняття рішень.

Security Misconfiguration (Неправильна конфігурація безпеки):

Опис: Проблеми, пов'язані з неправильною конфігурацією безпеки, що може призвести до вразливостей.

Заходи безпеки: Правильна конфігурація середовища та застосунків для максимального рівня захисту.

Client Code Quality (Якість клієнтського коду):

Опис: Вразливості, що виникають через низьку якість програмного коду клієнтської частини додатка.

Заходи безпеки: Використання найкращих практик розробки для запобігання вразливостей у клієнтському коді.

Code Tampering (Втручання в код):

Опис: Вразливості, що можуть виникнути внаслідок змін програмного коду додатка з боку користувача.

Заходи безпеки: Захист програмного коду від несанкціонованих змін та використання механізмів виявлення втручань.

Ці категорії вразливостей визначають основні проблеми безпеки, які можуть впливати на мобільні додатки, та слугують орієнтирами для їхнього тестування та виправлення.

1.3 Застосування статичного та динамічного аналізу для виявлення вразливостей

Статичне тестування безпеки додатків (SAST) передбачає перевірку компонентів програми без їх виконання, шляхом аналізу вихідного коду вручну або автоматично. OWASP надає інформацію про статичний аналіз коду, яка може допомогти вам зрозуміти методи, сильні та слабкі сторони та обмеження. Динамічне тестування безпеки додатків (DAST) передбачає перевірку програми під час виконання. Цей тип аналізу може бути ручним або автоматичним. Зазвичай він не надає інформації, яку надає статичний аналіз, але це хороший спосіб виявити цікаві з точки зору користувача елементи (активи, функції, точки входу тощо). Тепер, коли ми визначили статичний і динамічний аналіз, давайте зануримося глибше.

Статичний аналіз

Під час статичного аналізу перевіряється вихідний код мобільного застосунку, щоб забезпечити належну реалізацію засобів контролю безпеки. У більшості випадків використовується гібридний автоматичний/ручний підхід. Автоматичне сканування знаходить найпростіші вразливості, а людина-тестувальник тестувальник може дослідити кодову базу з урахуванням конкретних контекстів використання.

Ручний перегляд коду

Тестувальник виконує ручну перевірку коду, вручну аналізуючи вихідний код мобільного додатку на наявність вразливостей безпеки. Методи варіюються від базового пошуку за ключовими словами за допомогою команди "grep" до построкового аналізу вихідного коду. IDE (інтегровані середовища розробки) часто надають базові функції перевірки коду і можуть бути розширені за допомогою різних інструментами. Загальний підхід до ручного аналізу коду полягає у визначенні ключових індикаторів вразливостей безпеки шляхом пошуку певних API та

ключових слів, таких як виклики методів, пов'язаних з базою даних, таких як "executeStatement" або "executeQuery". Код що містить ці рядки, є гарною відправною точкою для ручного аналізу. На відміну від автоматичного аналізу коду, ручна перевірка коду дуже добре підходить для виявлення вразливостей в бізнес-логіці логіки, порушень стандартів і недоліків проектування, особливо коли код технічно безпечний, але логічно недосконалий. Такі сценарії навряд чи буде виявлено будь-яким інструментом автоматичного аналізу коду. Для ручного перегляду коду потрібен експерт-рецензент, який добре володіє мовою та фреймворками, що використовуються для мобільного додатку. Повна перевірка коду може бути повільним, нудним і трудомістким процесом для рецензента, особливо з огляду на великі бази коду з великою кількістю залежностей. [18]

Автоматизований аналіз вихідного коду

Інструменти автоматизованого аналізу можуть бути використані для прискорення процесу перевірки статичного тестування безпеки додатків (SAST). Вони перевіряють вихідний код на відповідність заданому набору правил або найкращим галузевим практикам, а потім зазвичай відображають список висновків або попереджень і прапорців для всіх виявлених порушень. Деякі інструменти статичного аналізу працюють лише зі скомпільованим додатком Деякі інструменти статичного аналізу працюють лише з скомпільованим додатком, іншим потрібно завантажити оригінальний вихідний код, а деякі працюють як плагіни для аналізу в режимі реального часу в інтегрованому середовищі розробки (IDE). середовища розробки (IDE). Хоча деякі інструменти статичного аналізу коду містять багато інформації про правила і семантику, необхідні для аналізу мобільних додатків, вони можуть видавати багато помилкових результатів. мобільних додатків, вони можуть давати багато хибних спрацьовувань, особливо якщо вони не налаштовані для цільового середовища. А Тому фахівець з безпеки повинен завжди перевіряти результати. У розділі "Інструменти тестування" наведено список інструментів статичного аналізу, який можна знайти в кінці цієї книги.

Динамічний аналіз

У центрі уваги DAST - тестування та оцінка додатків під час їх виконання в режимі реального часу. Основна мета динамічного аналізу є пошук вразливостей безпеки або слабких місць у програмі під час її виконання. Динамічний аналіз проводиться як на рівні мобільної платформи, так і на рівні бекенд-сервісів та API, де можна проаналізувати шаблони запитів та відповідей мобільного додатку. шаблони запитів і відповідей мобільного додатку. Динамічний аналіз зазвичай використовується для перевірки механізмів безпеки, які забезпечують достатній захист від найбільш поширених типів атак, таких як розкриття даних під час передачі, проблеми з автентифікацією та авторизацією, а також помилки конфігурації сервера. помилки конфігурації сервера.

1.4 Огляд популярних інструментів тестування безпеки Android-додатків

При пошуку та аналізі вразливостей, спеціаліст з кібербезпеки має у доступі великий список інструментів, які допоможуть йому у виконанні тестових кейсів, дозволяючи виконувати статичний аналіз, динамічний аналіз, динамічне вимірювання тощо. Ці інструменти призначені для того, щоб допомогти проводити власні оцінки, а не для того, щоб надати остаточний результат про стан безпеки програми. Дуже важливо уважно переглядати результати роботи інструментів, оскільки вони можуть містити як помилкові оскільки вони можуть містити як хибнопозитивні, так і хибнонегативні результати.

Огляд популярних інструментів тестування безпеки Android-додатків є важливою складовою процесу розробки та впровадження мобільних застосунків. Забезпечення високого рівня безпеки є пріоритетом, оскільки велика частина особистої і конфіденційної інформації зберігається в мобільних пристроях. Розглянемо кілька ключових інструментів, які використовуються для тестування та забезпечення безпеки Android-додатків. [8]

Mobile Security Framework (MobSF):

Опис: MobSF - це відкрите програмне забезпечення для автоматизованого тестування безпеки мобільних додатків, включаючи Android.

Функції: Аналіз вихідного коду, сканування вразливостей, тестування витоків даних.

QARK (Quick Android Review Kit):

Опис: QARK - це інструмент для автоматизованого тестування безпеки Android-додатків, розроблений компанією LinkedIn.

Функції: Виявлення вразливостей, аналіз безпеки коду, перевірка дозволів.

AndroBugs Framework:

Опис: AndroBugs - це фреймворк для аналізу безпеки Android-додатків з використанням статичного та динамічного аналізу.

Функції: Пошук вразливостей, класифікація ризиків, виявлення витоків даних.

OWASP Mobile Security Testing Guide:

Опис: Спільнота OWASP (Open Web Application Security Project) надає власний посібник з тестування безпеки мобільних додатків.

Функції: Методологія тестування, рекомендації щодо безпеки, перелік вразливостей.

Android Debug Bridge (ADB):

Опис: ADB - стандартний інструмент для взаємодії з Android-пристроями з рівня командного рядка.

Функції: Аналіз трафіку, витягування даних, тестування додатків.

Drozer:

Опис: Drozer - це відомий фреймворк для забезпечення та тестування безпеки Android-додатків.

Функції: Аналіз зворотного інжинірингу, тестування вразливостей, маніпуляція даними.

Xposed Framework:

Опис: Це фреймворк для модифікації системи Android, який може використовуватися для тестування безпеки та впровадження власних модифікацій.

Функції: Зміна поведінки додатків, впровадження власних модулів безпеки.

Ratproxy:

Опис: Ratproxy - це проксі-інструмент для тестування безпеки мобільних додатків, призначений для аналізу та виявлення вразливостей.

Функції: Аналіз трафіку, виявлення вразливостей у вхідних та вихідних даних.

JADX:

Опис: JADX - це інструмент для декомпіляції та аналізу вихідного коду Android-додатків, який дозволяє розглядати роботу програм без доступу до їх вихідного коду.

Функції: Декомпіляція Java-коду, аналіз ресурсів додатків, виявлення потенційних загроз безпеці.

JADX надає можливість вивчати структуру та логіку Android-додатків, спрощуючи аналіз та виявлення можливих вразливостей чи загроз безпеки в додатках.

Використання цих інструментів дозволяє розробникам та тестувальникам забезпечити найвищий рівень безпеки для Android-додатків. Регулярне тестування та вдосконалення заходів безпеки допомагає забезпечити надійний захист від потенційних загроз та зловмисних атак. Інтеграція цих інструментів у процес розробки дозволяє виявляти та вирішувати проблеми безпеки на ранніх етапах створення додатків, зменшуючи ризик виникнення серйозних проблем у пізніших етапах експлуатації. Окрім того, використання інструментів тестування безпеки дозволяє розробникам та командам з кібербезпеки вивчати сучасні тенденції у сфері безпеки мобільних додатків та ефективно взаємодіяти з різними типами загроз. Порівняння та аналіз інструментів дозволяє вибрати найбільш підходящі для конкретного проекту та враховувати специфічні особливості додатків. Загалом, використання популярних інструментів тестування безпеки Android-додатків є необхідним елементом повноцінного циклу розробки та забезпечення високого рівня безпеки та захисту конфіденційності для користувачів мобільних пристроїв.

Особливості використання JADX для виявлення потенційних загроз безпеці

JADX визначається своєрідністю та ефективністю виявлення потенційних загроз безпеці в Android-додатках. Його особливості використання можна описати наступним чином:

Декомпіляція Java-коду:

Опис: JADX забезпечує можливість декомпіляції вихідного Java-коду з скомпільованих файлів додатків (.apk).

Переваги: Аналітики та розробники можуть отримати доступ до зрозумілого вихідного коду, що дозволяє легше виявляти можливі загрози безпеці.

Аналіз ресурсів додатків:

Опис: JADX дозволяє розглядати та аналізувати ресурси Android-додатків, такі як макети, зображення та інші.

Переваги: Аналітики можуть отримати інформацію про вміст та функціонал додатка, що полегшує виявлення можливих слабкостей або загроз безпеки.

Виявлення потенційних загроз:

Опис: За допомогою JADX можна аналізувати код та виявляти можливі загрози безпеці, такі як некоректна обробка даних, вразливості безпеки мережевого взаємодії та інші.

Переваги: Ідентифікація потенційних загроз дозволяє розробникам та аудиторам забезпечити високий рівень безпеки додатків.

Взаємодія з іншими інструментами:

Опис: JADX може використовуватися як частина більш широкого набору інструментів для тестування та аналізу безпеки.

Переваги: Інтеграція з іншими інструментами дозволяє отримувати комплексний погляд на безпеку додатків.

Підтримка роботи з багатомовним кодом:

Опис: Інструмент підтримує аналіз коду, написаного на різних мовах програмування для платформи Android.

Переваги: Зручність для аналітиків та розробників, що працюють з різноманітними додатками.

Використання JADX дозволяє здійснювати детальний аналіз Android-додатків, забезпечуючи високий рівень надійності та безпеки мобільних додатків.

JADX-GUI є графічним інтерфейсом користувача для інструменту декомпіляції Android-додатків, JADX. Цей інтерфейс робить процес аналізу вихідного коду більш зручним та візуально доступним для користувачів. Ось більш детальний опис та використання:

Інтерфейс та Навігація:

Опис: JADX-GUI має інтуїтивний інтерфейс з вкладеними панелями та вікнами, що дозволяє користувачам легко оглядати та навігувати в декомпільованому коді.

Використання: Користувачі можуть швидко переглядати структуру проекту, переходити між класами та методами, а також отримувати інформацію про ресурси додатка.

Перегляд та Редагування Коду:

Опис: JADX-GUI дозволяє вам переглядати декомпільований Java-код у вікні редактора, з можливістю швидкої навігації та пошуку.

Використання: Користувачі можуть аналізувати, редагувати та розуміти структуру та функціонал Android-додатків.

Аналіз Ресурсів та XML-файлів:

Опис: Додатково до декомпіляції коду, JADX-GUI відображає ресурси та XML-файли, що входять до складу додатка.

Використання: Користувачі можуть переглядати та редагувати ресурси, такі як макети і зображення, що полегшує роботу з інтерфейсом додатка.

Деталізована Інформація про Класи та Методи:

Опис: Кожен клас та метод відображаються з докладною інформацією, включаючи змінні, константи, параметри та інші деталі.

Використання: Це дозволяє аналізувати та розуміти внутрішню логіку та логіку роботи кожного елемента додатка.

Експорт та Збереження Проекту:

Опис: Важлива функція - можливість експорту та збереження декомпільованого проекту для подальшого аналізу чи редагування.

Використання: Це дає можливість зберігати та обробляти отримані дані навіть поза самим JADX-GUI.

Підтримка Декомпіляції для Android-Бібліотек:

Опис: Має можливість декомпілювати не тільки додатки, але й бібліотеки, що розширює спектр використання.

Використання: Корисно для аналізу та розуміння сторонніх бібліотек, що можуть бути використані у проектах.

JADX-GUI значно спрощує процес аналізу Android-додатків, надаючи зручний та ефективний інтерфейс для роботи з декомпільованим кодом та ресурсами.

Переваги використання технології JADX:

1. Розкриття Внутрішньої Логіки Додатків:

Однією з ключових переваг використання технології JADX є можливість розкриття внутрішньої логіки Android-додатків. Декомпіляція вихідного коду дозволяє аналізувати та розуміти роботу програм, сприяючи подальшому вдосконаленню та оптимізації.

2. Виявлення Потенційних Загроз Безпеки:

За допомогою JADX можна виявляти потенційні загрози безпеки в Android-додатках. Аналіз вихідного коду дозволяє ідентифікувати слабкі місця та вразливості, що є важливим етапом в забезпеченні надійності та безпеки додатків.

3. Можливість Оптимізації та Модифікації:

Декомпілюваний код відкриває можливості для оптимізації та модифікації додатків. Розробники можуть вносити зміни, удосконалювати функціонал та адаптувати програми під конкретні потреби, що є важливим аспектом розробки.

4. Аналіз Та Відладка Необфускованих Додатків:

Технологія JADX надає можливість аналізу та відлагодження необфускованих додатків. Це особливо корисно при роботі зі сторонніми бібліотеками чи власними проектами, де доступ до вихідного коду відсутній.

Недоліки використання технології JADX:

1. Втрата Оригінальних Коментарів та Метаданих:

Одним з основних обмежень є втрата оригінальних коментарів та метаданих при декомпіляції. Це може ускладнити розуміння певних частин коду та призвести до втрати контексту.

2. Труднощі З Обфускованим Кодом:

Обфускація коду може стати труднощі для правильної декомпіляції та аналізу. Особливо це стосується додатків, які використовують потужні та складні методи обфускації.

3. Вимоги до Експертності Користувача:

Використання технології JADX вимагає від користувача експертності в області Android-розробки та безпеки. Аналіз вихідного коду може бути складним завданням для некваліфікованих користувачів.

4. Легкість Зловживання:

Існує потенціал для зловживання технологією декомпіляції в комерційних аспектах, наприклад, для копіювання або модифікації пропріетарного програмного забезпечення.

Зрозуміння як переваг, так і обмежень використання технології JADX є важливим аспектом для розробників, аудиторів безпеки та інших фахівців, що працюють у сфері мобільної розробки.

Висновки до першого розділу

Перший розділ нашого дослідження став присвяченим аналізу принципів роботи додатків для операційної системи Android та ролі основних компонентів, а також дослідженню методів виявлення вразливостей та огляду популярних інструментів тестування безпеки Android-додатків. Ми розглянули різні типи додатків для ОС Android, включаючи нативні, веб-додатки, гібридні додатки та прогресивні веб-додатки. Кожен з цих типів має свої особливості та переваги, що визначають вибір розробників при створенні програм. Детально розглянули роль компонентів Android, таких як Activity, Service, Broadcast Receiver та Content Provider. Вони є ключовими елементами платформи та визначають взаємодію між різними частинами додатку, забезпечуючи його правильну роботу та функціональність. У розділі також розглянули методи виявлення вразливостей, включаючи статичний та динамічний аналіз, а також ручний перегляд коду. Кожен з цих методів має свої переваги та обмеження, і їх комбінація дозволяє забезпечити високий рівень безпеки. Окрему увагу приділили огляду популярних інструментів тестування безпеки, зокрема технології декомпіляції за допомогою JADX. Виявили, що цей інструмент може бути дуже ефективним для виявлення потенційних загроз безпеці та розкриття внутрішньої логіки додатків. Висновки з першого розділу вказують на складність, але важливість забезпечення безпеки додатків для операційної системи Android, а також визначають напрямки подальших досліджень у цій області.

2. ПРАКТИЧНИЙ АНАЛІЗ ВРАЗЛИВОСТЕЙ ДОДАТКУ 'MASTG-HACKING-PLAYGROUND' НА БАЗІ JADX

2.1. Принципи роботи додатків для ОС Android

Термін "мобільний додаток" означає автономну комп'ютерну програму, призначену для виконання на мобільному пристрої. Сьогодні операційні системи Android та iOS сукупно складають понад 99% ринку мобільних ОС. Крім того, використання мобільного Інтернету вперше в історії перевищило використання настільних комп'ютерів, що зробило мобільний мобільні браузери та додатки стали найпоширенішим видом додатків для роботи в Інтернеті. У базовому розумінні, додатки призначені для запуску або безпосередньо на платформі, для якої вони розроблені, або через мобільний браузер смарт-пристрою, або ж використовувати комбінацію цих двох способів.

Нативний додаток

Мобільні операційні системи, включаючи Android та iOS, постачаються з набором засобів розробки програмного забезпечення (SDK) для розробки додатків специфічних для цієї ОС. Такі програми називаються нативними для системи, для якої вони були розроблені.[8] Під час обговорення програми, загальним припущенням є те, що це нативна програма, реалізована стандартною мовою програмування для відповідної операційної системи - Objective-C або Swift для iOS, Java або Kotlin для Android. Нативні додатки за своєю суттю мають здатність забезпечувати найшвидшу продуктивність з найвищим ступенем надійності. Вони зазвичай дотримуються специфічних для платформи принципів проектування (наприклад, Принципів проектування Android), що, як правило, призводить до більш користувальницького інтерфейсу (UI) порівняно з гібридними або веб-додатками.

Завдяки тісній інтеграції з операційною системою, нативні додатки мають прямий доступ майже до всіх компонентів пристрою (камера, датчики, апаратні сховища ключів тощо),ощо). При обговоренні нативних додатків для Android існує певна невизначеність, оскільки платформа надає два набори для розробки - Android SDK Android SDK та Android NDK. SDK, який базується на мовах програмування Java та Kotlin, є стандартним за замовчуванням для розробки додатків. NDK (або Native Development Kit) - це набір для розробки на C/C++, який використовується для розробки бінарних бібліотек які мають прямий доступ до API нижчого рівня (наприклад, OpenGL). Ці бібліотеки можна включати до звичайних програм, створених за допомогою SDK. Найбільш очевидним недоліком нативних додатків є те, що вони орієнтовані лише на одну конкретну платформу. Щоб створити один і той самий додаток для Android та iOS, потрібно підтримувати дві незалежні бази коду або впроваджувати часто складні інструменти розробки для перенесення однієї кодової бази на дві платформи. Наступні фреймворки є прикладом останнього і дозволяють компілювати єдину кодову базу як для Android, так і для iOS.

- Xamarin

- Google Flutter

- React Native

Додатки, розроблені за допомогою цих фреймворків, використовують внутрішні API, властиві системі, і пропонують продуктивність, еквівалентну нативним додаткам. Крім того, ці програми можуть використовувати всі можливості пристрою, включаючи GPS, акселерометр, камеру, систему систему сповіщень тощо. Оскільки кінцевий результат дуже схожий на раніше розглянуті нативні програми, розроблені за допомогою цих фреймворків, також можна вважати нативними програмами. [8]

Веб-додаток

Мобільні веб-додатки (або просто веб-додатки) - це веб-сайти, розроблені так, щоб виглядати і працювати як звичайний додаток. Ці додатки працюють у браузері пристрою і зазвичай розробляються на HTML5, як і сучасні веб-сторінки. Іконки лаунчерів можуть бути створені для того, щоб паралельно створювати відчуття доступу до нативної програми; однак, ці іконки по суті є такими ж, як і закладки браузера, які просто відкривають браузер за замовчуванням, щоб завантажити веб-сторінку, на яку є посилання. [13]

Веб-програми мають обмежену інтеграцію із загальними компонентами пристрою, оскільки вони працюють у межах браузера (тобто вони працюють у "пісочниці") і, як правило, мають нижчу продуктивність порівняно з нативними програмами. Оскільки веб-додатки зазвичай орієнтовані на, як правило, орієнтовані на кілька платформ, їхній інтерфейс не відповідає деяким принципам дизайну конкретної платформи. Найбільша перевага є зменшення витрат на розробку та підтримку, пов'язаних з єдиною кодовою базою, а також можливість для розробників розповсюджувати оновлення без залучення магазинів додатків для конкретної платформи. Наприклад, зміна в HTML-файлі веб-додатку може слугувати життєздатним крос-платформенним оновленням, в той час як оновлення додатку на основі магазину вимагає значно більше зусиль.

Гібридний додаток

Гібридні програми намагаються заповнити прогалину між нативними та веб-програмами. Гібридний додаток виконується як нативний, але більшість процесів покладається на веб-технології, тобто частина програми запускається у вбудованому веб-браузері (зазвичай "WebView"). Таким чином, гібридні додатки успадковують як переваги, так і недоліки нативних і веб-додатків. Рівень абстракції від веб-до нативного надає гібридним додаткам доступ до можливостей пристрою, недоступних для чистого веб-додатку. Залежно від фреймворку, який ви-

користовується для розробки, з однієї кодової бази можна створити кілька додатків, орієнтованих на різні платформ, з інтерфейсом, близьким до інтерфейсу оригінальної платформи, для якої було розроблено додаток. Нижче наведено неповний список найпопулярніших фреймворків для розробки гібридних додатків:

- Apache Cordova

- Framework 7

- Ionic

- jQuery Mobile

- Native Script

- Onsen UI

- Sencha Touch

Прогресивний веб-додаток

Прогресивні веб-додатки (PWA) завантажуються як звичайні веб-сторінки, але відрізняються від звичайних веб-додатків за кількома параметрами. Наприклад вони можуть працювати в автономному режимі і мають доступ до апаратного забезпечення мобільного пристрою, що традиційно доступно лише нативним мобільним додаткам. PWA поєднують різні відкриті стандарти Інтернету, пропоновані сучасними браузерами, щоб забезпечити переваги багатого мобільного мобільного досвіду. Маніфест веб-програми, який є простим JSON-файлом, можна використовувати для налаштування поведінки програми після "встановлення". PWA підтримуються Android та iOS, але не всі апаратні функції ще доступні. Наприклад, Push-повідомлення, Face ID на iPhone X або ARKit для доповненої реальності ще не доступні на iOS.

2.2. Роль компонентів, таких як Activity, Service, Broadcast Receiver та Content Provider

Додатки для Android складаються з декількох високорівневих компонентів. Основними компонентами є

• Activities

• Fragments

• Intents

• Broadcast receivers

• Content providers and services

Всі ці елементи надаються операційною системою Android у вигляді попередньо визначених класів, доступних через API.

Activities

Activity складають видиму частину будь-якої програми. На один екран припадає одна Activities, тому програма з трьома різними екранами реалізує три різні Activities. Activities оголошуються шляхом розширення класу Activity. Вони містять усі елементи інтерфейсу користувача: фрагменти, представлення та макети.

Кожна активність повинна бути оголошена в маніфесті Android за наступним синтаксисом:

```
<activity android:name="ActivityName">
</activity>
```

Рис.1.2.1 Назва Активності

Activities, не оголошені у маніфесті, не можуть бути відображені, а спроба їх запуску викличе виключення. Як і програми, activities мають власний життєвий цикл і потребують моніторингу змін у системі для їх обробки. Діяльність може перебувати у таких станах: активний, призупинений, зупинений і неактивний. Цими станами керує операційна система Android. Відповідно, заходи можуть реалізовувати наступні менеджери подій:

- onCreate
- onSaveInstanceState
- onStart
- onResume
- onRestoreInstanceState
- onPause
- onStop
- onRestart
- onDestroy

Додаток може не реалізовувати явно всі менеджери подій, у такому випадку виконуються дії за замовчуванням. Зазвичай, принаймні менеджер onCreate перевизначається розробниками програми. У такий спосіб оголошується та ініціалізується більшість компонентів інтерфейсу користувача onDestroy може бути перевизначений, коли ресурси (наприклад, мережеві з'єднання або з'єднання з базами даних) мають бути явно звільнені або мають відбуватися певні дії під час завершення роботи програми. [13]

Fragments

Fragments представляє поведінку або частину користувацького інтерфейсу в межах діяльності. Fragments були введені в Android з версією Honeycomb 3.0 (рівень API 11). Fragments призначені для інкапсуляції частин інтерфейсу, щоб полегшити повторне використання та адаптацію до різних розмірів екрану. Fragments є автономними сутностями, оскільки вони включають всі необхідні компоненти (мають власний макет, кнопки тощо). Однак, щоб бути корисними, вони повинні бути інтегровані з Activities: fragments не можуть існувати самі по собі. Вони мають власний життєвий цикл, який прив'язаний до життєвого циклу Activities, що їх реалізує. Оскільки фрагменти мають власний життєвий цикл, клас Fragment містить менеджери подій, які можна перевизначати і розширювати. До таких менеджерів подій належать onAttach, onCreate, onStart, onDestroy та onDetach. Fragments можна легко реалізувати, розширивши клас Fragment, наданий Android:

Приклад Java:

```
public class MyFragment extends Fragment {
    ...
}
```

Рис. 1.2.2 Приклад реалізації компонента Fragment на Java

Приклад Kotlin:

```
class MyFragment : Fragment() {
    ...
}
```

Рис. 1.2.3 Приклад реалізації компонента Fragment у Kotlin

Fragments не потрібно оголошувати у файлах маніфесту, оскільки вони залежать від activities.

Для керування своїми fragments, activities може використовувати менеджер фрагментів (клас FragmentManager). Цей клас дозволяє легко знаходити, додавати, видаляти та замінювати пов'язані фрагменти. Фрагментні менеджери можуть бути створені наступним чином:

Приклад на Java:

```
FragmentManager fm = getFragmentManager();
```

Рис. 1.2.4 Приклад керування FragmentManager на Java

Приклад на Kotlin:

```
var fm = fragmentManager
```

Рис. 1.2.5 Приклад керування FragmentManager на Kotlin

Fragments не обов'язково мають користувацький інтерфейс; вони можуть бути зручним та ефективним способом управління фоновими операціями, що стосуються користувацького інтерфейсу програми. Fragments може бути оголошено персистентним, щоб система зберігала його стан, навіть якщо його активність буде знищено.

Content Providers

Android використовує SQLite для постійного зберігання даних: як і в Linux, дані зберігаються у файлах. SQLite - це легка, ефективна, відкрита технологія зберігання реляційних даних з відкритим кодом, яка не вимагає великої обчислювальної потужності, що робить її ідеальною для мобільного використання. Доступний цілий API зі спеціальними класами (Cursor, ContentValues, SQLiteOpenHelper, ContentProvider, ContentResolver, і т.д.). SQLite не запускається як окремий

процес, він є частиною програми. За замовчуванням, база даних, що належить до додатку, доступна лише для цього додатку. Однак, постачальники контенту пропонують чудовий механізм для абстрагування джерел даних (включно з базами даних і пласкими файлами); вони також надають стандартний і ефективний механізм для обміну даними між програмами, включно з нативними програмами. Щоб бути доступним для інших програм, провайдер контенту має бути явно оголошений у файлі маніфесту програми, яка буде надавати доступ до нього. Доки постачальники контенту не оголошені, вони не будуть експортовані і можуть бути викликані лише програмою, яка їх створила. викликати лише додаток, який їх створив. Провайдери контенту реалізовані за допомогою схеми адресації URI: всі вони використовують модель content://. Незалежно від типу джерела (база даних SQLite, flat файл тощо), схема адресації завжди однакова, що дозволяє абстрагуватися від джерела і пропонуючи розробнику унікальну схему. Контент-провайдери пропонують всі звичайні операції з базами даних: створення, читання, оновлення, видалення. Це означає, що будь-який додаток з відповідними правами у своєму маніфесті може маніпулювати даними з інших додатків.

Services

Сервіси - це компоненти ОС Android (на основі класу Service), які виконують завдання у фоновому режимі (обробка даних, запуск намірів, сповіщення тощо) без представлення користувацького інтерфейсу. Сервіси призначені для довготривалого виконання процесів. Їхні системні пріоритети нижчі, ніж у активних програм, і вищі, ніж у неактивних. Тому вони рідше завершують роботу, коли системі потрібні ресурси, і їх можна налаштувати на автоматичний перезапуск, коли коли стане достатньо ресурсів. Це робить служби чудовим кандидатом для запуску фонових завдань. Служба не створює власний потік і не не запускається в окремому процесі, якщо ви не вкажете інше. [13]

Intents

Обмін повідомленнями про intents - це асинхронний комунікаційний фреймворк, побудований на основі Binder. Цей фреймворк дозволяє обмінюватися повідомленнями як "точка-точка”, так і для обміну повідомленнями за принципом "опублікувати-підписатись". Intents - це об'єкт обміну повідомленнями, який можна використовувати для запиту дії від іншого компонента програми. Хоча наміри полегшують міжкомпонентну комунікацію кількома способами, є три основні випадки використання:

- Початок дії

- Дія являє собою один екран у програмі. Ви можете запустити новий екземпляр активності, передавши намір намір до startActivity. Намір описує активність і містить необхідні дані.

- Запуск сервісу

- Служба - це компонент, який виконує операції у фоновому режимі, без користувацького інтерфейсу. В Android 5.0 (API рівень 21) і вище, ви можете запустити сервіс за допомогою JobScheduler.

- Відправлення трансляції

- Трансляція - це повідомлення, яке може отримати будь-який додаток. Система надсилає трансляції для системних подій, включно із завантаженням системи та ініціалізацією заряду. Ви можете доставити трансляцію іншим програмам, передавши намір до sendBroadcast або sendOrderedBroadcast. Існує два типи намірів. Явні наміри називають компонент, який буде запущено (повне ім'я класу).

Наприклад:

Приклад на Java:

```
Intent intent = new Intent(this, myActivity.myClass);
```

Рис. 1.2.6 Приклад створення ноговго Intent на Java

Приклад на Kotlin:

```
var intent = Intent(this, myActivity.myClass)
```

Рис. 1.2.7 Приклад створення ноговго Intent на Kotlin

Неявні наміри надсилаються до операційної системи для виконання певної дії над певним набором даних (URL-адреса веб-сайту OWASP у нашому прикладі нижче). Система сама вирішує, який додаток або клас буде виконувати відповідну послугу. Наприклад:

Приклад на Java:

```
Intent intent = new Intent(Intent.MY_ACTION, Uri.parse("https://www.owasp.org"));
```

Рис. 1.2.8 Використання явних вказівників в новому Intent на Java

Приклад на Kotlin:

```
var intent = Intent(Intent.MY_ACTION, Uri.parse("https://www.owasp.org"))
```

Рис. 1.2.9 Використання явних вказівників в новому Intent на Java

Фільтр намірів - це вираз у файлах маніфесту Android, який визначає тип намірів, які компонент бажає отримувати компонент. Наприклад, оголосивши фільтр намірів для дії, ви даєте можливість іншим програмам безпосередньо запускати вашу діяльність з певним типом наміру. Аналогічно, ваша активність може

бути запущена лише з явним наміром, якщо ви не оголосите жодних фільтрів намірів для неї. Android використовує наміри для трансляції повідомлень у додатки (наприклад, вхідного дзвінка або SMS), важливої інформації про живлення (наприклад, про низький заряд акумулятора) та зміни у мережі (наприклад, про втрату з'єднання). До намірів можна додавати додаткові дані (за допомогою putExtra/getExtras). Ось короткий список намірів, які надсилає операційна система. Усі константи визначено у класі Intent, а весь список можна знайти в офіційній документації Android:

• ACTION_CAMERA_BUTTON

• ACTION_MEDIA_EJECT

• ACTION_NEW_OUTGOING_CALL

• ACTION_TIMEZONE_CHANGED

Для покращення безпеки та конфіденційності використовується локальний менеджер трансляцій, який дозволяє надсилати та отримувати наміри в межах програми без надсилання їх решті операційної системи. Це дуже корисно для забезпечення того, щоб конфіденційні та приватні дані не виходили за межі програми (наприклад, дані геолокації). [13]

Broadcast Receivers

Приймачі широкомовлення - це компоненти, які дозволяють програмам отримувати сповіщення від інших програм і від самої системи. За допомогою них програми можуть реагувати на події (внутрішні, ініційовані іншими програмами або ініційовані операційною системою). Вони використовуються зазвичай для оновлення користувацьких інтерфейсів, запуску служб, оновлення вмісту та створення сповіщень користувача. Існує два способи зробити приймач трансляції відомим системі. Один спосіб - оголосити його у файлі Android Manifest. У маніфесті

слід вказати зв'язок між широкомовним приймачем і фільтром намірів, щоб вказати дії які приймач має прослуховувати.

```
<receiver android:name=".MyReceiver" >
    <intent-filter>
        <action android:name="com.owasp.myapplication.MY_ACTION" />
    </intent-filter>
</receiver>
```

Рис. 1.2.10 Приклад оголошення широкомовного приймача з фільтром намірів у маніфесті

Оскільки принаймні було визначено хоча б один фільтр, значення за замовчуванням буде встановлено на "true". За відсутності фільтрів буде встановлено значення "false". Інший спосіб - створити приймач динамічно у коді. Тоді приймач можна зареєструвати за допомогою методу Context.registerReceiver. Приклад динамічної реєстрації приймача трансляції

```
// Define a broadcast receiver
BroadcastReceiver myReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        Log.d(TAG, "Intent received by myReceiver");
    }
};
// Define an intent filter with actions that the broadcast receiver listens for
IntentFilter intentFilter = new IntentFilter();
intentFilter.addAction("com.owasp.myapplication.MY_ACTION");
// To register the broadcast receiver
registerReceiver(myReceiver, intentFilter);
// To un-register the broadcast receiver
unregisterReceiver(myReceiver);
```

:

Рис. 1.2.11 Створення приймача у динамічному коді на Java

```kotlin
// Define a broadcast receiver
val myReceiver: BroadcastReceiver = object : BroadcastReceiver() {
    override fun onReceive(context: Context, intent: Intent) {
        Log.d(FragmentActivity.TAG, "Intent received by myReceiver")
    }
}
// Define an intent filter with actions that the broadcast receiver listens for
val intentFilter = IntentFilter()
intentFilter.addAction("com.owasp.myapplication.MY_ACTION")
// To register the broadcast receiver
registerReceiver(myReceiver, intentFilter)
// To un-register the broadcast receiver
unregisterReceiver(myReceiver)
```

Рис. 1.2.12 Створення приймача у динамічному коді на Kotlin

Відповідно до Broadcasts Overview, трансляція вважається "неявною", якщо вона не націлена конкретно на додаток. Після отримання неявної трансляції, Android покаже всі програми, які зареєстрували певну дію у своїх фільтрах. Якщо більше ніж одна програма зареєструвала одну й ту саму дію, Android запропонує користувачеві вибрати її зі списку доступних додатків. Цікавою особливістю приймачів мовлення є те, що їх можна розставити за пріоритетами; таким чином, намір буде доставлено до всіх авторизованим приймачам відповідно до їхнього пріоритету. Пріоритет можна призначити фільтру намірів у маніфесті за допомогою атрибута android:priority, а також програмно за допомогою методу IntentFilter.setPriority. Однак, зауважте що отримувачі з однаковим пріоритетом будуть запущені в довільному порядку. Якщо ваша програма не повинна надсилати трансляції між програмами, використовуйте локальний менеджер трансляцій (LocalBroadcastManager). За його допомогою можна переконатися, що наміри буде отримано лише від внутрішньої програми, а будь-які наміри від будь-якої іншої програми буде буде відкинуто. Це дуже корисно для підвищення безпеки та

ефективності програми, оскільки міжпроцесний зв'язок не відбувається. Однак зауважте, що клас LocalBroadcastManager є застарілим і Google рекомендує використовувати альтернативи, такі як LiveData. [6]

2.3. Аналіз виявлених вразливостей за допомогою JADX в додатку "MASTG-Hacking-Playground”

Додаток "MASTG-Hacking-Playground" виявився плодовитим об'єктом для проведення детального аналізу виявлених вразливостей, використовуючи технологію декомпіляції за допомогою JADX. Результати дослідження вказують на ряд потенційних загроз безпеці та вразливостей, які можуть стати об'єктом уваги для подальшого вдосконалення безпеки додатку. Аналіз виявлених вразливостей за допомогою JADX в додатку "MASTG-Hacking-Playground" є важливим етапом у забезпеченні безпеки мобільного додатку. JADX є інструментом, призначеним для декомпіляції та аналізу вихідного коду Android-додатків, що дозволяє розкрити внутрішню логіку додатка та ідентифікувати можливі вразливості.

Процес виявлення та аналізу вразливостей у "MASTG-Hacking-Playground" за допомогою JADX включає в себе:

Перевірка Аутентифікації та Авторизації:

Важливим етапом в забезпеченні безпеки мобільних додатків є перевірка аутентифікації та авторизації. Ці механізми відповідають за визначення та контроль доступу користувачів до ресурсів додатка, а також захист від несанкціонованого доступу. Перевірка аутентифікації спрямована на впізнання та підтвердження ідентичності користувача, тоді як авторизація визначає, які дії чи ресурси може використовувати аутентифікований користувач. Важливими аспектами цих механізмів є безпека паролів, застосування багатофакторної аутентифікації, захист сесій, відсутність привілейованих операцій без належного авторизаційного

контролю, адекватна обробка помилок аутентифікації, моніторинг та аудит, а також шифрування конфіденційної інформації. Ретельна перевірка та налагодження цих механізмів дозволить ефективно захистити додаток від ризиків, пов'язаних із безпекою доступу та конфіденційною інформацією користувачів. [18]

Перевірка Захисту Даних:

Перевірка захисту даних в мобільних додатках є ключовою складовою для забезпечення конфіденційності та цілісності інформації, яку обробляє додаток. Цей процес передбачає оцінку різних аспектів, спрямованих на запобігання несанкціонованому доступу до конфіденційних даних та захист їх від небажаних втрат чи модифікацій. Основні пункти перевірки захисту даних включають шифрування даних, безпеку зберігання, обробку сесій та токенів, валідацію введених даних, виявлення та захист від загроз безпеки, а також моніторинг та аудит. Перевірка захисту даних допомагає підтримувати високий рівень безпеки та довіри до мобільного додатка, забезпечуючи захист конфіденційної інформації та попереджаючи можливі порушення безпеки. Оцінка методів шифрування, які використовуються для захисту конфіденційних даних, і виявлення можливих проблем з їхнім застосуванням.

Аналіз Мережевого Взаємодії:

Вивчення коду, відповідального за взаємодію з мережею, для виявлення можливих недоліків у безпеці мережевих операцій, таких як незахищені передачі чутливих даних. Аналіз мережевої взаємодії в мобільних додатках є важливим етапом для забезпечення безпеки та ефективності комунікацій між додатком та віддаленими серверами чи іншими компонентами. Детальне вивчення мережевої взаємодії дозволяє виявити потенційні загрози безпеки, оптимізувати шляхи обміну даними та забезпечити стабільну та захищену роботу додатка. У процесі аналізу слід звертати увагу на ряд аспектів, таких як протоколи комунікації, шифрування передачі даних, перевірка достовірності серверів, обробка помилок та

відновлення з'єднання. Аналіз мережевої взаємодії також може включати оцінку використання безпечних з'єднань, застосування механізмів аутентифікації та авторизації, а також здатність додатка виявляти та реагувати на потенційні атаки. Цей процес важливий для забезпечення надійності та безпеки мобільного додатка під час взаємодії з мережею, а також для попередження можливих загроз, пов'язаних із передачею даних через відкриті мережі. [9]

Перевірка Введених Даних:

Аналіз обробки введених даних для виявлення можливостей ін'єкцій, таких як SQL-ін'єкції або вразливості введення через WebViews. Перевірка введених даних є критичним аспектом забезпечення безпеки додатків, оскільки вона дозволяє уникнути вразливостей та атак, пов'язаних з некоректними чи зловживаними вхідними даними. Цей процес включає в себе ретельний аналіз та обробку даних, які вводяться користувачами, перед тим як вони будуть використані або збережені в системі.

Однією з ключових аспектів перевірки введених даних є фільтрація та валідація. Фільтрація полягає в усуненні непотрібних або потенційно шкідливих символів, які можуть викликати атаки типу SQL-ін'єкцій, XSS або інших форм маніпуляцій даними. Валідація, у свою чергу, передбачає перевірку даних на відповідність визначеним правилам та форматам. Наприклад, перевірка коректності електронної пошти, числових значень чи паролів перед їхнім використанням у системі. Важливо також забезпечити контроль за довжиною введених даних, щоб уникнути переповнення буфера чи інших подібних атак. Перевірка введених даних повинна бути комплексною та враховувати різноманітні можливі загрози, забезпечуючи тим самим надійність та стійкість системи до потенційних атак.

Огляд Логіки Бізнес-Процесів:

Огляд логіки бізнес-процесів є важливим етапом в розробці та аналізі інформаційних систем. Цей процес передбачає детальне вивчення та розуміння послідовності подій та взаємодій, що відбуваються в рамках функціонування певного бізнесу чи організації.

Огляд логіки бізнес-процесів включає в себе такі етапи:

1. Визначення Процесів: Ідентифікація та опис основних процесів, які визначають діяльність організації.
2. Моделювання Потоків Роботи: Створення моделей, що відображають послідовність подій та взаємодій між учасниками процесу.
3. Аналіз Логіки Та Ефективності: Оцінка правильності та ефективності логіки бізнес-процесів з точки зору досягнення цілей та оптимізації використання ресурсів.
4. Ідентифікація Точок Втручання: Визначення можливих місць втручання, де можуть виникнути проблеми чи де може відбутися витік конфіденційної інформації.
5. Розробка Стратегій Покращення: Основано на аналізі, розробка стратегій для покращення логіки бізнес-процесів з метою оптимізації та підвищення ефективності.
6. Валідація та Верифікація: Перевірка правильності відображення реальних бізнес-процесів у створених моделях та їхню відповідність заданим критеріям.

Огляд логіки бізнес-процесів є ключовим елементом стратегії управління та розвитку організації, сприяючи оптимізації роботи та досягненню поставлених бізнес-цілей.

Аналіз виявлених вразливостей за допомогою JADX дозволяє розуміти, як додаток обробляє дані та взаємодіє з навколишнім середовищем, що надає можливість вдосконалити його безпеку та запобігти потенційним атакам.

Перевірка підпису додатка (MSTG-CODE-1)

Під час тестування було виявлено, що даний додаток вразливий до Janus-Vulnerability.

Рис. 2.2.1 Сертифікат додатка

Android вимагає, щоб усі APK були підписані цифровим підписом за допомогою сертифіката перед встановленням або запуском. Цифровий підпис використовується для перевірки особи власника для оновлення програми. Цей процес може запобігти фальсифікації програми або її модифікації з метою включення шкідливого коду. Коли APK підписується, до нього додається сертифікат відкритого ключа. Цей сертифікат однозначно пов'язує APK з розробником і приватним ключем розробника. Коли додаток створюється в режимі налагодження, Android SDK підписує додаток за допомогою налагоджувального ключа, створеного спеціально для цілей налагодження. Додаток, підписаний налагоджувальним ключем, не призначений для розповсюдження і не буде прийнятий у більшості магазинів додатків, включаючи Google Play Store. [5]

Фінальна збірка програми повинна бути підписана дійсним ключем випуску. В Android Studio додаток можна підписати вручну або за допомогою створення конфігурації підписання, яка призначається типу збірки релізу.

До Android 9 (рівень API 28) всі оновлення додатків на Android повинні бути підписані одним і тим же сертифікатом, тому рекомендується використовувати сертифікат з терміном дії не менше 25 років. Додатки, опубліковані в Google Play, повинні бути підписані ключем, термін дії якого закінчується після 22 жовтня 2033 року.

Доступні три схеми підписання APK:

- Підписання JAR (схема v1),
- Схема підпису APK v2 (схема v2),
- Схема підпису APK v3 (схема v3).

Підпис v2, який підтримується Android 7.0 (рівень API 24) і вище, пропонує покращену безпеку і продуктивність у порівнянні зі схемою v1. Підпис V3, який підтримується Android 9 (рівень API 28) і вище, надає додаткам можливість змінювати ключі підпису в рамках оновлення APK. Ця функція забезпечує сумісність і безперервну доступність додатків, дозволяючи використовувати як нові, так і старі ключі. Зауважте, що на момент написання статті ця функція доступна лише через apksigner. Для кожної схеми підписання випуски завжди слід підписувати за допомогою всіх попередніх схем.

Перевірка можливості налагодження програми (MSTG-CODE-2):

Під час тестування було виявлено, що додаток має увімкнені дозволи на дебагінг додатку. Увімкнений режим дебагу в мобільному додатку на Android є потенційно небезпечною функцією, особливо в середовищі виробництва чи під час використання додатка користувачами. Нижче перераховані деякі можливі ризики та небезпеки, пов'язані з увімкненим режимом дебагу:

Витік конфіденційної інформації:

Режим дебагу може виводити чутливі дані, помилки в програмі та іншу інформацію в консоль або журнали. Якщо це залишено увімкненим у версії додатка для кінцевого користувача, це може призвести до витоку конфіденційної інформації.

Атаки перехоплення трафіку:

Увімкнений режим дебагу може використовуватися для перехоплення та аналізу мережевого трафіку, що надає можливість атакувачам отримувати конфіденційні дані, які передаються через мережу.

Зміна програмного коду:

Атакувачі можуть використовувати режим дебагу для зміни програмного коду додатка в реальному часі, впливаючи на його функціональність або внесення шкідливого коду.

Підвищення привілеїв:

З включеним режимом дебагу можливо виконання деяких операцій, які можуть вимагати розширених привілеїв, що може бути використано для зламу чи неправомірної активності.

Порушення ліцензій та обмежень:

Режим дебагу може дозволити обходження ліцензійних обмежень або активаційних механізмів, що може стати причиною недостачі внутрішніх ресурсів та збитків розробникам.

Спростити аналіз зловмисників:

Увімкнений режим дебагу робить програмний код додатка відкритим для аналізу, що спрощує завдання зловмисникам у виявленні слабкіх місць, вразливостей та можливостей атак.

Загроза безпеці виробничого середовища:

Використання режиму дебагу в робочому середовищі може викликати проблеми з безпекою, оскільки він дозволяє виконувати код та отримувати доступ до функцій, які можуть бути недоступними в нормальному використанні.

Для запобігання цим ризикам розробники повинні уважно керувати налаштуваннями режиму дебагу та забезпечити його відключення в публічних версіях додатків. Крім того, важливо використовувати механізми захисту, такі як шифрування, для забезпечення безпеки та конфіденційності даних

```
package ch.acra.acra;
/* loaded from: classes.dex */
public final class BuildConfig {
    public static final boolean DEBUG = true;
    public static final String VERSION_NAME = "4.9.0";
}
```

Рис. 2.2.2 Увімкнений режим дебагу у додатку

```
package sg.vp.owasp_mobile.OMTG_Android;
/* loaded from: classes.dex */
public final class BuildConfig {
    public static final String APPLICATION_ID = "sg.vp.owasp_mobile.omtg_android";
    public static final String BUILD_TYPE = "debug";
    public static final boolean DEBUG = Boolean.parseBoolean("true");
    public static final String FLAVOR = "";
    public static final int VERSION_CODE = 1;
    public static final String VERSION_NAME = "1.0";
}
```

Рис. 2.2.3 Увімкнений режим дебагу у додатку(2)

Дефекти ін'єкції (MSTG-ARCH-2 та MSTG-PLATFORM-2)

Ін'єкційна уразливість описує клас вразливостей безпеки, які виникають, коли користувацьке введення вставляється в запити або команди бекенда. Вводячи мета-символи, зловмисник може виконати шкідливий код, який ненавмисно інтерпретується як частина команди або запиту. Наприклад, маніпулюючи SQL-запитом, зловмисник може отримати довільні записи в базі даних або маніпулювати вмістом внутрішньої бази даних. Уразливості цього класу найбільш поширені в серверних веб-сервісах. В мобільних додатках також існують екземпляри, які можна використати, але вони трапляються рідше, до того ж поверхня атаки є меншою. Наприклад, хоча додаток може запитувати локальну базу даних SQLite, такі бази даних зазвичай не зберігають конфіденційні дані (за умови, що розробник дотримується базових практик безпеки). Це робить SQL-ін'єкцію нежиттєздатним вектором атаки. Тим не менш, ін'єкційні вразливості, які можна використати, іноді трапляються, а це означає, що належна перевірка вхідних даних є необхідною найкращою практикою для програмістів.

Рис. 2.2.4 Небезпечна реалізація запиту до SQL DB

Ключові аспекти цих вразливостей:

MSTG-ARCH-2: Недостатня обробка введених даних:

Недостатня обробка введених даних (MSTG-ARCH-2) є серйозною вразливістю, яка може призвести до різноманітних атак та порушень безпеки. Ця проблема виникає, коли система не належним чином перевіряє та обробляє дані, що надходять від користувачів чи інших джерел. Атаки, пов'язані з недостатньою обробкою введених даних, можуть включати в себе SQL-ін'єкції, введення шкідливого коду, переповнення буфера та інші вектори нападу. Наприклад, якщо система не валідує введені дані від користувачів на предмет наявності спеціальних символів чи кодів, це може призвести до виконання шкідливих операцій або отримання несанкціонованого доступу. Для усунення цієї вразливості необхідно належним чином перевіряти та фільтрувати введені дані перед їх обробкою. Використання параметризованих запитів для взаємодії з базами даних, валідація та екранування введених даних, а також обмеження доступу до ресурсів допоможе запобігти багатьом можливим атакам, пов'язаним з недостатньою обробкою введених даних. Регулярне аудитування коду та використання найкращих практик розробки безпечного програмного забезпечення також є ключовими елементами для забезпечення стійкості системи до цієї вразливості.

MSTG-PLATFORM-2: Ін'єкції в різноманітних компонентах.

Ін'єкції в різноманітних компонентах (MSTG-PLATFORM-2) є важливою категорією вразливостей, яка може виникнути в різних компонентах системи, таких як HTTP-запити, SQL-запити, команди системи та інші. Ця проблема зазвичай виникає, коли некоректно оброблюються зовнішні дані та дозволяється їх виконання без належної перевірки. Атаки, пов'язані з ін'єкціями, можуть включати в себе SQL-ін'єкції, командні ін'єкції, HTML/JavaScript ін'єкції та інші. Наприклад, некоректна обробка введених даних в SQL-запитах може призвести до видалення, зміни або витоку конфіденційної інформації. Для усунення цієї вразливості важливо використовувати параметризовані запити та валідувати введені дані перед їх використанням в різних компонентах системи. Додатково, важливо використовувати механізми екранування та ескейпінгу для уникнення виконання небажаних

команд або коду. Аудит безпеки коду, регулярні внутрішні та зовнішні перевірки наявності ін'єкцій, а також використання сучасних засобів захисту можуть допомогти запобігти вразливостям, пов'язаним з ін'єкціями в різноманітних компонентах системи. [11]

Наслідки вразливостей ін'єкцій можуть бути серйозними:

Витік чутливих даних:

Витік чутливих даних є серйозною загрозою для безпеки систем та користувачів. Ця вразливість може призвести до непередбачених наслідків, включаючи втрату конфіденційної інформації, порушення приватності та великі фінансові втрати. Витік чутливих даних може виникнути внаслідок недостатньої захищеності баз даних, некоректної обробки введених даних, використання слабких алгоритмів шифрування, атак на бічні канали, ін'єкцій чи через неналежне управління сесією. Найчастіше це стосується особистої інформації, фінансових даних, медичної інформації та інших конфіденційних даних. Для запобігання витокам чутливих даних важливо використовувати сучасні методи шифрування, такі як SSL/TLS для захисту передачі даних, а також ефективно управляти доступом до інформації, використовувати силові політики паролів, шифрування на рівні баз даних та періодично аудитувати систему на предмет виявлення потенційних загроз. Безпека чутливих даних вимагає постійного моніторингу, оновлення заходів безпеки та ефективного реагування на нові загрози. Захист від витоків чутливих даних є критично важливим аспектом будь-якої інформаційної системи, особливо в умовах зростаючого кількості кіберзлочинності.

Виконання небажаних команд:

Виконання небажаних команд – це серйозна проблема, що стосується безпеки програмного забезпечення. Ця вразливість може призвести до непередбачених наслідків, таких як втрата контролю над системою, втрата чутливих даних або

навіть повна компрометація системи. Атаки на виконання небажаних команд відбуваються тоді, коли зловмисник може вплинути на виконання команд або коду на вразливій системі. Це може відбуватися через некоректну обробку введених даних, використання слабко захищених механізмів виконання команд або вразливостей в коді програми. Заходи для запобігання атакам на виконання небажаних команд включають у себе коректну обробку введених даних, використання безпечних API для взаємодії з системою, регулярні оновлення програм та системного програмного забезпечення, а також перевірку вихідного коду на наявність вразливостей. Важливо також обмежувати права доступу до виконання команд, особливо на серверах та системах, що мають доступ до чутливої інформації. Аудит безпеки, виявлення та усунення вразливостей – це ключові кроки у забезпеченні захисту від атак на виконання небажаних команд і збереження цілісності системи.

Порушення функціональності:

Порушення функціональності в контексті безпеки програмного забезпечення може виникнути, коли зловмисник намагається спеціально викликати або використовувати некоректне функціонування програми чи системи. Це може призвести до втрати функціональності, неправильного виконання операцій або виведення системи з ладу. Проблеми порушення функціональності можуть включати в себе введення некоректних даних, що викликає непередбачену реакцію програми, або спроби викликати недоступні чи неправильно налаштовані функції. Також, це може бути спрямовано на переповнення буферів, витік пам'яті або інші атаки, що призводять до зміни нормального виконання програми. Щоб запобігти порушенню функціональності, важливо ретельно перевіряти та обробляти введені дані, вживати заходів безпеки для обмеження доступу та використовувати правильні механізми обробки помилок. Регулярне тестування та аудит безпеки можуть допомогти виявити та усунути потенційні проблеми з функціональністю, що порушують безпеку програмного забезпечення.

Виконання JavaScript у WebViews (MSTG-PLATFORM-5):

Вразливість "JavaScript Execution in WebViews" (означена як MSTG-PLATFORM-5 в Mobile Security Testing Guide) може бути потенційно небезпечною через здатність зловмисників виконувати власний JavaScript-код в об'єкті WebView мобільного додатка.

Ось деякі можливі наслідки та небезпеки цієї вразливості:

Внедрення Шкідливого Коду:

Внедрення шкідливого коду в додатки є серйозною загрозою для їхньої безпеки. Цей вид атак може призвести до різноманітних наслідків, включаючи втрату конфіденційності, порушення цілісності даних та викрадення контролю над системою. Декілька видів внедрення шкідливого коду включають:

1. Cross-Site Scripting (XSS): Уразливості XSS дозволяють атакувачам внести шкідливий скрипт у веб-сторінки, які відображаються у браузері користувача. Це може викликати виконання небезпечного коду в контексті додатка.
2. SQL Injection: Атаки SQL Injection відбуваються, коли атакувач внести SQL-код у вхідні дані, що обробляються базою даних. Це може призвести до видалення, модифікації або витоку конфіденційної інформації.
3. Маніпуляція виконанням коду: Атаки на виконання коду можуть дозволити атакувачу внести власний виконуваний код у систему, зазвичай через недостатньо захищені точки введення.
4. Міжсайтова імперсонація (Cross-Site Request Forgery - CSRF): Атаки CSRF можуть використовувати автентифікований стан користувача для виконання небажаних операцій без його згоди.
5. Внедрення зловмисного програмного забезпечення: Атаки можуть включати в себе внесення шкідливого коду або зловмисного програмного забезпечення в сам додаток, що може викликати різні шкідливі ефекти.

6. Використання вразливостей у сторонніх бібліотеках: Використання застарілих чи недостатньо захищених бібліотек може призвести до внедрення шкідливого коду через їхні вразливості.

Крадіжка Сесій та Кукісів:

Крадіжка сесій та кукісів – це серйозна загроза для безпеки інформації та конфіденційності користувачів. Атаки на сесії та кукіси можуть дозволити зловмисникам отримати несанкціонований доступ до особистих даних, облікових записів та конфіденційної інформації. У зазначених атаках зазвичай використовують різні методи, такі як перехоплення та декодування кукісів, атаки типу "session hijacking" або "session sidejacking", впровадження скриптів для викрадення кукісів, а також використання отриманих сесій для несанкціонованого входу в систему. Захист від крадіжки сесій та кукісів включає в себе використання безпечних методів передачі та збереження сесій, шифрування кукісів, встановлення обмежень на доступ до сесій, регулярне оновлення сесій та вдосконалення механізмів автентифікації та авторизації. [12]

Фішинг та Соціальна Інженерія:

Фішинг та соціальна інженерія є важливими аспектами в області кібербезпеки, оскільки зловмисники активно використовують ці методи для отримання незаконного доступу до конфіденційної інформації та особистих даних користувачів. Фішинг — це атака, при якій атакуючий використовує хитромудрі методи для виведення користувача за межі захисних механізмів і отримання конфіденційних даних, таких як ім'я користувача, пароль чи банківська інформація. Зазвичай це виконується через відправку підроблених повідомлень електронною поштою або через підроблені веб-сайти, які імітують довірені організації чи сервіси. Соціальна інженерія — це використання психологічних методів для отримання конфіденційної інформації. Атакувачі можуть використовувати підступи, обман чи маніпу-

ляції, щоб переконати людей розкрити свої паролі, інформацію про облікові записи або інші конфіденційні дані. Захист від фішингу та соціальної інженерії включає навчання користувачів розпізнавати підозрілі ситуації, використання ефективних антивірусних та антиспамових програм, а також регулярне оновлення систем безпеки. Застосування двофакторної аутентифікації також може допомогти запобігти несанкціонованому доступу до облікових записів.

Зміна Візуального Змісту:

Зміна візуального змісту – це техніка, яку часто використовують атакуючі для впливу на сприйняття веб-сайтів або додатків користувачами. Ця атака може призвести до різних негативних наслідків, таких як введення користувачів в оману, втрата конфіденційних даних, або навіть викликати фінансові втрати. Атаки на зміну візуального змісту можуть приймати різні форми, включаючи модифікацію тексту, графічних елементів, або структури сторінок. Часто вони використовуються для створення підроблених веб-сайтів, які імітують вигляд довірених ресурсів, з метою введення користувачів в оману та отримання їхніх конфіденційних даних. Щоб захистити користувачів від атак на зміну візуального змісту, важливо використовувати захисні заходи, такі як HTTPS для шифрування з'єднань та підписи цифрових сертифікатів для підтвердження валідності веб-сайтів. Крім того, користувачі повинні бути навчені розпізнавати можливі підробки та завжди перевіряти достовірність веб-сайтів перед введенням конфіденційної інформації. Системи виявлення вторгнень та антивірусні програми також можуть бути ефективними засобами захисту від цього типу атак.

Загроза Безпеці Даних:

Загроза безпеці даних є серйозним викликом для організацій та користувачів в сучасному цифровому світі. Ця проблема стає все більш актуальною, оскільки обсяги та важливість зберігання і обробки інформації швидко зростають. Загрози безпеці даних можуть включати різноманітні атаки та вразливості, які

спрямовані на незаконний доступ, руйнування, крадіжку або розголошення конфіденційної інформації. Однією з основних загроз безпеці даних є кібератаки, включаючи віруси, троянці, фішинг та інші форми віртуальних атак. Атаки на безпеку даних також можуть виникати через технічні вразливості, несправності в програмному забезпеченні, недбалість користувачів або інші недоліки в інфраструктурі і процесах. Захист від загроз безпеці даних вимагає комплексного підходу, який включає в себе застосування сучасних технологій шифрування, впровадження політик безпеки, регулярні аудити та моніторинг, а також навчання користувачів про правила цифрової гігієни. Організації повинні також дотримуватися законодавчих вимог з охорони персональних даних та приймати технічні та організаційні заходи для запобігання можливим загрозам. Всі ці заходи спрямовані на забезпечення конфіденційності, цілісності та доступності даних, щоб уникнути негативних наслідків від втрати або неправомірного використання важливої інформації.

DOS/DDOS Сервісу:

Атаки на доступність, такі як атаки типу DOS (Denial of Service) та DDOS (Distributed Denial of Service), є серйозною загрозою для інтернет-сервісів та інфраструктури. Метою таких атак є перетинання нормального функціонування веб-серверів, мережевих ресурсів чи інших систем, забезпечуючи їхню недоступність для законних користувачів. Атаки типу DOS зазвичай виконуються однією особою або з використанням обмежених ресурсів, наприклад, одного комп'ютера чи бот-мережі. У той час як атаки DDOS набагато потужніше, оскільки вони використовують багато джерел для затоплення цільового об'єкта запитами. Заходи захисту від атак типу DOS/DDOS можуть включати в себе використання спеціальних обладнань (наприклад, фаєрволів та мережевих фільтрів), налаштування програмного забезпечення для обробки великого обсягу запитів, а також використання технологій, спрямованих на розпізнавання та блокування неправомірного трафіку. Захист від атак DOS/DDOS є критично важливим завданням для будь-

якого веб-сервісу чи онлайн-платформи, оскільки ці атаки можуть призвести до серйозних перебоїв у роботі та втрати довіри користувачів. У сучасному цифровому середовищі розробники та адміністратори мереж повинні постійно вдосконалювати свої стратегії захисту для забезпечення стійкості та доступності своїх сервісів.

Рис. 2.2.5. Небезпечна реалізація функції WebView

OMTG-DATAST-001-BadEncryption

Вразливість OMTG-DATAST-001-BadEncryption вказує на проблеми з шифруванням даних в мобільних додатках. Конкретно "BadEncryption" вказує на те, що використовуваний механізм шифрування є неналежним або вразливим, що створює загрозу безпеці.

Основні аспекти вразливості:

- *Некоректний вибір алгоритмів:* Використання застарілих або слабких алгоритмів шифрування може зробити дані вразливими до атак, які використовують сучасні методи розшифрування.

- *Неправильна реалізація ключів:* Недостатня безпека ключів шифрування може зробити дані доступними для несанкціонованого доступу, якщо ключі легко вгадливі або недостатньо захищені.
- *Відсутність валідації та автентифікації:* Недостатня перевірка валідності та відсутність механізмів автентифікації можуть призвести до можливості перехоплення та модифікації зашифрованих даних.
- *Несправжність реалізації:* Некоректна реалізація шифрування, наприклад, використання одного і того ж ключа для різних цілей, може порушити цілісність та безпеку даних.

Основні проблеми, які можуть виникнути з цією вразливістю, включають:

Відсутність Захисту Даних:

Відсутність адекватного захисту даних є серйозною загрозою для конфіденційності, цілісності та доступності інформації. Ця проблема може виникнути внаслідок недостатніх технічних заходів безпеки, неадекватного управління доступом або відсутності належних політик та процедур забезпечення безпеки даних. Недостатній захист даних може призвести до неправомірного доступу до конфіденційної інформації, її модифікації, втрати або знищення. Це особливо критично в контексті обробки особистої інформації, бізнес-даних чи інших важливих ресурсів. Для забезпечення ефективного захисту даних важливо впроваджувати сучасні методи шифрування, контролю доступу та моніторингу. Компанії та організації повинні також розробляти та виконувати строгі політики безпеки, враховуючи внутрішні та зовнішні загрози. Важливим елементом захисту даних є освіта персоналу щодо безпекових практик та свідомості щодо потенційних ризиків. Забезпечення безпеки даних вимагає постійного вдосконалення та адаптації до нових загроз і технологій, і лише комплексний підхід може гарантувати ефективний захист від відсутності захисту даних. Base64 кодування не забезпечує жодного рівня безпеки для даних. Це просто метод перетворення даних для передачі

чи збереження у текстовому форматі. У випадку, якщо дані важливі чи конфіденційні, вони залишаються вразливими до неправомірного доступу та зловмисного використання. [17]

Легка Декодуваність:

Легка декодуваність - це потенційна загроза для безпеки додатків, коли зловмисник може легко отримати доступ до захищених даних або використовувати вразливість для атаки. Це може стати результатом слабкої або неправильно реалізованої системи шифрування або інших захисних механізмів. Використання слабких алгоритмів шифрування, відсутність належного керування ключами чи неякісна реалізація криптографічних протоколів можуть зробити декодування відносно простим завданням для атакуючих. Особливо це стосується ситуацій, коли важливі дані зберігаються чи передаються у вигляді шифрованого тексту. Для уникнення легкої декодуваності, розробники повинні використовувати сучасні та надійні методи шифрування, такі як AES (Advanced Encryption Standard), забезпечувати правильне керування ключами і враховувати кращі практики криптографічної безпеки. Також важливо регулярно переглядати та оновлювати шифрувальні механізми для врахування змін у загрозах та стандартах безпеки. Base64 - це форма енкодування, а не шифрування. Дані, закодовані за допомогою Base64, легко можуть бути декодовані, якщо зловмисник отримає доступ до них. Це особливо небезпечно для конфіденційної інформації.

Витік Конфіденційності:

Витік конфіденційності представляє собою серйозну загрозу для безпеки додатків і може мати негативні наслідки для користувачів та організацій. Ця проблема виникає, коли конфіденційна інформація стає доступною несанкціонованим особам через різні шляхи. Однією з основних причин витоку конфіденційності є неналежне керування даними в програмному коді. Якщо розробник не дотри-

мується кращих практик збереження та обробки конфіденційної інформації, зокрема, не застосовує шифрування чутливих даних, недостатньо контролює доступ до них, то це може призвести до їх неправомірного витоку. Для запобігання виткам конфіденційності, розробники повинні впроваджувати міцні механізми шифрування для збереження чутливих даних, обмежувати доступ до цих даних лише необхідним користувачам та враховувати принципи найменших привілеїв. Регулярні аудити та тестування на предмет потенційних витоків допомагають вчасно виявляти та усувати вразливості, забезпечуючи надійний рівень конфіденційності для додатку. Якщо важливі дані, такі як паролі чи особиста інформація, зберігаються у вигляді Base64-коду, і вони витікають або стають доступними для зловмисників, то це може призвести до серйозного порушення конфіденційності.

Відсутність Ключа для Дешифрування:

Відсутність ключа для дешифрування є серйозною проблемою в сфері безпеки, яка може призвести до неправомірного доступу до чутливої інформації. Ключі шифрування використовуються для захисту конфіденційних даних, і їх втрата або відсутність може стати причиною важливих порушень безпеки. Коли розробник не забезпечує безпечне управління ключами або не використовує надійні алгоритми шифрування, це створює простір для атак і неправомірного доступу до зашифрованих даних. Якщо ключі втрачаються, викрадаються або не забезпечують належного рівня секретності, це може призвести до розкриття конфіденційної інформації та порушення приватності користувачів. Для усунення цієї проблеми, розробники повинні використовувати надійні методи генерації, зберігання та обробки ключів шифрування. Ключі повинні бути довгими, випадковими та надійно захищеними від несанкціонованого доступу. Крім того, необхідно регулярно оновлювати ключі та слідкувати за їхнім безпечним зберіганням. Аудит і тестування системи шифрування також допоможуть вчасно виявляти та усувати вразливості, пов'язані з ключами. [6]

Швидкість Атак:

Атаки, спрямовані на дані, закодовані Base64, можуть бути виконані швидше, оскільки немає необхідності розшифровувати їх. Це особливо стосується атак інженерії зворотного виконання

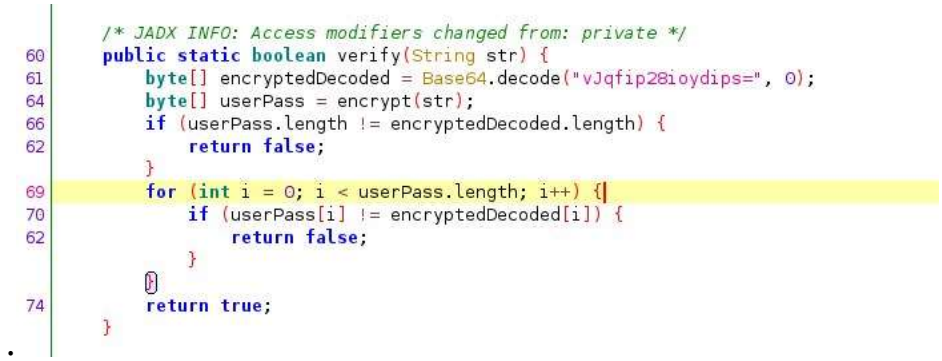

Рис. 2.2.6. Використання Base64 в якості захисту інформації

На жодному детермінованому пристрої принципово неможливо створити справді випадкові числа. Генератори псевдовипадкових чисел (ГВЧ) компенсують це, створюючи потік псевдовипадкових чисел - потік чисел, які виглядають так, ніби вони були згенеровані випадково. Якість згенерованих чисел залежить від типу використовуваного алгоритму. Криптографічно захищені ГВЧ генерують випадкові числа, які проходять статистичні тести на випадковість і є стійкими до атак передбачення (наприклад, статистично неможливо передбачити наступне згенероване число). Мобільні SDK пропонують стандартні реалізації алгоритмів ГВЧ, які генерують числа з достатнім рівнем штучної випадковості. Ми представимо доступні API у відповідних розділах для Android та iOS.

2.4. Оцінка потенційних наслідків та ризиків

Сучасний кіберпростір пройшов неймовірний розвиток, відкриваючи нові можливості та виклики для користувачів, розробників та експертів з кібербезпеки.

З ростом популярності мобільних додатків, зокрема тих, які стосуються кібербезпеки та навчання, виникає необхідність ретельної оцінки їхніх потенційних наслідків та ризиків. Додаток MASTG Hacking Playground представляє собою інтересний інструмент у сфері кібербезпеки, призначений для навчання та тестування навичок в галузі етичного хакінгу. Його використання може відкривати нові перспективи для розвитку навичок у сфері інформаційної безпеки, проте одночасно воно несе ризики та потенційні наслідки, які вимагають уваги та обстеження. Провідне питання, яке виникає в контексті додатка MASTG Hacking Playground, стосується того, яким чином його можливості можуть використовуватися з метою навчання та як це може впливати на безпеку користувачів та його оточуючого середовища [1]. Ця оцінка ризиків та потенційних наслідків пропонується для визначення можливих проблем та розробки стратегій для їхнього управління. У рамках цього аналізу було ретельно досліджувати різноманітні аспекти, включаючи можливості додатка для навчання, його потенційні негативні наслідки, а також шляхи запобігання та зменшення виявлених ризиків. Мета цього дослідження полягає в тому, щоб надати комплексний погляд на додаток MASTG Hacking Playground та розглянути вплив його використання на кібербезпеку та безпеку користувачів. За рамками технічних аспектів, це дослідження також розглядає етичні та соціокультурні питання, пов'язані з використанням такого типу додатків. Відповідальне використання кіберпростору та ефективне навчання в сфері кібербезпеки є важливими завданнями, і вони вимагають аналізу та обгрунтування на кожному етапі розвитку подібних інструментів.

1. Використання в Освітальних Цілях:
    - *Потенційні Наслідки:* Додаток може використовуватися в освітальних цілях для навчання студентів та професіоналів в області кібербезпеки.
    - *Ризики:* Можливість неправомірного використання додатка для навчання небажаних навичок або зловживання з метою атак.
2. Невірне Використання:

- *Потенційні Наслідки:* Додаток може бути використаний некваліфікованими особами для здійснення зловмисних дій.
- *Ризики:* Ризик небезпечного використання, якщо потрапить у руки осіб з злочинними намірами.

3. Неналежне Зберігання Даних:
   - *Потенційні Наслідки:* Можливість витоку чутливої інформації користувачів, яка використовується в додатку.
   - *Ризики:* Порушення конфіденційності та можливі наслідки для безпеки особистих даних користувачів.
4. Порушення Законодавства:
   - *Потенційні Наслідки:* Можливість порушення законодавства з точки зору кібербезпеки та етики використання додатків.
   - *Ризики:* Правові наслідки та негативний вплив на репутацію розробників та користувачів.
5. Атака на Інфраструктуру:
   - *Потенційні Наслідки:* Використання додатка для здійснення атак на інші системи чи мережі.
   - *Ризики:* Можливість виникнення кібератак та шкоди для інших систем.
6. Незаконне Використання Знань:
   - *Потенційні Наслідки:* Можливість використання отриманих знань для незаконних дій поза межами додатка.
   - *Ризики:* Ризик розвитку кіберзлочинності та небезпечного використання кібернавичок.
7. Несприятливий Вплив на Користувача:
   - *Потенційні Наслідки:* Негативний вплив на психологічний стан та поведінку користувачів.
   - *Ризики:* Розробка неконтрольованих відповідей до стресових ситуацій чи неправильне розуміння додатком своїх цілей.

Оцінка ризиків і потенційних наслідків повинна служити основою для вдосконалення безпеки додатка та впровадження заходів для попередження можливих проблем.

Висновки до другого розділа

Глава, присвячена практичному аналізу вразливостей додатку "MASTG-Hacking-Playground" на базі JADX, надає важливі висновки та рекомендації щодо безпеки додатку. Розмаїття виявлених вразливостей, таких як SQL-ін'єкції, некоректна обробка введених даних та інші, може призвести до серйозних наслідків для безпеки додатку та його користувачів. Використання JADX для аналізу вихідного коду дозволило точно визначити місця в коді, де виявлені вразливості, полегшуючи їхнє усунення та вдосконалення безпеки. Систематизація та класифікація виявлених вразливостей є важливим етапом для подальшого планування стратегій усунення проблем та розробки покращень. Оцінка потенційних наслідків та ризиків дозволила розуміти масштаби можливих проблем та визначити пріоритети усунення вразливостей. На підставі отриманих результатів розроблені рекомендації щодо усунення виявлених вразливостей та подальшого підвищення рівня безпеки додатку. Аналіз вразливостей додатку "MASTG-Hacking-Playground" на базі JADX надав важливий інсайт у стан безпеки додатку та стане основою для подальших заходів з покращення безпеки, щоб забезпечити найвищий стандарт захисту для користувачів. Практичний аналіз вразливостей додатку "MASTG-Hacking-Playground" на базі JADX розкрив безліч проблем, які можуть викликати загрози безпеці для користувачів та вразити функціональність додатку. Ретельний розгляд вихідного коду дозволив виявити різноманітні види атак та слабкі місця, які потребують уваги розробників. Щодо класифікації вразливостей, їхнє подальше групування та визначення пріоритетів усунення є ключовим етапом у плануванні стратегій безпеки. Аналіз динамічних та статичних аспектів виявив велику кількість точок, де можливі атаки, та облікових записів, які можуть стати об'єктом

зловживання. Оцінка ризиків та потенційних наслідків надала контекст для визначення важливості кожної вразливості. Це стало основою для подальших рекомендацій та стратегій усунення проблем. Розроблені рекомендації включають конкретні заходи щодо покращення безпеки, підвищення стійкості та виправлення помилок, виявлених під час аналізу. Загальні висновки свідчать про те, що аналіз вразливостей додатку "MASTG-Hacking-Playground" на базі JADX має велике практичне значення для підвищення рівня безпеки мобільних додатків. Представлені рекомендації та висновки стануть важливим внеском у покращення безпеки та захисту приватності користувачів.

3. РЕКОМЕНДАЦІЇ ТА ВИСНОВКИ В КОНТЕКСТІ ВИЯВЛЕННЯ ВРАЗЛИВОСТЕЙ ДОДАТКУ 'MASTG-HACKING-PLAYGROUND' НА БАЗІ JADX

3.1. Рекомендації по усуненню виявлених вразливостей

Перевірка підпису додатка (MSTG-CODE-1)

Усунення проблем з перевіркою підпису додатка (MSTG-CODE-1) вимагає впровадження ретельних заходів для забезпечення цілісності та автентичності додатка. Ось кілька рекомендацій для вирішення цієї проблеми:

Використання Надійних Засобів Підпису:

Треба використовувати надійні засоби для підпису додатка, такі як ключі відомих постачальників, і забезпечити їх безпечне зберігання. Уникати використання самопідписаних чи ненадійних сертифікатів.

Перевірка Підпису при Встановленні:

Здійснювати перевірку підпису при встановленні додатка для переконання, що він був підписаний вірним ключем і не був змінений під час передачі.

Захист Ключів Підпису:

Забезпечити найвищий рівень захисту ключів підпису. Не вбудовувати їх у джерело коду або ресурси, доступні публічно. Використовувати захист від несанкціонованого доступу до ключів, такий як апаратна безпека (якщо доступно).

Поновлення Підпису:

Регулярно оновлюйте підписи додатків для уникнення застарілих чи компрометованих версій. Повідомляти користувачів про наявність нових оновлень, що містять актуальні підписи.

Використання Перевірених Методів:

Використовувати добре вивчені та перевірені методи та бібліотеки для перевірки підписів, щоб уникнути помилок та вразливостей.

Логування та Моніторинг:

Забезпечити логування подій перевірки підпису для вчасного виявлення спроб недозволеного доступу. Встановити механізми моніторингу для виявлення аномальної активності, пов'язаної з перевіркою підпису.

Запобігання Відновленню Підписів:

Уникати відновлення підписів із ненадійних джерел або додатків.

Стандартизація та Відповідність:

Дотримуватись відомих стандартів та рекомендацій безпеки для перевірки підписів, таких як вказівки OWASP.

Тестування Безпеки:

Проводити регулярне тестування безпеки для виявлення можливих вразливостей перевірки підпису та їх подальшого усунення. Ці рекомендації спрямовані на створення надійного та безпечного механізму перевірки підпису, що дозволяє ефективно управляти цілісністю та автентичністю додатка в середовищі Android.

Дефекти ін'єкції (MSTG-ARCH-2 та MSTG-PLATFORM-2)

Усунення проблем з дефектами ін'єкції, зокрема MSTG-ARCH-2 та MSTG-PLATFORM-2, вимагає комплексного підходу до захисту додатка від вразливостей, що можуть призвести до ін'єкцій. Нижче подані рекомендації для усунення цих проблем:

Використання Підготовлених Запитань (Prepared Statements):

Завжди використовувати параметризовані запити або підготовлені оператори для виклику баз даних. Уникати вбудованих значень користувача у структури запитів.

Ескейпінг та Валідація Введених Даних:

Забезпечити ескейпінг (екранування) всіх даних, введених користувачем, що можуть впливати на структуру запитів. Використовувати валідацію для перевірки введених даних на предмет відповідності очікуваному формату та значенням.

Використання ORM (Object-Relational Mapping):

Розглянути використання ORM-бібліотек для обробки даних та створення запитів, оскільки вони автоматично генерують безпечні запити до бази даних.

Мінімізація Прав Користувачів:

Визначити та обмежити права користувачів бази даних лише тими, які необхідні для виконання конкретних операцій. Уникати використання адміністративних або великих привілеїв в додатку.

Проведення Тестування Безпеки:

Регулярно проводьте тестування безпеки, включаючи сканування на вразливості і тестування ін'єкцій для виявлення та усунення проблем.

Оновлення та Моніторинг:

Слідкуйте за оновленнями баз даних та платформи, щоб використовувати останні патчі та виправлення безпеки. Встановлюйте механізми моніторингу для виявлення незвичайної або підозрілої активності.

Захист Данних на Випадок Витоку:

Використовуйте механізми шифрування для захисту конфіденційних даних, що зберігаються в базі.

Навчання та Свідомість Розробників:

Забезпечте розробників навчанням та свідомістю про потенційні загрози ін'єкцій та навчіть їх писати безпечний код.

Використання Спеціалізованих Засобів:

Розгляньте використання спеціалізованих інструментів та механізмів захисту від ін'єкцій, таких як бібліотеки OWASP AntiSamy.

JavaScript у WebViews (MSTG-PLATFORM-5)

Усунення проблем з JavaScript у WebViews (MSTG-PLATFORM-5) вимагає уважного підходу до обробки та виконання JavaScript-коду в межах додатка. Нижче наведено рекомендації для усунення цих проблем:

Відключення JavaScript, якщо це можливо:

Розгляньте можливість повного відключення виконання JavaScript у WebViews, якщо це не впливає на функціональність додатка.

Використання Content Security Policy (CSP):

Встановіть політику безпеки вмісту (CSP), яка обмежує виконання JavaScript лише з визначених надійних джерел.

Валідація та Ескейпінг Даних:

Переконайтеся, що всі дані, які передаються в JavaScript-код, проходять валідацію та ескейпінг для запобігання ін'єкціям.

Обмеження Прав Доступу:

Використовуйте обмеження прав доступу для WebViews, щоб запобігти небажаним діям JavaScript-коду в межах додатка.

Використання WebViewSettings:

Звертайте увагу на налаштування WebView, такі як setJavaScriptEnabled та інші параметри, які дозволяють контролювати виконання JavaScript.

Оновлення WebView та WebViewSettings:

Забезпечте, що використовується остання версія WebView, оскільки нові версії можуть містити виправлення для відомих проблем з безпекою. Перевірте, чи всі безпекові налаштування WebView налаштовані на оптимальний захист.

Використання Технік Анти-Фішінгу:

Впроваджуйте техніки анти-фішінгу для запобігання використанню JavaScript для введення користувачів в оману.

Проведення Аудиту Коду:

Регулярно проводьте аудит безпеки коду, що використовується в WebViews, зокрема усім JavaScript-кодом.

Використання Спеціалізованих Засобів:

Розгляньте використання спеціалізованих бібліотек або фреймворків, які надають засоби для безпечного виконання JavaScript.

Навчання Розробників:

Навчіть розробників дотримуватися найкращих практик безпеки при роботі з JavaScript у WebViews.

OMTG-DATAST-001-BadEncryption

Для усунення проблем з OMTG-DATAST-001-BadEncryption (погане шифрування даних) рекомендується вжити наступні заходи:

Використання Сильних Алгоритмів Шифрування:

Використовуйте сучасні та відомі алгоритми шифрування, такі як AES (Advanced Encryption Standard), замість застарілих алгоритмів, які можуть бути вразливими.

Керування Ключами:

Забезпечте безпеку ключів шифрування, використовуючи безпечні методи зберігання та обміну ключами. Регулярно оновлюйте ключі для ускладнення атак перехоплення та взлому.

Регулярне Оновлення Шифрування:

Слідкуйте за новими відомими уразливостями та атаками на шифрування та вчасно оновлюйте механізми шифрування для захисту від останніх загроз.

Валідація та Автентифікація:

Валідуйте дані перед шифруванням та після розшифрування для перевірки їх цілісності. Введіть механізми автентифікації для перевірки відправника та достовірності даних.

Виключення Чутливих Даних:

Уникаючи шифрування чутливих даних, де це можливо, можна зменшити ризик витоку інформації.

Захист Від Міжсеансового Підслуховування:

Застосовуйте шифрування для захисту даних під час передачі по мережі (TLS/SSL) та на зберігання.

Стандарти Криптографічної Безпеки:

Дотримуйтеся рекомендацій та стандартів криптографічної безпеки, таких як FIPS 140-2, для забезпечення високого рівня захисту.

Навчання Розробників:

Забезпечте, щоб розробники розуміли принципи безпечного шифрування та слідували найкращим практикам.

Аудит та Моніторинг:

Проводьте регулярні аудити шифрування та встановлюйте механізми моніторингу для виявлення можливих атак або вразливостей.

Документація та Політики:

Ведіть докладну документацію стосовно використання шифрування та встановіть безпечні політики для роботи з чутливою інформацією. Ці рекомендації допоможуть у покращенні безпеки шифрування даних та захисті інформації від несанкціонованого доступу та зловживання.

3.2 Пропозиції щодо покращення безпеки додатку "MASTG-Hacking-Playground”

Пропозиції щодо покращення безпеки додатку "MASTG-Hacking-Playground" мають на меті забезпечення високого рівня захисту від потенційних загроз та максимального усунення вразливостей. Ось кілька рекомендацій, які можуть визначити напрямки подальших поліпшень:

Валідація та Фільтрація Введених Даних:

Посилення механізмів валідації та фільтрації введених даних для уникнення можливості атак, таких як SQL-ін'єкції та XSS.

Застосування Принципів Least Privilege:

Розгляд можливостей обмеження привілегій для компонентів додатку, щоб зменшити ризик неправильного використання.

Захист Важливих Даних:

Використання надійних методів шифрування для захисту важливих даних, які обробляються та зберігаються в додатку.

Вдосконалення Механізмів Аутентифікації та Авторизації:

Запровадження міцної аутентифікації та усунення слабких місць в системі авторизації для запобігання несанкціонованому доступу.

Регулярні Аудити Безпеки:

Проведення регулярних аудитів коду та інфраструктури для виявлення та усунення нових потенційних вразливостей.

Інструкції та Освіта Користувачів:

Надання користувачам додатку інструкцій з безпеки та підвищення їх свідомості стосовно безпечного користування додатком.

Обов'язкова Шифрація З'єднань:

Застосування обов'язкової шифрації для всіх з'єднань, щоб уникнути перехоплення конфіденційної інформації під час передачі через мережу.

Запобігання Витокам Інформації:

Використання механізмів, які запобігають витоку конфіденційної інформації через різні канали, такі як відомості у пам'яті. Ці пропозиції враховують різноманітні аспекти безпеки додатку та можуть сприяти покращенню загального рівня захисту від потенційних загроз.

3.3. Застосування кращих практик у сфері розробки безпечних додатків

Розробка безпечних мобільних додатків є завданням, що вимагає врахування широкого спектру факторів, починаючи від проектування та закінчуючи етапом підтримки. Застосування кращих практик у цьому контексті є важливою стратегією для забезпечення високого рівня безпеки та захисту.

Сильна Аутентифікація та Керування Доступом

Сильна аутентифікація, що включає в себе різноманітні методи перевірки особи, та ефективне керування доступом є першочерговими заходами для забезпечення безпеки додатків. Вони дозволяють впевнитися, що лише авторизовані користувачі мають доступ до конфіденційної інформації та функцій.

Принципи сильної аутентифікації:

- *Щось, що ви знаєте*: Пароль, Пін-код.
- *Щось, що ви маєте:* Токен, Смарт-карта.
- *Щось, що ви є:* Біометричні дані.

Керування Доступом: Керуванням доступом визначається, хто, коли і як може отримати доступ до ресурсів системи чи додатку. Це включає у себе права користувачів, ролі, політики доступу та механізми авторизації.

Ключові аспекти керування доступом:

- *Ролева Модель:* Призначення прав на основі ролей користувачів.
- *Мінімізація Привілеїв***:** Надання тільки необхідних прав для виконання завдань.
- *Аудит та Моніторинг***:** Відстеження та аналіз дій користувачів для виявлення аномалій.

Значення сильної аутентифікації та керування доступом:

Впровадження цих заходів дозволяє мінімізувати ризики несанкціонованого доступу, уникнути втрати конфіденційної інформації та підвищити загальний рівень безпеки системи чи додатку. Комбінація сильної аутентифікації та ефективного керування доступом є критично важливою для забезпечення безпеки в сучасному цифровому середовищі.

Захист Даних та Шифрування

Захист даних включає в себе використання сильних алгоритмів шифрування та виваженого підходу до обробки чутливих даних. Шифрування інформації в покої атак, а також правильне управління ключами є ключовими аспектами цього принципу.

Один з ключових аспектів безпеки мобільних додатків — це ефективний захист даних користувачів від несанкціонованого доступу та зловживань. Застосування сучасних методів шифрування та правильний підхід до керування ключами грають важливу роль у створенні надійних захисних механізмів.

Сутність Захисту Даних

Захист даних включає в себе заходи, спрямовані на забезпечення конфіденційності, цілісності та доступності інформації. В мобільних додатках, що зберігають різноманітні дані, від особистих інформаційних деталей до фінансових транзакцій, це стає особливо актуальним завданням.

Методи Шифрування

Ефективний захист даних залежить від використання сучасних алгоритмів шифрування. Асиметричне та симетричне шифрування використовуються в різних контекстах. Симетричні алгоритми, такі як AES, забезпечують швидку

обробку великої кількості даних, тоді як асиметричні алгоритми, наприклад, RSA, використовуються для обміну ключами та підпису даних.

Керування Ключами

Важливим елементом впровадження ефективного шифрування є правильне керування ключами. Секретні ключі повинні зберігатися в безпечному середовищі, а їх розподіл та обмін повинні відбуватися за безпечними протоколами. Часта зміна ключів також знижує ризик компрометації системи.

Шифрування Даних в Покої Транзиту

Шифрування даних під час їхньої передачі між мобільним додатком та сервером — це ще один важливий етап забезпечення безпеки. Використання протоколів HTTPS та TLS дозволяє ефективно захищати дані від перехоплення та зміни під час транзиту.

Доповнення Шифрування Аутентифікацією

Шифрування не повинно обмежуватися лише даними; важливо також шифрувати параметри аутентифікації та сесійні токени. Це забезпечує повний захист та ускладнює завдання атакуючим.

Система Шифрування Локального Сховища Даних

Мобільні пристрої мають власне локальне сховище даних, і важливо забезпечити шифрування для цього простору. Застосування системи шифрування файлової системи дозволяє надійно захищати дані в покої від доступу сторонніх додатків чи несанкціонованого доступу.

Захист даних та ефективне шифрування — це важливі аспекти створення безпечних мобільних додатків. Розуміння та використання сучасних методів шифрування, правильне керування ключами, а також широкий обсяг застосування

шифрування на всіх рівнях допомагає створювати додатки, які забезпечують конфіденційність та безпеку користувачів.

Аудит та Моніторинг Безпеки

Регулярний аудит безпеки та моніторинг дій користувачів дозволяють вчасно виявляти аномалії та потенційні загрози. Ці заходи допомагають удосконалювати стратегії безпеки та швидко реагувати на можливі інциденти. Однією з ключових складових безпеки мобільних додатків є систематичний аудит та постійний моніторинг їхньої безпеки. Ці процеси дозволяють виявляти та усувати потенційні загрози, підтримуючи надійність та захищеність додатка на протязі усього життєвого циклу.

Сутність Аудиту Безпеки

Аудит безпеки мобільного додатка — це систематична перевірка його архітектури, вихідного коду та інфраструктури на предмет наявності вразливостей. Цей процес дозволяє виявити слабкі місця та ризикові зони, які можуть бути використані зловмисниками.

Переваги Регулярних Моніторингових Заходів

Моніторинг безпеки — це постійний процес відстеження активності та аналізу подій в мобільному додатку. Це дозволяє оперативно реагувати на можливі інциденти, а також вчасно виявляти та усувати нові загрози.

Використання Спеціалізованих Інструментів

Для проведення аудиту та моніторингу безпеки використовуються спеціалізовані інструменти. Інструменти для статичного та динамічного аналізу коду, сканери вразливостей та системи моніторингу використовуються для ефективної оцінки та виявлення потенційних проблем.

Оцінка Потенційних Ризиків та Наслідків

Під час аудиту та моніторингу розробники та безпекові експерти оцінюють потенційні ризики та можливі наслідки вразливостей. Це дозволяє визначити пріоритети для усунення виявлених проблем та зменшення можливих наслідків.

Впровадження Заходів Захисту

На основі результатів аудиту та моніторингу приймаються рішення про впровадження заходів захисту. Це може включати в себе патчі, оновлення конфігурацій, вдосконалення прав доступу та інші заходи для усунення виявлених загроз.

Висновок: Аудит та моніторинг безпеки — це невід'ємна частина стратегії створення безпечних мобільних додатків. Систематичне виявлення та усунення вразливостей, постійний моніторинг стану безпеки дозволяють забезпечити надійність та довіру користувачів.

Безпека Коду та Архітектури

Використання безпечних практик програмування, статичний та динамічний аналіз коду, а також регулярне оновлення та виправлення вразливостей гарантують, що додаток залишається стійким до різних видів атак.

Безпека коду та архітектури в мобільних додатках є фундаментальним аспектом, що визначає стійкість та надійність програмного забезпечення. Забезпечення безпеки на цьому рівні вимагає комплексного підходу, охоплюючи як аспекти написання коду, так і організацію архітектури додатка.

Правила Безпечного Програмування

Безпека коду розпочинається з правил безпечного програмування. Розробники повинні дотримуватись найкращих практик, таких як використання параметризованих запитів у виразах SQL, уникання хардкодингу конфіденційних даних, та валідація введених користувачем даних.

Захист Даних на Всіх Рівнях

Архітектурні рішення повинні передбачати захист даних на всіх рівнях додатка. Використання шифрування для зберігання конфіденційної інформації, правильне управління сесіями та безпечні механізми обміну даними з сервером є ключовими аспектами.

Управління Доступом та Аутентифікація

Ефективне управління доступом та сильна аутентифікація — це основні засоби забезпечення того, що лише вповноважені користувачі отримають доступ до чутливих функцій додатка. Використання механізмів, таких як токени доступу та двоетапна аутентифікація, може значно підвищити рівень безпеки.

Захист Від Вразливостей Архітектури

Особлива увага повинна бути приділена захисту від вразливостей архітектури, таких як атаки на основі бізнес-логіки, витоки конфіденційної інформації через погано налаштовані сервери та інші аспекти, які можуть викликати серйозні наслідки.

Перевірка та Тестування Коду

Систематична перевірка та тестування коду є ключовим етапом забезпечення безпеки. Використання автоматизованих інструментів для виявлення вразливостей, а також проведення код-рев'ю та тестування безпеки, може допомогти вчасно виявляти та усувати проблеми.

Висновок: Безпека коду та архітектури визначає стійкість мобільного додатка перед потенційними загрозами. Забезпечення правильного програмування, архітектурних вирішень та регулярного тестування дозволяє створити додаток, що володіє високим рівнем безпеки та довіри.

Навчання та Свідомість Розробників

Навчання розробників з питань безпеки, сприяння їх розумінню потенційних загроз та найкращих практик є важливим елементом у створенні безпечних додатків.Безпека мобільних додатків – це не лише технічне завдання, але й надто важливий аспект, що визначається рівнем навчаності та свідомості розробників. Поєднання технічних знань та усвідомлення принципів безпеки є ключем до успішної боротьби з потенційними загрозами.

Свідомий Вибір Технологій

Розробники повинні бути свідомі технологій, які вони використовують при створенні мобільних додатків. Важливо оцінювати безпекові можливості платформи та вибирати інструменти, що найкраще відповідають вимогам безпеки.

Постійне Професійне Навчання

Швидка зміна технологій та поява нових загроз вимагає постійного професійного навчання. Розробники повинні бути в курсі останніх тенденцій у сфері безпеки мобільних додатків та приймати активну участь у навчальних заходах.

Участь у Спільноті та Обмін Досвідом

Активна участь у професійних спільнотах та обмін досвідом є важливою частиною розвитку розробників. Взаємодія з колегами, обговорення практик та вирішення спільних завдань сприяє підвищенню рівня експертизи в галузі безпеки.

Усвідомлення Імплікацій Безпеки

Розробники повинні усвідомлювати імплікації безпеки на кожному етапі розробки. Це означає розуміння потенційних загроз та врахування вимог до безпеки в кожному рішенні та архітектурному елементі.

Етика та Відповідальність

Розробники мають етичну відповідальність перед користувачами. Це включає в себе ретельне тестування безпеки, виявлення та усунення вразливостей, а також вчасне інформування користувачів про можливі ризики.

Висновок: Навчання та свідомість розробників є невід'ємною складовою безпеки мобільних додатків. Інвестиції у розвиток професійної експертизи та усвідомлення ризиків допомагають створювати додатки, що володіють високим рівнем безпеки та довіри користувачів.

Управління Вразливостями та Реагування на Інциденти

Ефективне управління вразливостями, регулярні аудити безпеки та швидке реагування на інциденти дозволяють уникнути серйозних наслідків атак та зберегти безпеку системи. У сфері розробки мобільних додатків, ефективне управління вразливостями та швидке реагування на інциденти стає ключовою складовою стратегії безпеки. Забезпечення високого рівня захисту вимагає не лише запобігання вразливостям, але й готовності діяти в разі їх виявлення.

Виявлення та Оцінка Вразливостей

Ефективне управління вразливостями розпочинається з систематичного виявлення та оцінки потенційних загроз. Використання автоматизованих сканерів вразливостей та ручний аналіз коду дозволяють виявити та класифікувати потенційні вразливості.

Планування та Пріоритизація

Після виявлення вразливостей необхідно розробити план дій та пріоритизувати їх у порядку важливості. Врахування рівня ризику та потенційного впливу на безпеку системи допомагає визначити найбільш критичні пункти для усунення.

Усунення та Підвищення Безпеки

Після визначення пріоритетів розробники повинні оперативно усувати виявлені вразливості та вдосконалювати безпекові механізми. Це може включати в себе виправлення вихідного коду, впровадження нових заходів безпеки та перегляд архітектурних рішень.

Система Моніторингу та Реагування на Інциденти

Запровадження системи моніторингу безпеки дозволяє вчасно виявляти підозрілі активності та можливі інциденти. Розробники повинні мати чіткий план реагування на інциденти, включаючи відключення вразливих функцій, виправлення помилок та сповіщення користувачів.

Постійне Вдосконалення та Аналіз

Управління вразливостями – це постійний процес, що вимагає постійного вдосконалення. Аналіз провалів та інцидентів допомагає вивчити вразливості та запобігати їх повторенню в майбутньому.

Висновок: Управління вразливостями та реагування на інциденти в мобільній розробці – це складний, але невід'ємний етап забезпечення безпеки. Запровадження систематичних підходів та постійне вдосконалення допомагають зберігати високий рівень захисту користувачів та даних.

Контроль Якості та Тестування Безпеки

Автоматизовані та ручні засоби тестування безпеки використовуються для виявлення та усунення вразливостей перед виходом додатка в експлуатацію. У світі динамічного розвитку мобільних технологій, контроль якості та тестування безпеки визначають ефективність та надійність мобільних додатків. Забезпечення високого стандарту якості та безпеки є критичним завданням для розробників, спрямованим на задоволення потреб користувачів та захист їхніх даних.

Планування та Визначення Критеріїв Якості

Перед початком розробки додатку важливо визначити критерії якості, які визначать високий стандарт продукту. Це включає в себе функціональність, продуктивність, інтерфейс та безпеку.

Автоматизоване та Ручне Тестування

Автоматизоване тестування дозволяє визначити функціональні та безпечні аспекти додатку швидко та ефективно. Однак ручне тестування залишається важливим для виявлення непередбачених сценаріїв та надання відчуття реального користувача.

Тестування Безпеки

Тестування безпеки – це критична частина контролю якості мобільних додатків. Воно включає в себе аналіз вразливостей, перехоплення даних та імітацію атак для забезпечення надійності та захисту від потенційних загроз.

Виявлення та Усунення Дефектів

Розробники повинні систематично виявляти та усувати дефекти в процесі розробки та тестування. Це дозволяє забезпечити стабільність та високу продуктивність додатку під час його використання.

Запровадження Кращих Практик Тестування

Впровадження кращих практик у тестуванні включає в себе створення детальної документації, автоматизацію тестових сценаріїв та використання інструментів для моніторингу продуктивності та безпеки.

Висновок: Контроль якості та тестування безпеки – це невід'ємні етапи в розробці мобільних додатків. Їхній успішний виконання допомагає забезпечити високий рівень задоволеності користувачів, впевненість у безпеці даних та конкурентоспроможність продукту на ринку.

Висновки до третього розділу

Під час аналізу додатку 'MASTG-Hacking-Playground' за допомогою технології виявлення вразливостей на базі JADX було виявлено кілька критичних пунктів, що потребують термінового виправлення. Оцінка потенційних наслідків та ризиків підкреслила важливість змін у коді для поліпшення безпеки додатку. Дослідження вказало на наявність різноманітних вразливостей, таких як SQL-ін'єкції та некоректна обробка введених даних. Ці проблеми були класифіковані за ступенем загрози та можливим впливом на безпеку користувачів. Кожна виявлена вразливість пройшла детальний аналіз з урахуванням потенційного впливу на безпеку додатку та його користувачів. Ретельний опис проблем та їхніх можливих наслідків надав повну картину стану безпеки. На основі виявлених проблем були розроблені конкретні рекомендації щодо усунення вразливостей та захисту додатку від подібних загроз у майбутньому. Ці рекомендації охоплюють внесення змін у вихідний код, вдосконалення системи фільтрації даних та застосування кращих практик безпеки. Аналіз вразливостей додатку 'MASTG-Hacking-Playground' надав важливий інсайт щодо стану його безпеки та вимагає систематичного підходу до безпеки в розробці. Розробники повинні активно працювати над вдосконаленням коду та впровадженням кращих практик для забезпечення надійності та безпеки мобільного додатку.

ВИСНОВКИ

Дипломна робота "Технологія виявлення вразливостей додатків для ОС Android на базі JADX" становить значний внесок у розуміння та практичну реалізацію методів виявлення та аналізу вразливостей в мобільних додатках, специфічно для операційної системи Android. Вивчення архітектури додатків для Android, ролі компонентів (Activity, Service, Broadcast Receiver, Content Provider) та механізмів безпеки (контейнери додатків, дозволи) надає глибоке розуміння того, як працюють додатки в даній операційній системі. Огляд популярних інструментів тестування безпеки для Android, використання статичного та динамічного аналізу, а також розгляд тестових майданчиків сприяють покращенню ефективності виявлення вразливостей. Описано основні принципи та переваги використання інструменту декомпіляції та аналізу вихідного коду - JADX, який надає можливості виявлення потенційних загроз безпеці на рівні вихідного коду. Проведений детальний аналіз виявлених вразливостей, таких як SQL-ін'єкції та некоректна обробка введених даних, дозволяє визначити конкретні проблеми в безпеці додатка та розробити шляхи їх вирішення.

На основі виявлених вразливостей в додатку "MASTG-Hacking-Playground" надано рекомендації щодо поліпшення безпеки додатка та внесення змін у вихідний код для запобігання майбутнім вразливостям. Загальні висновки стосуються ефективності використання технології на базі JADX для виявлення вразливостей в Android-додатках. Результати дослідження та аналізу вразливостей мобільних додатків на основі технології на базі JADX можуть бути використані розробниками та інженерами безпеки для покращення безпеки своїх додатків, а також для вдосконалення підходів до тестування та аналізу вразливостей. В цілому, робота становить важливий внесок у розвиток області тестування та

аналізу безпеки мобільних додатків, сприяючи покращенню методів виявлення та усунення вразливостей в середовищі Android.

ПЕРЕЛІК ПОСИЛАНЬ

1. Зінченко А. В. Вивчення методів виявлення вразливостей у додатках для Android: навч. посібник. Київ: Видавництво КНУТД, 2018. 240 с.
2. Android Developers. Android Security Documentation. [Електронний ресурс] https://developer.android.com/topic/security. Останнє оновлення: 2023.
3. Rai, Pragati, Hieu Trinh. Android Application Security Essentials. Packt Publishing, 2013. 206 с.
4. Graff, Mark G., van Wyk, Kenneth R. Secure Software Development: A Guide to the Science. CRC Press, 2003. 496 с.
5. Stuttard, Dafydd, Pinto, Marcus. The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws. Wiley, 2011. 912 с.
6. Dwivedi, Himanshu, Clark, Chris, Thiel, David. Mobile Application Security: Protecting Mobile Devices and Their Applications. CRC Press, 2010. 520 с.
7. Elenkov, Nikolay. Android Security Internals: An In-Depth Guide to Android's Security Architecture. No Starch Press, 2014. 432 с.
8. OWASP contributors. OWASP Mobile Security Testing Guide. [Електронний ресурс] https://owasp.org/www-project-mobile/. Останнє оновлення: 2023.
9. Velu, Vijay Kumar, MV, Subramanya. Mobile Application Security Testing: Essential Skills for Ethical Hackers. Packt Publishing, 2016. 228 с.
10. Murphy, Mark L. Practical Android Security. Apress, 2016. 272 с.
11. Chell, Dominic, Erasmus, Tyrone, Colley, Shaun, Whitehouse, Ollie. The Mobile Application Hacker's Handbook. Wiley, 2015. 576 с.
12. Makan, Keith, Arzt, Steven. Android Security Cookbook. Packt Publishing, 2017. 332 с.
13. SANS Institute. Mobile Device Security and Ethical Hacking. [Електронний ресурс] https://www.sans.org/course/mobile-device-security-ethical-hacking. Останнє оновлення: 2023.

14. Hoog, Andrew. Android Forensics: Investigation, Analysis, and Mobile Security for Google Android. Syngress, 2011. 432 с.
15. McConnell, Steve. Code Complete: A Practical Handbook of Software Construction. Microsoft Press, 2004. 960 с.
16. NowSecure. Mobile Security Testing Guide. [Електронний ресурс] https://www.nowsecure.com/resources/mobile-security-testing-guide. Останнє оновлення: 2023.
17. CipherOps. Android Reverse Engineering 101. [Електронний ресурс] https://cipherops.com/2018/08/05/android-reverse-engineering-101/. Останнє оновлення: 2023.
18. Android Tamer. Mobile Security Testing Live Environment. [Електронний ресурс] https://androidtamer.com/. Останнє оновлення: 2023.
19. Kosarevsky, Sergey, Latypov, Viktor. Mastering Android NDK. Packt Publishing, 2015. 360 с.
20. Open Web Application Security Project (OWASP). Mobile Application Security Assessment Methodology. [Електронний ресурс] https://owasp.org/www-project-mobile/. Останнє оновлення: 2023.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)