МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**

**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЗАХИСТУ ІНФОРМАЦІЇ
КАФЕДРА УПРАВЛІННЯ ІНФОРМАЦІЙНОЮ ТА КІБЕРНЕТИЧНОЮ БЕЗПЕКОЮ**

КВАЛІФІКАЦІЙНА РОБОТА

на тему: “АНАЛІЗ ТА ВДОСКОНАЛЕННЯ МЕТОДІВ ТЕСТУВАННЯ ЗАХИЩЕНОСТІ МОБІЛЬНИХ ДОДАТКІВ ДЛЯ ЗАБЕЗПЕЧЕННЯ КІБЕРБЕЗПЕКИ КОРИСТУВАЧІВ”

на здобуття освітнього ступеня бакалавра
зі спеціальності 125 Кібербезпека
освітньої програми Управління інформаційною та кібернетичною безпекою

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

_____ Лада СЛОБОДСЬКА
(підпис) *Ім’я, ПРІЗВИЩЕ здобувача*

Виконав: здобувачка вищої освіти гр. УБД-42

Лада СЛОБОДСЬКА
Ім’я, ПРІЗВИЩЕ

Керівник: Дмитро РАБЧУН
К.т.н. Ім’я, ПРІЗВИЩЕ

Рецензент: Галина ГАЙДУР
Д.т.н., проф. Ім’я, ПРІЗВИЩЕ

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут захисту інформації

Кафедра управління інформаційною та кібернетичною безпекою

Ступінь вищої освіти бакалавр

Спеціальність 125 Кібербезпека

Освітня програма Управління інформаційною та кібернетичною безпекою

ЗАТВЕРДЖУЮ
Завідувач кафедри УІКБ
_____ Світлана ЛЕГОМІНОВА
“____”_____2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Слободській Ладі Олегівні
(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи “Аналіз та вдосконалення методів захищенності мобільних додатків для забезпечення кібербезпеки користувачів”,
керівник кваліфікаційної роботи РАБЧУН Дмитро, к.т.н.
(ПРІЗВИЩЕ, Ім’я,, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від 27.02.24 № 36.

2. Строк подання кваліфікаційної роботи “20” травня 2024р.

3. Вихідні дані до кваліфікаційної роботи: *міжнародні стандарти, наукова та технічна література, методи та засоби тестування мобільних додатків, вразливості притаманні мобільним додаткам.*

4. Перелік питань, які мають бути розроблені:
4.1. Проаналізувати принципи функціонування мобільних дадатків та їх безпекову складову.
4.2. Дослідити основні методи тестування захищенності мобільних додатків.
4.3. Проаналізувати інструменти та методи тестування захищенності мобільних додатків, розробити практичні рекомендації.

5. Перелік ілюстративного матеріалу: *презентація PowerPoint*

6. Дата видачі завдання “11“ березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Етапи кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	Визначення об’єкту, предмету, мети та завдань дослідження.	18.03.2024	
2.	Збір та аналіз літератури.	29.03.2024	
3.	Аналіз принципів функціонування мобільних додатків	05.04.2024	
4.	Дослідження основних методів тестування захищенності мобільних додатків.	08.04.2024	
5.	Вивчення інструментів та методів покращення тестування на проникнення мобільних додатків.	13.05.2024	
6.	Розробка рекомендацій для тестування захищенності мобільних додатків	17.05.2024	
7.	Формулювання висновків за результатами проведеного дослідження.	20.05.2024	
8.	Оформлення роботи.	21.05.2024	
9.	Оформлення презентації.	03.06.2024	
10.	Отримання рецензії на роботу.	03.06.2024	
11.	Захист в ДЕК.	__.06.2024	

Здобувач вищої освіти _____ Лада СЛОБОДСЬКА
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Дмитро РАБЧУН
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

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

ПОДАННЯ ГОЛОВІ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ ЩОДО ЗАХИСТУ КВАЛІФІКАЦІЙНОЇ РОБОТИ на здобуття освітнього ступеня бакалавра

Направляється здобувачка Слободська Л.О. до захисту кваліфікаційної роботи
(прізвище та ініціали)
за спеціальністю 125 Кібербезпека
(код, найменування спеціальності)
освітньої програми Управління інформаційною та кібернетичною безпекою
(назва)
на тему: "Аналіз та вдосконалення методів тестування захищенності мобільних додатків для забезпечення кібербезпеки користувачів"

Кваліфікаційна робота і рецензія додаються.

Директор ННІЗІ _____ Віталій САВЧЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Висновок керівника кваліфікаційної роботи

Здобувачка СЛОБОДСЬКА Лада у кваліфікаційній роботі проаналізувала принципи функціонування мобільних додатків, дослідила основні методи тестування захищення мобільних додатків, вивчила інструменти та методи тестування захищення мобільних додатків, розробила практичні рекомендації за темою дослідження.

СЛОБОДСЬКА Лада показала розуміння проблеми дослідження та бачення основних теоретичних і практичних напрямів її вирішення, довела володіння методами наукового дослідження, проявила себе як організований, відповідальний виконавець. Результати дослідження апробовані на двох конференціях.

Все це дозволяє оцінити кваліфікаційну роботу здобувачки СЛОБОДСЬКОЇ Лади на оцінку "відмінно" та присвоїти їй кваліфікацію бакалавра з кібербезпеки за освітньою програмою Управління інформаційною та кібернетичною безпекою.

Керівник кваліфікаційної роботи _____ Дмитро РАБЧУН
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

"____" _____2024 року

Висновок кафедри про кваліфікаційну роботу

Кваліфікаційна робота розглянута. Здобувачка Слободська Л.О. допускається до захисту даної роботи в Екзаменаційній комісії.

Завідувач кафедри
управління інформаційною
та кібернетичною безпекою _____ Світлана ЛЕГОМІНОВА
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

ВІДГУК РЕЦЕНЗЕНТА
на кваліфікаційну бакалаврську роботу

здобувачки вищої освіти СЛОБОДСЬКОЇ Лади
на тему “Аналіз та вдосконалення методів тестування захищенності мобільних додатків для забезпечення кібербезпеки користувачів”

Актуальність. Стрімкий, розвиток мобільних технологій та збільшення кількості користувачів, які залежать від мобільних пристроїв у своєму повсякденному житті. Це зумовлює необхідність забезпечення високого рівня кібербезпеки, оскільки мобільні додатки часто містять конфіденційну інформацію та особисті дані. Тестування захищеності допомагає виявляти та усувати вразливості, знижуючи ризик кібератак та зловмисного використання даних.

Отже, дослідження методів тестування захищенності мобільних додатків є актуальним науковим завданням.

Позитивні сторони.

1. У роботі досліджено особливості тестування захищенності мобільних додатків з метою забезпечення кібербезпеки користувачів.
2. Кваліфікаційна робота оформлена відповідно до вимог. Виклад матеріалу здійснено відповідно до плану, зроблено логічні висновки. Ключові положення роботи представлено у вигляді рисунків.
3. Авторка опрацювала значну джерельну базу: близько 40 публікацій, в тому числі англомовних.
4. За результатами дослідження запропоновано рекомендації щодо ефективного тестування захищенності мобільних додатків.

Недоліки.

Доцільно було б приділити більше уваги практичному застосуванню розроблених рекомендацій для тестування захищенності мобільних додатків.

Однак, вищезгадані зауваження не впливають на загальну позитивну оцінку кваліфікаційної роботи.

Висновок: Кваліфікаційна робота виконана на належному науково-методичному рівні і заслуговує оцінки “відмінно”, а здобувачка СЛОБОДСЬКА Лада заслуговує присвоєння кваліфікації бакалавра з кібербезпеки за освітньою програмою Управління інформаційною та кібернетичною безпекою.

Рецензент: _____ Галина ГАЙДУР
к.т.н., доцент *підпис* Ім’я, ПРІЗВИЩЕ

РЕФЕРАТ

Кваліфікаційна робота присвячена аналізу та вдосконаленню методів тестування захищенності мобільних додатків. Робота складається зі вступу, трьох розділів, що містять 20 рисунків, висновків і списку використаних джерел із 40 найменувань. Загальний обсяг роботи становить 77 аркушів, з яких 6 аркуші займають перелік умовних скорочень та список використаних джерел.

Метою роботи є роботи є розробка рекомендацій для підвищення ефективності та якості тестування захищенності мобільних додатків на основі існуючих методів.

Об'єктом дослідження є методи тестування захищенності мобільних додатків.

Предмет дослідження – особливості виявлення вразливостей у мобільних додатках методом тестування на проникнення.

Методи дослідження. Для вирішення поставленого вище наукового завдання в роботі були використані методи аналізу, порівняння, класифікації, системного підходу до вдосконалення методів тестування захищенності мобільних додатків.

Як результат у роботі проаналізовано особливості функціонування мобільних додатків, досліджено основні принципи тестування захищенності мобільних додатків, вивчено методи тестування захищенносту та розроблено практичні рекомендації.

Галузь застосування. Розроблені рекомендації можуть бути використані при проведенні тестування захищенності мобільних додатків.

Ключові слова: ТЕСТУВАННЯ НА ПРОНИКНЕННЯ, МОБІЛЬНІ ДОДАТКИ, БЕЗПЕКА ДОДАТКІВ, ВРАЗЛИВОСТІ, ЕТИЧНИЙ ХАКІНГ, ТЕСТУВАННЯ ЗАХИЩЕННОСТІ.

ABSTRACT

The qualification work is devoted to the analysis and improvement of data encryption methods in cloud computing. The work consists of an introduction, three chapters containing 20 figures, conclusions and the list of references containing 40 items. The total volume of the work is 77 pages, of which 6 pages are occupied by the list of abbreviations and the list of references.

The purpose of the study is to develop recommendations for improving the efficiency and quality of testing the security of mobile applications based on existing methods.

The object the study is the methods of testing the security of mobile applications.

The subject of the study is the features of vulnerability detection in mobile applications by penetration testing.

Research methods. To solve the scientific task set above, methods of analysis, comparison, classification, and a systematic approach to improving the methods of testing the security of mobile applications were used in the work.

As a result, the work analyzed the features of the functioning of mobile applications, studied the main principles of testing the security of mobile applications, examined the methods of security testing, and developed practical recommendations.

Field of application. The developed recommendations can be used when conducting security testing of mobile applications.

Keywords: PENETRATION TESTING, MOBILE APPLICATIONS, APPLICATION SECURITY, VULNERABILITIES, ETHICAL HACKING, SECURITY TESTING.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

ОС	Операційна Система
SSL	Secure Sockets Layer
VM	Virtual Machine
API	Application Programming Interface
GPS	Global Positioning System
IDE	Integrated Development Environment
JVM	Java Virtual Machine
XML	eXtensible Markup Language
OAT	Android Runtime Optimized Application
APK	Android Package
JAR	Java Archive

ВСТУП

Актуальність теми. Стрімкий, розвиток мобільних технологій та збільшення кількості користувачів, які залежать від мобільних пристроїв у своєму повсякденному житті. Це зумовлює необхідність забезпечення високого рівня кібербезпеки, оскільки мобільні додатки часто містять конфіденційну інформацію та особисті дані. Тестування захищеності допомагає виявляти та усувати вразливості, знижуючи ризик кібератак та зловмисного використання даних.

Отже, дослідження методів тестування захищенності мобільних додатків є актуальним науковим завданням.

Метою роботи є роботи є розробка рекомендацій для підвищення ефективності та якості тестування захищенності мобільних додатків на основі існуючих методів.

Об’єктом дослідження є методи тестування захищенності мобільних додатків.

Предмет дослідження – особливості виявлення вразливостей у мобільних додатках методом тестування на проникнення.

Для досягнення цієї мети в роботі необхідно виконати наступні **завдання**:

1. Проаналізувати принципи функціонування мобільних дадатків та їх безпекову складову.
2. Дослідити основні методи тестування захищенності мобільних додатків.
3. Проаналізувати інструменти та методи тестування захищенності мобільних додатків, розробити практичні рекомендації.

Методи дослідження. Для вирішення поставленого вище наукового завдання в роботі були використані методи аналізу, порівняння, класифікації, системного підходу до вдосконалення методів тестування захищенності мобільних додатків.

Практичне значення одержаних результатів. Застосування напрацювань дасть змогу здійснити пошук вразливостей у мобільних додатках з

подальшим покращенням кібербезпеки користувачів цього додатку.

Апробація результатів кваліфікаційної роботи відбулася на Всеукраїнській науково-практичній конференції “Стратегії кіберстійкості: управління ризиками та безперервність бізнесу” 28 лютого 2024 року.

Розділ 1. ЗАГАЛЬНІ ВІДОМОСТІ ПРО ФУНКЦІОНУВАННЯ ТА БЕЗПЕКУ МОБІЛЬНИХ ДОДАТКІВ

1.1 Аналіз принципів функціонування мобільних додатків

Перш ніж розглядати структуру мобільних додатків, слід розуміти, як саме працює операційна система, на якій вони запускаються. Подальші пояснення будуть надані саме для ОС Android.

Android можна розглядати як програмний стек, що складається з різних рівнів, кожен з яких має чітко визначену поведінку та надає конкретні сервіси для рівнів, що знаходяться вище в ієрархії. На рисунку 1.1 зображено схему, яка ілюструє стек операційної системи Android та рівні, з яких вона складається.

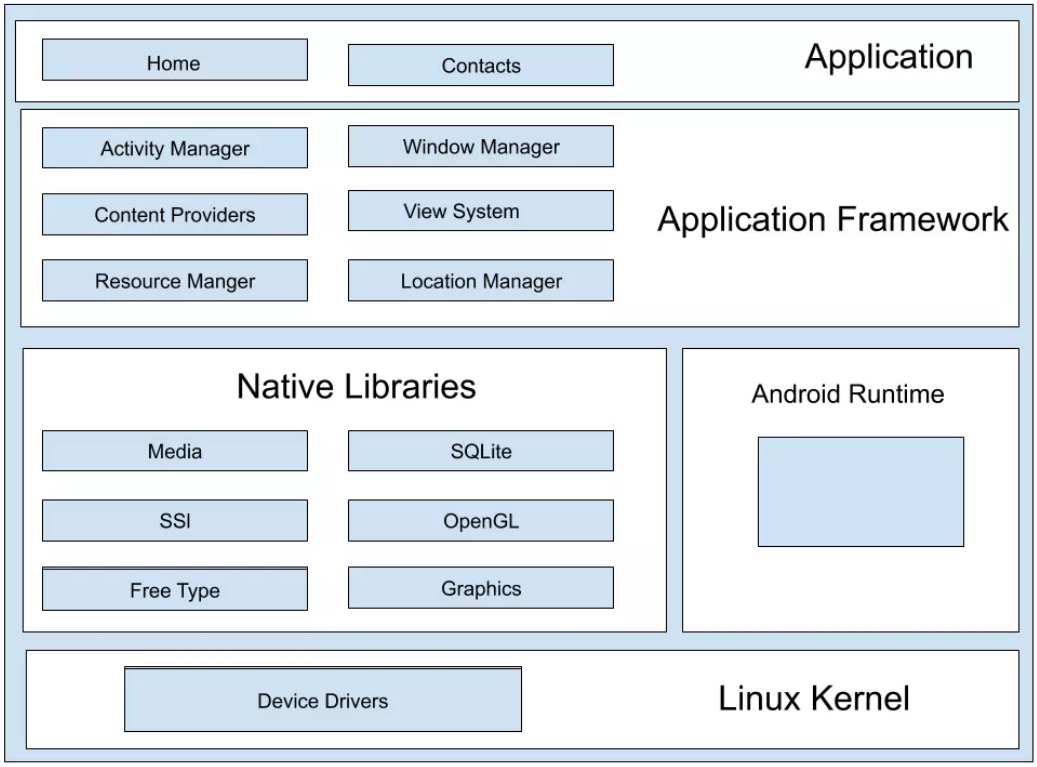

Рис. 1.1 Структура стеку операційної системи Android.

Операційна система Android базується на ядрі Linux, яке знаходиться на найнижчому рівні стеку та об'єднує всі вищі рівні. Ядро Linux забезпечує базові системні функції, такі як керування пам'яттю, процесами та апаратним забезпеченням (такі як камера, сенсори, клавіатура і т. д.) пристрою через їх драйвери, а також забезпечує безпеку операційної системи [1].

Наступний рівень стека складається з бібліотек, Android Runtime та Dalvik. Android містить у собі бібліотеки написані на C та C++, що використовуються різними компонентами Android, наприклад:

- ***Media Libraries*** - дозволяє відтворювати та записувати аудіо та відео у різних форматах.
- ***SQLite*** - надає реляційні бази даних, які можуть використовуватися додатками та системами.
- ***SSL*** - забезпечує підтримку типових криптографічних функцій.
- та інші.

Android Runtime можна розглядати як складову, що складається з двох різних компонентів: віртуальної машини Dalvik та основних бібліотек. Додатки для Android зазвичай пишуться мовою Java. Ці додатки потім компілюються у файли класів Java. Однак Android не виконує ці файли класів як є. Файли класів Java перекомпілюються в формат dex, що додає ще один крок до процесу перед виконанням додатків операційною системою. Потім формат Dex виконується в спеціальній реалізації віртуальної машини Java (VM) - Dalvik VM. Віртуальна машина Dalvik залежить від ядра Linux для надання функціонала на нижчому рівні (наприклад, керування пам'яттю). Кожен додаток, який працює на пристрої Android, має свою власну віртуальну машину Dalvik [2].

Android включає набір основних бібліотек, які забезпечують більшість функціонала, доступного в програмних інтерфейсах Java.

Рівнем вище йде рівень фреймворку додатків, цей рівень надає широкий набір класів, доступних розробникам через Java API для додатків. Це здійснюється за допомогою різних служб Менеджера додатків. Найважливіші компоненти на цьому рівні [3]:

- ***Менеджер активностей*** - керує життєвим циклом активностей додатків та різними компонентами додатків. Коли додаток запитує про запуск активності, наприклад, через startActivity(), Менеджер активностей забезпечує цю послугу.

- ***Менеджер ресурсів*** - надає доступ до ресурсів, таких як рядки, графіка та файлів макета.
- ***Менеджер місцеперебування*** - забезпечує підтримку оновлень місцеперебування (наприклад, GPS).
- ***Менеджер сповіщень*** - додатки, які цікавляться отриманням сповіщень про певні події, отримують цю послугу через менеджера сповіщень. Наприклад, якщо додаток цікавиться тим, коли надійшло нове електронний лист, він буде використовувати службу Менеджера сповіщень.

Останнє місце в стеку займає рівень додатків з якими й працюють кінцеві користувачі. Саме цей рівень і буде розглянутий далі.

Розробка додатків для Android включає кілька процесів, які відбуваються послідовно. Після написання вихідного коду, коли розробники натискають кнопку "Запустити" в Android Studio, розпочинається багато операцій та процесів на бекенді. Кожна операція, що відбувається у фоновому режимі, є важливим кроком і взаємозалежною. Інтегроване середовище розробки (IDE) компілює всі файли додатка, робить їх сумісними з пристроями для розгортання та гарантує успішний запуск додатка на пристрої. Далі будуть пояснені всі кроки, які проходить Android додаток, від написання коду до повноцінно працюючого додатка на пристрої [4].

Крок 1. Побудова файлу APK

Вихідні файли додатка для Android пишуться або на мові програмування Java (.java файли), або на мові Kotlin (.kt файли). Синтаксис написання коду на цих двох мовах відрізняється, але їх процес компіляції майже однаковий. Обидві мови програмування генерують код, який може бути скомпільований до байт-коду Java, що виконується на JVM (Java Virtual Machine). У середовищі Android процес починається з компіляції вихідного коду Java/Kotlin в файл класу Java. Файли класів мають розширення *.class і містять байт-код Java (представляє Java-асемблер). Це завдання компіляції виконується компіляторами javac та kotlinc для коду мов Java та Kotlin відповідно [5].

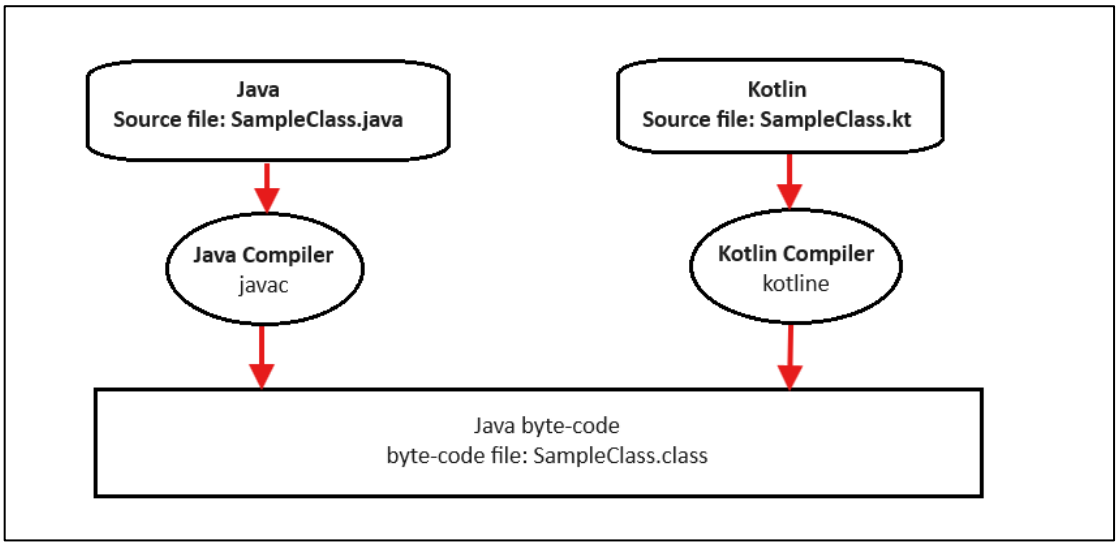

Рис. 1.2 Процес компіляції коду на прикладі мов Java та Kotlin.

Згенерований файл класу Java (*.class) на попередньому кроці виконується на віртуальній машині Java (JVM), оскільки містить стандартний байт-код Java Oracle JVM. Однак цей формат коду не підходить для пристроїв Android, і тому Android має свій власний унікальний формат байт-коду, відомий як байт-код Dalvik. Компілятор Dex перекладає байт-код Java в байт-код Dalvik, який є інструкціями машинного коду для теоретичного процесора. Під час процесу компіляції команда dx зв'язує всі файли .class, а також файли .jar разом і створює єдиний файл classes.dex, який записаний у форматі байт-коду Dalvik. Цей файл тепер може виконуватися на віртуальній машині в операційній системі Android, відомій як Android Runtime (або Dalvik Virtual Machine (DVM) для версій Android старше Kitkat).

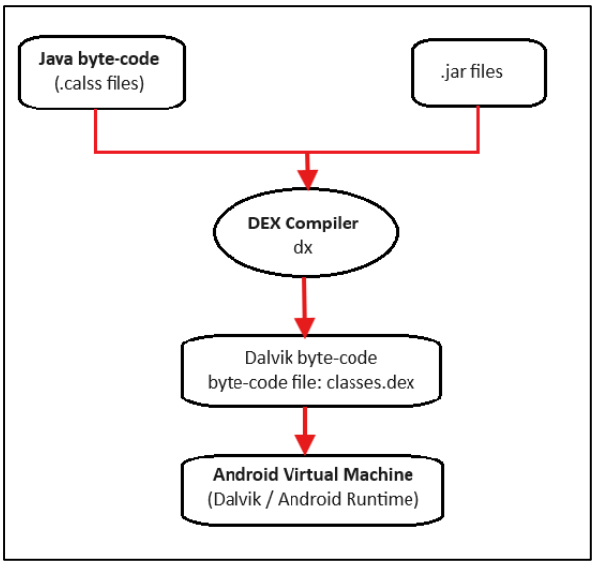

Рис. 1.3 Процес створення віртуальної машини.

Файли ресурсів додатка для Android, такі як зображення, шрифти, макети XML тощо, перетворюються в один скомпільований ресурсний блок за допомогою інструменту Android Asset Packaging Tool (aapt). Інструмент aapt також відповідає за створення файлу R.java додатка для Android. Далі, скомпільований ресурсний блок разом з файлом classes.dex стискається інструментом apkbuilder, і створюється файл подібний до zip, який називається пакетом Android (.apk файл). Згенерований файл .apk містить усі необхідні дані для запуску додатка для Android.

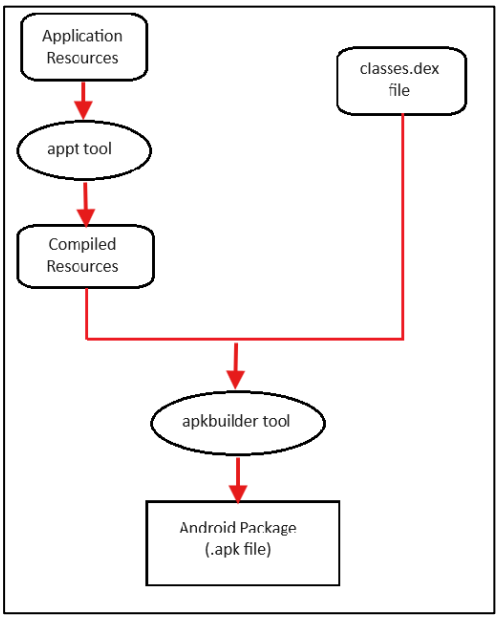

Рис. 1.4 Процес формування apk файлу.

Файл .apk, згенерований на попередньому кроці, є готовим пакетом додатка, і розробники можуть використовувати цей файл для розповсюдження додатка. Однак, для розповсюдження та публікації додатка через Google Play Store, розробники повинні підписати його. Додатки для Android повинні бути цифрово підписані сертифікатом, щоб їх можна було встановлювати користувачами. Сертифікат самопідписаний, і Android використовує його для ідентифікації автора додатка. Розробник/автор додатка зберігає приватний ключ сертифіката, і всі ці деталі зберігаються як додатковий файл у пакеті Android (.apk файл).

Oracle Java Development Kit (JDK) надає інструмент jarsigner для підписання файлів .jar і .apk. Крім того, стиснені частини підписаного файлу .apk повинні лежати на границях байтів таким чином, щоб Android OS могла читати їх без розпаковування файлу. Вирівнювання байтів файлів забезпечується за допомогою інструменту zipalign при виконанні підписаного файлу .apk.

Крок 2: Розгортання додатка

Android Debug Bridge (ADB) розгортає додаток на пристрої Android. Це інструмент командного рядка, який діє як інтерфейс і дозволяє розробникам взаємодіяти з пристроєм Android [6]. Для початку розгортання клієнт ADB спочатку перевіряє, чи вже запущений процес сервера ADB на пристрої. Якщо такого немає, процес сервера запускається за допомогою команди ADB. Сервер ADB починає роботу та прив'язується до локального TCP-порту 5037. Всі комунікації та команди передаються від сервера ADB до клієнтів ADB за допомогою порту 5037. Крім того, сервер налаштовує з'єднання з усіма запущеними пристроями. Він сканує всі порти, і коли сервер виявляє демона ADB (adbd: фоновий процес на емуляторі або пристрої), він налаштовує з'єднання з тим портом. Процес adbd, який збігається на пристрої Android, може спілкуватися з додатками, відлажувати їх та збирати їх вивід журналу.

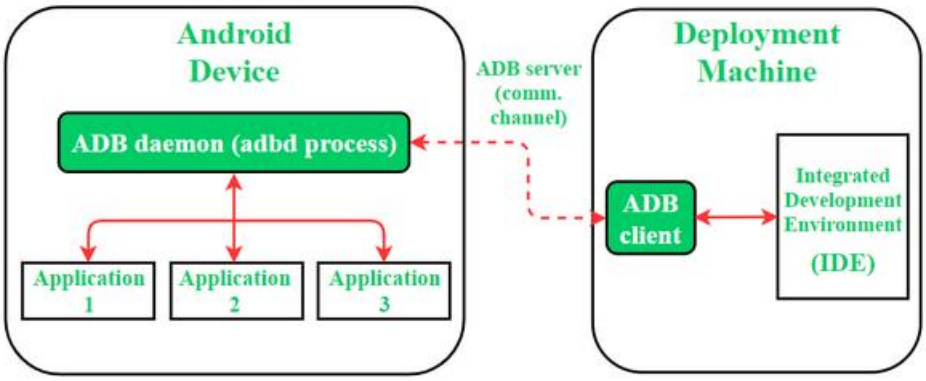

Рис. 1.5 Процес розгортання додатка.

ADB передає файл .apk в локальну файлову систему цільового пристрою Android. Розташування додатка в файловій системі пристрою визначається його іменем пакета. Наприклад, якщо пакет додатка - com.example.sampleapp, то його файл .apk буде розташований за шляхом /data/app/com.example.sampleapp.

Крок 3: Запуск додатка

Процес Zygote є батьківським для всіх додатків Android і запускає додаток, коли користувач подає на це запит. Zygote [7] є спеціальним типом процесу операційної системи Android, який дозволяє спільне використання коду між різними екземплярами, які працюють на різних віртуальних пристроях Android (Dalvik/Android Runtime). Ті ресурси, класи та кодові бібліотеки, які можуть знадобитися будь-якому додатку під час виконання, попередньо завантажуються в пам'ять процесу Zygote. Коли процес отримує запит на запуск нового додатка, він розділяється (створює копію) за допомогою виклику системи fork (Android - це система Linux) і починає новий додаток. Попередньо завантажені бібліотеки і ресурси - це причина ефективного і швидкого запуску додатків в Android.

Коли встановлюється новий додаток, Android оптимізує дані додатка і генерує відповідний файл OAT [8]. Цей файл створюється операційною системою Android для прискорення часу завантаження додатка. Процес створення файлу OAT починається з вилучення файлу classes.dex, який знаходиться всередині файлу .apk додатка. Файл classes.dex розміщується в окремій директорії, і Android компілює байт-код Dalvik за допомогою компіляції перед часом (AOT, також скорочено як OAT) у нативний машинний код. Система

Android використовує цей нативний файл OAT для поліпшення користувацького досвіду, швидко та плавно завантажуючи додаток.

Перед тим, як AOT з'явився на світі, інструмент dexopt використовувався для конвертації файлів .dex у файл .odex (оптимізований DEX), який містить оптимізований байт-код. З введенням AOT в Android, інструмент dex2oat конвертує і оптимізує файл .dex у формат файлу OAT, який містить машинний код, записаний у форматі ELF (Executable and Linkable Format) [9]. Ця нативна бібліотека потім відображається в пам'ять процесу додатка. Файли OAT, як правило, зберігаються на пристрої Android у директорії: /data/dalvik-cache/.

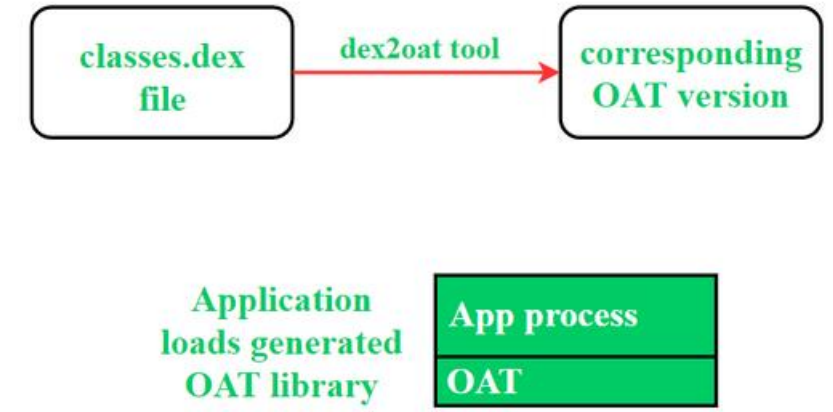

Рис. 1.6 Процес запуску додатка.

Після виконання всіх цих кроків і процедур додаток нарешті запуститься, і початкова активність додатка з'явиться на екрані пристрою.

Розглянемо детальніше про структурні компоненти набору для створення додатків Android (.apk) та їх призначення.

Файл .apk є Android-бінарним. APK - це скорочення від Android Package Kit (також Android Application Package) і є файловим форматом, який Android використовує для розподілу та встановлення додатків Android. Він містить всі елементи, які потрібні для правильної роботи додатка на вашому пристрої.

Додатки для Android упаковуються в один тип файлу, який називається APK (Android Package Kit). Вони мають розширення .apk, але насправді це просто файли ZIP. Якщо ви зміните це розширення, то зможете без перешкод

розпакувати їх вміст. Щоб бути точнішим, APK - це тип JAR (Java ARchive), який є типом ZIP. Вміст файлу .apk завжди має певну чітку структуру [10]:

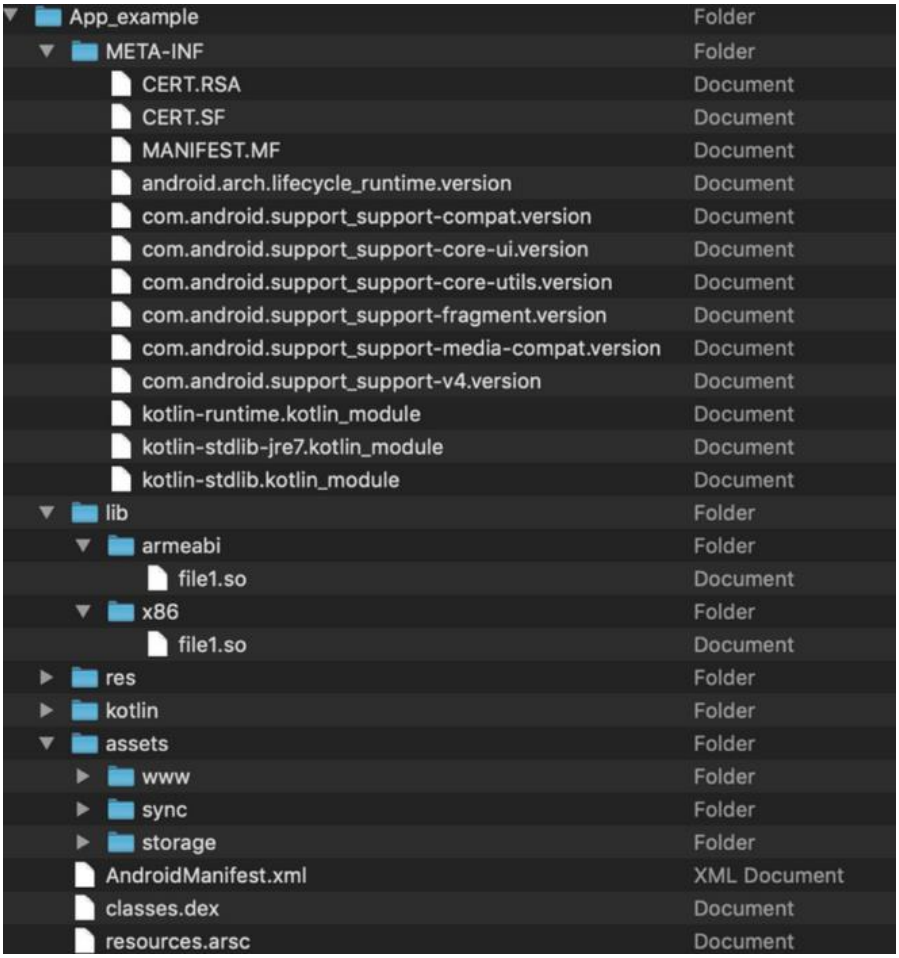

Рис. 1.7 Файлова структура APK-файлу.

- ***META-INF*** – ця тека містить інформацію для перевірки та генерується під час "підписування" додатка. Фактично це відбиток для кожного файлу, що міститься у APK, що означає, що будь-яка модифікація APK (навіть заміна значка) вимагає підписання APK. В іншому випадку операційна система відхилить встановлення [11]. Під час підписування додатків ці файли оновлюються. Зміст цієї теки включає:
    - *CERT.SF* (може мати інше ім'я файлу з тим самим суфіксом) – Список всіх файлів з їх хешами SHA-1.
    - *CERT.RSA* (може мати інше ім'я файлу з тим самим суфіксом) – Містить підписаний вміст

    - *CERT.RF* та використовується для перевірки цілісності додатка за допомогою публічного ключа.
- ***MANIFEST.MF*** – містить SHA-256-Digest всіх файлів у apk. Використовується для перевірки валідності кожного файлу у zip, що забезпечує зміну цього файлу або будь-яких інших файлів у zip-контейнері, анулює підпис та робить його недійсним [12].
- ***assets*** – розробник контролює ресурси додатка у ієрархії тек. Наприклад, якщо ваш додаток відображає відео або має деякі шаблони документів, вони будуть зберігатися у теки assets. Крім того, деякі фреймворки також використовують теку assets для зберігання коду та даних. Наприклад:
    - Cordova чи React-native додатки зберігають свій Javascript-код у теки assets.
    - Maui, Xamarin додатки зберігають файли DLL у теки зборок, які діють так само.
- ***AndroidManifest.xml*** – обов'язковий файл, який описує додаток. Маніфест містить ключові елементи інформації про додаток [13]. Ось частковий список:
    - Ім'я пакета додатка.
    - Усі компоненти додатка, такі як активності та ресурси.
    - Права, які потрібні для роботи цього додатка, а також права, необхідні для доступу інших додатків до інформації цього додатка.
    - Особливості сумісності (наприклад, мінімальна версія Android та підтримувані пристрої).
- ***classes.dex*** – пропрієтарний формат Google для їхньої версії Java VM, містить весь код Java/Kotlin, скомпільований до їхнього конкретного байт-коду, який називається Dalvik. Файли APK можуть містити більше одного файлу classes.dex через обмеження формату DEX. Додаткові файли будуть пронумеровані (наприклад, classes2.dex, classes3.dex і так далі).
- ***kotlin*** – ця тека присутня лише у разі, якщо додаток був написаний за допомогою Kotlin. Вона містить дані, специфічні для Kotlin.

- ***lib*** – тека, що містить нативні бібліотеки (машинний код). Оскільки Android є крос-платформеним, вона містить підтеки для кожного підтримуваного процесора:
    - *armabi* – бінарний код, який підтримує принаймні ARMv5TE, застарілий з моменту ndkr16 (грудень 2017 року) і видалений у ndkr17 (червень 2018 року).
    - *armeabi-v7a* – бінарний код, який підтримує ARMv7.
    - *arm64-v8a* – бінарний код, який підтримує ARMv8.
    - *x86* – бінарний код, який підтримує x86.
    - *x86_64* – бінарний код, який підтримує x86-64.
    - *mips* – бінарний код для MIPS, застарілий з моменту ndkr17.
- ***res*** – тека, яка містить ресурси (такі як активи), але з попередньо визначеною ієрархією тек, яку розробник не може змінити. Ці файли використовуються для надання альтернатив для різних орієнтацій екрана, версій операційних систем та підтримки багатомовності.
- ***resources.arsc*** – Це файл, який містить інформацію, що зв'язує код (classes.dex) з ресурсами (res). Наприклад, код може посилатися на текст діалогу, тоді як ресурси містять цей текст у всіх мовах. Операційна система Android потім обирає правильну мову відповідно до конфігурації локалі пристрою [14].

Android додатки також складаються з певних компонентів, кожен з них вважається точкою, через яку система може увійти до додатка. Не всі компоненти є фактичними точками входу для користувача, і деякі залежать один від одного, але кожен існує як власна сутність і відіграє певну роль - кожен з них є унікальним будівельним блоком, який допомагає визначити загальну поведінку додатка. Існують такі чотири компоненти додатка:

- ***Постачальник контенту (Content provider)*** - надає дані з одного додатка іншим за запитом. Через постачальника контенту інші додатки можуть запитувати або навіть змінювати дані (якщо постачальник контенту дозволяє це). Цей компонент також корисний у випадках, коли додаток

хоче поділитися даними з іншим додатком.Він дуже схожий на бази даних і має чотири методи: insert(), update(), delete(), query() [15].

- ***Активність (activity)*** - представляє собою один екран з користувацьким інтерфейсом. Наприклад, одна активність для входу та інша активність після успішного входу, це як новий екран.
- ***Служби (Services)*** - це компонент, який працює у фоновому режимі для виконання тривалих операцій або виконання роботи для віддалених процесів. Служба не надає користувацького інтерфейсу, жоден інший компонент, такий як активність, не може запускати службу та дозволяти їй працювати або пов'язуватися з нею для взаємодії з нею. Наприклад, служба може відтворювати музику в фоновому режимі, поки користувач перебуває в іншому додатку, або вона може отримувати дані через мережу без блокування взаємодії користувача з активністю.
- ***Приймач трансляцій (Broadcast Receiver)*** - реагує на системні трансляційні оголошення. Багато трансляцій походять від системи, наприклад, трансляція, що повідомляє, що екран вимкнено, акумулятор розряджений або зроблено знімок. Додатки також можуть ініціювати трансляції, наприклад, щоб інші додатки знали, що деякі дані були завантажені на пристрій і доступні для використання ними. Хоча приймачі трансляцій не відображають користувацький інтерфейс, вони можуть створювати сповіщення на панелі стану, щоб повідомити користувача, коли сталася подія трансляції. Однак частіше приймач трансляцій просто є "шлюзом" до інших компонентів і призначений виконувати дуже мінімальну кількість роботи. Наприклад, він може ініціювати службу для виконання деякої роботи на основі події. Додаток може зареєструвати приймач для повідомлення про низький рівень заряду акумулятора, наприклад, і змінити свою поведінку на основі цієї інформації [16].
- ***Інтент*** - за допомогою асинхронного повідомлення активує активності, служби та приймачі трансляцій. Інтенти пов'язують окремі компоненти між собою під час виконання (ви можете уявити їх як посла, який запитує

дію від інших компонентів), незалежно від того, чи належить компонент вашому додатку, чи іншому.

Не менш важливо розглянути заходи захисту додатків та конфіденційних даних користувачів, які надає сама операційна система Android. Серед найважливіших таких заходів можна виділити: модель дозволів Android, підпис додатків, перевірка додатків та пісочниця Android. Розглянемо їх детальніше:

- ***Модель дозволів Android*** - за замовчуванням у операційній системі Android є деякі захищені API, які може використовувати лише операційна система. Серед захищених API можна виділити:
    - Функції камери
    - Дані про місцезнаходження (GPS)
    - Функції Bluetooth
    - Телефонні функції
    - Функції SMS/MMS
    - Мережеві/дані з'єднання

  Якщо певний додаток потребує доступу до будь-якого з цих API, то він повинен зазначити цей дозвіл у файлі AndroidManifest.xml. Можливо, ви помітили, що при встановленні певного додатка з Google Play Store він запитує кілька потрібних дозволів; якщо ви не надаєте їх, то додаток не встановиться. Якщо ж користувач згоден надати ці дозволи, то операційна система Android надає доступ до цього захищеного API.

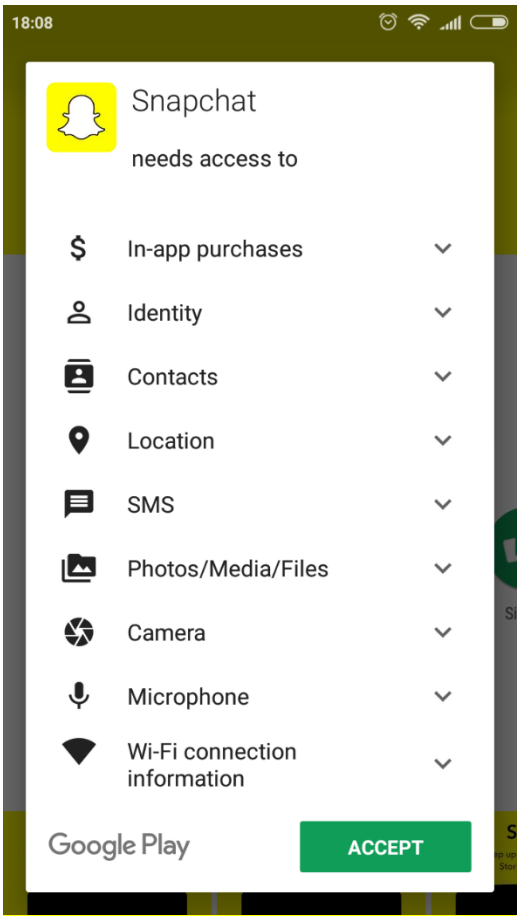

Рис. 1.8 Діалогове вікно дозволів під час встановлення додатку Snapchat.

- ***Підпис додатків*** - Android вимагає, щоб всі додатки були цифрово підписані сертифікатом перед їх встановленням, після чого вони використовуються для ідентифікації автора додатка. Для запуску додатка на пристрої він повинен бути підписаним. Під час встановлення додатка на пристрій менеджер пакунків перевіряє, чи був додаток належним чином підписаний сертифікатом у файлі apk. Додаток може бути підписаний самостійно або через сертифікаційний орган. Підпис додатків забезпечує, що один додаток не може отримати доступ до будь-якого іншого додатка, крім визначеного між ними міжпроцесного зв'язку, і також гарантує, що він передається на пристрій без змін.
- ***Перевірка додатків*** - починаючи з Android 4.2, підтримується перевірка додатків. Користувачі можуть вибрати опцію "Перевірка додатків" і мати можливість перевіряти додатки за допомогою перевірника додатків перед їх встановленням. Перевірка додатків може повідомляти користувача,

якщо вони намагаються встановити додаток, який може бути шкідливим; якщо додаток особливо поганий, він може блокувати встановлення.

- ***Android Sandbox (пісочниця)*** - після встановлення на пристрій кожний додаток Android існує у власному захищеному середовищі, відомому як пісочниця. Операційна система Android є багатокористувацькою системою Linux, в якій кожен додаток виступає як окремий користувач [17].

За замовчуванням система присвоює кожному додатку унікальний ідентифікатор користувача Linux (цей ідентифікатор використовується тільки системою і невідомий додатку). Система встановлює дозволи для всіх файлів у додатку так, щоб до них могли звертатися лише користувачі, яким було присвоєно ідентифікатор, призначений для цього додатка.

Кожен процес має власну віртуальну машину (VM), тому код додатка виконується в ізоляції від інших додатків.

Кожен додаток працює у власному процесі Linux. Android запускає процес, коли потрібно виконати будь-який з компонентів додатка, а потім завершує процес, коли він більше не потрібен або коли система має відновити пам'ять для інших додатків.

Таким чином, система Android реалізує принцип найменшого дозволу, тобто кожен додаток за замовчуванням має доступ лише до тих компонентів, які необхідні для його роботи, і не більше. Це створює дуже безпечне середовище, в якому додаток не може отримати доступ до частин системи, для яких він не має дозволу. Кожен додаток Android працює у власному середовищі пісочниці і за замовчуванням не може впливати на інші додатки, але два додатки можуть мати той самий ідентифікатор користувача Linux і також можуть спільно використовувати ту ж саму віртуальну машину Dalvik, якщо вони підписані тими самими сертифікатами.

1.2 Огляд найпоширеніших вразливостей у додатках

Безпека мобільних додатків повинна акцентувати увагу на тому, як мобільні додатки обробляють, зберігають і захищають чутливу інформацію. Вона повинна охоплювати ключові компоненти поверхні атак мобільного додатка, включаючи зберігання, криптографію, аутентифікацію та авторизацію, мережеву комунікацію, взаємодію з мобільною платформою, якість коду та зміцнення проти зворотного проектування та втручання.

Для того, щоб запезпечити безпеку всіх цих компонентів був створений Стандарт верифікації безпеки мобільних додатків від OWASP (MASVS). OWASP MASVS [18] став індустрійним стандартом безпеки мобільних додатків і є цінним ресурсом для розробників, власників додатків та фахівців з безпеки. OWASP MASVS є результатом багаторічних зусиль спільноти та зворотного зв'язку від індустрії. Він надає комплексний набір чітких та конкретних рекомендацій, найкращих практик і контролів безпеки, які можна використовувати для визначення та оцінки безпеки мобільних додатків на різних платформах (Android та iOS) у різних варіантах використання, з урахуванням конкретних загроз вашої індустрії та архітектури мобільного додатка.

OWASP в рамках проєкту "The OWASP Mobile Top 10" визначає 10 найпопулярніших вразливостей для мобільних додатків, серед них визначаються наступні:

M1: Неправильне Використання Облікових Даних - ця вразливість стосується вбудованихоблікових даних у вихідний код додатку або у конфігураційні файли [19].

- *Приклад 1:* Зловмисник виявив вбудовані облікові дані у коді додатка та використав їх для отримання неавтризованого доступу до конфіденціних даних у внутрішній системі (backend) додатку.
- *Приклад 2:* Зловмисник перехоплює незахищені облікові дані, що передавалися між дадатком та її серверною системою, в результаті, прехоплені облікові дані були використані для отримання несанкціонованого доступу до системи від імені іншого користувача.

- *Приклад 3:* Зловмисник отримує фізичний доступ до пристрою іншого користувача та витягує облікові дані з мобільного додатка, в результаті чого обліковий запис користувача був повністю скомпроментований.

M2: Недостатня Безпека Ланцюга Постачання - ця вразливість акцентується на "ланцюжку постачання", що означає діяльність, пов'язану зі створенням та розповсюдженням мобільного додатка.

- *Приклад 1:* Перед тим як APK-файл буде підписаний, зловмисник може змінити код додатка та розповсюджувати шкідливе програмне забезпечення під виглядом легітимного додатка.
- *Приклад 2:* Після підписання APK-файлу зловмисник може розповсюджувати модифікований додаток в ненадійних джерелах, таких як месенджери чи сторонні сайти [20].

M3: Ненадійна Аутентифікація/Авторизація - це одна з найпоширеніших вразливостей, що спостерігаються у мобільних додатках, під час експлуатації вразливостей такого типу зловмисники використовують автоматизовані атаки з використанням доступних інструментів [21].

M4: Недостатня Валідація Введених та Виведених Даних - недостатня валідація та санітизація даних зовнішніх джерел, таких як введення користувачів або мережеві дані, у мобільному додатку може призвести до серйозних вразливостей безпеки. Мобільні додатки, які не виконують належну валідацію та санітизацію таких даних, піддаються ризику експлуатації через атаки, характерні для мобільних середовищ, включаючи ін'єкцію SQL, ін'єкцію команд та атаки міжсайтового скриптінгу (XSS).

- *Приклад 1:* За допомогою створення зловмисного введення, що містить неочікувані символи, зловмисник експлуатує поведінку додатка. Через недостатню валідацію, додаток неправильно обробляє введення, що призводить до вразливостей. Атакуючий успішно виконує довільний код, отримуючи несанкціонований доступ до ресурсів пристрою та чутливих даних.

- *Приклад 2:* Атакуючуй експлуатує точку входу, де обробляється контент, що створений користувачем або ненадійні дані. Шляхом створення зловмисного вводу, що містить код або скрипти (наприклад, HTML, JavaScript, SQL), атакуючий використовує відсутню валідацію виводу. Надсилаючи створений ввід через взаємодію користувача, додаток не валідує або не санітує його, що дозволяє виконання вбудованого коду або непередбачених операцій. Атакуючий успішно виконує атаки ін'єкції, такі як міжсайтовий скриптінг (XSS) або внедрення SQL, компрометуючи цілісність додатка та отримуючи доступ до чутливої інформації.
- *Приклад 3:* Атакуючий виявляє мобільний додаток, який обробляє надані користувачем дані та генерує динамічний вивід. Він створює спеціально відформатовані дані, які експлуатують недостатню валідацію виводу додатка. Атакуючий надсилає зловмисні дані додатку, чи то через пряму взаємодію, чи використовуючи вразливий API. Додаток не валідує або не сані тує згенерований вивід, дозволяючи атакуючому створеній даний виконувати код або спричиняти непередбачені дії [22].

M5: Ненадійна Комунікація - Більшість сучасних мобільних додатків обмінюються даними з одним або кількома віддаленими серверами. Під час передачі даних вони зазвичай проходять через мережу оператора мобільного зв'язку та Інтернет. Зловмисник, який прослуховує зв'язок, може перехопити та модифікувати дані, якщо вони передаються у відкритому текстовому вигляді або за допомогою застарілого протоколу шифрування [23].

- *Приклад 1:* Якщо перевірка сертифікатів між сервером та мобільним додатком не вдалася, або якщо механізм закріплення сертифікатів може бути обійдений, це може призвести до використання експлойту, що викликає атаку типу "людина посередині" (MITM), витоку особистої ідентифікаційної інформації (PII), витоку облікових даних та ін.

M6: Недостатні Контролі Приватності - контролі приватності стосуються захисту особисто ідентифікованої інформації (PII), такої як імена та адреси,

інформація про кредитні картки, електронні адреси та IP-адреси, інформація про здоров'я, релігію та політичні погляди [24].

M7: Недостатній Захист Виконуваного Файлу - виконуваний файл містить всі елементи, які додаток потребує для правильної роботи на пристрої. Це може включати ключі API, захардкоджені облікові дані або криптографічні секрети [25].

M8: Неправильна Конфігурація Безпеки - неправильна конфігурація безпеки в мобільних додатках означає неправильне налаштування параметрів безпеки, дозволів та контролів, що може призвести до вразливостей та несанкціонованого доступу [26].

- *Приклад 1:* Будь-який мобільний додаток, який зберігає незашифровані дані у загальних налаштуваннях або дозволяє світовому читанню, вразливий для цього, що дозволяє іншим додаткам читати ці дані.

M9: Незахищене Зберігання Даних - Незахищене зберігання даних у мобільному додатку може привертати різних загрозних агентів, які мають на меті експлуатувати вразливості та отримувати несанкціонований доступ до чутливої інформації. Ці загрозні агенти включають в себе кваліфікованих атакуючих, які націлені на мобільні додатки для видобування цінної інформації, злоумисників всередині організації або команди розробників додатків, які зловживають своїми привілеями, державні органи, які здійснюють кібершпигунство, кіберзлочинці, які шукають фінансовий прибуток через крадіжку даних або вимагання викупу, непрофесійні атакуючі, які використовують готові інструменти для простих атак, брокери даних, які намагаються експлуатувати незахищене зберігання для продажу особистої інформації, конкуренти та індустріальні шпигуни, які мають на меті отримати конкурентну перевагу, а також активісти або хактивісти з ідеологічними мотивами [27].

M10: Недостатня Криптографія - Ця вразливість стосується зберігання даних без або зі слабким шифруванням на легко доступних місцях (у файловій системі в базі даних без захисту або у текстових файлах) [28].

Рис. 1.9 The OWASP Mobile Top 10.

У сфері мобільної безпеки "The OWASP Mobile Top 10" є лише початком. Розуміння цих ризиків становить основу для комплексної безпеки. Це надає розробникам та експертам можливість приймати превентивні заходи, дотримуватися найкращих практик та впроваджувати строгі вимоги. Тож, рухаючись вперед, давайте приймати це колективне зобов'язання до мобільної безпеки та разом навігувати у мобільному ландшафті з впевненістю та стійкістю.

1.3 Сучасні вимоги до безпеки мобільних додатків

Враховуючи ризики, які створюють вразливості у мобільних додатка були створені нормативні документи та стандарти, які вимагають від розробників притримуватися певних безпекових принципів під час розробки додатків. Серед таких документів:

- ISO/IEC 27034, яка розглядає концепції, принципи, основи, компоненти та процеси для допомоги організаціям у справі інтеграції неперервного захисту їхніх додатків упродовж усього життєвого циклу [29].
- NIST SP 800-163 - цей документ надає рекомендації щодо планування та впровадження процесу перевірки додатків, розробки вимог безпеки для мобільних додатків, вибору відповідних інструментів для тестування

мобільних додатків і визначення, чи є мобільний додаток прийнятним для розгортання на мобільних пристроях організації. Наводиться огляд технік, які зазвичай використовуються професіоналами з забезпечення якості програмного забезпечення, включаючи методи тестування на виявлення конкретних вразливостей програмного забезпечення та неправильних конфігурацій, пов'язаних з мобільним програмним забезпеченням [30].

- PCI Mobile Payment Acceptance Security Guidelines - надає важливі рекомендації та найкращі практики для забезпечення безпеки прийому мобільних платежів і дотримання стандартів PCI. Розроблений Радою зі стандартів безпеки індустрії платіжних карток (PCI SSC), цей документ корисний для розробників, торговців і постачальників послуг, оскільки він надає інструменти для захисту чутливих платіжних даних і забезпечення відповідності галузевим стандартам [31].

Ці нормативні документи та стандарти допомагають забезпечити надійний захист мобільних додатків, зменшити ризики вразливостей та підтримувати високий рівень безпеки в інформаційних системах організацій. PCI Mobile Payment Acceptance Security Guidelines, містить у собі всі актуальні способи захисту мобільних додатків, а також надає конкретні рекомендації щодо впровадження цих заходів безпеки. Саме цей документ буде розглянуто далі, адже він містить у собі ключові аспекти цього керівництва, включаючи вимоги до апаратного та програмного забезпечення, методи шифрування та інші важливі заходи безпеки, які необхідно враховувати під час розробки та використання мобільних платіжних рішень.

Метою цього документа є освіта зацікавлених сторін, відповідальних за архітектуру, дизайн і розробку мобільних додатків та їх середовище на мобільних пристроях, які можуть використовуватися торговцями для прийому платежів. Розробники та виробники можуть використовувати ці рекомендації для розробки відповідних заходів безпеки у своїх програмних та апаратних продуктах. Ці заходи можуть бути застосовані до середовища прийому мобільних платежів, підтримуючи впровадження більш безпечних рішень.

Цей документ зосереджується на двох аспектах: заходах безпеки, які можуть бути забезпечені технологією в сучасному середовищі, та заходах безпеки, призначених для надання рекомендацій і вказівок щодо проектування додатків для прийому мобільних платежів та їх середовища на мобільному пристрої.

Будь-який ризик, який існує на стандартному настільному комп'ютері або ноутбуці, також існує на мобільному пристрої. Крім того, мобільні пристрої можуть мати ширший набір функцій, ніж стандартні настільні комп'ютери та ноутбуки, що збільшує можливість появи нових вразливостей безпеки. Разом зі стандартними методами зв'язку, які використовуються в традиційних настільних комп'ютерах та ноутбуках, мобільні пристрої можуть також включати кілька стільникових технологій, GPS, Bluetooth, інфрачервоний і технології ближнього поля (NFC). Хоча це вже не є унікальним для мобільних пристроїв, датчики, такі як камери та мікрофони, також мають вразливості та ризики, пов'язані з їх використанням. Ризик також збільшується через використання знімних носіїв (наприклад, SIM-карт і SD-карт), внутрішньої електроніки, що використовується для тестування виробником, вбудованих датчиків і біометричних зчитувачів. Крім того, конфігурації логування та налагодження на рівні виробника та оператора мережі можуть додавати додаткові ризики.

Ризики безпеки також притаманні життєвому циклу розробки та інфраструктурі, пов'язаній з мобільними пристроями. Наприклад, виробник оригінального обладнання, розробник програмного забезпечення операційної системи, розробник додатків, інтегратор, реселер, оператор мобільної мережі і постачальник рішень для прийому мобільних платежів - усі вони відіграють роль у забезпеченні загальної безпеки мобільного пристрою. Деякі розробники беруть участь у кількох етапах процесу розробки, що потенційно полегшує їм вирішення різних аспектів безпеки пристрою від рівня кремнію до додатків, які працюють на операційній системі; інші зацікавлені сторони беруть участь лише в одному етапі розробки безпеки. Інші треті сторони можуть вводити ризики безпеки через драйвери пристроїв, мобільні додатки, периферійне обладнання та знімні носії. Усі ці фактори можуть бути потенційними шляхами для

несанкціонованого доступу до операцій пристрою або несанкціонованого розкриття даних рахунків. Визначення відповідальних за конкретні найкращі практики може бути складним через велику кількість учасників розробки мобільного пристрою.

Документ визначає три типи ризиків, що пов'язані з мобільними платіжними транзакціями:

1. Введення даних рахунку в пристрій;
2. Зберігання даних рахунку в пристрої;
3. Вихід даних рахунку з пристрою.

Для кожного з трьох ризиків наведено ціль з відповідними керівництвами:

- ***Запобігання перехопленню даних рахунку.*** Головна ціль - запобігти перехопленню даних рахунку при введенні їх у мобільний пристрій:
    - Переконатися, що дані рахунку адекватно шифруються перед введенням у мобільний пристрій.
    - Забезпечити наявність довірчого шляху між механізмами введення даних (наприклад, клавіатурою або читачем карт) та мобільним пристроєм. Це забезпечує, що дані рахунку не можуть бути перехоплені несанкціонованою стороною. Один із способів досягнення цього — використання надійного середовища виконання, що обмежує доступ між механізмом, що отримує дані рахунку, та захищеною пам'яттю всередині пристрою.
    - Якщо зовнішній пристрій використовується для введення даних рахунку у мобільний пристрій, забезпечити, що цей пристрій має засоби для підтвердження свого права на спілкування з мобільним пристроєм.
    - Якщо зовнішній пристрій працює по бездротовому каналу (наприклад, Wi-Fi або Bluetooth), забезпечити захист каналу сильною криптографією.
    - Незалежно від використовуваного процесу, забезпечити, що канал введення даних рахунку захищений від впроваджень на

клієнтському боці, таких як буферні переповнення, невідповідність типів даних, вбудований код або інші непередбачувані дані, а також зловживання або несанкціоновані додатки та сервіси на мобільному пристрої.

- Якщо є можливість введення PIN-коду, забезпечити, щоб воно здійснювалось лише через PIN-пристрій, схвалений PCI PTS, або рішення, схвалене платіжними брендами.

- ***Обробка даних рахунку в довірчому середовищі виконання.*** Головна ціль - забезпечити, що дані рахунку обробляються лише в межах довірчого середовища виконання:
  - Використовувати довірче середовище виконання, що забезпечує захист від витоку даних, зміна рівня безпеки може залежати від конкретної технології.
  - Запобігти доступу до даних рахунку за межі довірчого середовища виконання.
  - Тимчасове зберігання даних рахунку перед обробкою і авторизацією проводити в безпечному середовищі зберігання, наприклад, захищеному елементі, для запобігання підслуховуванню третіми особами.
  - Якщо дані рахунку зберігаються на мобільному пристрої після авторизації, забезпечити, що ці дані зберігаються у нерозчитуваному вигляді.
  - Усі пов'язані криптографічні ключі, які використовуються для шифрування даних рахунку, повинні управлятися таким чином, щоб вони не були доступні несанкціонованим особам, програмам або процесам.
- ***Запобігання перехопленню даних рахунку під час їх передачі.*** Головна ціль - запобігти перехопленню даних рахунку під час їх передачі з мобільного пристрою.

- Переконатися, що дані рахунку шифруються перед передачею, наприклад, за допомогою сильної симетричної або асиметричної криптографії, з довірчого середовища виконання мобільного пристрою.
- Забезпечити, що зашифровані дані рахунку передаються від довіреного джерела.
- Мобільні програми для прийому платежів піддаються різним типам атак, таким як MITM (Man-in-the-Middle) або пасивне підслуховування на компрометованому пристрої, погано реалізована криптографія або підслуховування через мобільну інфраструктуру.

Документ також визначає найкращі практики, яким необхідно слідувати розробникам. щоб забезпечити безпеку додатка:

1. Запобігти перехопленню даних рахунку під час введення їх у мобільний пристрій.
2. Запобігти компрометації даних рахунку під час їх обробки або зберігання у мобільному пристрої.
3. Запобігти перехопленню даних рахунку під час їх передачі з мобільного пристрою.
4. Запобігти несанкціонованому доступу до логічного пристрою.
5. Створити серверні контролі і повідомляти про несанкціонований доступ.
6. Запобігти підвищенню привілеїв.
7. Створити можливість віддаленого вимкнення платіжного додатку.
8. Виявляти крадіжку або втрату.
9. Підвищувати стійкість підтримуючих систем.
10. Переважати онлайн-транзакції.
11. Дотримуватися безпечного програмування, інженерії та тестування.
12. Захищати від відомих вразливостей.
13. Захищати мобільний пристрій від несанкціонованих додатків.
14. Захищати мобільний пристрій від шкідливих програм.
15. Захищати мобільний пристрій від несанкціонованих додатків.

16.Створити інструкційні матеріали для впровадження та використання.

17.Підтримувати безпечне зберігання чеків для продавців.

18.Забезпечити показник безпечного стану.

19.Забезпечити аудит і логування для доступу користувачів і пристроїв.

Ці практики також включають створення інструкційних матеріалів для користувачів, щоб забезпечити правильну і безпечну експлуатацію додатку. Вони орієнтовані на зменшення ризиків і негативних наслідків, пов'язаних з безпекою мобільного додатка, і спрямовані на забезпечення високого рівня захисту особистих даних і приватності користувачів. Такий комплексний підхід дозволяє ефективно впроваджувати сучасні стандарти безпеки і забезпечувати відповідність регуляторним вимогам та очікуванням користувачів.

Висновки до розділу 1

У цьому розділі було проведено детальний аналіз принципів функціонування мобільних додатків, що дозволило виявити ключові аспекти їх роботи та взаємодії з користувачем. Було розглянуто різноманітні атаки, які можуть бути здійснені на користувачів, та вразливості, які часто зустрічаються в мобільних додатках. Це дало змогу сформулювати рекомендації щодо підвищення безпеки та захисту даних. Також були розглянуті сучасні вимоги до безпеки мобільних додатків, що включають в себе захист від зовнішніх загроз та забезпечення конфіденційності користувачів. Висновки цього розділу підкреслюють необхідність комплексного підходу до розробки та тестування мобільних додатків, де безпека виступає як один з основних пріоритетів.

Розділ 2. ВИВЧЕННЯ СУЧАСНИХ МЕТОДІВ ТЕСТУВАННЯ ЗАХИЩЕНОСТІ МОБІЛЬНИХ ДОДАТКІВ

2.1 Мета та особливості тестування захищеності мобільних додатків.

У швидкозмінному цифровому світі сьогодення, мобільні додатки стали центральною частиною нашого щоденного життя, сприяючи всьому, від зв'язку та розваг до банківських послуг та медичної допомоги. Ця всеосяжна присутність підкреслює критичне значення ретельного тестування мобільних додатків для забезпечення їх функціональності, продуктивності, безпеки та безперешкодного користувацького досвіду.

Усі питання мобільної безпеки в основному стосуються зберігання особистих даних. Мобільний телефон, в прямому значенні слова, знає про людину все і зберігає багато чутливої інформації: зображення, відео, зразки голосу, нотатки, облікові дані, історію місць перебування та багато іншого. Сучасні мобільні операційні системи, такі як Android і iOS, надають розробникам багато можливостей для безпечного зберігання даних і мережевої взаємодії. А самі мобільні додатки використовують API, яке надається backend'ом і яке також потрібно тестувати разом з мобільним додатком. Проте, щоб ці інструменти були дійсно ефективними, їх потрібно використовувати грамотно. Зберігання даних, міжпроцесне взаємодія, правильне використання криптографічних ключів і безпечна мережева взаємодія — на будь-якому з цих етапів можна допустити помилку, яка обійдеться додатку безпекою. Пентест мобільних додатків допомагає зрозуміти, чи існує можливість несанкціонованого доступу до даних. Загалом термін "Тестування на проникнення", можна описати наступним чином: Тестування на проникнення - це метод оцінювання захищеності комп'ютерної системи чи мережі шляхом часткового моделювання дій зовнішніх зловмисників з проникнення у неї (які не мають авторизованих засобів доступу до системи) і внутрішніх зловмисників (які мають певний рівень санкціонованого доступу). Цей процес включає активний аналіз системи з виявлення будь-якої потенційної вразливості, що може виникати внаслідок неправильної конфігурації системи, відомих і невідомих дефектів апаратних засобів та програмного забезпечення, чи

оперативне відставання в процедурних чи технічних контрзаходах. Цей аналіз проводиться з позиції потенційного нападника і може включати активне використання вразливостей.

Під час тестування захищеності мобільного додатку важливо провести ряд етапів, які дозволяють ретельно перевірити його безпеку та надійність, серед них:

1. ***Підготовка***, яка є етапом визначення обсягу тестування безпеки, що включає ідентифікацію використовуваних засобів захисту, цілей тестування та конфіденційних даних. Загалом, етап підготовки є синхронізацією з клієнтом, під час якої досягається угода між тестувальником і клієнтом для захисту тестувальника від юридичних дій.
2. ***Збір інформації*** – це етап аналізу обсягу та архітектури додатка для отримання загального уявлення про додаток.
3. ***Аналіз додатка*** – наступний крок для завершення попереднього етапу, що включає автоматичне та ручне сканування додатків. Аналізя може забезпечити глибше розуміння додатка, який буде тестуватися, наприклад, входи, дані, що зберігаються, та інші потенційні основні слабкі місця.
4. ***Експлуатація*** – це етап, на якому тестувальники безпеки проникають у додаток, використовуючи виявлені під час попереднього процесу слабкі місця.
5. ***Звітність*** – важливий етап для клієнта, на якому тестувальники безпеки надають звіти про слабкі місця в додатках і можуть пояснити негативні наслідки слабких місць додатків.

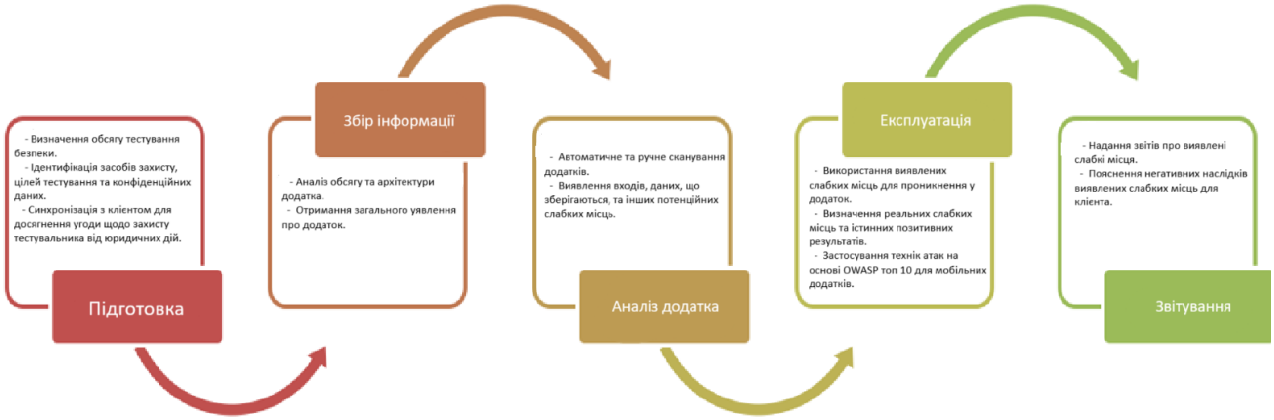

Рис. 2.1 Етапи тестування на проникнення.

Всі ці етапи разом створюють комплексний підхід до тестування захищеності мобільного додатка, що дозволяє виявити широкий спектр потенційних вразливостей. Вони допомагають забезпечити не лише виявлення вразливостей на різних рівнях додатка, а й зрозуміти їх кореневі причини та потенційні наслідки для безпеки та надійності програми. Такий глибокий аналіз дозволяє розробникам прийняти необхідні заходи для виправлення виявлених проблем і підвищення рівня безпеки мобільного додатка перед його випуском на ринок.

Не менш важливим аспектом є і метод, за яким проводиться тестування додатка, яке зазвичай розрізняють на три типи: Black-box, White-box та Gray-box. Кожен із них дозволяє досліджувати додаток з різних точок зору. Black-box тестування проводиться без попередньої інформації про додаток, іноді називаючи його "тестуванням з нульовими знаннями", щоб дозволити тестувальникові діяти як потенційному атакувальнику. White-box тестування, навпаки, передбачає повне знання про додаток, включаючи його вихідний код та документацію, що дозволяє створювати більш складні тестові сценарії. Gray-box тестування - це проміжний варіант, де тестувальнику надається певна обмежена інформація, що дозволяє зберегти баланс між ефективністю та охопленням тестування. Цей метод є найбільш поширеним у сфері безпеки програмного забезпечення.

Отже, якщо підсумувати, основною метою тестування на проникнення додатків є ідентифікація потенційних уразливостей, які можуть бути використані зловмисниками для несанкціонованого доступу до додатку або користувацьких даних. Цей процес включає в себе глибокий аналіз коду додатка, використання його функціональностей та комунікацій з серверами, а також дослідження можливих шляхів атаки.

2.2 Аналіз існуючих методологій тестування мобільних додатків.

Наразі існує декілька фреймворків, які допомагають аналізувати безпеку додатків для Android серед них Mobile Security Framework та Open Web Application Security Project Mobile Application Security Testing Guide. Ці фреймворки надають чіткі результати аналізу безпеки, які можуть бути основою для покращення безпеки для розробників додатків для Android.

Mobile Security Framework (MobSF) - є автоматизованим фреймворком для тестування мобільних додатків з відкритим кодом (Android, iOS і Windows). Цей фреймворк проводить аналіз на наявність шкідливого програмного забезпечення та тестування на проникнення, а також може виявляти вразливості, які можуть бути використані зловмисниками [33].

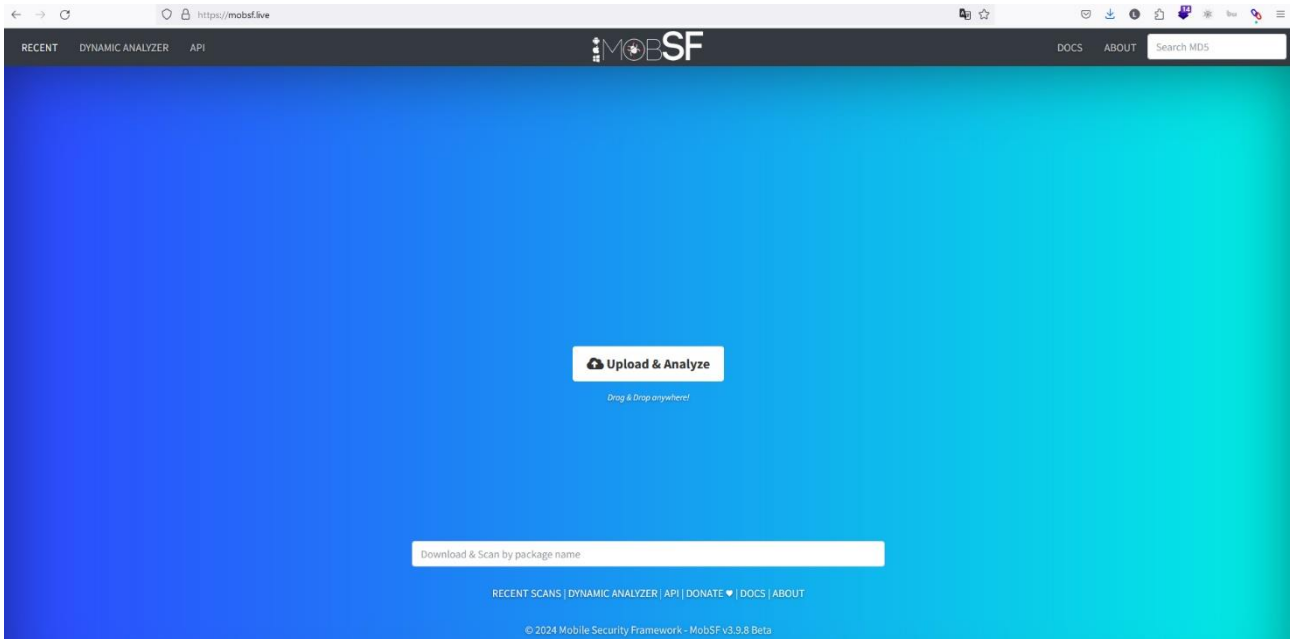

Рис. 2.2 Головна сторінка MobSF.

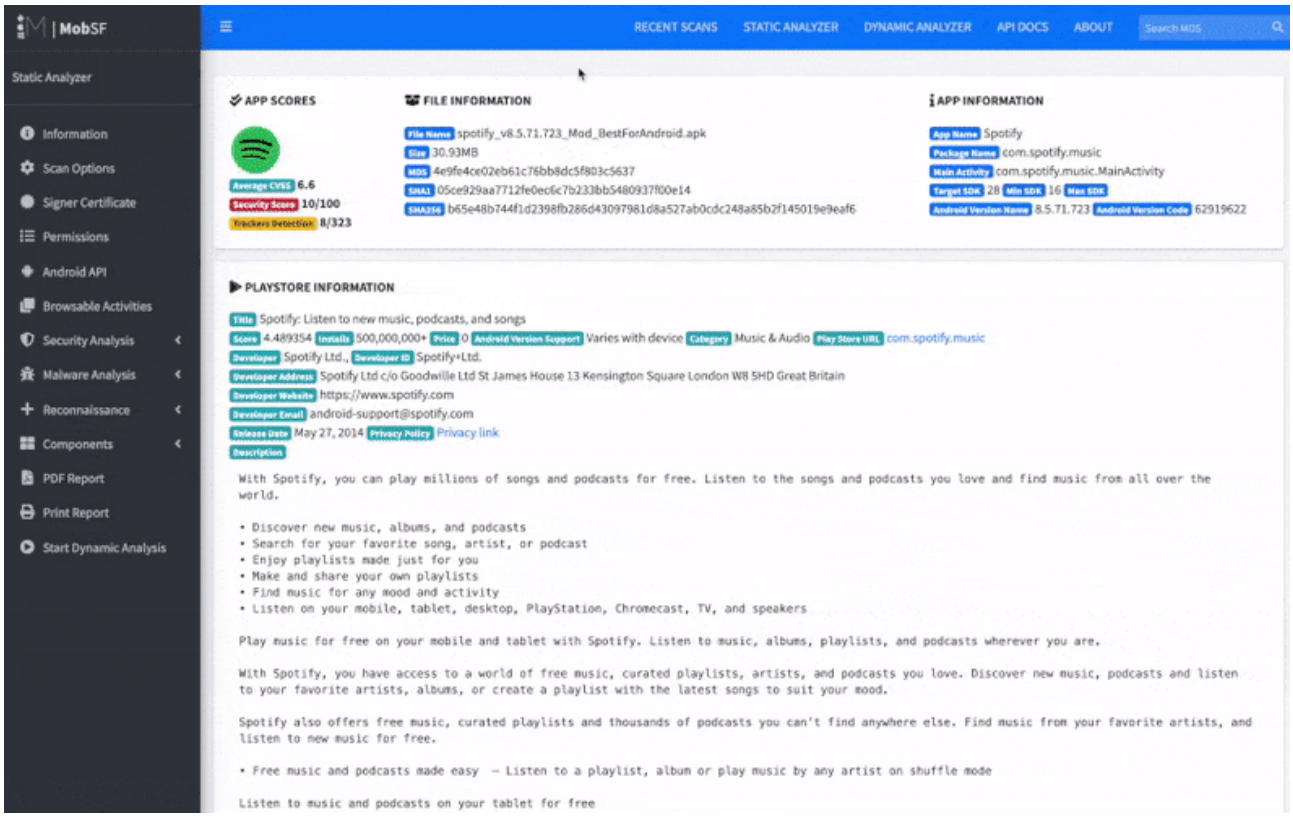

Рис. 2.3 Приклад роботи фреймворку.

OWASP MASTG [34] є стандартом безпеки, який використовується в мобільних додатках, і супроводжується комплексним керівництвом з тестування, яке включає процеси, техніки, інструменти та приклади проведення тестів безпеки мобільних додатків, він допомагає аналізувати та категоризувати ризики безпеки мобільних додатків на основі його рекомендацій. Цей фреймворк базується на двох документах створених OWASP:

- ***OWASP Risk Rating Methodology*** - це простий підхід до розрахунку та оцінки ризиків, пов'язаних із додатком. Використовуючи цей метод, можна вирішити, що робити з цими ризиками, розрахувавши визначені фактори відповідно до стандартів OWASP [35]. Методологія оцінки ризиків OWASP включає 6 етапів:
    - ***Ідентифікація ризику***: Це перший крок у процесі методології оцінки ризиків OWASP. На цьому етапі необхідно оцінити ризики безпеки та зібрати інформацію про агентів загроз, які будуть

використані атаки, вразливості та вплив успішних експлойтів на бізнес.

- ***Фактор оцінки ймовірності***: Це другий крок у процесі оцінки ризиків OWASP. Цей етап спрямований на прогнозування ймовірності успішної атаки з боку зловмисника. Процес оцінки проводиться після завершення ідентифікації ризику.
- ***Фактор оцінки впливу:*** Це третій крок у процесі оцінки ризиків OWASP. Цей етап спрямований на врахування впливу успішної атаки. OWASP визначає два типи впливу, які слід враховувати: технічний вплив і вплив на бізнес.
- ***Визначення серйозності ризику***: Це четвертий крок у процесі оцінки ризиків OWASP. Цей етап спрямований на визначення рівня серйозності ризику. OWASP надає шкалу від 0 до 9, розділену на 3 частини:
  - 0 до <3 - низький вплив
  - 3 до <6 - середній вплив
  - 6 до 9 - високий вплив
- ***Вирішення, що виправити:*** Цей крок включає прийняття рішення щодо того, які ризики потрібно виправити першочергово.

Це п'ятий крок у процесі методології оцінки ризиків OWASP. Цей етап спрямований на визначення, що потрібно виправити з раніше знайдених ризиків. Після класифікації ризиків для додатка буде створено пріоритетний список того, що потрібно виправити. В OWASP загальне правило полягає в тому, що першочергово слід виправляти найбільш серйозні ризики. OWASP доходить висновку, що не всі ризики варто виправляти, і деякі втрати не тільки очікувані, але й виправдані з огляду на вартість усунення проблеми.

- ***Налаштування моделі оцінки ризиків:*** Це необов'язковий крок у процесі методології оцінки ризиків OWASP. Цей етап спрямований на

встановлення налаштованої моделі оцінки ризиків, що є важливим для впровадження в бізнес-контексті.

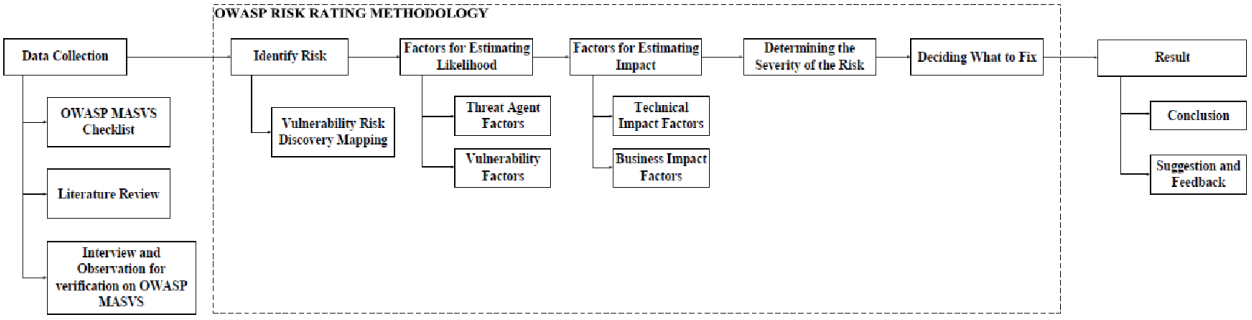

Рис. 2.4 Схематичне пояснення OWASP Risk Rating Methodology.

- ***OWASP MASVS*** - цей стандарт має на меті стандартизувати вимоги, використовуючи рівні верифікації, які підходять для різних сценаріїв загроз (Francisco та ін., 2019). MASVS може служити цінним орієнтиром при проведенні тестування безпеки додатків.У контрольному списку OWASP MASVS є 8 важливих вимог, які потрібно протестувати: архітектурний дизайн та архітектура загроз, зберігання даних та конфіденційність, криптографія, аутентифікація та управління сеансами, мережеве зв'язування, взаємодія з платформою, якість коду та налаштування збірки, стійкість [36].

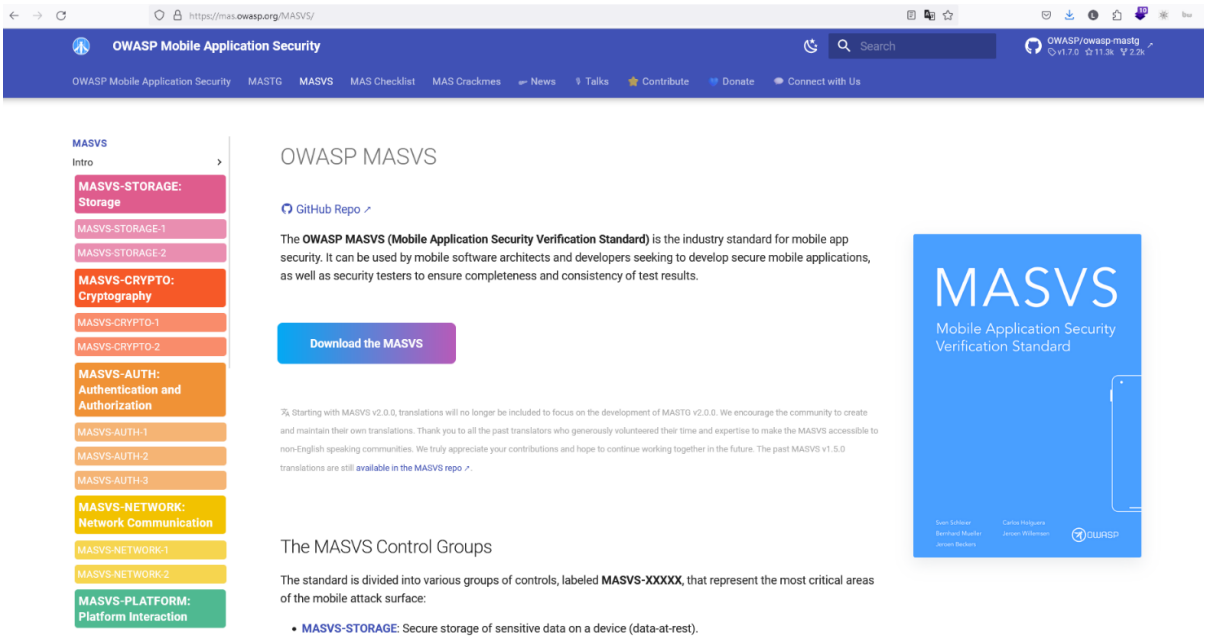

Рис. 2.5 Головна сторінка OWASP MASVS.

Загальна оцінка ефективності MobSF та OWASP MASTG підтверджує їх важливий внесок у покращення безпеки мобільних додатків для платформи Android. MobSF, як автоматизований фреймворк, проводить комплексний аналіз на наявність шкідливого програмного забезпечення, потенційні проникнення та вразливості, що можуть використовуватися зловмисниками.

З іншого боку, OWASP MASTG забезпечує стандартні процедури та керівництво для тестування безпеки мобільних додатків, використовуючи детальну методологію оцінки ризиків. Цей підхід дозволяє ідентифікувати загрози, оцінювати ймовірність та потенційний вплив атак, та раціонально вибирати пріоритетні заходи щодо зменшення ризиків. OWASP MASTG також створює міцну основу для реалізації вимог безпеки через стандартизацію верифікації, що відповідає різним сценаріям загроз.

Обидва фреймворки разом надають розробникам необхідні інструменти та керівництво для ефективного вдосконалення безпеки додатків для Android. Вони сприяють не лише зниженню загроз і ризиків, але й покращують якість програмного забезпечення, що відкриває шлях до підвищення довіри користувачів до мобільних додатків. Окрім того, використання цих фреймворків може значно зменшити потенційні витрати, пов'язані з реагуванням на безпекові інциденти, оскільки дозволяє виявляти і усувати проблеми ще до випуску додатків на ринок.

2.3 Аналіз існуючих програмних інструментів та автоматизованих сканерів з метою підвищення ефективності процесу тестування.

Під час тестування захищенності мобільних додатків тестувальники використовують несчисленну кількість утиліт та застосунків, які спрощують та скорочують проведення тестування додатку. Ці утиліти включають в себе різноманітні сканери вразливостей, які допомагають автоматизувати процес пошуку потенційних проблем безпеки, таких як недопущені відкриті порти, неправильне керування сесією або недоліки захисту даних. Крім того, тестувальники використовують спеціалізовані утиліти для перевірки

захищеності додатків від основних класів атак, таких як SQL-ін'єкції, перехоплення трафіку або вразливості в аутентифікаційних механізмах.

Деякі інструменти дозволяють глибоко аналізувати логіку додатку, здійснювати перехоплення трафіку між клієнтом і сервером, а також відтворювати складні атаки на сторону клієнта і сервера для оцінки вразливостей і стійкості додатку до них. Деякі інші утиліти дозволяють перевіряти веб-сервіси, на яких базується мобільний додаток, для виявлення проблем безпеки, пов'язаних з конфігурацією сервера або наявністю вразливостей в програмному забезпеченні.

Крім того, є спеціалізовані утиліти для аналізу витоку інформації та перевірки на наявність несанкціонованого збору даних. Ці утиліти можуть виявляти потенційні вразливості, які можуть призвести до порушення конфіденційності або цілісності даних, що обробляються мобільним додатком.

Всі ці інструменти і утиліти значно полегшують та прискорюють процес тестування захищенності мобільних додатків, дозволяючи зосередитися на виявленні та усуненні критичних вразливостей і забезпеченні найвищого рівня безпеки для користувачів.

Серед популярних утиліт можна виділити наступні:

1. ***MobSF*** **-** це відкритий фреймворк, призначений спеціально для тестування та аналізу безпеки мобільних додатків. Він надає широкий спектр функцій, включаючи статичний та динамічний аналіз, тестування веб-API та аналіз шкідливих програм. Сьогодні ми зосередимося на використанні можливостей статичного аналізу MobSF для виявлення потенційних вразливостей безпеки в мобільних додатках.

Автоматизоване сканування безпеки, яке забезпечує MobSF, включає в себе декілька значущих переваг. Перш за все, він зменшує ризики кіберзагроз шляхом автоматичної перевірки apk-файлів на предмет вразливостей, потоків даних та інших критичних питань безпеки. Такий підхід значно спрощує процес виявлення потенційних загроз ще до того, як додаток буде випущений.

Додатково, MobSF використовує динамічний аналіз, що означає, що він може аналізувати додаток в реальному часі або під час його виконання. Це дозволяє виявляти потенційні вразливості, які можуть виникнути тільки під час взаємодії з додатком у реальному середовищі.

Ще однією суттєвою перевагою є статичний аналіз, який використовується для перевірки вихідного коду, бінарних файлів та конфігурацій. Цей вид аналізу дозволяє виявити потенційні проблеми без необхідності виконувати додаток.

MobSF також надає можливість для реверс-інжинірингу, що дозволяє детально перевіряти код строка за строкою, а також проводити декомпіляцію, розбір та налагодження додатків. Це забезпечує глибокий інсайт у внутрішню структуру додатка і дозволяє виявити складні вразливості.

Крім того, MobSF має налаштовувану та адаптовану природу, яка дозволяє підлаштувати його до ваших конкретних потреб і стандартів безпеки. Це забезпечує відповідність вашого додатка найсучаснішим стандартам безпеки та захищає дані користувачів від потенційних загроз.

У підсумку, MobSF виявляється потужним інструментом для автоматизованого аналізу безпеки, який об'єднує в собі кілька методів аналізу, що дозволяють ефективно виявляти і вирішувати проблеми безпеки в додатках.

Як вже було сказано MobSF дозволяє проводити статичне сканування додатку та динамічне. Функція статичного аналізу дозволяє провести тестування коду мобільного додатка без його виконання, динамічне ж сканування означає аналіз поведінки мобільного додатка під час його виконання. Це допомагає ідентифікувати вразливості в реальному часі або під час роботи програми. Коли ми завантажуємо додаток до MobSF і встановлюємо його на пристрій чи емулятор, MobSF захоплює та аналізує мережевий трафік, який генерується додатком. Він досліджує відвідані URL-адреси, параметри в запитах і відповіді сервера. Результати аналізу надають детальні усвідомлення про мережеві взаємодії додатка, що допомагає в ідентифікації потенційних проблем з безпекою.

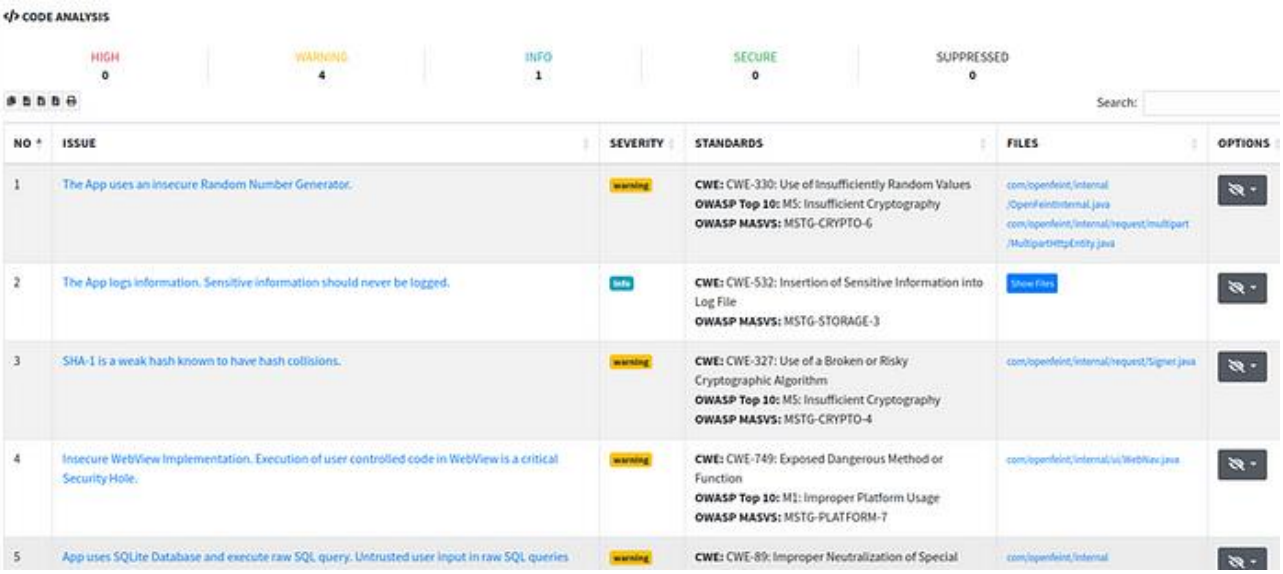

Рис. 2.6 Приклад звіту для статичного аналізу.

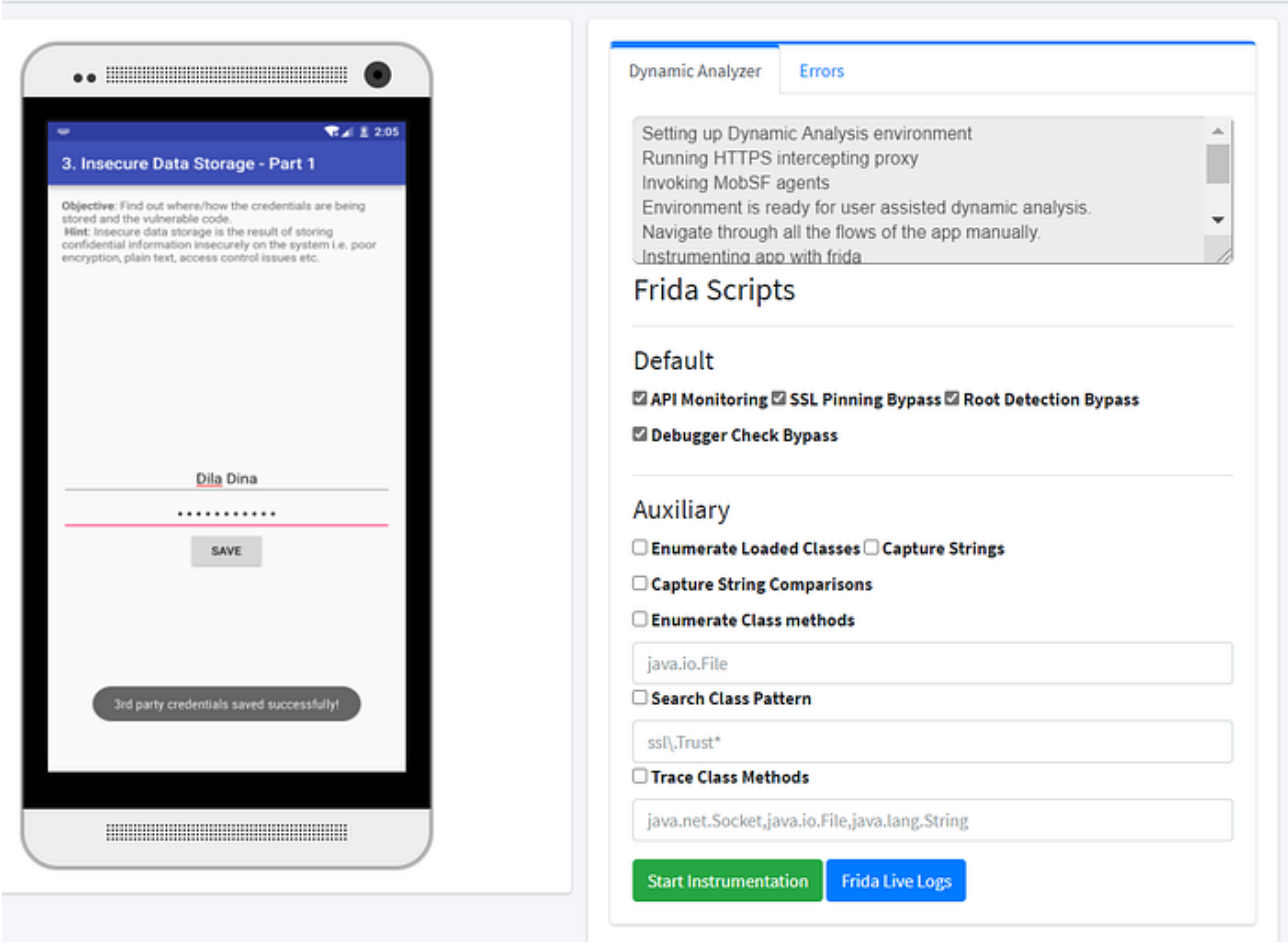

Рис. 2.7 Приклад роботи динамічного аналізу у MobSF.

2. ***Mariana Trench*** — це утиліта для автоматизованої верифікації безпеки додатків, розроблена командою інженерів Facebook. Вона призначена для

ідентифікації потенційних уразливостей безпеки, що можуть виникнути в додатках, що працюють на платформі Facebook. Утиліта використовує різні методи аналізу коду, такі як статичний аналіз, символьне виконання та аналіз протоколів, щоб виявляти проблеми з безпекою даних, неправильні перевірки доступу, потенційні XSS атаки та інші вразливості [37].

Основні особливості Mariana Trench включають автоматизацію процесу виявлення уразливостей, інтеграцію з іншими інструментами безпеки Facebook, підтримку як внутрішніх, так і зовнішніх розробників, а також генерацію автоматизованих звітів і рекомендацій з усунення виявлених проблем. Цей інструмент активно вдосконалюється та підтримується для того, щоб ефективно відповідати новим методам атак і вимогам до безпеки, забезпечуючи високий рівень захисту для додатків, які працюють на платформі Facebook.

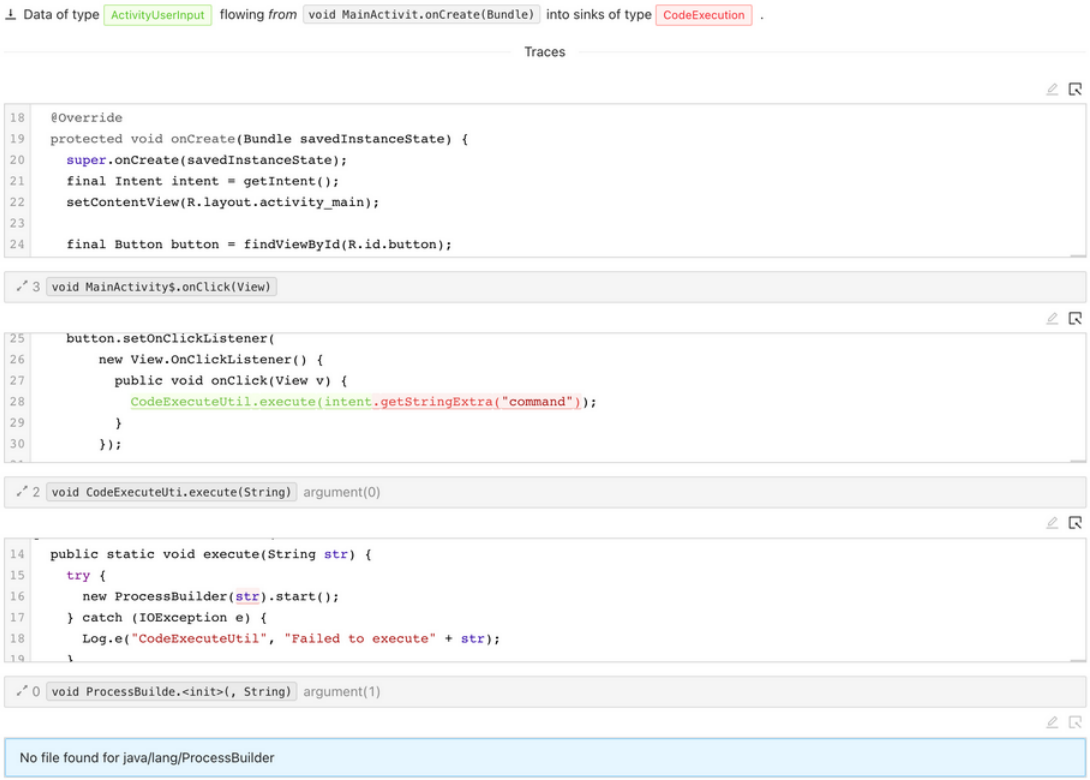

Рис. 2.8 Приклад звіту за результатами сканування.

3. ***Frida*** є висококласним інструментом для динамічної інструменталізації, який дозволяє розробникам, інженерам-зворотникам та дослідникам безпеки спостерігати та перепрограмувати програми в реальному часі на різних

платформах. Цей інструмент може вводити фрагменти JavaScript або власні бібліотеки в нативні додатки, що робить його подібним до Greasemonkey для нативних застосунків. Frida є безкоштовним програмним забезпеченням і постійно підтримується спільнотою, що робить його доступним для широкого кола користувачів [38].

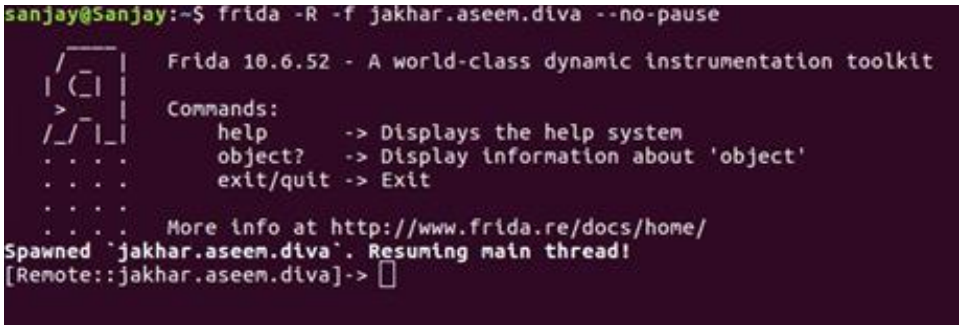

Рис. 2.9 Застосування утиліти Frida.

4. ***Objection*** - це інструмент, який працює на основі Frida і призначений для допомоги в автоматизації вразливих перевірок на мобільних пристроях, зокрема для iOS та Android. Він дозволяє виконувати різноманітні завдання, такі як перехоплення трафіку, маніпулювання середовищем виконання додатків та багато іншого, що робить його корисним інструментом для тестування безпеки мобільних додатків [39].

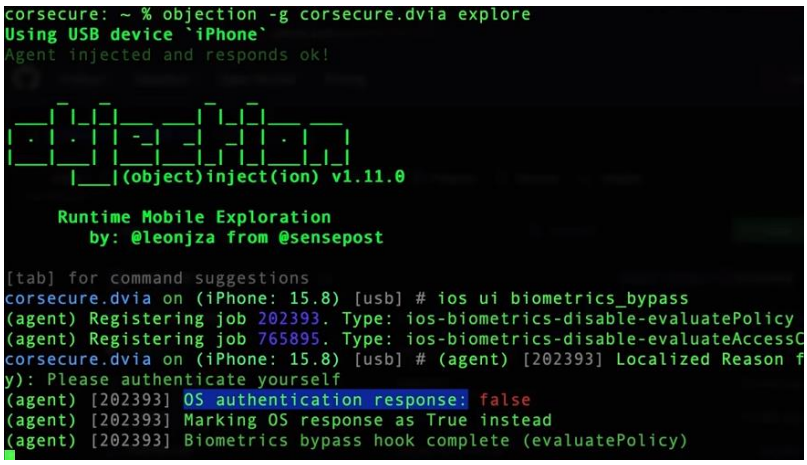

Рис. 2.10 Застосування утиліти Objection.

Обидва інструменти є потужними ресурсами для розробників та фахівців з безпеки, які хочуть глибше зрозуміти та вплинути на поведінку програмного

забезпечення в реальному часі. Вони надають можливість втручання в процеси, які виконуються, і зміни їх поведінки без необхідності доступу до вихідного коду, що є великою перевагою в сфері реверс-інженерії та безпеки.

5. ***APKiD*** - це інструмент для ідентифікації Android-додатків, який використовується в пентестінгу для визначення компіляторів, пакерів, обфускаторів та інших особливостей APK-файлів. Цей інструмент може бути корисним для виявлення піратських та шкідливих додатків, а також для аналізу захисних механізмів, які використовуються в додатках. APKiD допомагає розуміти, яким чином було створено APK, що може надати цінну інформацію під час проведення пентесту.

Використання APKiD дозволяє пентестерам визначити, чи було застосовано обфускацію коду, що може вказувати на спроби приховати потенційно шкідливу поведінку або захистити код від аналізу. Крім того, інструмент може виявити використання пакерів, які ускладнюють реверс-інженіринг APK-файлів, що також може бути ознакою шкідливих намірів розробників.

APKiD підтримується спільнотою і постійно оновлюється, щоб включати нові правила для виявлення найновіших методів обфускації та пакування. Це робить його незамінним інструментом у сучасному пентестінгу мобільних додатків. Він доступний під двома ліцензіями: комерційною, яка підходить для закритих проектів, та GPL-ліцензією, яка може бути використана в проектах з відкритим кодом [40].

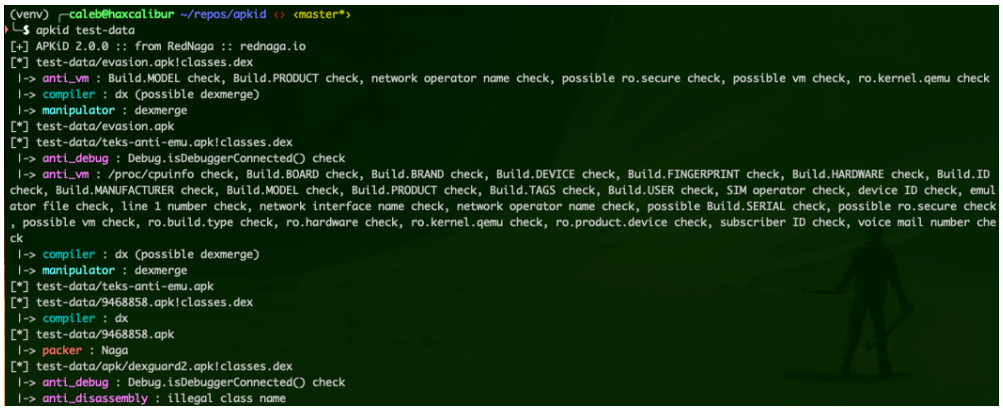

Рис. 2.11 Застосування утиліти APKiD.

Висновки до розділу 2

У цьому розділі кваліфікаційної роботи було розглянуто ключові аспекти тестування захищеності мобільних додатків. Встановлено, що мета тестування полягає не лише у виявленні потенційних вразливостей, а й у забезпеченні цілісності та конфіденційності даних користувачів. Аналіз існуючих методологій показав, що тестування на проникнення є важливим елементом у виявленні слабких місць, які можуть бути недоступні при стандартних методах тестування.

Дослідження програмних інструментів та автоматизованих сканерів, таких як MobSF, Mariana Trench, Frida, Objection та APKiD, демонструє, що використання спеціалізованих інструментів може значно підвищити ефективність тестування. Ці інструменти дозволяють проводити глибокий аналіз коду, виявляти складні вразливості та автоматизувати процес виявлення загроз, що в свою чергу сприяє підвищенню рівня безпеки мобільних додатків.

У підсумку, розділ підкреслює важливість комплексного підходу до тестування захищеності мобільних додатків, який включає в себе як теоретичний аналіз, так і практичне застосування спеціалізованих інструментів. Такий підхід дозволяє не тільки виявляти існуючі вразливості, а й прогнозувати потенційні загрози, забезпечуючи більш високий рівень захисту користувачів мобільних додатків.

Розділ 3. РОЗРОБКА РЕКОМЕНДАЦІЙ ДЛЯ ТЕСТУВАННЯ ЗАХИЩЕННОСТІ МОБІЛЬНИХ ДОДАТКІВ

3.1 Формулювання завдання

Для проведення якісного тестування на проникнення мобільних додатків необхідно мати чіткий та послідовний план, якому буде слідувати тестувальник. Тому було вирішено розробити рекомендації, які будуть побудовані на основі етапів тестування, відомих технік атак та популярних утиліт. Таким чином, завдання буде звучати наступним чином:

- розробити рекомендації для тестування мобільних додатків на проникнення, які будуть поєднувати у собі вже уснуючі етапи тестування, методології, програмні застосунки та утиліти, що в результаті дозволить тестувальникам на проникнення комплексно протестувати всі аспекти мобільного додатку, з метою забезпечення кібербезпеки користувачів додатку.

Розроблені рекомендації повинні охопити наступні аспекти:

- етапи тестування на проникнення мобільних додатків;
- методи тестування згідно з OWASP MASTG;
- способи використання практичних утиліт та сканерів для виявлення вразливостей;

3.2 Розробка рекомендацій для тестування захищеності мобільних додатків

Тестування мобільних додатків проходить етапами, тому логічно розділити рекомендації відповідно до етапів, які використовують тестувальники. Зазвичай виділяють такі етапи:

1. ***Розвідка*** - це первинний аналіз додатку, визначення його функціональності та потенційних вразливостей.
2. ***Реверсивний аналіз*** - це процес, який включає в себе розбір та дослідження програмного коду додатку з метою виявлення його структури, використаних алгоритмів та інших технічних особливостей.
3. ***Статичний аналіз*** - оцінка коду додатку без його виконання, щоб виявити можливі помилки та недоліки.
4. ***Динамічний аналіз*** - тестування додатку в роботі, щоб перевірити його поведінку в реальних умовах.

Кожен з цих етапів вимагає ретельного планування та виконання, а також спеціалізованих інструментів та методик. Тому подальші рекомендації будуть надані в межах окремих етапів.

3.2.1 Розробка рекомендацій для тестування захищеності мобільних додатків в межах етапу «Розвідка»

Розвідка в контексті тестування на проникнення - це процес збору інформації про цільову систему або мережу з метою ідентифікації потенційних слабких місць та вразливостей. Цей етап є критично важливим, оскільки він дозволяє тестувальникам на проникнення розробити стратегію, яка буде найбільш ефективною для конкретної системи.

Під час розвідки, фахівці можуть використовувати різноманітні методи та інструменти для збору інформації, такі як публічні бази даних, пошукові системи, соціальні мережі, сканування портів, визначення версій програмного забезпечення, та інші. Основною метою є визначення архітектури мережі, працюючих сервісів та додатків, конфігурацій системи безпеки, а також ідентифікація осіб, які можуть мати доступ до системи.

Зібрана інформація може допомогти виявити, які системи можуть бути легкою мішенню для атак, та які методи атаки будуть найбільш ефективними. Наприклад, якщо в процесі розвідки виявлено, що певна версія операційної системи містить відому вразливість, тестувальник може використати цю інформацію для планування атаки.

Таким чином, розвідка дозволяє тестувальникам на проникнення бути більш цілеспрямованими у своїх діях, зосереджуючись на найбільш перспективних об'єктах та використовуючи інформацію про систему для підвищення шансів успішного тестування. Вона також допомагає уникнути зайвого витрачання часу та ресурсів на менш вразливі або несуттєві компоненти системи.

Отже рекомендації для проведення розвідки можуть бути наступними:

- ***Розвідка. Рекомендація 1: Збір загальнодоступної інформації.***

  При зборі загальнодоступної інформації важливо використовувати різноманітні джерела, такі як відкриті бази даних, публічні записи та соціальні мережі. Основна увага має бути зосереджена на зборі даних, які можуть виявити потенційні слабкі місця або тенденції розвитку цільового об'єкта.

- ***Розвідка. Рекомендація 2: Аналіз розвитку додатку.***

  Аналіз розвитку додатку передбачає вивчення його історії версій, оновлень, змін у функціоналі та відгуків користувачів. Це допоможе зрозуміти логіку розвитку продукту та можливі напрямки для подальшої розвідки.

- ***Розвідка. Рекомендація 3: Перевірка на наявність відомих вразливостей.***

  Перевірка на наявність відомих вразливостей вимагає використання спеціалізованих інструментів та баз даних, таких як CVE. Важливо регулярно оновлювати ці інструменти та слідкувати за останніми звітами про безпеку.

- ***Розвідка. Рекомендація 4: Глибоке вивчення функціоналу.***

Глибоке вивчення функціоналу включає в себе тестування всіх можливостей продукту, аналіз його коду, якщо це можливо, та вивчення документації. Це дозволить виявити неочевидні слабкі місця та потенційні точки входу для розвідки.

Важливо розуміти, що розвідка є ключовим елементом тестування на проникнення, оскільки вона дозволяє ідентифікувати потенційні слабкі місця та вразливості в системі. Використання різноманітних методів і інструментів, таких як аналіз публічних баз даних, сканування портів та визначення версій ПЗ, допомагає визначити архітектуру мережі, активні сервіси та конфігурації системи безпеки. Це, в свою чергу, дозволяє тестувальникам на проникнення розробити цілеспрямовану стратегію атаки, зосереджуючись на найбільш уразливих компонентах системи та ефективно використовуючи ресурси. Рекомендації для розвідки включають збір загальнодоступної інформації з різних джерел та аналіз отриманих даних для виявлення потенційних цілей атаки.

3.2.2 Розробка рекомендацій для тестування захищеності мобільних додатків в межах етапу «Реверсивний аналіз»

Реверсивний аналіз — це процес дослідження програмного забезпечення шляхом розбору його на складові частини з метою виявлення його структури, функціональності та робочих алгоритмів. У контексті тестування на проникнення мобільних додатків для Android, реверсивний аналіз використовується для виявлення вразливостей та потенційних точок входу для зловмисників. Цей метод дозволяє тестувальникам краще розуміти, як додаток взаємодіє з операційною системою, які бібліотеки використовуються, та які можливі слабкі місця можуть бути використані для несанкціонованого доступу.

Використання реверсивного аналізу в тестуванні на проникнення є важливим, оскільки воно допомагає виявити та усунути потенційні ризики безпеки до того, як додаток потрапить до кінцевого користувача. Це також сприяє розробці більш захищених додатків, оскільки розробники можуть

використовувати результати реверсивного аналізу для покращення коду та архітектури додатку.

Рекомендаці для проведення реверсивного аналізу додатку можуть бути наступними:

- ***Реверсивний аналіз. Рекомендація 1: Використання утиліти APKiD.***
  Ця утиліта дозволяє швидко ідентифікувати використані компілятори, упаковувачі, обфускатори, а також різноманітні механізми безпеки в APK файлах. Ця інформація є критично важливою для розуміння того, як захищений додаток, і може вказувати на потенційні слабкі місця, які можуть бути використані під час пентесту. Використання APKiD допомагає пентестерам ефективно планувати свої атаки та визначати оптимальні стратегії тестування.
- ***Реверсивний аналіз. Рекомендація 2: Використання утиліти APKTool.***
  APKTool дозволяє розпаковувати APK-файли, щоб вивчити вихідний код і ресурси додатка, а також модифікувати їх для подальшого аналізу. Це дає можливість виявити потенційні вразливості та зрозуміти логіку роботи додатка, що є ключовим для ефективного тестування на проникнення.
- ***Реверсивний аналіз. Рекомендація 3: Використання утиліти Dex2Jar.***
  Ця утиліта дозволяє конвертувати .dex файли, які є виконавчими файлами Android, у .class файли, що можуть бути архівовані як JAR.
- ***Реверсивний аналіз. Рекомендація 4: Використання утиліти JADX.***
  JADX - це потужний інструмент для реверсивного інжинірингу, який дозволяє перетворювати скомпільований код Android-додатків (DEX/Smali) у вихідний код Java. Використання JADX у тестуванні на проникнення може бути надзвичайно корисним, оскільки воно дозволяє аналітикам безпеки глибше зрозуміти логіку роботи додатка, виявити потенційні вразливості та точки входу для атак. З його допомогою можна ефективно аналізувати код на предмет небезпечних функцій та ненадійних реалізацій безпеки, що є ключовим аспектом під час пентесту.

Реверсивний аналіз являє собою критично важливий процес у контексті пентестування мобільних додатків, оскільки він дозволяє глибоко зануритися в структуру та функціональність програми, виявляючи потенційні вразливості та точки входу для атак. Цей метод надає тестувальникам необхідні знання для розуміння взаємодії додатку з операційною системою, використовуваними бібліотеками та можливими слабкими місцями, що можуть бути експлуатовані.

3.2.3 Розробка рекомендацій для тестування захищеності мобільних додатків в межах етапу «Сатичний аналіз»

Статичний аналіз — це процес виявлення помилок та вразливостей у коді програми без її виконання. Він використовується для перевірки коду на наявність потенційних проблем, таких як помилки безпеки, витоки пам'яті, недотримання стандартів кодування та інші типи помилок, які можуть бути виявлені без запуску програми.

Основна мета статичного аналізу полягає в підвищенні якості програмного забезпечення та зменшенні часу, необхідного для виявлення та виправлення помилок у процесі розробки. Це особливо важливо для мобільних додатків, де виправлення помилок після розгортання може бути складним та дорогим.

Статичний аналіз дозволяє розробникам і тестувальникам виявляти проблеми на ранніх етапах розробки, що сприяє створенню більш безпечного та надійного програмного забезпечення. Він також може бути інтегрований у процес неперервної інтеграції, що дозволяє автоматизувати перевірку коду та забезпечити його відповідність стандартам якості перед його розгортанням.

Статичний аналіз може бути проведений з урахуванням наступних рекомендацій:

- ***Статичний аналіз. Рекомендація 1: Пошук цікавих рядків у файлах.***

  При тестуванні на проникнення мобільних додатків, важливо звернути увагу на рядкові дані в APK-файлі. Це може включати паролі, URL-адреси, API-ключі, методи шифрування, UUID Bluetooth, токени та інші цікаві елементи. Також варто захардкоджені облікові дані адміністратора, які можуть бути вбудовані в додаток.

- ***Статичний аналіз. Рекомендація 2: Перевірка правильного налаштування Firebase.***

  Налаштування Firebase можуть бути неправильно сконфігуровані, що відкриває потенційні вразливості. Переконайтеся, що правила доступу до бази даних Firebase строго обмежують доступ та не дозволяють несанкціоноване отримання, зміну або видалення даних. Також важливо перевірити, чи не включені в код додатка жорстко закодовані ключі доступу до Firebase, які можуть бути використані зловмисниками для отримання доступу до чутливих даних.

- ***Статичний аналіз. Рекомендація 3: Аналіз файлу Manifest.***

  При тестуванні на проникнення мобільних додатків, важливо звернути увагу на файли Manifest.xml та strings.xml, оскільки вони можуть містити вразливості безпеки. Ось деякі рекомендації:

  - *Debuggable Applications*: Уникайте встановлення додатків як налагоджуваних (debuggable="true"), оскільки це може сприяти з'єднанням, які можуть бути використані для експлуатації.
  - *Backup Settings*: Забезпечте, щоб атрибут android:allowBackup був встановлений як "false" для додатків, що обробляють конфіденційну інформацію, щоб запобігти несанкціонованому резервному копіюванню даних через adb.

- *Network Security*: Використовуйте налаштування безпеки мережі для вказівки деталей безпеки, таких як піни сертифікатів та налаштування HTTP-трафіку.
- *Exported Activities and Services*: Ідентифікація експортованих активностей та служб може вказувати на компоненти, які можуть бути неправильно використані.
- *Content Providers and FileProviders*: Переконайтеся, що провайдери контенту та FileProviders налаштовані таким чином, щоб запобігти несанкціонованому доступу або зміні даних.
- *Broadcast Receivers and URL Schemes*: Зверніть увагу на компоненти, які можуть бути використані для експлуатації, особливо на управління URL-схемами для вразливостей вводу.
- *SDK Versions*: Атрибути minSdkVersion, targetSDKVersion та maxSdkVersion вказують підтримувані версії Android, тому важливо забезпечити, що додаток сумісний з останніми стандартами безпеки.

- ***Статичний аналіз. Рекомендація 4: Перевірка вразливості до Tapjacking.***

  Для виявлення мобільних додатків, що є вразливими до атаки типу "Tapjacking", необхідно перевірити експортовані активності в маніфесті Android. Зверніть увагу, що активність з інтент-фільтром автоматично вважається експортованою за замовчуванням. Після виявлення експортованих активностей слід перевірити, чи потрібні для них дозволи. Це важливо, оскільки зловмисна програма також потребуватиме цього дозволу. Таким чином, ви зможете ідентифікувати потенційні точки входу для атаки та вжити необхідних заходів безпеки для їх усунення.

- ***Статичний аналіз. Рекомендація 4: Перевірка вразливості до Hijacking.***

  У випадку, коли активність у додатку встановлена з параметром launchMode у singleTask і при цьому не визначено taskAffinity, існує ризик "захоплення завдання" (task hijacking). Це означає, що зловмисний додаток

може бути встановлений і, якщо він буде запущений до оригінального додатку, він може захопити його завдання. Таким чином, користувач може взаємодіяти зі зловмисним додатком, вважаючи, що він користується справжнім.

- ***Статичний аналіз. Рекомендація 5: Перевірка внутрішнього сховища.***
  У файлів, що зберігаються в інтерній пам'яті Android, є призначення бути доступними виключно для додатку, який їх створив. Цей захід безпеки забезпечується операційною системою Android і зазвичай задовольняє потреби в безпеці більшості додатків. Проте, розробники іноді використовують режими, такі як MODE_WORLD_READABLE та MODE_WORLD_WRITABLE, для спільного використання файлів між різними додатками. Однак, ці режими не обмежують доступ до цих файлів іншими додатками, включаючи потенційно шкідливі.
- ***Статичний аналіз. Рекомендація 6: Перевірка зовнішнього сховища.***
  При роботі з файлами на зовнішньому носії, такому як SD-карти, слід дотримуватися певних заходів безпеки. Файли на зовнішньому носії доступні для читання та запису всіма, що означає, що будь-який додаток або користувач може отримати до них доступ. З огляду на легкість доступу, не рекомендується зберігати конфіденційну інформацію на зовнішньому носії, оскільки він може бути вилучений або доступний для будь-якого додатку, що робить його менш безпечним. Завжди виконуйте перевірку вхідних даних для даних, отриманих із зовнішнього носія, оскільки ці дані походять з ненадійного джерела. Настійно не рекомендується зберігати виконувані файли або класи на зовнішньому носії для динамічного завантаження. Якщо додатку необхідно отримувати виконувані файли з зовнішнього носія, переконайтеся, що ці файли підписані та криптографічно перевірені перед їх динамічним завантаженням.
  - Доступ до зовнішнього сховища можна отримати за шляхом: /storage/emulated/0 , /sdcard , /mnt/sdcard

- Shared preferences: Android дозволяє кожній програмі легко зберігати файли xml в шляху/даних/даних/< packagename >/shared _ prefs/і іноді можна знайти конфіденційну інформацію в цій папці.
- Бази даних: Android дозволяє кожному додатку легко зберігати бази даних sqlite в шляху/data/data/< packagename >/databases/, і іноді можна знайти конфіденційну інформацію в цій папці.

- ***Статичний аналіз. Рекомендація 7: Перевірка як додаток приймає сертифікати.***

  При тестуванні на проникнення мобільних додатків, однією з виявлених вразливостей є прийняття всіх сертифікатів без перевірки. Розробники іноді налаштовують додатки таким чином, що вони приймають будь-які сертифікати SSL, навіть якщо ім'я хоста не відповідає сертифікату, що може бути продемонстровано за допомогою коду, який ігнорує перевірку імені хоста. Це створює потенційні ризики безпеки, оскільки додаток може стати вразливим до атак типу "людина посередині". Для виявлення такої вразливості можна використовувати проксі-сервери, такі як Burp Suite, для перехоплення трафіку, не авторизуючи сертифікат CA проксі-сервера на пристрої. Крім того, можливо створити сертифікат для іншого імені хоста за допомогою Burp Suite та використовувати його для перевірки реакції додатка на неправильні сертифікати.

- ***Статичний аналіз. Рекомендація 8: Перевірка управління ключами шифрування.***

  У деяких мобільних додатках розробники зберігають конфіденційні дані в локальному сховищі, шифруючи їх за допомогою ключа, який жорстко закодовано або передбачувано в коді програми. Такий підхід є небезпечним, оскільки зловмисники можуть використовувати методи реверс-інжинірингу для отримання доступу до цих даних. Це створює значні ризики для безпеки, оскільки витік конфіденційної інформації може призвести до серйозних наслідків для користувачів та компаній. Важливо,

щоб розробники використовували більш надійні методи управління ключами шифрування, щоб забезпечити захист даних.

- ***Статичний аналіз. Рекомендація 9: Використання застарілих алгоритмів.***

  У сфері мобільних додатків, використання застарілих або небезпечних алгоритмів для перевірки авторизації, зберігання та передачі даних може призвести до серйозних проблем з безпекою. Алгоритми, такі як RC4, MD4, MD5 та SHA1, вже давно вважаються ненадійними та легко піддаються атакам злому. Наприклад, при зберіганні паролів за допомогою хешування, слід використовувати методи, стійкі до атак грубої сили, з обов'язковим додаванням "солі", щоб забезпечити додатковий рівень захисту. Це допоможе запобігти можливості витоку конфіденційної інформації та забезпечить більш надійний захист даних користувачів.

- ***Статичний аналіз. Рекомендація 10: Аналіз виконання коду та використання нативних функцій.***

  Під час тестування на проникнення мобільних додатків, особливу увагу слід приділити механізмам виконання коду та використанню нативних функцій. Функції, які дозволяють виконання коду, такі як Runtime.exec(), ProcessBuilder(), та системні виклики через native code:system(), можуть бути потенційно вразливими точками, що дозволяють зловмисникам виконувати довільний код. Також, функції для відправлення SMS, включаючи sendTextMessage та sendMultipartTestMessage, потребують ретельного аналізу, оскільки вони можуть бути використані для неправомірного розсилання повідомлень. Нативні функції, оголошені як public native, System.loadLibrary, та System.load, також вимагають детального дослідження, адже неправильне їх використання може призвести до серйозних проблем з безпекою. Важливо звернути увагу на ці аспекти, щоб забезпечити надійність та безпеку мобільних додатків.

- ***Статичний аналіз. Рекомендація 11: Використання автоматизованого сканеру MobSF для статичного аналізу.***

  Статичний аналіз мобільних додатків за допомогою автоматизованого сканера MobSF є важливим етапом у процесі тестування на проникнення, який дозволяє виявити потенційні вразливості на ранніх стадіях розробки. Цей інструмент аналізує код додатка, бібліотеки та інші елементи без необхідності запуску додатка, надаючи розробникам можливість оцінити безпеку коду до його впровадження. MobSF використовується для ідентифікації загальновідомих вразливостей та недоліків у конфігурації безпеки, що допомагає командам розробників уникнути потенційних ризиків безпеки в майбутньому.

- ***Статичний аналіз. Рекомендація 12: Використання автоматизованого сканеру Mariana Trench для статичного аналізу.***

  Цей інструмент дозволяє ідентифікувати потенційні вразливості шляхом аналізу коду без його виконання, що є ефективним способом виявлення слабких місць на ранніх етапах розробки. Сканер здатен виявляти широкий спектр загроз, включаючи недоліки управління пам'яттю, небезпечні виклики функцій та інші проблеми, які можуть призвести до несанкціонованого доступу або витоку даних. Використання Mariana Trench дозволяє розробникам більш ефективно оцінювати безпеку мобільних додатків та забезпечувати вищий рівень захисту ще до їх випуску на ринок.

3.2.4 Розробка рекомендацій для тестування захищеності мобільних додатків в межах етапу «Динамічний аналіз»

Динамічний аналіз є процесом тестування програмного забезпечення, який виконується шляхом його реального запуску та виконання. Він дозволяє виявити помилки, які можуть не бути очевидними під час статичного аналізу, коли код не виконується. Динамічний аналіз важливий під час тестування на проникнення, оскільки він допомагає імітувати реальні умови експлуатації програми та виявляти потенційні вразливості, які можуть бути використані зловмисниками.

Під час тестування на проникнення, динамічний аналіз використовується для оцінки безпеки системи шляхом моделювання атак, які можуть бути здійснені зовнішніми або внутрішніми зловмисниками. Це включає в себе активний пошук вразливостей, які можуть виникнути через неправильну конфігурацію, дефекти обладнання чи програмного забезпечення, а також через застарілі процедури безпеки. Динамічний аналіз дозволяє також оцінити реальний вплив потенційних атак на бізнес та визначити необхідність технічних та процедурних контрзаходів для зниження ризиків.

В контексті мобільних додатків, динамічний аналіз може включати тестування на вразливості, які специфічні для мобільних платформ, такі як небезпечні API, проблеми зберігання даних, ненадійну обробку сесій та інші. Це допомагає забезпечити, що мобільні додатки є безпечними для кінцевих користувачів та не містять вразливостей, які можуть бути експлуатовані для отримання несанкціонованого доступу або витоку конфіденційної інформації.

Рекомендації для проведення динамічного аналізу можуть бути наступні:

- ***Динамічний аналіз. Рекомендація 1: Аналіз процесу логування додатку.*** Під час розробки мобільних додатків важливо уникати публічного викриття інформації для налагодження, оскільки це може призвести до витоку конфіденційних даних. Для моніторингу журналів додатків та ідентифікації захисту чутливої інформації рекомендуються інструменти pidcat та adb logcat. Pidcat забезпечує простоту використання та читабельність, що робить його переважним вибором для цих завдань.
- ***Динамічний аналіз. Рекомендація 2: Аналіз роботи додатку з буфером обміну.***

  Функціональність копіювання та вставки, яку забезпечує фреймворк буфера обміну Android, є зручною для користувачів, але водночас створює потенційні ризики безпеки. Це пов'язано з тим, що інші додатки можуть отримати доступ до буфера обміну та прочитати з нього дані, що може призвести до витоку конфіденційної інформації. Тому важливо вимкнути

функції копіювання/вставки в чутливих розділах додатків, таких як поля для введення даних кредитних карток, для запобігання витоку даних.

- ***Динамічний аналіз. Рекомендація 3: Аналіз процесу логування помилок додатку.***

  При виникненні збою в мобільному додатку та збереженні журналів, ці журнали можуть стати в нагоді зловмисникам, особливо якщо додаток не піддається реверсивному інжинірингу. Журнали можуть містити чутливу інформацію, яка допоможе атакуючим у виявленні потенційних вразливостей. Щоб зменшити цей ризик, слід уникати ведення журналів при збоях. У випадку, коли передача журналів через мережу є необхідною, важливо забезпечити їх передачу через захищений SSL канал, що гарантує безпеку переданих даних. Такий підхід дозволить знизити ризик компрометації інформації та підвищити загальний рівень безпеки мобільного додатку.

- ***Динамічний аналіз. Рекомендація 4: Аналіз витоку інформації.***

  У сучасному світі мобільні додатки часто інтегруються з сервісами, такими як Google Adsense, що може призвести до ненавмисного витоку конфіденційних даних через неправильну реалізацію розробниками. Щоб виявити потенційні витоки даних, важливо перехоплювати трафік додатка та перевіряти, чи не передається конфіденційна інформація третім особам. Цей процес є ключовим елементом тестування на проникнення, який допомагає забезпечити захист даних користувачів та підвищити загальний рівень безпеки мобільних додатків.

- ***Динамічний аналіз. Рекомендація 5: Аналіз бази даних додатку.***

  Під час тестування на проникнення мобільних додатків особливу увагу слід приділити внутрішнім базам даних SQLite, які використовуються для зберігання інформації. Необхідно перевірити створені бази даних, назви таблиць та стовпців, а також всі збережені дані, оскільки можна виявити чутливу інформацію, що є вразливістю. Бази даних зазвичай розташовані за шляхом /data/data/ім'я_пакета/databases, наприклад,

/data/data/com.mwr.example.sieve/databases. Якщо база даних зберігає конфіденційну інформацію та зашифрована, але пароль до неї можна знайти всередині додатку, це також вважається вразливістю. Для переліку таблиць використовуйте команду .tables, а для переліку стовпців таблиць - .schema <назва_таблиці>.

- ***Динамічний аналіз. Рекомендація 5: Аналіз сервісів та content provider.***
  Drozer — це інструмент для тестування на проникнення, який дозволяє вам взаємодіяти з мобільними додатками, виконуючи роль одного з них. Він надає можливість виконувати всі дії, на які здатний встановлений додаток, включаючи використання механізму міжпроцесного взаємодії (IPC) Android та взаємодію з базовою операційною системою. Drozer ефективний для виявлення вразливостей у експортованих активностях, сервісах та провайдерах контенту, що є ключовим для глибокого аналізу безпеки мобільних додатків. Завдяки цьому інструменту, фахівці з кібербезпеки можуть ідентифікувати та аналізувати потенційні слабкі місця в захисті додатків, що є важливим етапом у процесі забезпечення їх надійності.
- ***Динамічний аналіз. Рекомендація 6: Аналіз Keystore.***
  Під час тестування на проникнення мобільних додатків Android, особлива увага приділяється Keystore, оскільки це вважається найбезпечнішим місцем для зберігання конфіденційних даних. Проте, навіть з високим рівнем привілеїв, існує можливість доступу до цих даних. Враховуючи тенденцію додатків зберігати чутливу інформацію у відкритому тексті в Keystore, під час пентестів необхідно перевіряти цю вразливість, використовуючи права користувача root або фізичний доступ до пристрою, щоб виявити потенційну можливість крадіжки даних.

- ***Динамічний аналіз. Рекомендація 7: Перевірка на можливість обходу біометричної аутентифікації.***

  Одним із способів захисту є біометрична аутентифікація, яка часто включає сканування відбитків пальців. Проте, використання скриптів Frida може виявити потенційні вразливості, дозволяючи обійти цей рівень захисту в додатках Android. Це відкриває шлях до зловмисного доступу до захищених секцій додатку, що може призвести до несанкціонованого доступу до особистих даних користувачів. Така можливість обходу підкреслює необхідність ретельного тестування на проникнення та розробки більш надійних методів аутентифікації, які можуть протистояти подібним атакам.

- ***Динамічний аналіз. Рекомендація 7: Перевірка на можливість витоку даних через знімки екрану.***

  При переміщенні мобільного додатка в фоновий режим, Android створює знімок екрану додатка, щоб при поверненні його на передній план зображення завантажувалось до самого додатка, створюючи ілюзію більш швидкого завантаження. Однак, якщо знімок містить конфіденційну інформацію, особа з доступом до файлової системи пристрою може викрасти ці дані. Зазвичай, знімки зберігаються у директорії /data/system_ce/0/snapshots і для доступу до них потрібні права root. Android надає можливість запобігти створенню знімків екрану шляхом встановлення параметра FLAG_SECURE у компонуванні вікна. Цей прапорець забезпечує вважання вмісту вікна захищеним, запобігаючи його відображенню у знімках екрану або на незахищених дисплеях.

- ***Динамічний аналіз. Рекомендація 8: Перевірка захищенності інтентів у мобільних додатках.***

  У процесі розробки мобільних додатків, розробники часто створюють проксі-компоненти, такі як активності, сервіси та приймачі широкомовних повідомлень, які обробляють Intents та передають їх до методів на кшталт startActivity(...) чи sendBroadcast(...). Це може бути ризиковано, оскільки

існує загроза, що атакуючі зможуть активувати компоненти додатка, які не призначені для експорту, або отримати доступ до чутливих провайдерів контенту, спрямувавши ці Intents неправильно. Особливо вразливим є компонент WebView, який перетворює URL-адреси на об'єкти Intents за допомогою Intent.parseUri(...) та виконує їх, що може призвести до зловмисних ін'єкцій Intents.

- ***Динамічний аналіз. Рекомендація 8: Використання скриптів для утиліти Frida.***

  Під час тестування на проникнення мобільних додатків, використання скриптів для утиліти Frida є ключовим елементом для виявлення вразливостей. Frida дозволяє тестувальникам виконувати динамічний аналіз, втручаючись та змінюючи виконання коду в реальному часі, що може виявити потенційні слабкі місця, які не були б очевидні під час статичного аналізу. Це інструментально важливо для ідентифікації проблем, пов'язаних з безпекою, таких як ненадійне зберігання даних, недоліки в системі аутентифікації, або небезпечне використання API. Використання Frida вимагає глибокого розуміння внутрішньої структури додатку та може бути використане для підтвердження вразливостей, виявлених іншими методами тестування.

- ***Динамічний аналіз. Рекомендація 9: Використання утиліти Objection.***

  Під час тестування на проникнення мобільних додатків було виявлено, що утиліта Objection може бути використана для виявлення вразливостей у безпеці додатків. Цей інструмент дозволяє виконувати динамічний аналіз в реальному часі, що дає можливість тестувальникам швидко ідентифікувати потенційні слабкі місця в захисті додатку. Використання Objection допомагає виявити такі проблеми, як ненадійне зберігання даних, вразливості на стороні клієнта та неправильне використання криптографічних протоколів. Цей інструмент є корисним для проведення глибокого аналізу безпеки, що є важливим аспектом розробки надійних мобільних додатків.

- ***Динамічний аналіз. Рекомендація 10: Динамічне сканування з утилітою MobSF.***

  Динамічне сканування мобільних додатків за допомогою утиліти MobSF є ключовим елементом у виявленні потенційних вразливостей. Цей процес включає аналіз додатку в реальному часі під час його виконання, що дозволяє тестувальникам на проникнення виявляти проблеми, які можуть не бути очевидними під час статичного аналізу коду. MobSF автоматизує процес виявлення вразливостей, надаючи детальні звіти про безпеку, які можуть включати в себе виявлені вразливості, зловмисні скрипти та інші потенційні ризики. Використання MobSF допомагає забезпечити глибший рівень аналізу безпеки, що є важливим для розробки надійних та безпечних мобільних додатків.

Висновки до розділу 3

У цьому розділі було виконано завдання розробки рекомендацій для проведення якісного тестування на проникнення мобільних додатків. Ці рекомендації базуються на вже відомих етапах тестування, методологіях, програмних застосунках та утилітах, що дозволяють тестувальникам на проникнення комплексно протестувати всі аспекти мобільного додатку з метою забезпечення кібербезпеки користувачів додатку.

Розроблені рекомендації охоплюють етапи тестування на проникнення мобільних додатків, методи тестування згідно з OWASP MASTG, а також способи використання практичних утиліт та сканерів для виявлення вразливостей. Це дає можливість тестувальникам систематично і послідовно проводити тестування, використовуючи найкращі практики та інструменти, що вже є на ринку.

Цей підхід до тестування на проникнення мобільних додатків дозволяє не тільки виявити потенційні вразливості, але й забезпечити високий рівень захисту користувачів додатку, що є основною метою кібербезпеки.

ВИСНОВКИ

Мобільні застосунки продовжують відігравати значну роль як у повсякденному житті людей, так і в бізнес-процесах великих та малих організацій. Проте, незважаючи на численні переваги, мобільні пристрої несуть із собою нові загрози. Досягти високого рівня захищеності застосунків можна, використовуючи найкращі практики написання коду та регулярно проводячи тестування на проникнення.

Завданням кваліфікаційної роботи було розробити рекомендації, що враховує всі аспекти тестування на проникнення мобільних додатків. Для досягнення мети було проведено всебічний аналіз функціонування мобільних додатків, включаючи детальний огляд принципів їх роботи та потенційних вразливостей, які можуть бути використані зловмисниками для атак на користувачів. Було визначено сучасні вимоги до безпеки мобільних додатків, що дозволяє розробникам краще розуміти, як захистити свої продукти від потенційних загроз. Також проведено порівняльний аналіз існуючих методологій. Це дозволило виявити найбільш ефективні підходи до тестування та визначити потенційні напрямки для подальшого вдосконалення процесу.

У результаті дослідження був розроблений список рекомендацій, який враховує всі аспекти тестування мобільних застосунків на базі Android. Розроблені рекомендації охоплюють етапи тестування на проникнення мобільних додатків, методи тестування згідно з OWASP MASTG, а також способи використання практичних утиліт та сканерів для виявлення вразливостей. Це дає можливість тестувальникам на проникнення систематично і послідовно проводити тестування, використовуючи найкращі практики та інструменти, що вже є на ринку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Android application security part 2-understanding android operating system – aditya agrawal. *Aditya Agrawal – Security Researcher. RailFan. Foodie*. URL: https://manifestsecurity.com/android-application-security-part-2/ (date of access: 21.05.2024).
2. Abhishek D. Mobile app hackers handbook. CRC Press, 2013. 303 p. [2, с. 40]
3. Abhishek D. Mobile app hackers handbook. CRC Press, 2013. 303 p. [2, с. 45]
4. Android runtime and dalvik | android open source project. *Android Open Source Project*. URL: https://source.android.com/docs/core/runtime (date of access: 21.05.2024).
5. Build APKs for Android Emulators and devices. *Expo Documentation*. URL: https://docs.expo.dev/build-reference/apk/ (date of access: 21.05.2024).
6. Abhishek D. Mobile app hackers handbook. CRC Press, 2013. 303 p. [2, с. 56]
7. Android zygote startup - elinux.org. *eLinux.org*. URL: https://elinux.org/Android_Zygote_Startup (date of access: 21.05.2024).
8. 10 - Android formats – LIEF Documentation. *LIEF*. URL: https://lief.re/doc/latest/tutorials/10_android_formats.html (date of access: 21.05.2024).
9. What is an ELF file? | baeldung on linux. *Baeldung on Linux*. URL: https://www.baeldung.com/linux/executable-and-linkable-format-file (date of access: 21.05.2024).
10. Reduce your app size | App quality | Android Developers. *Android Developers*. URL: https://developer.android.com/topic/performance/reduce-apk-size (date of access: 21.05.2024).
11. JAR file specification. *Moved*. URL: https://docs.oracle.com/javase/8/docs/technotes/guides/jar/jar.html (date of access: 21.05.2024).
12. Understanding the JAR manifest file | baeldung. *Baeldung*. URL: https://www.baeldung.com/java-jar-manifest (date of access: 21.05.2024).

13. App manifest overview | Android Developers. *Android Developers*. URL: https://developer.android.com/guide/topics/manifest/manifest-intro (date of access: 21.05.2024).

14. ARSC - android package resource table file. *File Format Docs*. URL: https://docs.fileformat.com/programming/arsc/ (date of access: 21.05.2024).

15. Content providers | android developers. *Android Developers*. URL: https://developer.android.com/guide/topics/providers/content-providers (date of access: 21.05.2024).

16. Broadcast receiver in android with example - geeksforgeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/broadcast-receiver-in-android-with-example/ (date of access: 21.05.2024).

17. What is Android Privacy Sandbox? | AppsFlyer mobile glossary. *AppsFlyer*. URL: https://www.appsflyer.com/glossary/android-privacy-sandbox/ (date of access: 21.05.2024).

18. OWASP MASVS - OWASP mobile application security. *OWASP Mobile Application Security*. URL: https://mas.owasp.org/MASVS/ (date of access: 21.05.2024).

19. M1: improper credential usage | OWASP foundation. *OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation*. URL: https://owasp.org/www-project-mobile-top-10/2023-risks/m1-improper-credential-usage.html (date of access: 21.05.2024).

20. M2: inadequate supply chain security | OWASP foundation. *OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation*. URL: https://owasp.org/www-project-mobile-top-10/2023-risks/m2-inadequate-supply-chain-security.html (date of access: 21.05.2024).

21. M3: insecure authentication/authorization | OWASP foundation. *OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation*. URL: https://owasp.org/www-project-mobile-top-10/2023-risks/m3-insecure-authentication-authorization.html (date of access: 21.05.2024).

22. M4: insufficient input/output validation | OWASP foundation. *OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation*. URL: https://owasp.org/www-project-mobile-top-10/2023-risks/m4-insufficient-input-output-validation.html (date of access: 21.05.2024).

23. M5: insecure communication | OWASP foundation. *OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation*. URL: https://owasp.org/www-project-mobile-top-10/2023-risks/m5-insecure-communication.html (date of access: 21.05.2024).

24. M6: inadequate privacy controls | OWASP foundation. *OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation*. URL: https://owasp.org/www-project-mobile-top-10/2023-risks/m6-inadequate-privacy-controls.html (date of access: 21.05.2024).

25. M7: insufficient binary protection | OWASP foundation. *OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation*. URL: https://owasp.org/www-project-mobile-top-10/2023-risks/m7-insufficient-binary-protection.html (date of access: 21.05.2024).

26. M8: security misconfiguration | OWASP foundation. *OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation*. URL: https://owasp.org/www-project-mobile-top-10/2023-risks/m8-security-misconfiguration.html (date of access: 21.05.2024).

27. M9: insecure data storage | OWASP foundation. *OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation*. URL: https://owasp.org/www-project-mobile-top-10/2023-risks/m9-insecure-data-storage.html (date of access: 21.05.2024).

28. M10: insufficient cryptography | OWASP foundation. *OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation*. URL: https://owasp.org/www-project-mobile-top-10/2023-risks/m10-insufficient-cryptography.html (date of access: 21.05.2024).

29. ДСТУ ISO/IEC 27034-2:2017 Інформаційні технології. Методи захисту. Безпека прикладних програм. Частина 2. Нормативна структура організації

(ISO/IEC 27034-2:2015, IDT). *БУДСТАНДАРТ Online - нормативні документи будівельної галузі України*. URL: https://online.budstandart.com/ua/catalog/doc-page?id_doc=75943 (дата звернення: 21.05.2024).

30. SP 800-163 rev. 1, vetting the security of mobile applications | CSRC. *NIST Computer Security Resource Center | CSRC*. URL: https://csrc.nist.gov/pubs/sp/800/163/r1/final (date of access: 21.05.2024).

31. *PCI Security Standards Council – Protect Payment Data with Industry-driven Security Standards, Training, and Programs*. URL: *https://listings.pcisecuritystandards.org/documents/PCI_Mobile_Payment_Acceptance_Security_Guidelines_for_Developers_v2_0.pdf* (дата звернення: 21.05.2024).

32. Dark wolf | android security research playbook. *Dark Wolf | Android Security Research Playbook*. URL: https://asrp.darkwolf.io/ASRP-Plays/dynamic (date of access: 21.05.2024).

33. Automated mobile application security assessment with mobsf – MAS. *OpSecX*. URL: https://opsecx.com/index.php/product/automated-mobile-application-security-assessment-with-mobsf/ (date of access: 21.05.2024).

34. OWASP MASTG - OWASP mobile application security. *OWASP Mobile Application Security*. URL: https://mas.owasp.org/MASTG/ (date of access: 21.05.2024).

35. OWASP risk rating methodology | OWASP foundation. *OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation*. URL: https://owasp.org/www-community/OWASP_Risk_Rating_Methodology (date of access: 21.05.2024).

36. OWASP MASVS - OWASP mobile application security. *OWASP Mobile Application Security*. URL: https://mas.owasp.org/MASVS/ (date of access: 21.05.2024).

37. Mariana trench | mariana trench. *Mariana Trench | Mariana Trench*. URL: https://mariana-tren.ch/ (date of access: 21.05.2024).

38. Frida A world-class dynamic instrumentation toolkit. *Frida A world-class dynamic instrumentation toolkit*. URL: https://frida.re/ (date of access: 21.05.2024).

39. IOS pen testing with objection. *Redfox Security*. URL: https://redfoxsec.com/blog/ios-pen-testing-with-objection-mastering-vulnerabilities/ (date of access: 21.05.2024).

40. Android APK Checklist | HackTricks | HackTricks. *HackTricks | HackTricks | HackTricks*. URL: https://book.hacktricks.xyz/mobile-pentesting/android-checklist (date of access: 21.05.2024).