МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**

**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ КІБЕРБЕЗПЕКИ ТА ЗАХИСТУ ІНФОРМАЦІЇ
КАФЕДРА УПРАВЛІННЯ КІБЕРБЕЗПЕКОЮ ТА ЗАХИСТОМ ІНФОРМАЦІЇ**

КВАЛІФІКАЦІЙНА РОБОТА

на тему: “МЕТОДИ ТА ІНСТРУМЕНТИ ОЦІНКИ ЗАХИЩЕНОСТІ МОБІЛЬНИХ ЗАСТОСУНКІВ ФІНАНСОВОГО СЕКТОРУ”

на здобуття освітнього ступеня магістра
зі спеціальності 125 Кібербезпека та захист інформації
освітньо-професійної програми Управління інформаційною та кібернетичною безпекою

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

_____ Олександр РОМАНОВ

(підпис) *Ім’я, ПРІЗВИЩЕ здобувача*

Виконав: Здобувач вищої освіти гр. УБДМ-61
Олександр РОМАНОВ

Керівник: Дмитро РАБЧУН
К.т.н.

Рецензент:

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут кібербезпеки та захисту інформації

Кафедра Управління кібербезпекою та захистом інформації
Ступінь вищої освіти магістр
Спеціальність 125 Кібербезпека та захист інформації
Освітньо-професійна програма Управління інформаційною та кібернетичною безпекою

ЗАТВЕРДЖУЮ
Завідувач кафедрою УКБЗІ
_____ Світлана ЛЕГОМІНОВА
“____” _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студенту Романову Олександру Андрійовичу
(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи: “Методи та інструменти оцінки захищеності мобільних застосунків фінансового сектору”
керівник кваліфікаційної роботи Дмитро РАБЧУН, к.т.н.
(Ім’я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від “30” жовтня 2025 р. № 467.
2. Строк подання кваліфікаційної роботи “11” грудня 2025 р.
3. Вихідні дані до кваліфікаційної роботи: оцінювання захищеності мобільних застосунків, методи технічного аналізу вразливостей, стандарти безпеки мобільних фінансових сервісів.
4. Перелік питань, які потрібно розробити:
    1. Здійснити аналіз архітектури мобільних застосунків фінансового сектору та вимог до їх захищеності.
    2. Дослідити методи, стандарти й інструменти оцінки безпеки мобільних мобільних застосунків (MASVS, OWASP, CVSS, PCI DSS тощо).
    3. Провести практичну оцінку захищеності мобільного застосунку Damn Vulnerable Bank із використанням статичного, динамічного аналізу та аналізу API.
    4. Оцінити результати тестування з позиції міжнародних стандартів і розробити рекомендації щодо підвищення рівня безпеки мобільних застосунків фінансового призначення.
5. Перелік ілюстративного матеріалу: *презентація.*
6. Дата видачі завдання “02” жовтня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назви етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Визначення об’єкту, предмету, мети та завдань дослідження.	10.10.2025	
2.	Збір та аналіз літератури.	23.10.2025	
3.	Аналіз методів, стандартів та інструментів оцінювання безпеки мобільних фінансових застосунків і формування теоретичної моделі дослідження.	27.10.2025	
4.	Проведення практичної оцінки захищеності мобільного застосунку на основі статичного, динамічного аналізу та дослідження API, інтерпретація отриманих результатів.	10.11.2025	
5.	Визначення технічних і процесних напрямів підвищення рівня безпеки мобільних застосунків фінансового сектору з урахуванням результатів практичного аналізу та вимог міжнародних стандартів.	15.11.2025	
6.	Формулювання висновків за результатами дослідження.	22.11.2025	
7.	Оформлення роботи.	09.12.2025	
8.	Оформлення презентації.	10.12.2025	
9.	Отримання рецензії на роботу.	11.12.2025	
10.	Захист в ЕК.	20.01.2026	

Здобувач вищої освіти _____ Олександр РОМАНОВ
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

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

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

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

Направляється здобувач Романов О. А. до захисту кваліфікаційної роботи
(прізвище та ініціали)
за спеціальністю 125 Кібербезпека та захист інформації
(код, найменування спеціальності)
Освітньо-професійної програми Управління інформаційною та кібернетичною безпекою
(назва)
на тему: “Методи та інструменти оцінки захищеності мобільних застосунків фінансового сектору”
Кваліфікаційна робота і рецензія додаються.

Директор ННІКБЗІ _____ Євгенія ІВАНЧЕНКО
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

Висновок керівника кваліфікаційної роботи

Здобувач **РОМАНОВ Олександр** у кваліфікаційній роботі здійснив комплексне дослідження методів і інструментів оцінювання захищеності мобільних застосунків фінансового сектору. У роботі проаналізовано архітектурні особливості мобільних фінансових сервісів, досліджено сучасні стандарти та підходи до їх безпеки, зокрема MASVS, OWASP Mobile, OWASP API Security, PCI DSS, а також методологію оцінювання вразливостей за CVSS v3.1. Проведене практичне дослідження із застосуванням статичного, динамічного та мережевого аналізу дозволило сформувати об’єктивну картину ризиків та визначити основні напрями їх усунення.

РОМАНОВ Олександр показав високу теоретичну і практичну підготовку, володіння науково-дослідницькими методами, вміння самостійно знаходити шляхи вирішення проблеми дослідження. Результати дослідження апробовані на конференції “Стратегії кіберстійкості: управління ризиками та безперервність бізнесу” 27 лютого 2025 року

Все це дозволяє оцінити кваліфікаційну роботу здобувача **РОМАНОВА Олександра** на оцінку “добре” та присвоїти йому кваліфікацію “Магістр з кібербезпеки та захисту інформації за освітньо-професійною програмою Управління інформаційною та кібернетичною безпекою”.

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

“___“_____2025 року

Висновок кафедри про кваліфікаційну роботу

Кваліфікаційна робота розглянута. Здобувач Романов О. А. допускається до захисту даної роботи в Екзаменаційній комісії.

Завідувач кафедрою
Управління кібербезпекою та захистом
інформації _____ Світлана ЛЕГОМІНОВА
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

ВІДГУК РЕЦЕНЗЕНТА
на кваліфікаційну магістерську роботу

здобувача вищої освіти Романова Олександра Андрійовича
на тему “Методи та інструменти оцінки захищеності мобільних застосунків фінансового сектору”

Актуальність У сучасних умовах мобільні застосунки стали ключовим інструментом взаємодії користувачів із фінансовими послугами, що зумовило зростання вимог до їхньої безпеки. Кількість кіберзагроз, спрямованих на мобільні платформи, постійно збільшується, а компрометація мобільного банківського застосунку може призвести до прямих фінансових збитків, витоку персональних даних та порушення цілісності фінансових операцій. У цьому контексті дослідження методів і інструментів оцінювання захищеності мобільних застосунків фінансового сектору є актуальним і має важливе науково-практичне значення.

Позитивні сторони

1. У роботі проведено всебічний аналіз архітектурних особливостей мобільних фінансових застосунків, сучасних стандартів безпеки (MASVS, OWASP Mobile, OWASP API Security, PCI DSS) та методів оцінювання вразливостей. Автор продемонстрував глибоке розуміння принципів мобільної безпеки та логічно обґрунтував вибір дослідницьких підходів.
2. Значною перевагою є практична частина, у якій здійснено комплексне тестування мобільного застосунку з використанням статичного, динамічного та мережевого аналізу. Отримані результати систематизовано, формалізовано за CVSS v3.1 та зіставлено з вимогами міжнародних стандартів, що свідчить про високий рівень методичної підготовки здобувача.
3. Кваліфікаційна робота належно структурована, матеріал викладено послідовно й аргументовано, зроблено логічні висновки. Оформлення відповідає встановленим вимогам. Джерельна база є достатньою і включає сучасні наукові публікації й технічні стандарти, у тому числі англомовні.

Недоліки

1. Доцільно було б деталізувати порівняльний аналіз ефективності окремих інструментів мобільного тестування, що дозволило б ширше відобразити їхні можливості на різних етапах оцінювання безпеки.

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

Висновок: Кваліфікаційна робота виконана на належному науково-методичному рівні і заслуговує позитивної оцінки, а здобувач Романов Олександр Андрійович заслуговує присвоєння кваліфікації “Магістр кібербезпеки за освітньо-професійною програмою “Управління інформаційною та кібернетичною безпекою”.

Рецензент: _____

підпис

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 88 стор., 11 рис., 1 табл., 40 джерел.

Метою роботи **є** встановлення ефективних методів і практичних підходів до оцінки захищеності мобільних застосунків фінансового сектору з метою виявлення потенційних вразливостей та підвищення рівня кіберзахисту.

Об'єктом дослідження є процес оцінки захищеності мобільних застосунків у сфері фінансових технологій.

Предметом дослідження є методи, інструменти та критерії, що застосовуються для виявлення вразливостей і оцінки рівня безпеки мобільних застосунків, які використовуються у фінансовому секторі.

Методи дослідження ґрунтуються на поєднанні методів системного аналізу, технічного аудиту безпеки, методологій статичного й динамічного тестування, аналізу API, а також практичного застосування інструментів мобільного пентесту. Теоретичну основу роботи становлять стандарти й моделі безпеки (MASVS, OWASP Mobile, OWASP API Security, PCI DSS), що дозволяють формалізувати вимоги та критерії оцінювання.

Короткий зміст роботи. У роботі проведено комплексний аналіз архітектури мобільних застосунків фінансового сектору та ключових вимог до їх захисту. Досліджено сучасні методи й міжнародні стандарти оцінювання безпеки мобільних застосунків, зокрема MASVS, OWASP Mobile, OWASP API Security, CVSS та PCI DSS. У практичній частині виконано повний цикл оцінки захищеності мобільного застосунку з використанням статичного, динамічного та мережевого аналізу, що дозволило виявити критичні вразливості на різних рівнях від клієнтської логіки до API та бізнес-процесів. На основі отриманих результатів сформовано технічні та організаційні рекомендації щодо підвищення рівня безпеки мобільних фінансових сервісів та узгодження їх із міжнародними вимогами.

Галузь застосування. Отримані результати можуть бути використані під час розроблення, тестування та аудиту мобільних застосунків фінансового сектору, а також у процесі впровадження політик безпеки, побудови Secure SDLC і створення системи управління кіберзагрозами для фінансових мобільних сервісів.

КЛЮЧОВІ СЛОВА: МОБІЛЬНА БЕЗПЕКА, ВРАЗЛИВІСТЬ, СТАТИЧНИЙ ТА ДИНАМІЧНИЙ АНАЛІЗ, MASVS, OWASP MOBILE, CVSS, ФІНАНСОВИЙ МОБІЛЬНИЙ ЗАСТОСУНОК, ОЦІНКА ЗАХИЩЕНОСТІ.

ABSTRACT

The text part of the qualification work for obtaining a master's degree: 88 pages, 11 figures, 1 table, 40 sources.

The purpose of the thesis is to establish effective methods and practical approaches for assessing the security of mobile applications in the financial sector in order to identify potential vulnerabilities and enhance the level of cybersecurity.

Object of research is the process of evaluating the security of mobile applications within the field of financial technologies.

Subject of research includes methods, tools, and criteria used to identify vulnerabilities and assess the security level of mobile applications employed in the financial sector.

The research method is based on a combination of system analysis techniques, security auditing methods, static and dynamic testing approaches, API analysis, as well as practical application of mobile penetration testing tools. The theoretical foundation of the work includes security standards and models (MASVS, OWASP Mobile, OWASP API Security, PCI DSS), which allow the formalization of requirements and evaluation criteria.

Brief content of research. The work presents a comprehensive analysis of the architecture of mobile applications in the financial sector and the key requirements for their protection. Modern methods and international standards for assessing the security of mobile applications, including MASVS, OWASP Mobile, OWASP API Security, CVSS, and PCI DSS, are examined. The practical part includes a full security assessment cycle of a mobile application using static, dynamic, and network analysis, which enables the identification of critical vulnerabilities at multiple levels from client-side logic to API and business processes. Based on the findings, technical and organizational recommendations were formulated to improve the security of mobile financial services and ensure their compliance with international requirements.

Field of research. The obtained results may be used in the development, testing, and security auditing of mobile applications in the financial sector, as well as in the implementation of security policies, the construction of a Secure SDLC, and the creation of cybersecurity management systems for mobile financial services.

KEYWORDS: MOBILE SECURITY, VULNERABILITY, STATIC AND DYNAMIC ANALYSIS, MASVS, OWASP MOBILE, CVSS, FINANCIAL MOBILE APPLICATION, SECURITY ASSESSMENT.

ЗМІСТ

ВСТУП

Актуальність теми. Мобільні застосунки стали ключовим елементом інфраструктури фінансового сектору, забезпечуючи доступ до банківських послуг, управління рахунками та виконання транзакцій у режимі реального часу. Водночас стрімке зростання їх функціональності супроводжується збільшенням кількості кіберзагроз, спрямованих на компрометацію фінансових даних, порушення бізнес-логіки та отримання несанкціонованого доступу до облікових записів користувачів. Зважаючи на високу цінність інформації та критичність операцій, мобільні фінансові застосунки стають пріоритетними цілями зловмисників, а їхній захист стратегічним завданням для фінансових установ.

У таких умовах особливого значення набуває розроблення науково обґрунтованих підходів до оцінювання безпеки мобільних застосунків, здатних забезпечити виявлення вразливостей на різних рівнях від архітектури до бізнес-логіки. Актуальність теми зумовлена необхідністю комплексного підходу, що поєднує практичні методи тестування, використання спеціалізованого інструментарію та орієнтацію на міжнародні стандарти безпеки. Саме тому тема даної кваліфікаційної роботи є актуальною й має практичне значення для суб’єктів фінансового сектору.

Метою роботи є встановлення ефективних методів і практичних підходів до оцінки захищеності мобільних застосунків фінансового сектору з метою виявлення потенційних вразливостей та підвищення рівня кіберзахисту.

Об’єктом дослідження є процес оцінки захищеності мобільних застосунків у сфері фінансових технологій.

Предметом дослідження є методи, інструменти та критерії, що застосовуються для виявлення вразливостей і оцінки рівня безпеки мобільних застосунків, які використовуються у фінансовому секторі.

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

1. Здійснити аналіз архітектури мобільних застосунків фінансового сектору та вимог до їх захищеності;

2. Дослідити методи, стандарти та інструменти оцінки безпеки мобільних застосунків (MASVS, OWASP, CVSS, PCI DSS тощо);
3. провести практичну оцінку захищеності мобільного застосунку Damn Vulnerable Bank із використанням статичного, динамічного аналізу та аналізу API;
4. Оцінити результати тестування з позиції міжнародних стандартів та розробити рекомендації щодо підвищення рівня безпеки мобільних застосунків фінансового призначення.

Методи дослідження ґрунтуються на поєднанні методів системного аналізу, технічного аудиту безпеки, методологій статичного й динамічного тестування, аналізу API, а також практичного застосування інструментів мобільного пентесту. Теоретичну основу роботи становлять стандарти й моделі безпеки (MASVS, OWASP Mobile, OWASP API Security, PCI DSS), що дозволяють формалізувати вимоги та критерії оцінювання.

Наукова новизна одержаних результатів полягає у формуванні комплексного підходу до оцінки захищеності мобільних фінансових застосунків, який поєднує практичну методику тестування з формалізованим аналізом відповідності міжнародним стандартам і дозволяє систематизовано визначати критичні вектори атак та їхній вплив на безпеку. Запропонований підхід може бути використаний для удосконалення методів аудиту фінансових мобільних систем.

Практичне значення одержаних результатів полягає в можливості безпосереднього застосування розроблених методичних рекомендацій для тестування, аналізу та вдосконалення мобільних застосунків фінансового призначення. Результати дослідження можуть бути використані фахівцями з інформаційної безпеки, розробниками та аудиторами для підвищення стійкості мобільних сервісів до кіберзагроз.

Апробація результатів кваліфікаційної роботи було оприлюднено на Всеукраїнській науковій конференції “Стратегії кіберстійкості: управління ризиками та безперервність бізнесу”.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ОЦІНКИ ЗАХИЩЕНОСТІ МОБІЛЬНИХ ЗАСТОСУНКІВ ФІНАНСОВОГО СЕКТОРУ

Стрімке поширення мобільних технологій у фінансовому секторі зумовило трансформацію традиційних моделей надання банківських та платіжних послуг. Мобільні застосунки стали ключовим інструментом взаємодії клієнтів з фінансовими установами, забезпечуючи доступ до персональних даних, управління рахунками та здійснення фінансових операцій у режимі реального часу. Така концентрація чутливої інформації та критичних бізнес-процесів у мобільному середовищі водночас створює значні ризики, оскільки застосунки працюють у неповністю контрольованих умовах і перебувають під постійною загрозою з боку зловмисників [1].

Необхідність оцінки захищеності мобільних застосунків фінансового сектору обумовлена комплексністю їхньої архітектури та багаторівневістю моделі загроз. На відміну від традиційних інформаційних систем, мобільні застосунки поєднують клієнтську логіку, залежність від операційної системи, мережеву взаємодію та серверні механізми обробки даних, кожен з яких може бути вразливим. У сучасних умовах забезпечення безпеки фінансових мобільних сервісів потребує не лише технічних засобів, а й системного підходу до аналізу ризиків, нормативних вимог і міжнародних стандартів, що встановлюють обов’язкові критерії для захисту цифрових фінансових продуктів.

У цьому розділі розглядаються фундаментальні положення, необхідні для подальшого практичного дослідження. Аналіз особливостей архітектури мобільних застосунків фінансового сектору дозволяє визначити ключові компоненти, які потребують посиленого захисту. Дослідження підходів і стандартів міжнародного рівня дає змогу сформувати нормативно-методичну базу для оцінки безпеки. Окрему увагу приділено методам та інструментам, що застосовуються на практиці для виявлення вразливостей і оцінки рівня ризиків.

Сукупність цих аспектів створює теоретичний фундамент, на якому ґрунтуватиметься подальша практична частина дослідження [2].

1.1. Особливості архітектури мобільних застосунків у фінансовій сфері

Розвиток фінансових технологій супроводжується стрімким зростанням використання мобільних застосунків як основного каналу доступу користувачів до банківських та платіжних послуг. За останнє десятиліття мобільні фінансові сервіси стали домінуючою платформою для здійснення транзакцій, управління рахунками, обробки персональних і платіжних даних. У багатьох країнах їхня частка перевищує 60–70% від загального обсягу цифрових операцій.

Така динаміка робить мобільні застосунки одним з найважливіших елементів фінансової інфраструктури. Разом з тим, вони є й однією з найбільш атакованих цілей, оскільки поєднують у собі такі фактори ризику:

1. робота у потенційно незахищеному середовищі користувацького пристрою;
2. зберігання чутливих даних або доступ до них через токени;
3. взаємодія з мережевими API та бекенд-системами;
4. залежність від механізмів захисту операційної системи;
5. високий рівень фрагментації екосистеми (особливо Android).

На відміну від веб-додатків, мобільні застосунки у фінансовому секторі мають більш розгалужену модель ризиків: загрози надходять не лише через мережеву взаємодію, але й через локальне середовище виконання, файлова система, механізми біометрії, апаратні модулі та канали комунікації між компонентами ОС [3].

Клієнтська частина мобільного застосунку у фінансовій сфері виконує комплекс взаємопов’язаних функцій, які забезпечують одночасно взаємодію користувача з сервісом, локальну обробку даних, роботу криптографічних механізмів та інтеграцію з операційною системою. Передусім саме клієнтський

компонент відповідає за формування інтерфейсу та забезпечення коректної взаємодії з користувачем, але його роль значно ширша, адже будь-яка операція, яка на перший погляд здається “зовнішньою” наприклад, автентифікація чи ініціювання транзакції фактично запускається на рівні мобільного застосунку, що означає необхідність обробки попередніх даних, створення криптографічних підписів або перевірки певних параметрів середовища.

Клієнтська частина завжди включає елементи локальної логіки. Це може бути попередня перевірка коректності введених даних, обробка частини бізнес-правил або формування запитів до серверної сторони. У фінансових застосунках ця логіка частіше обмежена мінімумом, адже критичні рішення мають прийматися на сервері, проте навіть той мінімум, який залишається в мобільному застосунку, може становити істотний ризик у разі некоректної реалізації. Особливо це стосується криптографічних операцій, що передбачають генерацію ключів, формування токенів доступу, шифрування тимчасових даних або реалізацію алгоритмів апаратної біометричної автентифікації. На практиці саме клієнтський компонент найчастіше містить помилки у реалізації криптографії наприклад, використання жорстко заданих ключів або слабких генераторів випадкових чисел.

Важливою складовою архітектури є взаємодія застосунку з операційною системою. Така взаємодія визначає рівень доступу до апаратних можливостей пристрою, механізмів ізольованого зберігання, системних журналів, параметрів мережевого з’єднання та біометричних сенсорів. У фінансовому середовищі вимоги до цієї взаємодії незрівнянно суворіші, оскільки будь-які небезпечні дозволи, неконтрольовані виклики API або неправильна обробка помилок операційної системи можуть призвести до витоку чутливих даних або можливості виконання стороннього коду.

Безпека клієнтської частини значною мірою залежить від того, чи забезпечує застосунок захищене зберігання конфіденційної інформації. Токени автентифікації, сеансові маркери, криптографічні ключі, коди підтвердження

транзакцій усе це може зберігатися локально, хоча й у різних формах. Фінансовий застосунок має використовувати спеціалізовані механізми операційної системи: ключові сховища Android Keystore або iOS Keychain, а за можливості апаратні модулі типу Secure Enclave чи StrongBox. Порушення цих принципів наприклад, зберігання ключів у відкритому вигляді в локальних файлах одразу відкриває шлях до компрометації всіх механізмів автентифікації.

Окремим аспектом є протидія реверс-інжинірингу. Через те, що мобільні застосунки поширюються у вигляді виконуваних пакетів, які встановлюються безпосередньо на пристрій користувача, зловмисник має можливість отримати до них фізичний доступ, декомпілювати їх, дослідити внутрішню логіку або навіть модифікувати її. Саме тому фінансові застосунки повинні включати багаторівневий захист: обфускацію коду, механізми виявлення модифікацій, перевірку цілісності, блокування виконання на рутованих або джейлбрейкнутих пристроях, а також виявлення спроб налагодження процесу. Відсутність цих механізмів дозволяє зловмиснику зчитати структуру API, обійти механізми автентифікації або модифікувати логіку транзакцій.

Відмінності між платформами Android та iOS істотно впливають на модель загроз. Android, будучи відкритою системою з можливістю встановлення сторонніх прошивок, модифікованих системних компонентів та root-доступу, створює значно ширше поле для атак. Висока фрагментація версій ОС, наявність пристроїв без регулярних оновлень та різноманіття виробників ускладнюють передбачення середовища виконання. У свою чергу iOS забезпечує жорсткішу ізоляцію застосунків і контролює цілісність системних компонентів централізовано, проте й вона не усуває ризики повністю. Навіть на захищених пристроях можлива підміна сертифікатів, використання емуляційних середовищ, маніпуляції provisioning-профілями, ін’єкція динамічних бібліотек при наявності джейлбрейку або експлойтів нульового дня [4].

Таким чином, клієнтська архітектура мобільного застосунку у фінансовій сфері формує складний ландшафт безпекових викликів, у якому поєднуються

особливості операційної системи, ризики локальної обробки даних, чутливість криптографічних операцій та необхідність протидії реверс-інжинірингу. Врахування цих аспектів є передумовою для побудови коректної моделі загроз і визначення методів оцінки захищеності застосунків, що використовуються у фінансовому секторі.

Серверна частина мобільного фінансового застосунку становить основу всієї системи обробки даних та бізнес-логіки і функціонує як центральний елемент, що забезпечує збереження, цілісність та узгодженість фінансових транзакцій. На відміну від клієнтського компонента, сервер працює у контрольованому середовищі, що дозволяє застосовувати більш жорсткі механізми безпеки. Однак саме серверна частина несе основну відповідальність за дотримання коректності фінансових операцій та забезпечення захисту інформації, яка передається між компонентами системи.

У типовій архітектурі мобільного застосунку сервер реалізує декілька критично важливих функцій. Передусім це виконання бізнес-логіки, що включає перевірку правильності транзакцій, підтвердження балансових операцій, верифікацію даних користувача, а також застосування правил безпеки та регуляторних вимог. Сервер також забезпечує керування доступом, здійснюючи автентифікацію та авторизацію користувачів, перевіряючи валідність токенів, аналізуючи історію їх застосування та контролюючи контекст виконання запитів. Усі процеси, пов’язані з керуванням доступом, мають особливе значення у фінансових системах, оскільки будь-яка помилка в логіці доступу може призвести до несанкціонованих операцій і, як наслідок, до фінансових збитків.

Окрему роль відіграють модулі, що відповідають за обробку транзакцій. Вони повинні працювати за принципами ідемпотентності, точності та незворотності, адже некоректна реалізація механізмів транзакцій може спричинити підміну, дублювання або втрату даних. Усі транзакції мають супроводжуватися аудитом, а система журналювання фіксувати ключові

параметри кожної операції, включно з ідентифікацією користувача, часом виконання, типом операції та можливими відхиленнями у поведінці.

Центральним елементом взаємодії між мобільним застосунком і сервером є API. Саме API виступає точкою входу до всієї бекенд-логіки, тому його безпека визначає загальний рівень стійкості системи. API-шлюзи зазвичай реалізують контроль доступу, перевірку автентичності запитів, фільтрацію параметрів, застосування політик обмеження швидкості та додаткові механізми, спрямовані на захист від автоматизованих атак.

Слабкість API неодмінно призводить до масштабних порушень безпеки, оскільки через API мобільний застосунок отримує практично всі можливості серверної логіки. Найпоширеніші проблеми пов’язані з недостатньою перевіркою прав доступу до об’єктів. Зокрема, неправильно реалізована авторизація може дозволити користувачеві отримати доступ до ресурсів інших користувачів, що є типовим проявом уразливості рівня IDOR. Якщо сервер не перевіряє коректно належність об’єкта конкретному користувачу, зловмисник може змінювати параметри запитів і отримувати дані, які не повинні бути доступні.

У фінансових застосунках нерідко виявляються проблеми, пов’язані з передбачуваністю токенів або сесійних маркерів. Якщо такі маркери не використовують криптографічно стійкі механізми генерації або мають надто тривалий строк життя, можливість їх підбору або перехоплення стає значно вищою. Недостатня кількість перевірок на стороні сервера може також призвести до того, що сесія продовжить існувати навіть після виконання ризикових дій, таких як зміна пристрою або одночасне підключення з декількох локацій.

Ще однією критичною помилкою є відсутність обмежень на кількість запитів. Ситуації, коли сервер не застосовує rate-limiting, дозволяють зловмиснику здійснювати атаки перебору на токени, паролі або операції підтвердження транзакцій, оскільки система не стримує велику кількість

автоматизованих запитів. Такі недоліки часто поєднуються з надмірно інформативними повідомленнями про помилки, що дає змогу зловмиснику адаптувати свої атаки відповідно до поведінки серверної системи.

Не менш важливою проблемою є недостатній контроль за параметрами, які надходять до API. Якщо сервер довіряє даним від клієнта і не виконує належної валідації, відкривається можливість для ін’єкцій, некоректних маніпуляцій з бізнес-логікою або обходу перевірок, які мали виконуватися на серверному рівні. У фінансових застосунках це може стосуватися підміни реквізитів транзакції, некоректної зміни параметрів переказу, або обходу внутрішніх обмежень щодо сум та частоти операцій.

Журналювання і аудит на сервері теж часто виявляються недостатніми. Якщо система не фіксує критичні події або фіксує їх у неповному вигляді, стає складно встановити контекст атаки чи визначити точні дії зловмисника. Недостатній аудит також знижує можливість виявлення аномалій, оскільки поведінкові відхилення залишаються непоміченими.

Таким чином, безпека серверної архітектури мобільного фінансового застосунку визначається тим, наскільки детально та коректно реалізовані механізми контролю доступу, обробки транзакцій, перевірки параметрів і журналювання. Сам API є ключовою ланкою цього процесу, адже саме він забезпечує взаємодію між клієнтом і бекендом. Будь-яка помилка в його реалізації може стати не лише локальною вразливістю, а й точкою повного компрометування системи. Для фінансових сервісів, де ціна помилки є надзвичайно високою, забезпечення належної стійкості API є обов’язковою умовою функціонування.

Формування моделі загроз для мобільних застосунків у фінансовій сфері потребує врахування кількох взаємопов’язаних рівнів впливу, кожен з яких формує власний набір ризиків та потенційних сценаріїв атак. На відміну від традиційних веб-систем, де більшість загроз концентрується навколо мережевої взаємодії та серверних компонентів, мобільні застосунки створюють значно

ширше поле можливостей для зловмисника. Це пов’язано з тим, що виконання частини логіки відбувається безпосередньо на пристрої користувача, а тому обмежується лише тими механізмами захисту, які вбудовані у застосунок або надаються операційною системою.

Однією з ключових площин ризику є саме клієнтське середовище. Мобільний пристрій, на якому працює застосунок, не є повністю контрольованим елементом. Користувач може встановлювати стороннє програмне забезпечення, вносити зміни в операційну систему або використовувати модифіковані прошивки. У разі наявності root-доступу або jailbreak-середовища зникає основна перевага мобільної платформи ізоляція додатків одне від одного та захист критичних системних функцій. У такому середовищі зловмисник може перехоплювати виклики API, модифікувати процеси застосунку, впроваджувати сторонній код або відстежувати роботу криптографічних функцій, що у звичайному режимі було б неможливо. Крім того, клієнтське середовище може бути скомпрометоване шкідливим програмним забезпеченням від клавіатур-логерів до шпигунських модулів, здатних зчитувати екран, маніпулювати мережею або підмінювати системні компоненти.

Другою важливою складовою моделі загроз є мережевий рівень, на якому відбувається обмін даними між мобільним застосунком і сервером. Незважаючи на широке впровадження протоколу TLS, мережеві атаки залишаються одним із найбільш розповсюджених і ефективних способів впливу. Зловмисник може намагатися підмінити маршрут трафіку, змусивши застосунок встановлювати з'єднання з фальшивим сервером, що дозволяє перехоплювати або модифікувати дані. Техніки типу SSL-stripping або DNS spoofing дають змогу маніпулювати процесом встановлення захищеного каналу, а відсутність жорсткого контролю сертифікатів або механізмів SSL pinning полегшує реалізацію таких атак. Небезпечність цих векторів у фінансовому секторі полягає у можливості впливати безпосередньо на транзакційні дані,

перехоплювати токени автентифікації або здійснювати повну підміну серверних відповідей.

Третя площина ризику стосується серверного середовища та його взаємодії з API. Навіть ідеально захищений клієнт не здатний компенсувати недоліки в авторизаційних механізмах або логіці роботи API, якщо вони реалізовані неналежним чином. Вразливості типу некоректної перевірки прав доступу, недостатньої валідації параметрів, використання передбачуваних токенів або недосконалих криптографічних механізмів на сервері створюють умови, за яких зловмисник може отримати доступ до даних інших користувачів або навіть повністю змінити логіку виконання транзакцій. У таких випадках саме API стає центральною точкою компрометації, оскільки мобільний застосунок, будучи клієнтом, змушений довіряти відповідям сервера [5].

У межах трьох зазначених площин виникають конкретні загрози, що формують основу реальної моделі атак. Однією з найбільш критичних є можливість експлуатації слабких або неправильно реалізованих криптографічних механізмів. У разі якщо застосунок зберігає ключі або токени у відкритому вигляді або формує криптографічні параметри з використанням передбачуваних значень, зловмисник може відтворити авторизаційні процедури та фактично виконувати операції від імені користувача.

Не менш небезпечною є загроза зчитування або підміни інформації через сайд-канали, тобто канали, що виникають поза межами типової взаємодії між застосунком і сервером. До таких каналів може належати доступ до екрану під час введення платіжних даних, зчитування буфера обміну або перехоплення системних сповіщень.

Розвиток інструментів динамічної ін’єкції коду, таких як Frida або Objection, зробив можливим втручання у внутрішні механізми застосунку без необхідності модифікації самого файлу пакету. Зловмисник може перехоплювати функції, змінювати їх поведінку під час виконання, підмінювати дані, що передаються на сервер, або обходити механізми контролю цілісності.

Це значно ускладнює захист застосунків, оскільки традиційні методи протидії реверс-інжинірингу, орієнтовані на статичний аналіз, стають недостатніми.

Одним із найсерйозніших ризиків залишається повна підміна серверної логіки через атаку типу MITM. Якщо застосунок не перевіряє автентичність сервера належним чином, зловмисник може створити підроблений сервер, який відповідатиме на запити так, щоб змінити зміст транзакцій або обманути користувача щодо результатів операції. У фінансовому секторі наслідки такої атаки можуть бути катастрофічними, оскільки вона надає можливість керувати транзакціями без жодної видимої для користувача ознаки компрометації.

Модель загроз мобільного фінансового застосунку, таким чином, має розглядатися як система взаємопов’язаних рівнів, у якій вразливості одного рівня здатні змінювати або посилювати вплив іншого. Саме тому ефективна оцінка захищеності не може обмежуватися дослідженням лише клієнтської частини або API. Вона має охоплювати весь життєвий цикл взаємодії мобільного застосунку з користувачем, операційною системою, мережевими каналами та серверною логікою. Тільки комплексний підхід дозволяє сформувати повну картину ризиків та визначити оптимальні механізми їх мінімізації.

1.2. Підходи та стандарти до оцінки безпеки мобільних застосунків

Міжнародні стандарти у сфері кібербезпеки відіграють визначальну роль у формуванні вимог до мобільних застосунків фінансового сектору. На відміну від звичайних цифрових сервісів, фінансові системи працюють у контексті жорстких регуляторних обмежень, де найменша помилка може призвести до порушення законодавства, значних фінансових втрат або компрометації конфіденційних даних користувачів. Тому стандарти, які визначають правила захисту інформації, безпосередньо впливають на проєктування архітектури,

реалізацію функціональних компонентів і вибір технічних рішень у мобільних застосунках.

Одним із найважливіших міжнародних документів, що регламентують безпеку фінансових сервісів, є стандарт PCI DSS, спрямований на захист платіжних карткових даних. Хоча він переважно орієнтований на серверні системи та мережеву інфраструктуру, його вимоги прямо стосуються мобільних застосунків, якщо вони зберігають, передають або обробляють карткові дані або токени, що заміщують такі дані. У контексті мобільних застосунків особливе значення мають вимоги щодо використання криптографічних механізмів із підтвердженою стійкістю, забезпечення цілісності програмних компонентів і недопущення прямого доступу до чутливої інформації. Для мобільної архітектури важливо також те, що PCI DSS зобов’язує мінімізувати обсяг даних, які можуть бути скомпрометовані в разі порушення, і таким чином стимулює перехід до моделей, де застосунок не обробляє критичну інформацію, обмежуючись роботою з токенами.

Європейська директива PSD2 у поєднанні з технічними стандартами RTS формує ще один важливий пласт вимог, який безпосередньо стосується мобільних застосунків. Основна ідея PSD2 полягає у забезпеченні сильної клієнтської автентифікації, яка повинна гарантувати, що доступ до фінансових ресурсів може отримати лише легітимний користувач. Для мобільних застосунків це означає обов’язкове впровадження багатофакторних механізмів, у яких кожен фактор має незалежну природу. Мова може йти про поєднання біометрії з криптографічним токеном, прив’язаним до конкретного пристрою, або використання одноразових кодів, що генеруються в ізольованому середовищі. Вимоги PSD2 також поширюються на захист транзакцій, підпис яких повинен засновуватися на параметрах операції таким чином, щоб підробка або модифікація даних між клієнтом і сервером була неможливою без компрометації кількох незалежних елементів системи.

Ще одним вагомим джерелом вимог є рекомендації NIST SP 800-63, які визначають рівні достовірності цифрової ідентичності. Цей документ формує концептуальну основу для визначення того, яким має бути рівень автентифікації залежно від критичності операцій. Мобільні фінансові застосунки зазвичай працюють на найвищих рівнях, оскільки мають справу з ресурсами, що мають значну цінність. Це означає, що механізми автентифікації повинні не лише перевіряти користувача в момент входу, але й забезпечувати стійкість сесії, контролювати її прив’язку до пристрою, а також перевіряти відповідність параметрів сесії очікуваним характеристикам. Окреме місце в NIST SP 800-63 займає визначення вимог до криптографічних токенів, які повинні зберігатися в апаратно-захищених контейнерах і бути невідтворюваними поза межами оригінального пристрою [6].

Сукупність міжнародних стандартів формує цілісну систему вимог, яка суттєво впливає на архітектурні рішення та вибір технологій під час розробки мобільних застосунків у фінансовій сфері. PCI DSS зосереджується на мінімізації площі атаки та забезпеченні захищеного середовища для обробки платіжних даних. PSD2 наголошує на автентифікації користувача та захисті транзакцій від підміни. NIST SP 800-63 формує вимоги до стійкості цифрової ідентичності та механізмів автентифікації. Разом ці стандарти створюють нормативно-методологічне підґрунтя, яке визначає, яким повинен бути рівень безпеки сучасних мобільних фінансових застосунків, і формують орієнтири для створення моделей оцінки захищеності, що будуть використані в подальших розділах роботи.

У сучасній практиці оцінювання безпеки мобільних застосунків саме стандарти OWASP MASVS і Mobile Security Testing Guide (MSTG) відіграють ключову роль, оскільки вони інтегрують у собі системний підхід до аналізу архітектури, поведінкових характеристик та механізмів захисту програмного забезпечення. MASVS позиціонується як універсальний набір вимог, що визначає цільовий рівень безпеки для мобільного застосунку ще на етапі його

проєктування. На відміну від абстрактних нормативних документів, які часто визначають лише загальні вимоги до інформаційної безпеки, MASVS фокусується на конкретних аспектах мобільної розробки і забезпечує структурований підхід до оцінки здатності застосунку протистояти сучасним типам атак [7].

Цей стандарт поділяє рівні захищеності застосунків залежно від їх критичності. Базовий рівень передбачає мінімальний набір механізмів безпеки, яких достатньо для застосунків, що не обробляють чутливі дані. У випадку фінансових сервісів вимоги значно посилюються, оскільки саме від коректності реалізації захисних механізмів залежить не лише конфіденційність даних, але й безпосередня можливість здійснення транзакцій. Тому застосунки, що працюють у фінансовому секторі, повинні відповідати підвищеному рівню захищеності та демонструвати стійкість до реверс-інжинірингу, який є одним із найпоширеніших засобів аналізу та модифікації мобільних продуктів зловмисниками.

MASVS структуровано таким чином, щоб охоплювати всі основні аспекти мобільної безпеки: від коректної організації архітектури до забезпечення надійності криптографічних модулів та контролю доступу. Принципове значення має те, що стандарт не обмежується статичними вимогами, а передбачає перевірку взаємодії застосунку з операційною системою та мережевою інфраструктурою. Для фінансових застосунків це особливо важливо, оскільки саме неправильна інтеграція з платформними механізмами найчастіше призводить до критичних вразливостей, зокрема витоку ключового матеріалу або обходу автентифікації.

Методологія оцінювання MSTG є практичним продовженням MASVS і фактично слугує дорожньою картою для дослідників безпеки, розробників і аудиторів. MSTG містить докладний опис технік тестування, які дозволяють виявляти недоліки у реалізації криптографії, логіці авторизації, зберіганні даних, мережевих комунікаціях і поведінці застосунку під час виконання. У

його межах пропонується комплексний підхід, який передбачає паралельне застосування статичного аналізу, динамічного аналізу, реверс-інжинірингу та дослідження API. Особливу увагу MSTG приділяє моделюванню поведінки зловмисника, що дозволяє оцінити здатність застосунку протистояти реальним сценаріям атак, а не лише відповідати формальним вимогам.

Особливого значення стандарти OWASP набувають саме у контексті фінансових застосунків, оскільки вони забезпечують можливість уніфікованої оцінки захищеності незалежно від платформи та типу реалізації сервісу. Завдяки MASVS організація може однозначно визначити рівень безпеки, який має бути досягнутий застосунком, а MSTG отримати інструментарій для перевірки цього рівня на практиці. Такий підхід дозволяє оцінювати мобільний застосунок не лише як окремий продукт, але як частину цілісної фінансової екосистеми, де будь-який недолік у клієнтській компоненті може мати критичні наслідки для всієї інфраструктури.

Таким чином, стандарти OWASP MASVS та MSTG формують методологічний фундамент, який дозволяє системно підходити до оцінювання безпеки мобільних застосунків. Вони встановлюють чіткі критерії, пропонують узгоджені методи тестування та дозволяють оцінити мобільний застосунок у контексті реальних загроз. У фінансовому секторі, де рівень ризику є одним із найвищих, відповідність цим стандартам стає не рекомендацією, а необхідністю, без якої неможливо гарантувати безпеку транзакцій, даних та цифрової ідентичності користувачів [8].

Підходи, які застосовуються у промисловості для аналізу безпеки мобільних застосунків, формувалися поступово, у відповідь на зростання складності програмних систем та розвитку методів атак. Сучасні фінансові сервіси працюють у середовищі, де обсяг програмної логіки, кількість взаємодіючих компонентів і швидкість оновлень значно перевищують можливості виключно ручного аудиту. Тому індустрія виробила декілька

взаємодоповнюючих методологій, кожна з яких оцінює безпеку з власної позиції й дозволяє охопити різні класи вразливостей.

Одним із основних підходів є статичний аналіз програмного забезпечення. Ця методика полягає у дослідженні коду або бінарних артефактів без їх запуску, що дає змогу виявляти структурні недоліки, логічні помилки, небезпечні конструкції та неправильне використання криптографічних механізмів. У контексті мобільних застосунків статичний аналіз охоплює декомпіляцію клієнтської частини, аналіз конфігураційних файлів, виявлення жорстко заданих ключів, перевірку коректності дозволів та аналіз взаємодії між компонентами застосунку. Цей підхід дозволяє отримати загальне уявлення про якість реалізації застосунку та рівень уваги, який розробники приділили безпеці ще на етапі написання коду [9].

Динамічний аналіз розглядає безпеку з протилежної перспективи. Застосунок запускається в контрольованому середовищі, а дослідник спостерігає за його поведінкою під час виконання. Такий аналіз особливо цінний для мобільних фінансових застосунків, оскільки дозволяє оцінити коректність обробки мережевих з’єднань, реакцію на спроби втручання у середовище, стійкість до MITM-атак та здатність застосунку протистояти маніпуляціям у реальному часі. Динамічний підхід дає змогу виявити ті вразливості, які неможливо ідентифікувати суто статичними засобами, наприклад, помилки у логіці автентифікації, некоректну обробку виняткових ситуацій або небажану передачу даних третім сторонам.

Зі збільшенням складності мобільних застосунків і появою нових векторів атак відбулося природне об’єднання статичного та динамічного підходів у ширшу методологію, відому як Mobile Application Security Testing. У межах цього підходу застосунок досліджується одночасно з кількох боків: аналізується структура клієнтської частини, логіка взаємодії з сервером, поведінка під час виконання, а також механізми захисту від реверс-інжинірингу. MAST охоплює роботу з інструментами декомпіляції, мережевого аналізу, ін’єкції коду, емуляції

середовищ та тестування API. Завдяки цьому дослідник отримує повну картину того, як застосунок поводиться у реальних умовах, які механізми захисту в ньому присутні, та наскільки ефективно вони протидіють сучасним атакам.

Доповненням до технічних методик виступає підхід, орієнтований на ризики. У традиційній моделі безпеки фокус робиться на пошуку конкретних уразливостей, але така стратегія не завжди дозволяє адекватно оцінити вплив проблеми на бізнес-процеси фінансової установи. Методологія оцінювання ризиків зміщує акцент із технічних деталей на значущість даних, критичність функцій і можливі наслідки експлуатації вразливості. У фінансовому секторі ця логіка набуває особливої ваги, адже вразливість, яка на перший погляд здається незначною, може призвести до суттєвих фінансових втрат або порушення регуляторних вимог, якщо вона стосується транзакційних процесів або механізмів автентифікації. Аналіз на основі ризиків дозволяє розставити пріоритети у виправленні помилок, оцінити доцільність інвестицій у певні механізми захисту та сформувати стратегічне бачення безпеки мобільних застосунків у контексті їх реального використання.

Таким чином, індустріальні підходи до оцінки безпеки мобільних застосунків не існують у вигляді ізольованих методик. Вони взаємодіють і доповнюють одне одного, утворюючи цілісну систему аналізу, що враховує багатовимірну природу мобільних фінансових сервісів. Статичний аналіз закладає основу для розуміння внутрішньої структури застосунку, динамічний демонструє його поведінку у реальному середовищі, MAST створює інтегровану картину технічних характеристик безпеки, а ризик-орієнтований підхід забезпечує відповідність захисних заходів реальним потребам бізнесу. Сукупне застосування цих методик дозволяє досягти необхідної глибини оцінювання та сформувати обґрунтовані висновки щодо стійкості мобільного застосунку до сучасних загроз.

Критерії оцінки захищеності мобільного застосунку формують основу для системного аналізу його стійкості до сучасних кіберзагроз. У мобільному

середовищі складність оцінювання значно вища, ніж у класичних веб- або серверних системах, оскільки застосунок функціонує у напіввідкритому середовищі, де значна частина логіки виконується на стороні користувача. Саме тому критерії безпеки охоплюють не один, а кілька взаємопов'язаних рівнів архітектурний, криптографічний, мережевий, поведінковий та платформний. Кожен із них визначає свою частину вимог, і лише сукупна оцінка дозволяє сформувати об'єктивне уявлення про рівень захищеності фінансового застосунку [10].

Першим напрямом оцінювання є відповідність вимогам MASVS, які служать фактичним галузевим стандартом для мобільної безпеки. Їхня значущість полягає в тому, що MASVS визначає не лише рівень наявних захисних механізмів, але й відповідність загальної архітектури застосунку принципам безпечної розробки. Якщо застосунок не демонструє хоча б базового рівня відповідності MASVS, це свідчить про системні прогалини в проєктуванні, що неминуче призводитиме до критичних вразливостей незалежно від інших факторів.

Другим фундаментальним критерієм є якість реалізації криптографічних механізмів. Тут оцінюється не лише стійкість використаних алгоритмів, але й коректність управління ключовим матеріалом, захищеність місць його зберігання, наявність апаратної підтримки для криптографії та безпечність процедур генерації випадкових чисел. У фінансових застосунках саме порушення у криптографічних модулях найчастіше призводить до компрометації авторизаційних токенів або можливості підробки транзакційних підписів. Тому криптографія розглядається як критично важлива складова будь-якої системи оцінювання безпеки.

Наступним напрямом є дослідження взаємодії мобільного застосунку з API. Оцінка зосереджується на тому, наскільки серверна частина коректно перевіряє правомірність запитів і забезпечує свою стійкість до маніпуляцій. Уразливості на кшталт IDOR, SSRF чи ін'єкцій виникають саме через

некоректне трактування даних, отриманих від клієнта. Для фінансових сервісів такі недоліки є особливо небезпечними, оскільки можуть дозволити зловмиснику переглядати або змінювати дані інших користувачів, виконувати несанкціоновані операції або обходити транзакційні обмеження.

Оцінювання авторизаційних механізмів доповнює аналіз API. Тут увага приділяється не лише перевірці користувача, але й тому, як система підтримує його сесію в подальшому. Наявність механізмів оновлення токенів, їхня короткочасність, прив'язка до пристрою, коректність перевірки контексту усе це є невід'ємними складовими безпечної авторизації. Якщо хоча б один із цих параметрів реалізований неналежним чином, мобільний застосунок стає вразливим до атак, які дозволяють видавати себе за легітимного користувача або викрадати сеансові дані під час мережевої взаємодії.

Окремий критерій стосується здатності застосунку протистояти реверс-інжинірингу. У мобільному середовищі зловмисник може отримати фізичний доступ до виконуваного файлу, дослідити його або змінити певні компоненти. Тому важливо оцінити наявність обфускації, механізмів перевірки цілісності, блокування виконання на модифікованих пристроях та здатність застосунку виявляти інструменти динамічної ін'єкції коду. Захист від реверс-інжинірингу є особливо важливим для фінансових застосунків, адже будь-яка можливість модифікувати логіку або перехоплювати внутрішні виклики може дати зловмиснику контроль над транзакційними процесами.

Ще одним важливим аспектом є стійкість до атак у мережевому середовищі. Захищений канал зв'язку повинен гарантувати цілісність і конфіденційність даних, що передаються між мобільним застосунком і сервером. У цьому контексті оцінюється правильність реалізації TLS, наявність SSL-pinning, перевірка сертифікатів і здатність протистояти MITM-атакам. Якщо застосунок демонструє довіру до будь-якого сертифіката або не перевіряє структуру ланцюга довіри, ризик перехоплення даних суттєво зростає.

Зрештою, вагоме значення має відповідність вимогам PSD2 щодо сильної клієнтської автентифікації. Для фінансових застосунків недостатньо просто мати багатофакторний механізм автентифікації він повинен забезпечувати незалежність факторів, захищеність їх зберігання та прив’язку до конкретного пристрою. Оцінювання відповідності вимогам SCA дозволяє визначити, наскільки мобільний застосунок здатний протистояти атакуючому, який намагається обійти автентифікацію або підмінити елементи, що формують транзакцію.

Сукупність цих критеріїв формує комплексну модель оцінювання безпеки мобільного застосунку. Вона дає змогу розглядати продукт не фрагментарно, а як цілісну систему, у якій захищеність кожного компонента визначає надійність усього фінансового сервісу. Саме така багатоаспектність є ключовою передумовою для побудови ефективної методики практичного аналізу, що буде застосована у подальших розділах роботи.

1.3. Методи дослідження та інструментарій оцінки захищеності

Аналіз безпеки мобільних застосунків спирається на комплекс методів, кожен з яких дає змогу розглядати застосунок із різних точок зору та виявляти специфічні типи вразливостей. Особливість мобільного середовища полягає в тому, що застосунок поєднує локальну логіку, платформні механізми операційної системи та постійну взаємодію з віддаленим сервером. Відповідно, ефективна оцінка його безпеки не може обмежуватися аналізом одного рівня, а вимагає поєднання статичних, динамічних і поведінкових підходів. Саме ця багаторівневість зумовлює використання різних методів аналізу, кожен з яких виконує власну функцію в загальній структурі дослідження [11].

Статичний аналіз є вихідною точкою у процесі дослідження мобільного застосунку. Він здійснюється без запуску програми і зосереджується на вивченні її внутрішньої структури, що включає дослідження бінарних файлів,

декомпіляцію коду, аналіз конфігураційних елементів та виявлення потенційно небезпечних залежностей. У фінансовому секторі статичний аналіз дозволяє виявити такі критичні ризики, як наявність жорстко заданих ключів, нешифроване зберігання токенів, використання застарілих криптографічних алгоритмів або неправильно налаштовані дозволи. На цьому етапі дослідник отримує перше уявлення про рівень захищеності застосунку, якість реалізації безпекових механізмів і потенційні слабкі місця в архітектурі.

Динамічний аналіз дає змогу розглянути застосунок у середовищі, що максимально наближене до реального. Основна увага зосереджується на тому, як мобільний застосунок поводиться під час виконання, які операції здійснює, як взаємодіє з операційною системою, яким чином управляє даними та як реагує на виклики, що надходять із мережі. Цей метод дозволяє зафіксувати реальну поведінку застосунку і виявити вразливості, які не проявляються у статичному коді, але виникають через некоректну логіку або непередбачувані сценарії використання. Динамічний аналіз є особливо важливим для фінансових мобільних застосунків, оскільки дозволяє оцінити стійкість до MITM-атак, перевірити коректність виконання криптографічних операцій та визначити, чи не відбувається витік конфіденційної інформації під час комунікації з сервером.

Реверс-інжиніринг доповнює попередні методи і фокусується на вивченні внутрішньої структури застосунку з метою визначення того, наскільки добре він захищений від спроб несанкціонованого аналізу або модифікації. В умовах мобільної платформи користувач або зловмисник може отримати доступ до виконуваного файлу застосунку, декомпілювати його та проаналізувати логіку роботи. Тому реверс-інжиніринг є не лише засобом виявлення вразливостей, але й методом оцінювання надійності механізмів захисту, таких як обфускація, перевірка цілісності, захист від дебагу та перепакування. Для фінансових застосунків цей аспект є критичним, адже недостатній захист дозволяє зловмиснику виявити функції генерації токенів, внутрішні структури API або механізми перевірки транзакцій.

Окрему роль відіграє аналіз API як невід'ємної частини мобільного застосунку. API фактично є точкою доступу до всієї логіки, що виконується на сервері, а отже, гарантує або підриває захищеність усієї системи. Аналіз API полягає у вивченні маркерів автентифікації, структури запитів і відповідей, перевірці прав доступу, а також у моделюванні взаємодії між клієнтом і сервером за умов, максимально наближених до реальної зловмисної активності. Особливу увагу приділяють перевірці того, чи не можна обійти авторизацію, змінити параметри іншого користувача, вплинути на логіку виконання транзакцій або змусити сервер обробляти запити в небезпечний спосіб. Якщо клієнтський застосунок функціонує у відносно неконтрольованому середовищі, то API це саме той рівень, де має бути реалізована основна оборона [12].

У сукупності ці методи формують комплексний підхід, необхідний для оцінювання захищеності мобільного застосунку фінансового спрямування. Жоден із них не є самодостатнім: статичний аналіз дозволяє побачити внутрішню структуру, але не дає уявлення про поведінку під час виконання; динамічний аналіз демонструє реакцію застосунку на реальні сценарії, але не пояснює глибинних причин вразливостей; реверс-інжиніринг виявляє слабкість механізмів захисту, але не завжди дозволяє оцінити наслідки для бізнес-логіки; аналіз API визначає межі доступу, але не пояснює можливості обходу перевірок на клієнтському рівні. Лише їхнє інтегроване застосування забезпечує достатню глибину дослідження та дозволяє сформувати адекватну оцінку того, наскільки мобільний застосунок відповідає сучасним вимогам безпеки.

Статичний аналіз мобільного застосунку передбачає вивчення його внутрішньої структури та програмного коду без запуску програми. Цей підхід дає змогу отримати системне уявлення про архітектуру клієнтської частини, логіку взаємодії між компонентами, якість реалізації механізмів безпеки та наявність потенційно небезпечних рішень. У контексті фінансових мобільних застосунків статичний аналіз має особливе значення, оскільки саме на цьому етапі можна виявити помилки, що виникли внаслідок неправильного

проєктування або недбалості під час розробки, а також визначити слабкі місця, які надалі можуть бути використані зловмисниками під час атак.

Одним із ключових інструментів, що застосовуються для статичного аналізу, є MobSF комплексна платформа, яку широко використовують як спеціалісти з безпеки, так і розробники. Перевага MobSF полягає в тому, що вона здатна автоматично декомпілювати мобільний застосунок, аналізувати його структуру, визначати критичні конфігураційні елементи та виявляти відомі патерни небезпечної поведінки. Інструмент дозволяє отримати детальні звіти, які охоплюють характеристики застосунку, параметри безпеки, аналіз дозволів, підозрілу активність у коді, а також відповідність вимогам MASVS. Для фінансових застосунків така автоматизація є особливо корисною, оскільки надає змогу швидко оцінити загальний рівень захищеності й визначити ділянки, що потребують глибшого аналізу.

Декомпілятори, такі як JADX та apktool, розширюють можливості дослідника, дозволяючи працювати безпосередньо з вихідними компонентами застосунку. Вони забезпечують доступ до структури проєкту, включно з файлами ресурсів, конфігураційними файлами та частково відновленим вихідним кодом. Завдяки цьому стає можливим виявлення жорстко закодованих секретів ключів, паролів, строк підключення до API чи інших параметрів, які за належної реалізації не повинні зберігатися у клієнтській частині. У фінансових застосунках подібні помилки можуть мати катастрофічні наслідки, оскільки компрометація ключового матеріалу відкриває шлях до підробки автентифікаційних даних та маніпуляції транзакціями [13].

Статичний аналіз дозволяє також оцінити захищеність механізмів зберігання даних, що використовуються застосунком. Дослідження способу роботи з файловою системою, базами даних SQLite або механізмами кешування дає змогу визначити, чи забезпечується належний рівень шифрування та чи дотримано стандартів обробки конфіденційної інформації. Неправильна робота із збереженням даних може стати критичною загрозою, оскільки на мобільних

пристроях з рутованим доступом або під час зняття резервних копій такі дані можуть бути прочитані без жодних обмежень.

Не менш важливою складовою статичного аналізу є перевірка механізмів криптографії. Дослідник має можливість оцінити правильність вибору алгоритмів, коректність їх конфігурації та дотримання вимог щодо генерації ключів і початкових векторів. У фінансових застосунках криптографічні модулі відіграють роль основного механізму захисту транзакційних даних, і будь-яке порушення у цій сфері призводить до втрати довіри до системи. Статичний аналіз допомагає визначити, чи використовуються сучасні алгоритми, чи не містять вони відомих вразливостей та чи не порушено логіку побудови криптографічних операцій.

Статичний аналіз є незамінним також у процесі оцінювання архітектурних рішень. Досліджуючи структуру застосунку, можна визначити, наскільки продумано розмежовано відповідальності між компонентами, чи є надмірна відкритість інтерфейсів, та чи реалізовано механізми, що забороняють неконтрольовану взаємодію між модулями. У фінансовому секторі неправильне архітектурне рішення може дозволити зловмиснику вплинути на ключові функції застосунку або навіть підмінити дані на ранніх етапах обробки.

Нарешті, статичний аналіз дає змогу оцінити наявність та ефективність механізмів захисту від реверс-інжинірингу. Обфускація, контроль цілісності, виявлення модифікацій пакету чи засобів налагодження усе це може бути перевірено без запуску застосунку. Якщо такі механізми відсутні або реалізовані формально, дослідник може зробити висновок про високу ймовірність того, що зловмисник зможе дослідити внутрішню логіку застосунку та скомпрометувати чутливі елементи, які мають залишатися недоступними.

У результаті статичний аналіз формує базове уявлення про рівень захищеності мобільного застосунку задовго до того, як дослідник перейде до динамічного або поведінкового аналізу. Він дозволяє визначити ключові вектори ризику, оцінити архітектурні рішення, перевірити відповідність

формальним стандартам і виявити технічні помилки, які залишилися поза увагою під час розробки. У фінансовому секторі, де застосунок взаємодіє з критично важливою інфраструктурою, статичний аналіз є першою і однією з найбільш значущих складових загальної методології оцінювання безпеки.

Динамічний аналіз мобільного застосунку передбачає дослідження його поведінки під час виконання, коли програма працює у середовищі, максимально наближеному до реального використання. На відміну від статичного аналізу, який дає змогу оцінити структуру та внутрішню будову застосунку, динамічний підхід дозволяє спостерігати за тим, як застосунок реагує на конкретні дії, мережеві події, системні виклики та різноманітні взаємодії з операційною системою. Для фінансових мобільних сервісів цей вид аналізу є особливо важливим, оскільки саме в процесі реальної роботи найчастіше проявляються критичні помилки, логічні недоліки та поведінкові аномалії, які можуть бути використані зловмисником [14].

Одним із ключових аспектів динамічного аналізу є дослідження мережевої активності застосунку. У фінансових системах передача даних між клієнтською та серверною частинами є постійним процесом, що передбачає відправлення та отримання конфіденційних відомостей, зокрема токенів автентифікації, параметрів транзакцій або даних про користувача. Аналізуючи мережевий трафік, дослідник може визначити, чи використовує застосунок захищені канали зв’язку, як обробляє TLS-сертифікати, чи коректно застосовує механізми перевірки цілісності та автентичності сервера. Виявлення ситуацій, у яких застосунок некритично приймає сторонні сертифікати або неправильно обробляє TLS-помилки, майже завжди свідчить про можливість реалізації атак типу “людина посередині”, що у фінансовому контексті може призвести до перехоплення транзакційних даних і повної компрометації системи.

Інструменти на кшталт Burp Suite дозволяють проводити такі дослідження на практиці, надаючи можливість перехоплювати, аналізувати та модифікувати мережеві запити в реальному часі. Завдяки цьому дослідник може

оцінити, як сервер реагує на некоректні параметри, чи існують механізми захисту від підміни даних та чи дотримується мобільний застосунок принципу захисту від повторного використання пакетів. У фінансовому середовищі будь-яка здатність модифікувати запит до сервера без належної перевірки на стороні сервера або застосунку може означати можливість для несанкціонованого здійснення операцій.

Динамічна ін’єкція коду становить інший важливий напрям динамічного аналізу. Використання інструментів, таких як Frida або Objection, дозволяє досліднику втручатися у виконання застосунку, перехоплювати системні виклики, змінювати функціональність та обходити механізми захисту. Ця категорія інструментів має важливе значення тому, що вона моделює дії реального зловмисника, який намагається обійти обмеження, накладені розробником. Якщо застосунок не перевіряє цілісність свого коду, не блокує підключення дебагера або не виявляє характерні ознаки інструментів маніпуляції, це створює умови для підробки авторизаційних процесів та модифікації критичної логіки. У фінансовому контексті це може означати зміну суми транзакції, підміну реквізитів або обхід механізмів підтвердження операцій.

Динамічний аналіз також дає змогу оцінити взаємодію застосунку з операційною системою. Багато мобільних загроз виникають через неправильне використання системних ресурсів, надмірні дозволи або доступ до небезпечних компонентів середовища. Під час виконання можна зафіксувати, до яких ресурсів звертається застосунок, які системні виклики генерує, чи намагається зчитувати інформацію, доступ до якої він не повинен мати. Таке дослідження дозволяє виявити приховані функції або небажані побічні ефекти роботи програми, що можуть свідчити про наявність вразливостей або потенційно небезпечної поведінки.

Особливе значення у динамічному аналізі має здатність застосунку протистояти модифікації середовища виконання. У випадку мобільних

фінансових застосунків важливо зрозуміти, як програма реагує на запуск на рутованому пристрої, у емуляторі або в середовищі, де відсутні критичні системні функції. Якщо застосунок продовжує працювати без обмежень, це може означати, що він не вживає достатніх заходів контролю, що створює можливість атак з боку шкідливого ПЗ або користувача із підвищеними правами. У багатьох випадках фінансові застосунки повинні відмовлятися працювати у таких умовах або вимагати додаткових кроків для перевірки автентичності середовища.

Таким чином, інструменти динамічного аналізу дозволяють глибоко дослідити поведінку мобільного застосунку в реальних умовах, оцінити коректність взаємодії з сервером, визначити можливості для перехоплення або модифікації даних, а також виявити слабкі місця у механізмах захисту від маніпуляцій. На відміну від статичного аналізу, який демонструє лише потенціал безпекових недоліків, динамічний підхід показує, як ці недоліки проявляються під час реальної експлуатації. У контексті фінансових сервісів динамічний аналіз є незамінним, оскільки він дозволяє виявити ті вразливості, які можуть мати безпосередній вплив на транзакції, доступ до рахунків та взаємодію користувача із фінансовими інструментами.

Оцінювання результатів аналізу мобільного застосунку завершує процес дослідження й дозволяє перетворити виявлені технічні дані на цілісне уявлення про стан його безпеки. На цьому етапі важливо не лише перелічити знайдені вразливості, а й визначити їхню природу, взаємозв’язки та реальний вплив на функціонування фінансової інфраструктури. У мобільному середовищі, де клієнтська логіка, серверна частина та мережеві процеси тісно переплетені, ізольований розгляд окремих недоліків не дає повної картини. Тому першим кроком є узагальнення результатів таким чином, щоб виявити системні проблеми, що охоплюють декілька компонентів застосунку.

Ключовим аспектом оцінювання є визначення рівня ризику кожної виявленої вразливості. Це передбачає аналіз не лише технічної складності її

експлуатації, але й потенційних наслідків для конфіденційності даних, цілісності транзакцій та загальної доступності сервісу. У фінансових застосунках навіть відносно прості вразливості можуть мати критичний вплив, якщо вони стосуються авторизаційних токенів, управління сесіями або механізмів підтвердження операцій. Тому оцінювання ризиків завжди повинно враховувати сценарії реального використання та можливості зловмисника діяти в різних середовищах від легітимного пристрою до рутованих чи скомпрометованих систем [15].

Важливим елементом оцінювання є співставлення отриманих результатів із вимогами галузевих стандартів, насамперед MASVS, PCI DSS, NIST SP 800-63 і PSD2. Таке порівняння надає об’єктивний орієнтир для визначення того, наскільки застосунок відповідає сучасним вимогам безпеки. Якщо певна вразливість суперечить конкретному положенню стандарту, це вказує не лише на технічну прогалину, але й на недотримання нормативних вимог, що є особливо важливим у фінансовому секторі.

Завершальним кроком є формування висновку про загальний рівень захищеності застосунку та визначення пріоритетів щодо усунення недоліків. Такий висновок повинен відображати як технічні аспекти безпеки, так і рекомендації щодо архітектурних чи процесних змін, необхідних для підвищення рівня стійкості. У фінансових системах особливу увагу приділяють механізмам автентифікації, захисту ключового матеріалу, коректності роботи API та стійкості до атак у мережевому середовищі. Оцінювання результатів аналізу фактично перетворює окремі знахідки на цілісну модель ризиків, що слугує основою для подальших рекомендацій та визначення напрямів удосконалення безпеки.

Висновки до розділу 1

У першому розділі досліджено теоретичні засади, на яких базується сучасний підхід до оцінки захищеності мобільних застосунків фінансового сектору. Аналіз особливостей архітектури мобільних застосунків дозволив визначити ключові компоненти, що формують модель загроз: клієнтське середовище, мережеву взаємодію та серверну інфраструктуру. Було встановлено, що саме багаторівневий характер цієї архітектури створює широкий спектр можливих векторів атак, які охоплюють як внутрішню логіку застосунку, так і механізми автентифікації, зберігання даних та обробки транзакцій.

Подальший огляд міжнародних стандартів безпеки показав, що сучасні фінансові мобільні застосунки повинні відповідати комплексним нормативним вимогам, серед яких особливе місце займають PCI DSS, PSD2/RTS, NIST SP 800-63 та рекомендації OWASP MASVS і MSTG. Ці стандарти не лише визначають обов’язкові вимоги щодо захисту даних і цифрової ідентичності, але й задають рамки для побудови архітектури, вибору криптографічних механізмів і реалізації систем контролю доступу. Дотримання цих стандартів є критично важливим для забезпечення стійкості фінансових сервісів до сучасних кіберзагроз.

Розгляд методів та інструментарію оцінювання захищеності показав, що жоден окремий метод не може забезпечити повного охоплення всіх можливих ризиків. Лише поєднання статичного аналізу, динамічного тестування, реверс-інжинірингу та дослідження API дозволяє сформувати всебічну і об’єктивну оцінку. Крім того, було встановлено, що результати аналізу повинні оцінюватися не лише з технічної точки зору, але й через призму реальних ризиків та нормативних вимог, що особливо актуально у фінансовому секторі.

Таким чином, теоретичний розділ сформував цілісну методологічну основу, що визначає ключові аспекти безпеки мобільних фінансових

застосунків, характерні моделі загроз і критерії їх оцінювання. Ця основа є необхідною передумовою для переходу до практичної частини дослідження, у якій будуть застосовані відповідні методи аналізу для оцінювання реального мобільного застосунку та інтерпретації отриманих результатів.

РОЗДІЛ 2. ПРАКТИЧНА ОЦІНКА ЗАХИЩЕНОСТІ МОБІЛЬНОГО ЗАСТОСУНКУ НА ОСНОВІ СУЧАСНОГО ІНСТРУМЕНТАРІЮ

Практична оцінка захищеності мобільного застосунку є ключовим етапом дослідження, адже саме вона дозволяє перевірити на реальному прикладі ефективність методів, стандартів та підходів, розглянутих у попередньому розділі. Теоретичні положення набувають повноцінного змісту лише тоді, коли їх можна співставити з фактичними результатами тестування та аналізу поведінки застосунку в умовах, наближених до реальної експлуатації. У мобільному фінансовому середовищі це особливо важливо, оскільки значна частина ризиків проявляється не лише на рівні архітектури або коду, а у способі взаємодії застосунку з платформою, сервером та користувачем.

Розділ зосереджується на виборі об’єкта дослідження, визначенні цілей і задач практичної оцінки безпеки, а також на застосуванні сучасного інструментарію для виявлення та аналізу вразливостей. Особлива увага приділяється інструментам, здатним виявити слабкі місця в роботі застосунку як у статичному стані, так і під час виконання. Це дозволяє комплексно охопити ті аспекти безпеки, які найчастіше стають об’єктом атак у фінансових сервісах: зберігання ключового матеріалу, автентифікаційні механізми, роботу з API, мережеві транзакції та реакцію на спроби втручання у середовище.

У межах розділу буде проведено практичне тестування обраного мобільного застосунку, інтерпретовано результати та співставлено їх із вимогами міжнародних стандартів, зокрема MASVS, MSTG та нормативами, характерними для фінансового сектору. Це дозволить визначити сильні та слабкі сторони реалізації його механізмів безпеки, а також сформувати основу для рекомендацій, що будуть наведені у подальшому розділі. Таким чином, практичний етап дослідження не лише підтверджує або спростовує теоретичні припущення, але й є критично важливим для формування об’єктивної оцінки захищеності мобільного фінансового застосунку.

2.1. Вибір об’єкта дослідження та постановка задач оцінки безпеки

Для практичної частини роботи було обрано мобільний застосунок Damn Vulnerable Bank (DVB), спеціально створений як навчальне середовище для дослідження вразливостей, характерних для фінансових мобільних сервісів. На відміну від реальних банківських застосунків, DVB навмисно містить широкий набір типових помилок у галузі мобільної безпеки від некоректної роботи з автентифікацією до слабких механізмів захисту даних та API. Завдяки цьому він є оптимальним інструментом для відпрацювання методів аналізу, описаних у теоретичному розділі, та дозволяє на практиці продемонструвати, як саме формуються вразливості у фінансових застосунках і які наслідки вони можуть мати.

Вибір саме DVB обумовлений тим, що цей застосунок моделює реальні сценарії роботи банківського клієнта: авторизацію, перегляд балансу, виконання транзакцій, взаємодію з сервером та опрацювання конфіденційних даних. Це забезпечує можливість дослідити як клієнтські механізми, так і серверні компоненти, включно з API та логікою обробки запитів. Наявність відомих вразливостей не лише спрощує проведення аналізу, а й дозволяє зосередитися на демонстрації методології тестування, правильному оформленні результатів та інтерпретації їхнього впливу на безпеку [16].

Метою тестування є комплексна оцінка рівня захищеності банківського мобільного застосунку, моделювання можливих сценаріїв атак та визначення того, наскільки застосунок здатний протистояти загрозам, характерним для фінансових сервісів. Тестування має показати, чи забезпечує DVB належний рівень захисту персональних даних, банківської інформації та транзакцій, а також чи відповідає він принципам безпечної мобільної розробки, визначеним у першому розділі.

Особливе значення має саме банківська природа застосунку, оскільки у такому контексті будь-яка вразливість може прямо вплинути на цілісність фінансових операцій, конфіденційність даних клієнтів та надійність механізмів

автентифікації. Тому тестування спрямовується не лише на пошук технічних недоліків, але й на оцінку ризиків, які вони створюють для банківського середовища.

Завдання тестування полягає у проведенні комплексного дослідження мобільного застосунку з використанням статичного, динамічного та поведінкового аналізу, що дозволяє охопити всі ключові аспекти його роботи. У межах цього дослідження необхідно виявити вразливості, пов’язані зі зберіганням конфіденційних даних, механізмами автентифікації, роботою криптографічних модулів та обробкою транзакцій, а також проаналізувати взаємодію застосунку з API й визначити, наскільки коректно реалізовано серверну логіку. Важливо оцінити стійкість застосунку до атак типу «людина посередині», спроб реверс-інжинірингу та маніпуляцій під час виконання, оскільки саме ці вектори є найбільш критичними для фінансових мобільних сервісів. Узагальнення отриманих результатів дає змогу визначити загальний рівень захищеності DVB та виявити ті компоненти, які становлять найбільший ризик для безпеки банківського застосунку [17].

Тестування мобільного застосунку здійснюватиметься поетапно, охоплюючи всі ключові аспекти методології аналізу безпеки. Перший етап передбачає підготовку середовища: налаштування інфраструктури для тестування, встановлення DVB та підготовку інструментів статичного й динамічного аналізу, таких як MobSF, JADX, Burp Suite і Frida. На цьому ж етапі моделюються умови, максимально наближені до роботи реального банківського застосунку, що дозволяє коректно оцінити поведінку системи у типовому середовищі.

Подальший статичний аналіз спрямований на дослідження внутрішньої структури застосунку, декомпіляцію коду та оцінку його архітектури з точки зору безпеки. На цьому етапі визначається, як організовано зберігання чутливих даних, чи коректно застосовано криптографічні механізми та наскільки безпечно реалізовано взаємодію з операційною системою.

Динамічний аналіз дозволяє перейти до дослідження фактичної поведінки застосунку під час виконання. Особлива увага приділяється роботі мережевих компонентів, зокрема реакції на спроби підміни сертифікатів, стійкості до MITM-атак, коректності обробки запитів та здатності протидіяти динамічній ін’єкції коду. Це дає змогу визначити, наскільки надійно захищений застосунок у реальних умовах експлуатації.

Окремим етапом є аналіз API, у межах якого оцінюється взаємодія клієнтської частини із серверними компонентами. Досліджуються механізми автентифікації та авторизації, перевіряється коректність параметрів транзакцій і здатність серверної логіки належним чином контролювати доступ та дані, що надходять від клієнта.

Завершальним етапом є інтерпретація отриманих результатів. Виявлені вразливості класифікуються за рівнем ризику, аналізуються з позиції банківської безпеки та узагальнюються для формування висновків щодо загального стану захисту. Такий поетапний підхід забезпечує послідовний рух від базового розуміння архітектури до глибокої оцінки критичних механізмів, які визначають безпеку мобільного банківського сервісу [18].

Оскільки DVB моделює функціональність реального банківського застосунку, особливу увагу приділено тим компонентам, що найчастіше стають об’єктом атак у фінансових системах: механізмам автентифікації, роботі із сесійними токенами, обробці транзакційних параметрів, захисту API та стійкості клієнтської частини до зовнішніх маніпуляцій. Це дозволяє сформувати всебічне уявлення про рівень безпеки застосунку та визначити ключові напрями його вдосконалення.

2.2. Використання інструментів для виявлення вразливостей

У межах практичного дослідження для виявлення вразливостей мобільного застосунку Damn Vulnerable Bank було використано набір

інструментів, що забезпечують повноцінне охоплення як статичного, так і динамічного аналізу. Основою тестового середовища слугував Android-пристрій, бажано з правами root, або емулятор на кшталт Genymotion чи MEmu, який дозволяє відтворити поведінку застосунку в умовах, наближених до реальних сценаріїв використання банківських сервісів. Наявність рутованого середовища є важливою, оскільки саме в таких умовах проявляються численні загрози, характерні для мобільних фінансових застосунків, зокрема можливість доступу до захищених сховищ, маніпулювання середовищем виконання та перехоплення внутрішніх викликів.

Інструмент adb застосовувався для взаємодії з пристроєм на низькому рівні: встановлення пакунків, зняття логів, дослідження файлової системи та управління процесами. Це дозволило оцінити роботу застосунку у різних середовищах, дослідити його реакцію на модифікацію середовища та отримати доступ до внутрішніх артефактів, які можуть містити конфіденційну інформацію або службові параметри [19].

Для аналізу поведінки застосунку під час виконання було використано Frida, яка дозволила перехоплювати та модифікувати виклики функцій у реальному часі. Завдяки цьому стало можливим перевірити стійкість DVB до динамічної ін’єкції коду, обійти механізми виявлення root-доступу та антидебагу, а також оцінити коректність роботи автентифікаційних та криптографічних механізмів. У контексті банківських мобільних застосунків такі перевірки є критичними, оскільки демонструють, чи може зловмисник змінити поведінку застосунку без доступу до його вихідного коду.

Інструменти apkx і apktool використовувалися для декомпіляції та аналізу структури пакунка застосунку. Вони дають змогу відтворити ресурси, конфігураційні файли та частини коду, необхідні для виявлення слабких місць у зберіганні даних, обробці дозволів або конфігурації WebView. Особливо цінним є те, що такий аналіз дозволяє виявити неправильно реалізовану бізнес-логіку, слабку політику паролів або наявність чутливих даних у незашифрованому

вигляді фактори, які мають критичне значення саме для банківських застосунків.

Для поглибленого дослідження логіки роботи застосунку на рівні машинного коду застосовувалася Ghidra, що дозволила виконати реверс-інжиніринг складних компонентів, які не піддаються повній декомпіляції стандартними інструментами. У випадку фінансових мобільних застосунків це особливо важливо, оскільки деякі механізми автентифікації, перевірки підписів або роботи з криптографією можуть бути реалізовані у вигляді нативних бібліотек, захованих за межами звичайного Android-коду.

Сукупне використання цих інструментів забезпечує системний огляд безпеки застосунку і дозволяє дослідити його з усіх необхідних ракурсів: від файлової системи та архітектури пакунка до динамічної поведінки і взаємодії з сервером. Саме такий комплексний підхід є необхідним для оцінки безпеки банківського мобільного застосунку, де кожен компонент криптографія, автентифікація, логування, обробка транзакцій або робота API може стати потенційною точкою компрометації.

2.2.1 Проведення статичного аналізу застосунку та виявлення вразливостей

Статичний аналіз є першим етапом дослідження безпеки мобільного застосунку і дозволяє оцінити його архітектуру, конфігурацію та логіку роботи без виконання програми. У межах аналізу DVB застосовувалися інструменти apktool, apkx та Ghidra, що забезпечили доступ до внутрішньої структури пакунка, дозволів, ресурсів та вихідних компонентів коду. Це дало можливість дослідити організацію зберігання даних, реалізацію функцій автентифікації, використання системних API та механізми взаємодії із серверною частиною.

У процесі статичного аналізу було встановлено, що після успішного проходження автентифікації застосунок зберігає токен доступу у файлі

SharedPreferences. Декомпіляція коду та дослідження внутрішньої структури пакунка показали, що механізм збереження не використовує жодних додаткових засобів шифрування або захисту цілісності. Це означає, що токен, який фактично виконує роль ключа доступу до облікового запису користувача, зберігається у відкритому вигляді і може бути прочитаний будь-ким, хто отримує доступ до файлової системи пристрою [20].

Така реалізація створює типову вразливість класу Insecure Data Storage, яка полягає в тому, що чутливі дані автентифікаційні токени, облікові параметри, особиста інформація зберігаються у місцях, що не забезпечують належного рівня захисту. На рутованих пристроях або у випадку шкідливих програм сторонні процеси можуть безперешкодно отримати доступ до директорії застосунку, переглянути вміст файлів конфігурації або експортувати дані для подальшої експлуатації. У контексті банківського мобільного застосунку це є критичною проблемою, оскільки доступ до незахищеного токена дозволяє зловмиснику увійти в систему без знання облікових даних користувача.

```
public void a(JSONObject jSONObject) throws JSONException {
    BankLogin bankLogin;
    Intent intent;
    try {
        JSONObject jSONObject2 = new JSONObject(e.a(jSONObject.get("enc_data").toString()));
        if (jSONObject2.getJSONObject("status").getInt("code") != 200) {
            Toast.makeText(BankLogin.this.getApplicationContext(), "Error: " + jSONObject2.getJSONObject("data").getString("message"), 0).show();
            bankLogin = BankLogin.this;
            intent = new Intent(BankLogin.this, (Class<?>) BankLogin.class);
        } else {
            String string = jSONObject2.getJSONObject("data").getString("accessToken");
            SharedPreferences sharedPreferences = BankLogin.this.getSharedPreferences("jwt", 0);
            Log.d("accesstoken", string);
            sharedPreferences.edit().putString("accesstoken", string).apply();
            sharedPreferences.edit().putBoolean("isloggedin", true).apply();
            bankLogin = BankLogin.this;
            intent = new Intent(BankLogin.this, (Class<?>) Dashboard.class);
        }
        bankLogin.startActivity(intent);
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
```

Рисунок 2.1 Фрагмент коду, що демонструє небезпечне зберігання токена доступу у SharedPreferences

У випадку Damn Vulnerable Bank збереження токена здійснюється у файлі shared_prefs/jwt.xml, розташованому в робочій директорії пакунка. На рутованому пристрої цей файл може бути прочитаний без будь-яких додаткових дозволів, що легко підтверджується через інтерфейс adb. Доступ до вмісту

файлу можна отримати за допомогою команди: cat /data/data/{identifier name}/shared_prefs/jwt.xml.

Рисунок 2.2 Зберігання сесійного токену

Команда повертає вміст XML-файлу, у якому токен доступу записано в явному вигляді, без шифрування або маскування. Це демонструє недостатність реалізованого механізму захисту та підтверджує, що застосунок не дотримується базових вимог до безпечного зберігання конфіденційних даних, визначених стандартами MASVS та рекомендаціями OWASP щодо роботи з чутливою інформацією на клієнтському пристрої.

У ході статичного аналізу було встановлено, що мобільний застосунок здійснює некоректну обробку журналів подій під час виконання операцій автентифікації. Зокрема, у модулі, відповідальному за процес входу, реалізовано механізм логування, який фіксує службові дані разом із токеном доступу, сформованим після успішної автентифікації користувача. Це означає, що чутлива інформація зокрема access token фактично виводиться до системного журналу, який за природою є незахищеним і може бути прочитаний сторонніми процесами або користувачем на рутованому пристрої.

Подібна поведінка відповідає класичному прояву вразливості Insecure Logging, що виникає у випадках, коли застосунок помилково записує конфіденційні дані до логів, призначених для налагодження або діагностики. Логи не призначені для зберігання безпеково значущої інформації, оскільки вони не шифруються, можуть бути передані стороннім сервісам та залишаються у файловій системі навіть після завершення сеансу користувача. У контексті банківського застосунку наслідки такої вразливості є особливо значущими, оскільки access token фактично відкриває повний доступ до облікового запису

користувача і може бути використаний для несанкціонованого виконання операцій [21].

```
public void a(JSONObject jSONObject) throws JSONException {
    BankLogin bankLogin;
    Intent intent;
    try {
        JSONObject jSONObject2 = new JSONObject(e.a(jSONObject.get("enc_data").toString()));
        if (jSONObject2.getJSONObject("status").getInt("code") != 200) {
            Toast.makeText(BankLogin.this.getApplicationContext(), "Error: " + jSONObject2.getJSONObject("data").getString("message"), 0).show();
            bankLogin = BankLogin.this;
            intent = new Intent(BankLogin.this, (Class<?>) BankLogin.class);
        } else {
            String string = jSONObject2.getJSONObject("data").getString("accessToken");
            SharedPreferences sharedPreferences = BankLogin.this.getSharedPreferences("jwt", 0);
            Log.d("accesstoken", string);
            sharedPreferences.edit().putString("accesstoken", string).apply();
            sharedPreferences.edit().putBoolean("isloggedin", true).apply();
            bankLogin = BankLogin.this;
            intent = new Intent(BankLogin.this, (Class<?>) Dashboard.class);
        }
        bankLogin.startActivity(intent);
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
```

Рисунок 2.3 Фрагмент коду, що демонструє небезпечне зберігання токена

Для підтвердження факту небезпечного логування було проведено динамічний огляд журналів за допомогою adb. Логи застосунку можна проаналізувати безпосередньо під час його виконання за допомогою команди:

adb logcat | grep $(adb shell ps | grep {identifier name} | awk '{print $2}')

Після виконання команди виводяться записи журналу, у яких міститься токен доступу, сформований під час успішного входу користувача. Наявність цієї інформації у журналі підтверджує, що застосунок не здійснює належної фільтрації або маскування чутливих даних і порушує вимоги до безпечного логування, встановлені рекомендаціями OWASP MASVS (зокрема MASVS-STORAGE та MASVS-AUTH).

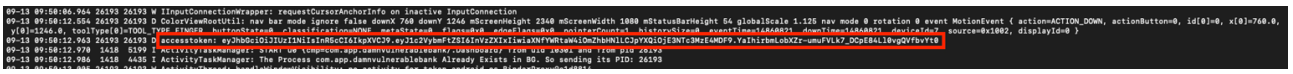

Рисунок 2.4 Небезпечне зберігання сесійного токена

Виявлена вразливість свідчить про недостатній контроль над роботою діагностичних механізмів та демонструє ризик витоку даних у середовищах, де зловмисник має можливість переглядати системні журнали. У реальному банківському застосунку така помилка могла б дозволити стороннім особам отримати непрямий доступ до фінансової інформації або повністю

компрометувати обліковий запис користувача без необхідності обходити механізми автентифікації.

У процесі статичного аналізу було встановлено, що одна з активностей застосунку зокрема CurrencyRates містить механізм обробки зовнішніх deeplink-посилань, які можуть передавати URL для відображення у вбудованому WebView. Аналіз конфігурації AndroidManifest та декомпільованого коду показав, що застосунок не здійснює жодної перевірки походження, структури або безпечності переданого URL. У результаті WebView представляє будь-який отриманий URI як довірений, автоматично завантажуючи його без попередньої валідації.

Таке функціонування створює класичну вразливість WebView-компонента, що виникає тоді, коли застосунок приймає зовнішні посилання та передає їх у WebView без фільтрації. За відсутності контролю над параметрами deeplink зловмисник може сформувати довільний URL, включно з адресами шкідливих сайтів або посиланнями, що містять JavaScript-навантаження. У контексті банківського застосунку це становить значний ризик, оскільки WebView працює всередині довіреного середовища застосунку, а користувач може вважати отриманий контент частиною легітимного функціоналу.

Детальніше дослідження activity CurrencyRates показало, що вхідний параметр URL передається у WebView без перевірки домену, без обмеження дозволених схем URI та без будь-яких фільтрів. У такій конфігурації WebView може рендерити будь-який контент, зокрема виконувати ін’єкції JavaScript. Це дозволяє реалізувати XSS-атаку у WebView, яка в нормальних умовах не повинна бути можлива у локальному мобільному контексті [22].

Для підтвердження цього припущення було змодельовано сценарій атаки, який передбачав передачу спеціально сформованого посилання через локальний HTML-файл. Посилання містило JavaScript-навантаження у форматі javascript:alert('hacked'), інкапсульоване в параметр зовнішнього URL. Після

збереження HTML-файлу на пристрої та його відкриття у браузері, подальший перехід за сформованим посиланням і вибір відкриття його через DVB призвели до того, що WebView без будь-яких перевірок виконав переданий JavaScript-код. Це підтвердило наявність вразливості та продемонструвало, що застосунок допускає виконання довільного скрипту в межах власного інтерфейсу.

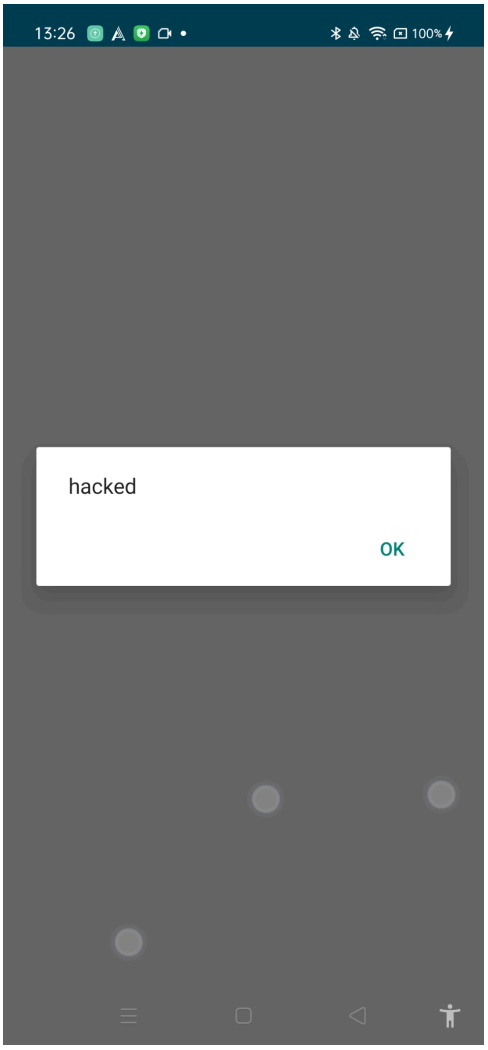

Рисунок 2.5 Підтвердження вразливості

У реальних банківських застосунках подібні помилки у конфігурації WebView мають серйозні наслідки. Зловмисник може змусити застосунок завантажити фішинговий ресурс, підмінити логіку відображення даних, отримати доступ до елементів веб-інтерфейсу або здійснювати атаки соціальної інженерії, використовуючи довіру користувача до інтерфейсу легітимного банківського застосунку. Таким чином, виявлена у DVB вразливість демонструє

важливість строгого контролю вхідних параметрів deeplink та необхідність обмеження функціональності WebView, відповідно до рекомендацій MASVS-PLATFORM та MASVS-ARCH.

Проведений статичний аналіз дозволив окреслити низку вразливостей, що безпосередньо впливають на рівень захищеності мобільного банківського застосунку. Насамперед було виявлено порушення принципів безпечного зберігання даних: токен доступу зберігається у відкритому вигляді у SharedPreferences, що робить його доступним на рутованих пристроях і створює можливість несанкціонованого доступу до облікового запису. Дослідження логувальних механізмів також підтвердило наявність витоку чутливої інформації, оскільки застосунок записує access token до системного журналу, що суперечить вимогам безпечної обробки конфіденційних даних.

Крім того, аналіз конфігурації WebView продемонстрував відсутність належної валідації зовнішніх посилань, що призвело до можливості відкриття небезпечних URI та виконання JavaScript-коду в контексті застосунку. Така поведінка створює додатковий вектор атак, пов’язаний із маніпуляцією інтерфейсом, фішинговими сценаріями та потенційним виконанням небажаних скриптів.

Загалом цей етап показав, що навіть на рівні статичного аналізу DVB демонструє суттєві відхилення від вимог безпечної мобільної розробки. Виявлені проблеми охоплюють різні аспекти від захисту даних та логування до конфігурації WebView і свідчать про необхідність подальшого поглибленого дослідження в межах динамічного аналізу та тестування API.

Загалом проведений статичний аналіз продемонстрував свою ефективність як початковий етап оцінювання безпеки мобільного застосунку. Саме аналіз внутрішньої структури пакунка, декомпіляція коду та дослідження конфігураційних файлів дали можливість виявити критичні недоліки, які залишалися б непомітними під час поверхневого тестування. Незахищене зберігання токенів було виявлено завдяки аналізу файлової структури та логіки

роботи SharedPreferences; небезпечне логування стало очевидним після перегляду журналювальних функцій та їхніх викликів; відсутність валідації deeplink-посилань у WebView була встановлена через аналіз активностей і відповідних фрагментів клієнтського коду. Таким чином, статичний аналіз не лише дозволив ідентифікувати вразливості різної природи, але й сформував основу для подальшого динамічного тестування, окресливши ті компоненти застосунку, які потребують поглибленої перевірки під час виконання.

2.2.2 Проведення динамічного аналізу застосунку та виявлення вразливостей

Динамічний аналіз є наступним етапом оцінювання безпеки мобільного застосунку і передбачає вивчення його роботи під час реального виконання. На відміну від статичного дослідження, що дає змогу оцінити архітектуру та внутрішню структуру програми, динамічний підхід дозволяє побачити фактичну поведінку застосунку у взаємодії з операційною системою, мережевою інфраструктурою та серверними компонентами. Саме під час виконання проявляються ті властивості, які неможливо повністю оцінити через аналіз коду: реакція на зміну середовища, механізми контролю цілісності, обробка помилок, робота криптографічних функцій і коректність мережевої комунікації.

У межах динамічного аналізу застосунку Damn Vulnerable Bank тестування проводилося у середовищі, наближеному до реальних умов використання банківського мобільного сервісу. Це включало запуск застосунку на рутованому пристрої або емуляторі, налаштування інструментів для перехоплення та модифікації трафіку, а також використання інструментів динамічної ін’єкції коду. Такий підхід дозволяє оцінити не лише стійкість застосунку до зовнішніх впливів, але й його здатність забезпечувати цілісність

даних і коректну обробку транзакцій у зміненому або потенційно небезпечному середовищі.

Особливе значення динамічний аналіз має для банківських мобільних застосунків, де критичною є кожна операція, пов’язана зі збереженням або передачею чутливих даних. Наявність слабких місць у мережевій взаємодії, контролі автентифікації або роботі клієнтської логіки може безпосередньо вплинути на фінансову безпеку користувачів. Саме тому проведення динамічного аналізу є необхідним для всебічної оцінки захищеності DVB та подальшої інтерпретації поведінкових характеристик застосунку в умовах потенційного впливу зловмисника.

Під час динамічного аналізу було встановлено, що мобільний застосунок містить реалізацію механізму *root detection*, призначеного для виявлення пристроїв Android, у яких користувач отримав привілейований доступ до системних ресурсів. Наявність такого механізму є типовою для банківських застосунків, оскільки рутоване середовище створює умови для перехоплення, модифікації або підміни чутливих даних і може бути використане зловмисником для повного контролю над виконанням програми. У нормальних умовах застосунок має обмежувати свою роботу або повністю припиняти її на рутованому пристрої, тим самим зменшуючи можливі ризики компрометації.

Дослідження показало, що реалізований у DVB механізм перевірки root можна обійти кількома способами. Один з них передбачає використання Magisk популярного фреймворку для керування root-доступом. Завдяки функції *DenyList* Magisk дозволяє приховати факт рутування від конкретного застосунку. Додавши пакет DVB до списку виключення, застосунок перестає отримувати індикатори модифікованого середовища, і перевірка root перестає спрацьовувати. У такому випадку застосунок продовжує роботу, навіть якщо середовище є потенційно небезпечним.

Другий спосіб обходу передбачає використання Frida інструмента для динамічного втручання у виконання програми. Виконання готового

Frida-скрипту, розміщеного у відкритих репозиторіях, дозволило перехопити та модифікувати функції, які визначають наявність root-доступу. Запуск команди: frida --codeshare d3tonator/ssl-and-root-detection-bypass -f {identifier name} -U.

призводить до ін’єкції коду, який підміняє результат перевірок на користь «безпечного» середовища. Це підтверджується тим, що після запуску застосунку не з’являється попереджувальне повідомлення *“Phone is rooted”*, яке зазвичай виконується при успішному виявленні рутованого пристрою.

Крім використання готових скриптів, можливо також створити власний фрагмент коду для обходу перевірки. Для цього за допомогою JADX визначається конкретна функція, що реалізує root detection, після чого відповідний фрагмент логіки модифікується в індивідуальному Frida-скрипті. Створений таким чином файл із розширенням .js може бути виконаний через команду: frida -l {script-bypass.js} -f {identifier name} -U.

Після цього застосунок знову запускається без попереджувального повідомлення про рутований пристрій, що підтверджує успішне обходження механізму.

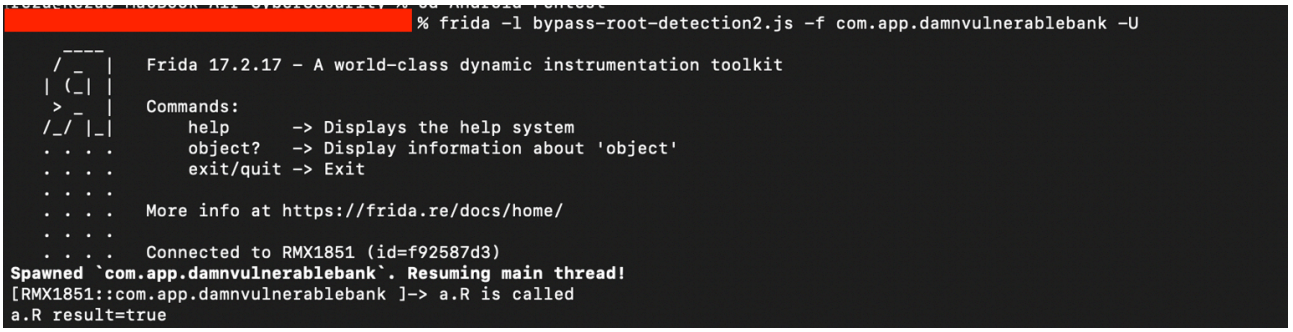

Рисунок 2.6 Використання утиліти

Результати тестування засвідчили, що хоча DVB і містить базовий механізм виявлення небезпечного середовища, він не забезпечує достатньої стійкості до динамічних атак. Механізм легко обходиться як засобами модифікації середовища, так і втручанням у логіку виконання програми. Для банківського застосунку подібна вразливість має суттєві наслідки, оскільки

відкриває можливість подальших атак, таких як підміна токенів, перехоплення даних або маніпулювання транзакційною логікою.

Крім перевірки наявності root-доступу, застосунок Damn Vulnerable Bank також реалізує механізм *anti-debugging*, призначений для виявлення спроб динамічного втручання у процес виконання. Подібні перевірки часто застосовуються у банківських застосунках з метою ускладнення аналізу їхньої логіки, перехоплення внутрішніх викликів або модифікації поведінки через інструменти на кшталт Frida. Під час первинного динамічного аналізу було відзначено, що навіть після успішного обходу механізму root detection за допомогою Frida програма припиняє роботу, виводячи повідомлення, яке свідчить про виявлення активного дебагера. Це означає, що перевірка anti-debugging виконується незалежно й блокує подальший аналіз.

Одним зі способів обходу цієї перевірки є використання Magisk. На відміну від Frida, Magisk працює на рівні модифікації середовища, що дозволяє приховати як факт рутування, так і ознаки втручання в роботу процесу без потреби у запуску додаткових інструментів під час виконання застосунку. У таких умовах програма не отримує індикаторів, які вона очікує для визначення наявності відладчика або ін’єкції сторонніх бібліотек, і продовжує роботу у звичайному режимі.

Поряд із використанням Magisk було досліджено можливість обходу anti-debugging перевірок через Frida за допомогою створення власного динамічного скрипту, аналогічного тому, що застосовувався для обходу root detection. Аналіз декомпільованого коду за допомогою JADX дозволив визначити методи, відповідальні за виявлення підключення дебагера та перевірку сигнатур Frida. Механізм перевірки виявився порівняно простим: застосунок звертається до системних API, які повертають інформацію про наявність активного інструмента налагодження. Відповідно, створений Frida-скрипт модифікує ці функції у момент виконання, підміняючи їх результати на значення, що сигналізують про безпечний стан середовища.

Після поєднання логіки обходу root detection та антидебагу в одному скрипті з’явилася можливість одночасно нейтралізувати обидва механізми захисту. Збережений у вигляді .js-файла скрипт був завантажений у процес застосунку за допомогою команди: frida -l {script-bypass-root-anti-debugging-detection.js} -f {identifier name} -U.

Після ін’єкції програма успішно запустилася, не відображаючи жодних попереджень про рутований пристрій чи присутність Frida, що підтвердило ефективність виконаних маніпуляцій.

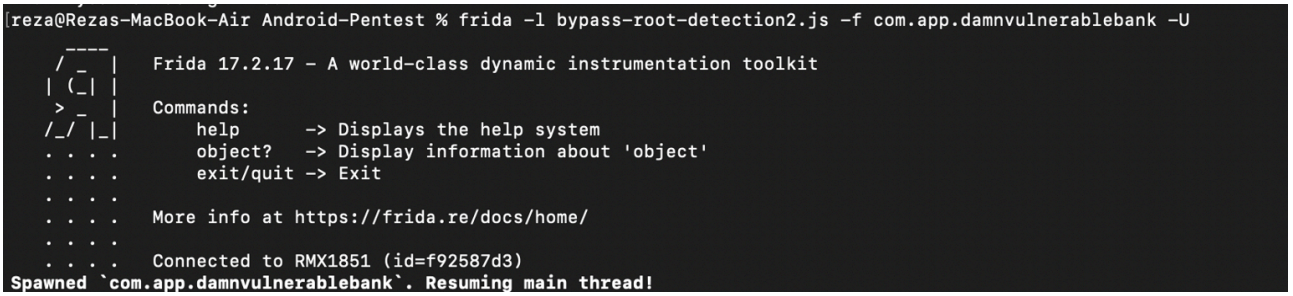

Рисунок 2.7 Використання утиліти

Отримані результати свідчать про те, що механізм anti-debugging у DVB є недостатньо стійким до атак, які використовують динамічну модифікацію поведінки програми. Для банківських застосунків це становить суттєвий ризик, оскільки у разі обходу контролю середовища зловмисник отримує можливість перехоплювати внутрішні виклики, змінювати критичні параметри транзакцій та здійснювати подальші атаки на компоненти безпеки. Подібні недоліки підкреслюють важливість багаторівневого захисту та необхідність застосування перевірених механізмів протидії інструментам динамічного аналізу.

Під час динамічного аналізу було встановлено, що мобільний застосунок шифрує дані як у вихідних запитах, так і у відповідях, отриманих від серверної частини. Усі перехоплені HTTP-пакети у Burp Suite відображалися у вигляді зашифрованих рядків, що ускладнювало безпосередню інтерпретацію змісту трафіку. Щоб проаналізувати фактичний зміст переданих даних, необхідно було визначити механізм шифрування, який використовується клієнтом.

Декомпіляція застосунку показала, що операції шифрування та дешифрування реалізовані у вигляді двох окремих функцій e.b() (шифрування) та e.a() (дешифрування), розміщених у пакеті c.b.a. Подальший розбір коду дозволив встановити, що реалізований механізм є симетричним і використовує просту операцію XOR у поєднанні з кодуванням Base64. Такий підхід не відповідає жодним криптографічним стандартам і вважається слабким та передбачуваним, оскільки XOR із фіксованим ключем легко відтворити та обійти.

Рисунок 2.8 Дані передаються у закодованому вигляді

Для отримання відкритого тексту було створено Frida-скрипт, який перехоплює виклики функції e.a() у момент виконання програми. Скрипт дає змогу виводити результат дешифрування до консолі ще до того, як дані будуть знову зашифровані або передані далі. Після запуску Frida із цим скриптом мережеві запити та відповіді стали доступними у відкритому вигляді під час роботи застосунку. Це підтвердило, що дані, які виглядали зашифрованими у трафіку, можна повністю відтворити без взаємодії із сервером та без знання його внутрішніх механізмів.

Додатковий аналіз логіки шифрування підтвердив, що спочатку виконується операція XOR із використанням статичного секретного ключа "amazing", після чого результат кодується у Base64. Дешифрування, відповідно, здійснюється у зворотному порядку: декодування Base64 і повторна операція XOR із тим самим ключем. Така схема є криптографічно нестійкою і може бути

відтворена будь-ким, хто має доступ до APK-файлу або результатів динамічного аналізу.

Рисунок 2.9 Розкодування даних

Отримані результати демонструють, що хоч застосунок і формально застосовує шифрування, використаний механізм не забезпечує захисту чутливих даних. Будь-який зловмисник, який має можливість запускати застосунок у контрольованому середовищі або перехоплювати трафік, може повністю відновити зміст усіх запитів і відповідей. Це створює критичні ризики для мобільного банківського застосунку, де конфіденційність даних та їхня цілісність є ключовими вимогами [23].

Під час динамічного аналізу було виявлено порушення в роботі механізму керування сесією, яке полягає в тому, що сесійний токен залишається дійсним навіть після явного завершення роботи користувача через функцію logout. У мобільних застосунках, особливо фінансових, коректне завершення сесії є критично важливим елементом безпеки, оскільки токен автентифікації є еквівалентом активного доступу до аккаунта користувача. Якщо такий токен продовжує діяти після виходу із системи, це створює можливість несанкціонованого використання його сторонніми особами.

Для перевірки цього припущення було виконано експеримент. Спершу здійснювався вхід у застосунок, після чого один із доступних API-запитів надсилався через Burp Suite до модуля Repeater, що дозволяє повторно відправляти збережені запити без участі клієнта. Після фіксації запиту користувач у застосунку здійснював вихід із системи за допомогою відповідної кнопки logout, що за правильної реалізації повинно призвести до негайної інвалізації сесійного токена на сервері.

Однак повторне надсилання збереженого запиту через Burp Repeater показало, що сервер продовжує приймати цей токен як чинний та повертає

коректну відповідь. Це свідчить про те, що серверна частина не здійснює інвалізації або прив’язки токена до поточного стану сесії. Іншими словами, токен, який мав би втратити чинність після завершення сесії, залишився повністю функціональним.

Така поведінка становить критичну вразливість для банківського застосунку. Якщо зловмисник отримає доступ до токена, наприклад через перехоплення трафіку, витік логів або будь-яку іншу вразливість, він зможе продовжувати виконувати операції від імені користувача навіть після того, як реальний власник облікового запису завершив роботу. Це повністю нівелює механізм logout і суперечить вимогам MASVS-AUTH та принципам захищеної авторизації, що передбачають негайне припинення дії всіх активних токенів.

Отримані результати демонструють системну проблему в реалізації логіки обробки сесій на серверній стороні. Відсутність інвалізації токена після logout створює суттєві ризики для цілісності та конфіденційності фінансових даних користувачів, а також відкриває шлях до несанкціонованого доступу і потенційного зловживання банківськими функціями.

Динамічний аналіз дозволив виявити вразливості, які не проявляються під час статичного дослідження та стають очевидними лише в процесі реального виконання застосунку. Завдяки використанню інструментів перехоплення та модифікації трафіку вдалося проаналізувати поведінку клієнтської частини у взаємодії з сервером і дослідити, як саме формуються запити, які параметри передаються та яким чином сервер реагує на змінені або неочікувані значення. Перехоплення HTTP-комунікації через Burp Suite дало змогу відтворювати окремі операції повторно, оцінювати вплив їхніх параметрів та перевіряти, чи застосовує сервер належний контроль доступу та валідацію запитів.

Використання інструментів динамічної ін’єкції коду, зокрема Frida, дозволило втручатися у внутрішню логіку застосунку під час його роботи. Це забезпечило можливість перехоплювати та модифікувати виклики функцій, змінювати параметри перед їх відправленням на сервер, обходити захисні

механізми на кшталт root detection та anti-debugging, а також отримувати доступ до проміжних значень, які не передаються через мережу у відкритому вигляді. У такий спосіб стало можливим дослідити механізми шифрування, обробку токенів і використання ключових структур даних, що у звичайному режимі виконання залишаються недоступними.

Модифікація параметрів безпосередньо під час виконання дала змогу встановити, чи існує у застосунку належна перевірка коректності даних як на клієнтській, так і на серверній стороні. Зокрема, зміна автентифікаційних токенів, підміна параметрів транзакцій або повторне надсилання попередніх запитів допомогли виявити недоліки у механізмах керування сесіями, автентифікації та бізнес-логіці API. У багатьох випадках застосунок або сервер приймали некоректні та модифіковані параметри, що свідчило про відсутність критично важливих перевірок або про неправильну реалізацію обмежень доступу.

Таким чином, динамічний аналіз дав змогу комплексно оцінити поведінку застосунку в реальному середовищі та підтвердити наявність вразливостей, які виникають не лише через помилки у коді, але й внаслідок недостатнього контролю над потоками даних та відсутності належних серверних перевірок. Саме поєднання перехоплення трафіку, ін’єкції коду та модифікації параметрів забезпечило повне розуміння того, як застосунок реагує на маніпуляції та де розташовані критичні точки, що можуть бути використані для атаки.

2.2.3 Аналіз API та дослідження логіки серверної взаємодії

Аналіз API є одним із ключових етапів оцінювання безпеки мобільного застосунку, оскільки саме через серверну взаємодію реалізуються більшість критичних операцій, пов’язаних із автентифікацією, авторизацією, фінансовими транзакціями та обробкою конфіденційних даних. У контексті банківського застосунку цей компонент має вирішальне значення: будь-яка помилка на рівні

API може призвести до компрометації облікових записів, підміни транзакцій або обходу механізмів контролю доступу.

У межах цього підрозділу буде застосовано поєднання методологій динамічного тестування та аналізу поведінки HTTP-запитів за допомогою інструментів Burp Suite, Frida та adb. Використання Burp Suite у режимі перехоплення та повторної відправки запитів дозволяє оцінити стабільність і коректність механізмів автентифікації, поведінку токенів, а також перевірити наявність контролю доступу до критичних ендпойнтів. Frida забезпечує можливість втручання в логіку клієнта, зокрема модифікацію параметрів запитів перед їх відправленням, що є необхідним для ідентифікації вразливостей бізнес-логіки. adb використовується для збирання діагностичних даних, роботи з внутрішніми файлами застосунку та підтвердження фактичної поведінки компонентів [24].

Методологічно аналіз API передбачає: оцінку автентифікаційних механізмів, перевірку послідовності та коректності обробки транзакцій, дослідження поведінки серверних ендпойнтів у відповідь на маніпулятивні або некоректні параметри, а також моделювання сценаріїв, які типово використовують зловмисники для обходу контролю доступу. Такий комплексний підхід дозволяє встановити, наскільки серверна логіка відповідає вимогам безпечної обробки даних і чи достатньо надійно реалізовані обмеження на виконання банківських операцій.

У процесі динамічного аналізу з використанням інструмента Drozer було встановлено, що частина активностей мобільного застосунку може бути запущена без попередньої автентифікації. Виявлено, що кілька компонентів не мають належно налаштованих дозволів, а значення параметра android:exported у їхній конфігурації встановлено в true. Це означає, що відповідні activity можуть бути викликані будь-яким іншим застосунком, встановленим на пристрої, незалежно від наявності або відсутності дійсної сесії.

Такий недолік є суттєвою вразливістю у контексті банківських мобільних застосунків, оскільки exported activity фактично перетворюється на неконтрольовану точку входу. Якщо доступ до критичних функцій, як-от перегляд рахунку чи ініціювання транзакції, не обмежується автентифікацією, зловмисник може скористатися цим для запуску внутрішніх екранів або бізнес-функцій, оминаючи логіку входу.

Для підтвердження цього припущення було виконано запуск низки activity за допомогою Drozer. Інструмент дає змогу безпосередньо звернутися до будь-якої заявленої компоненти застосунку. Команда: run app.activity.start --component {identifier name} {identifier name.activity} дозволила запустити відповідну activity без проходження процедури автентифікації. Аналогічний результат було отримано під час використання інструмента ADB, який також підтримує запуск компонентів напряму: adb shell am start -n {identifier name}/{identifier name.activity}.

У обох випадках застосунок відкривав внутрішню сторінку зокрема сторінку *Send Money* без будь-яких запитів на автентифікацію чи перевірки сесії. Це свідчить про те, що доступ до критичної функціональності жодним чином не контролюється, а механізм автентифікації фактично можна обійти через прямий виклик exported activity.

Ця вразливість має особливо високий рівень ризику, оскільки дозволяє зловмиснику не тільки переглядати внутрішні інтерфейси застосунку, але й потенційно ініціювати небажані операції, якщо activity взаємодіє з API без додаткових перевірок. Банківські застосунки покладаються на сувору взаємодію між автентифікацією, авторизацією та бізнес-логікою, тому наявність відкритих activity свідчить про фундаментальний недолік у реалізації моделі безпеки.

Виявлена проблема демонструє важливість коректного налаштування компонентів Android та перевірки того, що всі критичні activity потребують автентифікації й не доступні для виклику ззовні. В іншому випадку навіть

найсильніший механізм входу не забезпечує реального захисту, якщо логіка застосунку може бути обійдена через exported компоненти.

У процесі аналізу API реєстрації було встановлено, що застосунок демонструє поведінку, характерну для вразливості *user enumeration*. Цей тип вразливості виникає тоді, коли система повертає різні повідомлення залежно від того, чи існує вже обліковий запис із вказаним ім’ям користувача. Якщо відповіді сервера можна розрізнити, зловмисник здатний визначити, які імена користувачів є дійсними, навіть без спроби автентифікації.

Тестування показало, що під час спроби зареєструвати користувача із вже існуючим ім’ям сервер повертає однозначне повідомлення *"Username already taken"*. Таким чином, застосунок розкриває наявність певного облікового запису у системі. Якщо ж ім’я користувача не існує, сервер повертає інший тип відповіді, що дозволяє легко відрізнити дійсні облікові записи від неіснуючих. Подібна поведінка створює передумови для масового перебирання імен користувачів та подальшого зловживання отриманою інформацією.

У безпековому контексті мобільних банківських застосунків така вразливість є небезпечною з кількох причин. По-перше, вона дозволяє зловмиснику сформувати список дійсних користувачів, що значно спрощує подальші атаки, зокрема brute-force на сторінці входу або цілеспрямовані фішингові кампанії. По-друге, наявність можливості визначити існування конкретного облікового запису може супроводжуватися ризиком витоку непрямої персональної інформації, оскільки ім’я користувача нерідко корелює із даними про реальну особу.

Для запобігання цьому типу вразливості необхідно забезпечити уніфіковану та нейтральну відповідь сервера на всі помилкові запити під час реєстрації незалежно від того, чи введений користувач існує в системі. Типовими варіантами є повідомлення *"Registration failed"* або *"An error occurred. Please try again later"*, які не містять жодної інформації про валідність імені користувача. Подібний підхід унеможливлює аналіз поведінки API з боку

зловмисників і відповідає принципам безпечної обробки автентифікаційних даних, визначених у MASVS-AUTH.

Таким чином, виявлена вразливість свідчить про відсутність належного контролю над інформаційними потоками під час реєстрації та створює умови для ідентифікації існуючих користувачів системи. Її усунення є необхідним для підтримання конфіденційності даних та запобігання подальшим атакам на механізми автентифікації.

У ході аналізу механізму автентифікації було встановлено, що сторінка входу мобільного застосунку не реалізує жодного обмеження на кількість спроб введення неправильних облікових даних. Це створює типову вразливість *no rate limit*, яка дозволяє зловмиснику необмежено багато разів надсилати запити автентифікації, змінюючи параметри логіна та пароля до моменту, поки не буде знайдено дійсну комбінацію. У банківських застосунках відсутність захисту такого типу є критичним недоліком, оскільки уможливлює прямі brute-force атаки та може призвести до компрометації облікового запису.

Для підтвердження наявності вразливості було відтворено реалістичний сценарій атаки. Спочатку було перехоплено вихідний запит до ендпойнта входу, після чого його передано до Burp Suite Repeater. Оскільки застосунок використовує XOR-шифрування з подальшим кодуванням Base64, отримані попередньо дані в «чистому» вигляді (plaintext) за допомогою Frida дозволили відтворити структуру JSON-параметрів, що передаються у запиті. Ці дані були повторно сформовані в Repeater та підготовлені до автоматизованого тестування.

Для моделювання brute-force атаки застосовано модуль Burp Intruder. За допомогою Hackvertor було налаштовано автоматичне застосування того самого алгоритму шифрування, що й у клієнтському застосунку, а саме виконання XOR із фіксованим ключем "amazing" та кодування в Base64. У результаті Intruder отримав можливість надсилати до серверу повністю коректні, хоч і автоматично згенеровані зашифровані запити на автентифікацію.

Під час тестування виявлено, що сервер не застосовує жодних обмежень на кількість запитів із неправильною комбінацією логіна та пароля. Було здійснено понад сотню запитів без будь-яких затримок, блокувань або реакцій захисту. На 101-му запиті Intruder успішно підібрав валідні облікові дані та отримав доступ до інтерфейсу застосунку. Це однозначно підтверджує відсутність rate limiting і демонструє, що система не відстежує частоту спроб входу ані за обліковим записом, ані за IP-адресою, ані за сеансовими параметрами.

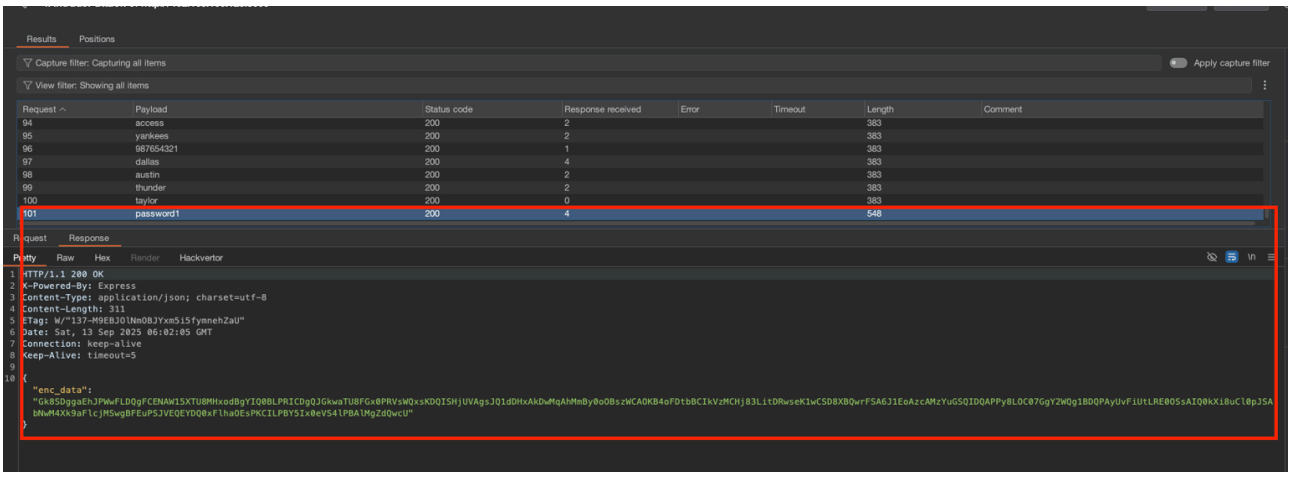

Рисунок 2.10 Автоматизування атаки

Подібна поведінка свідчить про суттєвий недолік у реалізації механізмів автентифікації. За відсутності контролю інтенсивності запитів зловмисник може виконувати автоматизовані атаки, що не лише загрожують компрометацією користувацьких облікових даних, але й створюють ризик перевантаження серверної інфраструктури, що може призвести до часткової або повної недоступності сервісу. У випадку фінансових застосунків такий недолік суперечить вимогам MASVS-AUTH та загальноприйнятим принципам захисту від brute-force атак [25].

У процесі аналізу серверної взаємодії було виявлено суттєву вразливість у бізнес-логіці API, відповідального за виконання фінансових переказів. Замість того щоб забезпечувати повну перевірку параметрів транзакції, серверна частина застосунку покладається на значення, які надсилає клієнт, не

здійснюючи жодної валідації допустимості переданих сум. Унаслідок цього зловмисник може сформувати запит, який порушує базові правила фінансового обліку, зокрема передати від'ємну суму, що призводить до зменшення балансу рахунку жертви.

Під час експерименту було обрано два облікові записи один у ролі зловмисника, інший як потенційна жертва. Спершу було зафіксовано поточний баланс жертви для подальшого порівняння. Далі з облікового запису зловмисника було ініційовано стандартний запит на переказ коштів, який було перехоплено в Burp Suite. Після перехоплення стало очевидно, що дані запиту перебувають у зашифрованому вигляді, тому для коректної модифікації було застосовано ту саму методику розшифрування та повторного шифрування, яку використовували на попередніх етапах аналізу (XOR → Base64).

Використавши Frida та Hackvertor, вдалося отримати оригінальну структуру JSON-параметрів запиту. Після відтворення їх у відкритому вигляді параметр amount був змінений на від'ємне значення, наприклад -2000. Наступним кроком стала повторна генерація коректного зашифрованого формату запиту та його надсилання до серверної частини через Burp Repeater.

Результат продемонстрував повну відсутність перевірок на стороні сервера: транзакція була прийнята, додана до історії операцій та оброблена як легітимна. У загальному балансі облікового запису жертви з'явилося незаконне зменшення коштів, що підтверджує критичний дефект бізнес-логіки. Серверська частина не здійснила перевірку наявності достатньої суми, коректності математичної операції та відповідності правилам проведення фінансових транзакцій.

Ця вразливість становить особливу небезпеку, оскільки не є технічною помилкою кодування чи криптографії вона виникає на рівні логіки продукту. У реальному банківському середовищі подібна відсутність валідації могла б призвести до прямої фінансової шкоди, підміни транзакцій, обходу лімітів,

створення “віртуальних” коштів або викривлення балансових показників системи.

Виявлена проблема демонструє фундаментальну необхідність впровадження на серверному рівні повних та незалежних від клієнта перевірок параметрів транзакцій. Будь-які значення, що надходять від клієнтської частини, повинні розглядатися як недовірені, а всі обчислення та обмеження мають бути реалізовані виключно на сервері. В іншому разі система стає вразливою до широкого спектра атак, що експлуатують логічні помилки в бізнес-процесах.

У межах аналізу серверної взаємодії було встановлено, що механізм автентифікації, який має застосовуватися під час виконання фінансових транзакцій, реалізований некоректно. За задумом, перед здійсненням операції переказу користувач повинен пройти додаткову перевірку автентифікацію за допомогою PIN-коду або біометрії (fingerprint). Такий підхід відповідає вимогам підвищеного рівня безпеки та є типовим для банківських застосунків, де підтвердження транзакцій компонентами SCA (Strong Customer Authentication) є обов’язковим.

Однак у процесі тестування було виявлено, що механізм підтвердження транзакції реалізований лише на рівні клієнтського інтерфейсу та не підкріплений відповідними перевірками на сервері. Під час роботи застосунку виконання операції дійсно блокується до моменту проходження PIN або fingerprint-перевірки, однак це обмеження не поширюється на серверну частину API. Безпосереднє надсилання запиту на переказ коштів до ендпойнта /transfer дозволяє виконати операцію, навіть якщо користувач не проходив жодної автентифікації під час поточної сесії.

Експериментально це було підтверджено шляхом перехоплення легітимного запиту на переказ у Burp Suite та подальшого його повторного надсилання без виконання PIN або fingerprint-перевірки. Незважаючи на те, що клієнтська частина застосунку вимагала підтвердження транзакції, сервер приймав і обробляв модифіковані запити, виконуючи перекази без жодного

додаткового контролю. Більше того, така операція могла бути повторена неодноразово, оскільки логіка автентифікації не була прив’язана до виконання конкретної транзакції і не контролювалася сервером.

У результаті було доведено, що механізм автентифікації є формальним і не забезпечує реального захисту в контексті фінансових операцій. Основна проблема полягає у відсутності перевірки прав доступу на стороні сервера: API не визначає, чи має користувач право ініціювати транзакцію, не перевіряє, чи виконано додаткову автентифікацію, та не застосовує жодних обмежень, характерних для Strong Customer Authentication. Це дає змогу зловмиснику за умови наявності токена автентифікації виконувати фінансові операції без відома та участі реального користувача.

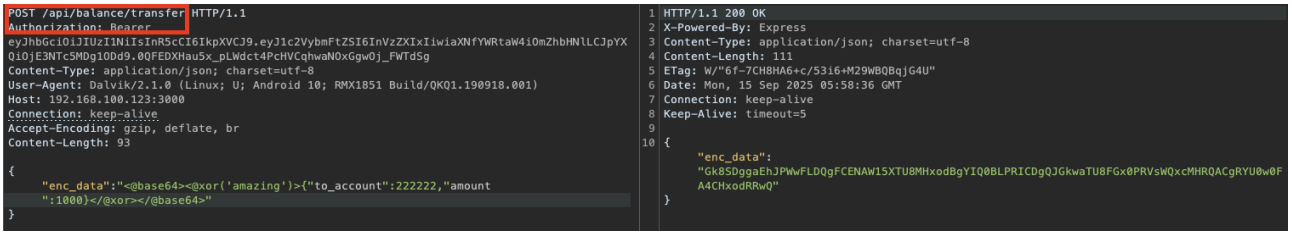

Рисунок 2.11 Експлуатація вразливості

Наявність такої вразливості є критичною для банківського застосунку, оскільки вона фактично дозволяє повністю обійти транзакційний рівень автентифікації та робить систему вразливою до несанкціонованих переказів, підміни операцій та фінансового шахрайства. Вона також свідчить про порушення вимог PSD2, MASVS-AUTH та інших стандартів, що регламентують порядок підтвердження транзакцій у фінансових системах.

Проведений аналіз API та серверної логіки мобільного застосунку засвідчив, що критичні елементи захисту, пов’язані з автентифікацією, авторизацією та обробкою фінансових операцій, реалізовані неповно або відсутні повністю. Виявлені вразливості охоплюють різні рівні взаємодії від початкової реєстрації користувача до виконання транзакцій і демонструють системні порушення принципів побудови безпечних фінансових сервісів.

Одним із ключових недоліків є відсутність належних механізмів захисту при взаємодії з API, що стало очевидним під час перевірки поведінки системи щодо частоти запитів, валідності автентифікаційних токенів та контролю доступу до критичних ендпойнтів. Система не застосовує обмежень на кількість спроб входу, дозволяючи brute-force атаки; не уніфікує помилки під час реєстрації, розкриваючи інформацію про існуючі облікові записи; а також приймає запити на фінансові операції без перевірки того, чи пройшов користувач транзакційну автентифікацію. Достатньо лише наявності чинного токена без жодних додаткових підтверджень щоб виконати переказ коштів.

Подальший аналіз виявив, що серверна частина не виконує фундаментальних бізнес-перевірок, пов’язаних із транзакціями, зокрема не контролює допустимість введених сум та не гарантує цілісність параметрів, що надходять від клієнта. Це дозволило здійснити операцію із від’ємним значенням суми, що призвело до незаконного зменшення балансу на рахунку жертви. Така поведінка свідчить про серйозний дефект бізнес-логіки, який не залежить від технологічних аспектів, але прямо впливає на фінансову безпеку користувачів.

Додатково було встановлено, що частина activity застосунку є доступною для виклику ззовні через параметр android:exported=true, що дозволило обійти механізм автентифікації на клієнтському рівні. Ця вразливість підкреслює відсутність належної узгодженості між клієнтськими та серверними перевірками, оскільки API приймає запити незалежно від того, яким шляхом користувач потрапив до відповідного функціоналу.

Узагальнюючи результати дослідження, можна стверджувати, що API застосунку не забезпечує жодного з базових принципів серверної безпеки: не перевіряє достовірність автентифікації, не контролює авторизацію, не валідує дані транзакцій та не захищає бізнес-процеси від маніпуляцій. Виявлені проблеми є критичними у контексті банківського середовища, оскільки відсутність цих механізмів може призвести до компрометації рахунків, підміни транзакцій або прямої фінансової шкоди. Така ситуація демонструє необхідність

повного перегляду архітектури серверної логіки, обов’язкової реалізації перевірок SCA, а також впровадження стандартів MASVS та відповідних вимог PSD2 до захисту клієнтських і серверних компонентів.

Дослідження взаємодії клієнтської частини застосунку з API стало ключовим етапом у виявленні вразливостей, пов’язаних із механізмами автентифікації, авторизації та логікою обробки фінансових операцій. Аналіз HTTP-запитів за допомогою Burp Suite, у поєднанні з динамічним втручанням у виконання застосунку через Frida, дозволив відстежити, як саме клієнт формує параметри запитів, як сервер реагує на змінені або некоректні значення, та чи виконується належна перевірка прав доступу під час критичних операцій.

Отримані результати показали, що серверна частина застосунку покладається на дані, отримані від клієнта, не здійснюючи необхідних перевірок. Це проявляється у можливості обходу автентифікації під час виконання транзакцій, коли API приймає запити на переказ коштів навіть без проходження PIN-або біометричної перевірки. Крім того, зміна параметрів запиту зокрема суми транзакції не призводить до відмови з боку сервера, що вказує на відсутність контролю допустимості значень. Виявлена можливість передати від’ємну суму і спровокувати незаконне зменшення балансу на рахунку жертви підтвердила критичні порушення бізнес-логіки та відсутність внутрішніх обмежень на рівні API.

Не менш показовою стала поведінка системи під час тестування стійкості до автоматизованих атак. Сервер не застосовував жодних обмежень щодо кількості спроб автентифікації, що дозволило виконати масовий перебір паролів і довести відсутність rate limiting, критично важливого для захисту банківських сервісів. Водночас аналіз механізму реєстрації виявив, що API повертає різні повідомлення для існуючих і неіснуючих імен користувачів, що дає змогу здійснювати user enumeration і збирати валідні облікові записи для подальших атак.

Перехоплення та модифікація параметрів запитів також дозволили встановити, що окремі activity клієнтського застосунку доступні для запуску ззовні, а сервер не контролює відповідність HTTP-запиту активному стану автентифікованої сесії. Це створює можливість доступу до внутрішніх функцій застосунку без автентифікації, якщо зловмисник викличе відповідний компонент напряму через ADB або Drozer. Така поведінка свідчить про загальну несумісність клієнтських та серверних механізмів контролю доступу, а також про відсутність цілісної політики авторизації.

Узагальнюючи отримані результати, можна стверджувати, що використання поєднання інструментів Burp Suite для аналізу трафіку, Frida для динамічної модифікації виконання, adb та Drozer для взаємодії з компонентами середовища дозволило всебічно оцінити поведінку застосунку як на клієнтському, так і на серверному рівні. Кожен із методів виявляв окремий клас вразливостей: статичний аналіз показав структурні недоліки та помилки конфігурації; динамічний аналіз виявив слабкість механізмів автентифікації та контролю середовища виконання; тестування API продемонструвало порушення бізнес-логіки та відсутність належних перевірок на стороні сервера.

Саме комплексний підхід, що поєднує різні інструменти й методології, дав змогу сформувати повну картину ризиків та визначити критичні точки у взаємодії між клієнтом і сервером. Це дозволяє адекватно оцінити стійкість застосунку до сучасних загроз і створює підґрунтя для наступного етапу роботи інтерпретації результатів та визначення пріоритетів щодо підвищення рівня безпеки мобільного банківського застосунку.

2.3. Оцінка захищеності мобільного застосунку на основі отриманих результатів

Отримані під час статичного, динамічного та API-аналізу результати дозволяють перейти до формальної оцінки рівня захищеності мобільного застосунку. На цьому етапі окремі виявлені вразливості розглядаються не

ізольовано, а через призму їхнього реального впливу на конфіденційність, цілісність і доступність даних. Для цього застосовується систематизація знайдених недоліків, визначення їхнього імпакту та обчислення показників критичності за методикою CVSS.

Паралельно проводиться оцінювання відповідності застосунку вимогам міжнародних стандартів безпеки, включно з MASVS, OWASP Mobile Top 10 та галузевими нормами на кшталт PCI DSS і PSD2. Це дозволяє визначити, наскільки реалізовані механізми відповідають базовим практикам захисту мобільних фінансових сервісів. Сукупність цих методів формує підґрунтя для об’єктивного висновку про поточний стан безпеки застосунку та визначає напрямки його подальшого вдосконалення.

Результати тестування показали, що вразливості охоплюють усі ключові рівні мобільного банківського застосунку від локального зберігання даних до серверної бізнес-логіки. Для подальшої оцінки їхнього впливу вразливості були згруповані за характером ризиків, які вони створюють.

Першу групу становлять вразливості, пов’язані з компрометацією конфіденційних даних. До неї належать Insecure Data Storage та Insecure Logging on Login, що дають змогу отримати доступ до токенів автентифікації, збережених у відкритому вигляді, або перехопити їх через журналювання. Ці недоліки безпосередньо впливають на захист облікових записів і створюють передумови для несанкціонованого доступу.

Другу групу формують вразливості платформного рівня, які дозволяють обійти механізми контролю середовища виконання. Bypass Root Detection та Bypass Anti-Debugging Check демонструють, що застосунок не здатний надійно відрізнити безпечне середовище від скомпрометованого. Це дає змогу зловмиснику вільно модифікувати поведінку програми, перехоплювати функції та маніпулювати даними.

До третьої групи належать вразливості, що стосуються мережевої взаємодії та обробки даних, зокрема Decrypt Request & Response. Вони свідчать

про використання нестійких криптографічних механізмів та можливість повного відновлення даних, які передаються між клієнтом і сервером.

До четвертої групи входять проблеми керування сесією та автентифікацією, які мають безпосередній вплив на цілісність облікового доступу. Session Not Expired After Logout, No Rate Limit on Login, User Enumeration on Register та Bypass Login свідчать про відсутність системних механізмів контролю сесій, захисту від brute-force атак та уніфікації помилок під час реєстрації. У сукупності вони значно знижують стійкість застосунку до атак на автентифікаційний механізм.

Окрему категорію складають вразливості бізнес-логіки та авторизації, які мають найбільш критичний вплив у контексті фінансових операцій. Business Logic Vulnerability on Transfer та Broken Authentication on Transfer демонструють відсутність перевірки допустимості параметрів транзакції та неефективність механізму підтвердження переказів. Їхня експлуатація дозволяє виконувати операції з від’ємними сумами або ініціювати фінансові транзакції без проходження PIN чи біометричної автентифікації. Ці недоліки є найбільш небезпечними, оскільки відкривають можливість прямої фінансової шкоди для користувачів.

Зрештою, вразливість WebView via Deeplink формує групу інтерфейсних та платформних загроз, що дозволяють виконувати JavaScript у контексті застосунку та маніпулювати контентом, який бачить користувач. Це створює ризики фішингу, підміни інтерфейсу та атак соціальної інженерії.

Узагальнена класифікація демонструє, що вразливості охоплюють як низькорівневі механізми зберігання та журналювання, так і критичні операції, що стосуються фінансових транзакцій та авторизації. Така ширина спектра ризиків свідчить про низький загальний рівень захищеності мобільного застосунку та потребує комплексного перегляду його архітектури безпеки, насамперед на рівні серверної логіки й управління доступом.

2.3.1 Оцінювання вразливостей за CVSS v3.1

Для кількісного вимірювання ризиків, які створюють виявлені вразливості, було застосовано методологію CVSS v3.1. Ця система дозволяє формалізувати критичність недоліків на основі їхнього впливу на конфіденційність, цілісність і доступність, а також на основі складності експлуатації. Аналіз показав, що найвищі базові оцінки отримали вразливості, пов’язані з обходом автентифікації та відсутністю контролю кількості спроб входу, оскільки вони безпосередньо загрожують доступу до облікових записів і порушують фундаментальні механізми контролю доступу. Значну небезпеку також становлять недоліки керування сесіями та шифруванням трафіку, які створюють можливості для перехоплення або підміни даних. Вразливості середнього рівня, включно з проблемами WebView, локального зберігання та журналювання, формують додаткові вектори атаки та підсилюють критичні недоліки. Сукупність отриманих балів CVSS відображає низький загальний рівень безпеки застосунку й вказує на необхідність перегляду його архітектури захисту [26].

Common Vulnerability Scoring System (CVSS) версії 3.1 є загальновизнаним стандартом для визначення рівня небезпеки програмних вразливостей. Його мета надати уніфіковану, порівнювану та об’єктивну оцінку ризику, яка може бути використана як розробниками, так і аналітиками безпеки для визначення пріоритетів реагування. CVSS оцінює не лише технічну складність експлуатації вразливості, але й потенційний вплив на конфіденційність, цілісність і доступність системи.

Система складається з трьох груп метрик. Базові метрики (Base Metrics) описують фундаментальні характеристики вразливості, які не змінюються залежно від контексту. До них належать Attack Vector, Attack Complexity, Privileges Required, User Interaction та показники впливу на Confidentiality, Integrity й Availability. Саме ці метрики визначають ключовий рівень небезпеки.

Тимчасові метрики (Temporal Metrics) уточнюють оцінку з огляду на доступність експлойтів, наявність способів пом’якшення наслідків та ступінь підтвердженості інформації про вразливість. Ці метрики дозволяють адаптувати оцінку до актуального стану загроз.

Контекстні або екологічні метрики (Environmental Metrics) враховують специфіку середовища, у якому експлуатується вразливість: критичність активів, можливі наслідки для бізнесу, залежність від певних конфігурацій або обмежень. На цьому рівні оцінка може суттєво змінюватися залежно від того, чи йдеться про мобільний банківський застосунок, корпоративну інфраструктуру чи ізольовану лабораторну систему.

Використання CVSSv3.1 дозволяє перетворити якісні характеристики вразливостей на кількісні показники, що дає змогу порівнювати їх між собою та формувати пріоритети усунення. Для мобільного банківського застосунку це особливо важливо, оскільки різні вразливості впливають на різні компоненти системи від локального зберігання даних до бізнес-логіки фінансових операцій. CVSS допомагає визначити, які з них становлять негайну загрозу та мають найвищий пріоритет для усунення.

Для формування об’єктивної картини рівня ризику, який становлять виявлені вразливості, їх було попередньо класифіковано, після чого здійснено оцінювання за методологією CVSS v3.1. Використання цього стандарту дає змогу перетворити якісні характеристики кожної вразливості на кількісні метрики, що дозволяють порівнювати їх між собою, визначати рівень потенційної шкоди та встановлювати пріоритети щодо усунення.

У контексті мобільного банківського застосунку CVSS є особливо корисним, оскільки дає змогу оцінити не лише технічний аспект експлуатації, але й реальний вплив на конфіденційність даних, цілісність фінансових операцій та доступність критичних сервісів. Таблиця нижче узагальнює ключові вразливості, виявлені під час тестування DVB, із коротким описом їхнього впливу та полем для подальшого зазначення інтегральної CVSS-оцінки.

Таблиця 2.1

Перелік вразливостей та їх попередня характеристика для оцінювання CVSSv3.1

Назва вразливості	**Опис вразливості**	**Оцінка CVSS v3.1**
Insecure Data Storage	Токен доступу зберігається у відкритому вигляді, що дозволяє його витік та несанкціонований доступ до акаунта. Вплив на Confidentiality та Authorization.	4.4 – Medium
Небезпечне зберігання даних	Токен автентифікації потрапляє у системні логи, що створює ризик компрометації облікових даних.	4.4 – Medium
Вразливість WebView через неконтрольовані deeplink-посилання	Можливість виконання JavaScript у WebView через неконтрольовані deeplink-посилання. Вплив на Integrity та UI Manipulation.	5.4 – Medium
Обхід перевірки root-доступу	Дозволяє працювати у небезпечному середовищі, що дає можливість модифікувати логіку застосунку та перехоплювати дані.	4.4 – Medium
Обхід перевірок протидії дебагу	Дозволяє відладку та ін’єкцію коду під час виконання, створюючи умови для повного контролю над застосунком.	0.0 – None
Можливість дешифрування запитів і відповідей	Дозволяє розшифрувати мережевий трафік та переглядати чутливі дані в plaintext.	7.5 – High
Сесія не інвалізується після виходу з облікового запису	Сесійний токен залишається чинним після logout, що дозволяє виконувати дії від імені користувача.	8.1 – High
Обхід механізму автентифікації	Можливість доступу до функціоналу без автентифікації шляхом прямого звернення до відповідних компонентів або API.	9.1 – Critical
Розкриття наявності користувача під час реєстрації	Сервер розкриває інформацію про існуючих користувачів, що полегшує bruteforce та таргетовані атаки.	5.3 – Medium
Відсутність обмеження кількості спроб входу	Відсутність обмеження кількості спроб входу дозволяє ефективний bruteforce атак.	9.1 – Critical

Продовження таблиці 2.1

Назва вразливості	**Опис вразливості**	**Оцінка CVSS v3.1**
Вразливість бізнес-логіки під час виконання переказів	Відсутність перевірок транзакцій дозволяє маніпулювати балансом шляхом введення некоректних сум.	6.5 – Medium
Некоректна автентифікація транзакцій	Сервер приймає транзакції без PIN/біометричної перевірки, що порушує авторизацію операцій.	6.5 – Medium

Оцінювання знайдених вразливостей за CVSS v3.1 показало суттєву різницю у рівнях ризику між окремими категоріями недоліків. Найвищі бали отримали вразливості, які безпосередньо впливають на механізми автентифікації та доступ до критичного функціоналу. Bypass Login (9.1) та No Rate Limit on Login (9.1) становлять критичний ризик, оскільки дають можливість повністю обійти або зламати механізм входу в систему. У випадку банківського мобільного застосунку це означає фактичну втрату контролю над автентифікаційним контуром, що відкриває шлях до підміни транзакцій та компрометації рахунків.

До групи високого ризику належать вразливості, які безпосередньо впливають на цілісність даних сесії та мережеву конфіденційність. Decrypt Request & Response (7.5) та Session Not Expired After Logout (8.1) створюють можливість несанкціонованого доступу до даних і виконання операцій від імені користувача навіть після завершення сесії. Такі недоліки свідчать про відсутність базових механізмів захисту як на рівні криптографічної обробки трафіку, так і на рівні управління життєвим циклом сесії.

Вразливості середнього рівня ризику Insecure Data Storage, Insecure Logging, WebView via Deeplink, User Enumeration, а також дефекти бізнес-логіки та транзакційної авторизації демонструють структурні і архітектурні недоліки, що формують сприятливі умови для більш складних атак. Хоча їхній базовий

CVSS-бал нижчий, саме ці вразливості забезпечують передумови для експлуатації критичних недоліків у майбутньому.

Вразливості Bypass Root Detection та Bypass Anti-Debugging Check формально не мають CVSS-імпакту (0.0), однак вони відіграють важливу роль у реальній моделі загроз. Хоча ці механізми належать до категорії «defense-in-depth», їх обходження суттєво полегшує експлуатацію інших вразливостей і знижує загальний рівень стійкості застосунку до динамічних атак.

У сукупності результати CVSS-оцінювання дозволяють сформувати узагальнену картину безпеки мобільного застосунку DVB. Найбільш ураженими виявилися компоненти, пов’язані з автентифікацією, керуванням сесіями та серверною бізнес-логікою. Критичні недоліки в цих областях переважують інші типи загроз і свідчать про системну відсутність механізмів контролю доступу на серверному рівні. Вразливості середнього рівня зокрема пов’язані з WebView, локальним зберіганням даних та журналюванням формують додаткові шляхи ескалації атаки, а відсутність платформних перевірок робить застосунок чутливим до аналізу та модифікації.

З позиції CVSS загальний рівень захищеності DVB можна охарактеризувати як низький, оскільки наявні в ньому вразливості не лише численні, але й належать до різних критичних областей автентифікації, авторизації, сесій, криптографії та бізнес-логіки. Це створює ефект кумулятивного ризику, коли експлуатація однієї вразливості значно підсилює наслідки інших, що є типовим для комплексних атак на фінансові мобільні застосунки.

2.3.2 Відповідність вимогам MASVS

Порівняння результатів тестування з вимогами MASVS підтверджує, що застосунок системно не відповідає жодному з рівнів стандарту. Порушення

виявлено в усіх ключових доменах: архітектурний рівень не забезпечує моделі довіреного середовища; засоби зберігання даних не гарантують захисту токенів і конфіденційних значень; автентифікація та керування сесіями працюють неналежним чином і допускають обходження контролю доступу; механізми підтвердження транзакцій відсутні або реалізовані виключно на клієнті; інтеграція з платформою Android є неповною, а WebView і exported activity створюють додаткові ризики; мережеве шифрування не відповідає мінімальним вимогам і дозволяє відновити вміст трафіку. У цілому результати показують, що застосунок не досягає навіть базового рівня MASVS-L1, а вимоги MASVS-L2, орієнтовані на фінансові сервіси, порушені системно.

Оцінювання застосунку відповідно до вимог MASVS показало, що виявлені вразливості охоплюють практично всі сфери безпеки, визначені стандартом. На рівні архітектури застосунок не демонструє належного моделювання загроз: обхід перевірок root-доступу та механізмів антидебагу свідчить про відсутність ефективних засобів контролю оточення, що порушує принципи побудови довіреної моделі виконання. Проблеми із зберіганням даних та журналюванням показують, що застосунок не дотримується базових вимог щодо захисту конфіденційної інформації. Токени автентифікації зберігаються у відкритому вигляді, логування містить чутливі дані, а локальні компоненти платформи не мають жодного додаткового рівня захисту [27].

Механізми автентифікації та керування сесіями також не відповідають вимогам MASVS. Відсутність обмежень на кількість спроб входу, можливість обходу автентифікації через пряме звернення до API та збереження чинності сесії після виходу користувача демонструють системну неспроможність застосунку забезпечити контроль доступу. Особливо критичною є відсутність перевірки прав доступу під час виконання фінансових транзакцій, що робить систему уразливою до підміни операцій та обходу транзакційної автентифікації.

Порушення вимог стосується також використання можливостей платформи, оскільки відкриті для зовнішнього виклику компоненти та

неконтрольоване використання WebView створюють додаткові вектори атаки. Неправильне налаштування Android-компонентів і відсутність валідації вхідних параметрів дозволяють запускати внутрішні екрани та виконувати небезпечний код у межах інтерфейсу застосунку.

Окремою проблемою є безпека мережевої взаємодії. Застосунок не використовує жодних сучасних криптографічних механізмів, натомість покладаючись на просту XOR-схему, що дозволяє повністю розшифровувати передані дані. Це свідчить про повну невідповідність вимогам щодо захисту мережевого каналу і робить конфіденційність інформації вразливою навіть за наявності TLS.

У сукупності ці недоліки показують, що застосунок не відповідає базовому рівню MASVS-L1, а вимоги MASVS-L2 орієнтовані на фінансові мобільні застосунки порушені системно. Результати аналізу вказують на необхідність повного перегляду архітектури безпеки, оскільки окремі механізми не можуть бути виправлені локально без змін у моделі доступу, принципах обробки даних та серверній логіці.

2.3.3 Відповідність OWASP Mobile Top 10 та OWASP API Security

Зіставлення виявлених вразливостей із категоріями OWASP Mobile Top 10 та OWASP API Security підтверджує їхнє віднесення до найпоширеніших класів загроз мобільних і серверних платформ. Проблеми зберігання та журналювання конфіденційних даних відповідають категоріям, що охоплюють неправильну обробку чутливої інформації. Відсутність належного мережевого шифрування узгоджується з категоріями, які описують слабкі або неправильно реалізовані криптографічні механізми. Недоліки автентифікації, такі як необмежена кількість спроб входу, обхід логіну або неправильне завершення сесії, повністю корелюють із категоріями, що описують уразливі механізми авторизації та підтримання сесій. Виявлені порушення бізнес-логіки узгоджуються з

категоріями API6 та API7, які визначають ризики некоректної валідації параметрів і відсутність контролю бізнес-правил. Поведінкові проблеми WebView відповідають вразливостям ін’єкційного типу, описаним у M7. Сукупно ці результати підтверджують, що застосунок містить повний спектр типових ризиків, окреслених у OWASP.

Порівняння результатів тестування з вимогами OWASP Mobile Top 10 та OWASP API Security показує, що виявлені вразливості безпосередньо відповідають найпоширенішим і найкритичнішим категоріям загроз для мобільних і серверних компонентів. Проблеми збереження та журналювання конфіденційних даних узгоджуються з категорією M2, яка охоплює неправильне зберігання чутливої інформації на пристрої й відсутність механізмів її належного захисту. Недоліки у шифруванні мережевого трафіку та можливість розшифрування запитів і відповідей повністю відповідають категорії M3 та API8, де описуються загрози, пов’язані з недостатньою захищеністю каналу зв’язку та використанням неякісних криптографічних підходів [28].

Усі вразливості, пов’язані з автентифікацією зокрема обходження механізму входу, відсутність обмеження кількості спроб, некоректне завершення сесії або нездатність перевіряти права доступу під час транзакцій корелюють із категорією M4 та API2, які акцентують на відсутності надійної автентифікації та пропускних механізмів підтримання сесії. Ці недоліки визначають фундаментальні проблеми, що роблять систему вразливою до компрометації облікових даних і несанкціонованих дій.

Окрему групу становлять порушення бізнес-логіки, виявлені під час тестування механізму фінансових переказів. Вони узгоджуються з категоріями API6 та API7, які описують недостатню валідацію бізнес-правил, відсутність захисту транзакційних операцій та некоректну інтерпретацію параметрів запитів на серверному рівні. Такі вразливості створюють прямий ризик фінансових маніпуляцій, навіть якщо інші механізми безпеки залишаються формально активними.

До клієнтських проблем належить можливість виконання довільного JavaScript у WebView через некоректну обробку deeplink-посилань. Цей випадок відповідає категорії M7, яка стосується ін’єкційних атак та неналежної ізоляції виконуваного коду всередині мобільного застосунку. Подібні помилки дозволяють підмінювати контент, вводити користувача в оману та створювати вектори соціальної інженерії.

Таким чином, практично кожна з виявлених вразливостей має прямий відповідник у OWASP Mobile Top 10 або OWASP API Security, що підтверджує системний характер недоліків і дозволяє оцінювати рівень загроз не лише в контексті конкретного застосунку, але й у ширшому масштабі поширених ризиків сучасних мобільних фінансових сервісів.

2.3.4 Відповідність вимогам PCI DSS

Аналіз сумісності поведінки застосунку з вимогами PCI DSS свідчить про повну невідповідність ключовим безпековим принципам, необхідним для захисту платіжних даних. Незахищене зберігання токенів автентифікації, відсутність коректного журналювання та використання нестійких криптографічних алгоритмів показують порушення основних положень щодо захисту даних у спокої та під час передання. Механізми контролю доступу не відповідають вимогам строгої автентифікації та авторизації: можливість обходу логіну, відсутність rate limiting і відсутність транзакційної автентифікації суперечать практиці багаторівневого підтвердження доступу. Обхід захисних механізмів середовища виконання робить неможливим забезпечення цілісності пристрою, що теж є обов’язковим у PCI DSS. У результаті застосунок не демонструє здатності гарантувати безпеку фінансових даних і не може розглядатися як придатний до роботи у середовищах, де очікується обробка карткової інформації.

Співвіднесення результатів тестування з вимогами PCI DSS показує, що застосунок не відповідає базовим принципам забезпечення безпеки фінансових даних. Хоча DVB не працює з реальними картковими реквізитами, сам характер вразливостей дозволяє однозначно оцінити, що в разі обробки платіжної інформації застосунок не зміг би виконати вимоги до її захисту. Проблеми із зберіганням даних свідчать про порушення фундаментальних вимог стандарту: токени автентифікації зберігаються у відкритому вигляді, чутлива інформація потрапляє до логів, а механізми шифрування є нестійкими та легко відтворюваними. У термінах PCI DSS це означає відсутність надійного захисту карткових даних «у спокої» та під час передавання [29].

Аналіз також показав відсутність дієвого контролю доступу. Обхід автентифікації, відсутність обмежень на кількість спроб входу й некоректна робота серверної авторизації суперечать вимогам PCI DSS щодо суворого розмежування доступу та застосування багатофакторної автентифікації для критичних операцій. Механізми підтвердження транзакцій не прив’язані до серверної логіки, що порушує принципи «need-to-know» і «least privilege», визначені стандартом.

Журналювання також реалізовано неналежним чином. Логи містять чутливі дані, зокрема токени, що прямо порушує вимоги PCI DSS щодо недопущення витоку персональних і фінансових даних через журнальні механізми. Крім того, застосунок не забезпечує захищеного середовища виконання: механізми root detection і anti-debugging обходяться без зусиль, що робить неможливим дотримання вимог щодо цілісності середовища, у якому обробляються платіжні операції [30].

У сукупності ці результати показують, що застосунок не лише не досягає рівня відповідності PCI DSS, але й не має фундаментальних механізмів, необхідних для обробки платіжних даних у будь-якому середовищі. Це вказує на необхідність глибокої архітектурної перебудови, оскільки окремі виправлення не зможуть усунути системні порушення вимог стандарту.

Висновки до розділу 2

Інтегрована оцінка результатів тестування показує, що критичні технологічні та архітектурні недоліки охоплюють усі рівні функціонування мобільного застосунку. Порівняння CVSS-оцінок із вимогами MASVS, OWASP Mobile та PCI DSS демонструє, що процеси автентифікації, керування сесіями, захисту бізнес-логіки, конфіденційності даних та контроль цілісності середовища не відповідають базовому рівню зрілості безпеки, очікуваному від фінансового застосунку. Критичні вразливості, пов’язані з обходом автентифікації, відсутністю rate limiting та коректних транзакційних перевірок, свідчать про те, що серверна частина не встановлює меж довіри та не контролює поведінку клієнта. Вразливості середнього рівня, включно з проблемами WebView, зберігання даних та журналюванням, вказують на відсутність базових принципів захисту на стороні мобільного клієнта.

Сукупність цих факторів дозволяє зробити висновок, що рівень безпеки застосунку є низьким. Системним коренем більшості знайдених проблем є неправильно побудована модель авторизації та відсутність чіткої архітектури контролю доступу на серверному рівні. До цього додається відсутність захищених криптографічних механізмів, слабка інтеграція платформних можливостей безпеки Android та відсутність захищеного SDLC, що приводить до повторюваних помилок у різних частинах застосунку. У такому вигляді система є вразливою до широкого спектра атак, а експлуатація однієї критичної вразливості здатна посилити наслідки інших, створюючи ефект каскадної компрометації. Це підтверджує необхідність глибинної перебудови архітектури безпеки та впровадження системних підходів до її забезпечення.

Підсумовуючи результати оцінювання, можна стверджувати, що застосунок містить сукупність критичних ризиків, які безпосередньо впливають на конфіденційність даних, цілісність фінансових операцій та надійність механізмів автентифікації. Найбільш вразливими виявилися компоненти,

відповідальні за керування сесіями, авторизацію, обробку транзакцій і захист локальних даних, що свідчить про системну недостатність реалізованої моделі безпеки. Виявлені недоліки не існують ізольовано: слабкі криптографічні механізми, відсутність серверних перевірок, порушення бізнес-логіки й некоректне використання платформних можливостей підсилюють один одного, створюючи середовище для каскадних атак. У такому вигляді рівень захищеності застосунку є явно недостатнім для фінансових сервісів, а загальна картина вказує на фундаментальні архітектурні помилки, що потребують переосмислення. Саме ці результати формують основу для подальших рекомендацій, спрямованих на усунення ключових класів вразливостей і впровадження цілісної моделі безпеки, яка відповідатиме вимогам галузевих стандартів.

РОЗДІЛ 3. РЕКОМЕНДАЦІЇ ЩОДО ВДОСКОНАЛЕННЯ БЕЗПЕКИ МОБІЛЬНИХ ЗАСТОСУНКІВ ФІНАНСОВОГО ПРИЗНАЧЕННЯ

3.1. Розроблення технічних заходів для підвищення рівня захищеності мобільних застосунків

Результати проведеного дослідження показали, що в мобільних застосунках фінансового призначення поєднуються широкий спектр технічних і архітектурних ризиків, які безпосередньо впливають на надійність механізмів автентифікації, цілісність транзакцій і стійкість до атак на локальні та мережеві компоненти. Виявлені недоліки мають системний характер: окремі технічні помилки підсилюються слабкою логікою контролю доступу, недостатнім використанням можливостей платформи та відсутністю послідовних процесів забезпечення безпеки на етапах розробки й тестування.

У цих умовах підвищення рівня захищеності не може ґрунтуватися на точковому усуненні виявлених вразливостей. Необхідний комплексний підхід, який поєднує впровадження технічних контрзасобів, формування ефективної політики безпечної розробки та адаптацію міжнародних практик. Саме таке поєднання дозволяє забезпечити стійкість мобільних фінансових сервісів до сучасних загроз і запобігти повторенню аналогічних недоліків у майбутньому.

Подальші підрозділи спрямовані на формування практичних рекомендацій, що охоплюють як технічні аспекти побудови захищених мобільних застосунків, так і організаційні процеси, без яких неможливо досягнути системного, довгострокового рівня захисту [31].

Першочерговим напрямом удосконалення є впровадження технічних заходів, спрямованих на усунення виявлених під час дослідження вразливостей. Йдеться про корекцію критично важливих механізмів автентифікації, авторизації та керування сесіями, які повинні працювати виключно на стороні серверної логіки й не покладатися на дані, отримані від клієнта. Потрібно забезпечити безперервну перевірку повноважень користувача під час кожної

операції, включно з транзакціями, а механізми підтвердження дій (PIN або біометрія) мають бути інтегровані в серверну частину відповідно до принципів Strong Customer Authentication. Необхідно відновити коректність бізнес-логіки API: впровадити перевірку допустимості параметрів транзакцій, контролювати позитивність і коректність сум, фіксувати спроби маніпуляцій. Додатково слід забезпечити уніфіковані повідомлення для сценаріїв автентифікації і реєстрації, запровадити обмеження кількості спроб входу, інвалізацію токенів після виходу користувача й примусову ревокацію сесій у разі виявлення аномалій. Такий комплексний підхід дає можливість мінімізувати ризики, пов’язані з несанкціонованим доступом і некоректною обробкою фінансових операцій.

Окремого опрацювання потребують компоненти клієнтської частини, відповідальні за захист локальних даних, журналювання та обробку контенту. Необхідно впровадити безпечні механізми зберігання конфіденційної інформації, використовуючи криптографічні засоби на основі апаратних модулів (наприклад, Android Keystore) та уникаючи збереження токенів у відкритому вигляді. Журналювання повинно бути обмежене до мінімально необхідного обсягу службових даних; чутливі параметри мають бути повністю вилучені з логів. Конфігурації WebView потребують жорсткої ізоляції: контроль дозволених доменів, заборона виконання довільного JavaScript, фільтрація deeplink-посилань, а також вимкнення застарілих або небезпечних API. Середовище виконання має бути посилене: перевірки на root, ін’єкцію коду та відладку повинні поєднувати детекцію на клієнті з незалежним контролем на сервері, щоб унеможливити обхід захисту за допомогою магістральних інструментів. Ці заходи зменшують кількість шляхів атак і забезпечують більшу стійкість застосунку до клієнтських маніпуляцій.

Третьою групою пріоритетних заходів є вдосконалення серверної частини застосунку, оскільки саме вона визначає коректність автентифікації, авторизації та транзакційної логіки. Передусім необхідно впровадити чітку модель доступу, у якій кожен запит клієнта проходить незалежну серверну перевірку прав

користувача, а всі критичні операції виконуються виключно після успішної автентифікації та підтвердження особи. Будь-які механізми, що покладаються на клієнтську сторону для прийняття рішень, мають бути перенесені на сервер, включно з перевіркою транзакцій, визначенням дозволених дій та оцінкою ризиків [32].

Сервер також повинен забезпечувати повний контроль життєвого циклу сесії. Токени автентифікації мають інвалізуватися негайно після виходу користувача, а повторні використання попередніх токенів повинні блокуватися. Необхідно впровадити механізми rate limiting та захисту від brute-force, а також детектування аномальної поведінки, зокрема повторюваних спроб входу або великої кількості транзакцій за короткий проміжок часу. Важливим елементом є коректна робота з токенами: їхній строк дії повинен бути обмежений, а для чутливих операцій мають застосовуватися окремі короткоживучі токени підтвердження.

Окрему увагу слід приділити транзакційній логіці. Усі параметри запитів повинні проходити сувору серверну валідацію, включно з перевіркою допустимості сум, наявності достатнього балансу, відповідності отримувача, зв’язності операції з автентифікованою сесією та факту виконання транзакційної автентифікації. Фінансова операція не може виконуватися лише на основі клієнтського запиту; вона має підтверджуватися сервером як така, що відповідає встановленим правилам і політикам безпеки. Впровадження цих заходів забезпечує стійкість системи до маніпуляцій із параметрами запитів та унеможливлює експлуатацію помилок бізнес-логіки.

Четвертим напрямом удосконалення є переосмислення архітектури застосунку таким чином, щоб запобігти повторному виникненню вразливостей, подібних до виявлених під час дослідження. Ключовою вимогою є відокремлення відповідальності між клієнтською та серверною частинами: мобільний застосунок не повинен містити логіки, що визначає умови виконання критичних операцій, перевірку прав користувача чи валідацію транзакційних

параметрів. Усі рішення, які впливають на безпеку, повинні прийматися виключно сервером, а клієнт має виконувати роль інтерфейсу, що посилає запити, але не формує бізнес-логіку. Такий підхід мінімізує ризик маніпуляцій на рівні клієнта та знижує залежність від цілісності середовища мобільного пристрою.

Архітектура також повинна передбачати багаторівневий контроль доступу, де кожний елемент системи має чітко визначені повноваження. Для цього необхідно впровадити централізовану модель авторизації, у якій усі API-ендпойнти декларують свої вимоги до автентифікації, рівня довіри та контексту виконання. Така модель унеможливлює виконання критичних операцій без відповідних підтверджень і значно спрощує аудит безпеки. Додатково архітектура повинна враховувати принцип найменших привілеїв: мобільний застосунок не повинен мати змоги ініціювати дії, які не відповідають його функціональному профілю або стану сесії.

Окреме значення має впровадження захищеної криптографічної підсистеми. Використання нестійких алгоритмів, таких як XOR із фіксованим ключем, повинно бути повністю виключене з архітектури. Натомість необхідно використовувати перевірені механізми підписання та шифрування, де криптографічні операції виконуються або сервером, або спеціалізованими апаратними модулями мобільної платформи. Це дає змогу гарантувати цілісність, автентичність і конфіденційність даних незалежно від стану пристрою.

Також важливо передбачити ізоляцію критичних компонентів клієнтської частини. Бізнес-логіка, токени, ключі та інші чутливі елементи не повинні бути безпосередньо доступні у декомпільованому вигляді. Потрібно застосовувати методи мінімізації поверхні атаки: обмеження exported-компонентів, контроль WebView, блокування виконання довільного коду, застосування перевірок цілісності та захисту від модифікації. Архітектурний захист має включати і

механізми моніторингу, що дозволяють виявляти аномалії та вчасно реагувати на спроби порушення політик [33].

Комплексність цих заходів формує новий структурний підхід, у якому безпека розглядається не як сукупність точкових механізмів, а як системна властивість мобільного фінансового сервісу. Така архітектура здатна не лише усунути існуючі вразливості, але й запобігти їх повторному виникненню у нових версіях застосунку.

Підсумовуючи викладені технічні рекомендації, можна стверджувати, що підвищення рівня захищеності мобільних застосунків фінансового призначення потребує не лише усунення окремих вразливостей, а й системного перегляду фундаментальних механізмів їх побудови. Розглянуті заходи охоплюють як критично важливі аспекти серверної логіки автентифікацію, керування сесіями, перевірку транзакцій і захист бізнес-правил так і компоненти клієнтської частини, пов’язані із зберіганням даних, журналюванням, використанням WebView та контролем цілісності середовища виконання. Окреме значення має впровадження архітектурних принципів, що чітко розмежовують відповідальність між клієнтом і сервером та гарантують, що всі рішення щодо безпеки приймаються на довірених рівнях системи.

У сукупності ці рекомендації створюють основу для побудови мобільних фінансових сервісів, здатних протидіяти сучасним загрозам і забезпечувати належний рівень захисту персональних і транзакційних даних. Їхня реалізація є необхідною передумовою для переходу від реактивного усунення вразливостей до превентивного, системного підходу до безпечної розробки. Саме тому наступні підрозділи зосереджуються на формуванні процесів і політик, що дозволяють закріпити ці технічні заходи на організаційному рівні та забезпечити їх довгострокове функціонування.

3.2. Формування політики та процесів безпечної розробки і тестування мобільних застосунків

Ефективність технічних заходів забезпечення безпеки значною мірою залежить від того, наскільки послідовно та системно вони впровадлюються у процеси розробки й тестування. Навіть добре спроєктовані механізми захисту не гарантують належного рівня безпеки, якщо вони не підтримуються відповідною політикою, внутрішніми регламентами та культурами роботи команди. Саме тому важливим елементом підвищення захищеності мобільних фінансових застосунків є формування цілісної системи організаційних та процесних заходів, які забезпечують безпечний життєвий цикл програмного забезпечення.

У цьому підрозділі розглядаються принципи побудови безпечного SDLC, вимоги до проведення регулярного тестування, інтеграція перевірок безпеки в кожний етап розробки та формування внутрішніх процедур, що унеможливлюють появу системних вразливостей. Особлива увага приділяється стандартизації підходів, автоматизації критичних перевірок і створенню таких умов, за яких безпека стає природною частиною продуктового процесу, а не реакцією на окремі інциденти.

Формування ефективної політики безпеки для мобільних застосунків починається з визначення правил і процедур, які регулюють усі етапи роботи над продуктом від початкового проєктування до супроводу в експлуатації. Внутрішні регламенти мають встановлювати чіткі вимоги до роботи з конфіденційними даними, принципи доступу до середовищ розробки та тестування, порядок управління секретами, а також стандарти кодування, що забороняють використання небезпечних практик. Важливо забезпечити повторюваність і передбачуваність процесів: прозоре документування рішень, контроль змін, аудит критичних компонентів і регулярний перегляд політик відповідно до актуальних загроз. Такий підхід створює основу для циклічного

підтримання безпеки та мінімізує залежність від індивідуальних рішень розробників або зовнішніх чинників.

Наступним елементом є впровадження принципів Secure SDLC, які передбачають включення перевірок безпеки в кожний етап життєвого циклу програмного забезпечення. На етапі проєктування необхідно враховувати вимоги до моделі загроз і визначати зони довіри; під час розробки дотримуватися безпечних практик програмування й проводити регулярні код-рев'ю з фокусом на безпеці; на етапі тестування застосовувати як статичні, так і динамічні методи аналізу, включно з тестуванням API та бізнес-логіки. Особливу роль відіграє автоматизація: використання інструментів SAST, DAST, інтеграція сканерів у пайплайн CI/CD, контроль залежностей і регулярне оновлення компонентів. Завдяки цьому безпека перестає бути точковою активністю й перетворюється на постійний процес, що супроводжує продукт на всіх стадіях його життєвого циклу [34].

Важливою складовою організаційної моделі безпеки є формування чітких вимог до тестування, яке має охоплювати не лише окремі функціональні елементи, а й цілісну логіку роботи застосунку. Статичний аналіз дозволяє на ранніх етапах виявляти структурні недоліки, помилки конфігурації та небезпечні практики програмування, тоді як динамічне тестування забезпечує перевірку фактичної поведінки системи, взаємодії з платформою, реакції на порушення середовища виконання та стійкість до атак під час реального використання. Тестування API має проводитися окремо й систематично, оскільки саме серверні ендпойнти визначають коректність автентифікації, авторизації та обробки транзакцій. Особливу увагу необхідно приділяти аудиту бізнес-логіки, який часто виходить за межі стандартних інструментів та потребує аналізу правил, що регулюють фінансові операції.

Для забезпечення стабільності таких процесів мають проводитися регулярні перевірки відповідності вимогам MASVS, OWASP та іншим релевантним стандартам. Це дозволяє оперативно виявляти відхилення від

найкращих практик і попереджати накопичення критичних помилок у нових версіях програмного забезпечення. У сукупності ці підходи забезпечують системне й повторюване тестування безпеки, яке формує передумови для довгострокової стійкості мобільних фінансових сервісів.

Ефективна політика безпеки неможлива без впровадження централізованих механізмів управління секретами, токенами, ключами та іншими чутливими артефактами, які визначають довіру між компонентами системи. У мобільних фінансових застосунках особливо важливо уникати децентралізованого зберігання таких елементів у коді або конфігураційних файлах, оскільки це робить їх доступними для реверс-інжинірингу та подальшої експлуатації. Централізована система управління секретами має забезпечувати генерацію, ротацію, відкликання й аудит використання ключів. Її інтеграція з серверною частиною дозволяє контролювати життєвий цикл токенів і виключати можливість їхнього використання поза визначеним контекстом.

Окрему роль відіграє використання спеціалізованих рішень, таких як апаратні модулі захисту ключів або сховища, що підтримують політику access control на рівні сервісів. Подібні механізми забезпечують розмежування прав, мінімізують вплив компрометації окремого компонента та гарантують, що критичні криптографічні операції виконуються у захищеному середовищі. На стороні мобільного клієнта слід покладатися на засоби Android Keystore та прив’язку ключів до конкретного пристрою, що унеможливлює їхнє відтворення в іншому середовищі [35].

Упровадження таких практик створює керовану, прозору й передбачувану модель роботи з секретами, яка знижує ймовірність експлуатації вразливостей, пов’язаних із витоком токенів, компрометацією ключів або підміною критичних параметрів. У довгостроковій перспективі централізоване управління секретами є необхідною складовою безпечної розробки та експлуатації мобільних фінансових сервісів.

Успішність будь-якої політики безпеки залежить не лише від технічних засобів, а й від рівня підготовки команди та наявності усталених внутрішніх стандартів роботи. Практики безпечної розробки мають закріплюватися через навчання, регулярне підвищення кваліфікації та систематичне ознайомлення фахівців із новими моделями атак і вимогами галузевих стандартів. Внутрішні стандарти кодування повинні містити вимоги щодо роботи з чутливими даними, використання криптографії, обмеження доступу до платформних API та правила уникнення типових уразливостей. Обов’язкове код-рев’ю з акцентом на безпеку дозволяє виявляти потенційні проблеми ще до етапу тестування, а автоматизація перевірок у CI/CD-процесах дає змогу оперативно виявляти порушення політик та запобігати появі системних помилок у нових версіях програмного забезпечення. Усе це формує сталу культуру безпечної розробки, у якій команда діє не реактивно, а превентивно, закладаючи вимоги до безпеки в основу продукту.

Підсумовуючи, формування політики та процесів безпечної розробки є ключовою передумовою для усунення системних уразливостей і забезпечення сталого рівня захищеності мобільного фінансового сервісу. Технічні рішення, розглянуті у попередньому підрозділі, матимуть стійкий ефект лише за умови їх підтримки організаційними механізмами та усталеними процесами контролю. Впровадження системного Secure SDLC, стандартизація підходів, автоматизація перевірок та підвищення компетентності команди створюють середовище, у якому безпека інтегрується в життєвий цикл продукту й перестає бути окремим етапом або разовою активністю. Це забезпечує довготривалу стійкість застосунку до загроз та формує основу для інтеграції міжнародних практик, що розглядатимуться в наступному підрозділі.

3.3. Інтеграція міжнародних практик і стандартів у систему забезпечення безпеки мобільних фінансових сервісів

Побудова стійкої системи безпеки мобільних фінансових сервісів неможлива без опори на міжнародні практики, що акумулюють багаторічний досвід протидії сучасним кіберзагрозам. Технічні заходи та внутрішні процеси, розглянуті у попередніх підрозділах, формують основу захисту, однак їх ефективність значно зростає тоді, коли вони узгоджуються з усталеними глобальними стандартами та підходами. Саме стандарти визначають рівень вимог до архітектури, програмних компонентів і процедур контролю, забезпечуючи відтворюваність, прогнозованість і перевірюваність системи безпеки.

У цьому підрозділі увага зосереджена на тому, як міжнародні рамки зокрема MASVS, OWASP Mobile, OWASP API Security, PCI DSS та інші галузеві рекомендації можуть бути інтегровані у практичну діяльність з побудови й супроводу мобільних фінансових застосунків. Йдеться не лише про формальну відповідність вимогам, а про створення цілісної моделі, у якій архітектурні рішення, процеси розробки, тестування та операційна експлуатація узгоджуються в єдину систему захисту. Такий підхід дозволяє не просто зменшувати ризики, а й забезпечувати довгострокову стійкість сервісу до нових класів атак [37].

Міжнародні стандарти відіграють ключову роль у формуванні цілісної моделі безпеки мобільних фінансових застосунків, оскільки вони узагальнюють перевірені підходи та встановлюють вимоги, що забезпечують стійкість системи до сучасних загроз. MASVS визначає базові та підвищені рівні захисту для мобільних застосунків, охоплюючи архітектуру, зберігання даних, мережеву взаємодію, автентифікацію та роботу з платформою. Його застосування дозволяє визначити мінімальний перелік технічних характеристик, які повинен мати будь-який фінансовий застосунок, та виявити розриви між поточним станом і очікуваними вимогами.

OWASP Mobile Top 10 та OWASP API Security надають більш деталізовані орієнтири щодо найпоширеніших класів атак, що виникають на рівні мобільного клієнта та серверних ендпойнтів. Інтеграція цих стандартів дозволяє систематизувати ризики, зосередитися на найбільш вразливих областях і забезпечити, щоб усі критичні вектори атак були враховані під час проектування та тестування. Це підсилює технічні заходи, впроваджені в попередніх підрозділах, і формує практичний перелік перевірок, які повинні виконуватися регулярно.

PCI DSS і стандарти NIST розширюють цю модель, формуючи вимоги до захисту платіжних даних, управління ключами, контролю доступу та моніторингу. Їхнє застосування дозволяє інтегрувати захист не лише на рівні програмного забезпечення, але й у контексті операційних процесів, інфраструктури та управління середовищем виконання. Разом ці стандарти створюють багаторівневу рамку, у якій безпека розглядається як взаємопов’язана система, що включає технічні механізми, процеси та політики. Саме така інтеграція дозволяє сформувати послідовну модель безпеки, здатну забезпечити високий рівень захисту мобільних фінансових сервісів у реальних умовах експлуатації [38].

Внутрішні політики компанії мають спиратися на ключові положення міжнародних стандартів, адаптуючи їх до особливостей конкретного мобільного фінансового сервісу. Елементи MASVS слід використовувати як базову матрицю вимог до архітектури та функціональних компонентів застосунку: обмеження доступу до платформних можливостей, надійні механізми зберігання ключів і токенів, вимога до криптографічного захисту даних у всіх станах, чітке розмежування відповідальності між клієнтом і сервером. Ці положення формують основу політики безпеки продукту та визначають, які технічні рішення є допустимими на етапах планування й реалізації.

Зі свого боку, OWASP Mobile Top 10 та OWASP API Security визначають набір категорій ризиків, які повинні бути обов’язково враховані під час

розробки політик і процедур. Йдеться про вимоги до валідації вхідних даних, контролю автентифікації й авторизації, обмеження обсягу і характеру логування, впровадження механізмів rate limiting, забезпечення цілісності сесій, а також постійний контроль над незахищеними інтерфейсами, такими як WebView і exported-компоненти. Положення цих стандартів можуть бути трансформовані у список заборонених практик та обов'язкових контролів, що мають бути застосовані до всіх майбутніх релізів.

Елементи PCI DSS і NIST також повинні бути інтегровані у внутрішні політики, оскільки вони визначають вимоги до керування ключами, захисту середовища виконання, моніторингу подій і контролю доступу. Зокрема, політики повинні передбачати ротацію ключів, обмеження прав доступу до серверів, ізоляцію середовищ розробки та тестування, вимоги до журналювання без витоку чутливих даних і наявність безперервного моніторингу активності. Усе це формує фундамент, на якому будується комплексна внутрішня система безпеки, що регламентує не лише технічну реалізацію, а й процеси, які забезпечують її сталу роботу [39].

Практичне впровадження міжнародних стандартів потребує чіткого операційного механізму, який забезпечує контроль архітектури, оцінювання ризиків та регулярний аудит безпеки. Архітектурні рішення повинні проходити попередню експертизу на відповідність ключовим вимогам MASVS, OWASP та PCI DSS, а зміни в системі оцінку з погляду потенційного впливу на моделі загроз. Ризик-орієнтований підхід дає змогу впорядкувати пріоритети: найбільшу увагу слід приділяти компонентам, які безпосередньо впливають на автентифікацію, транзакції, зберігання даних і взаємодію з API. Регулярні аудити, зокрема зовнішні, дозволяють перевірити відповідність установленим стандартам і виявити слабкі місця, які могли бути пропущені під час внутрішніх перевірок. Для систем, що опрацьовують фінансові дані, доцільно включати також сертифікаційні вимоги, які формують додатковий рівень контролю та підтверджують, що процеси відповідають галузевим нормам.

У цьому контексті особливо важливо дотримуватися багаторівневого підходу до безпеки, що поєднує принципи defense-in-depth, zero trust і secure-by-design. Ідея багаторівневості полягає в тому, що жоден окремий механізм захисту не повинен розглядатися як достатній. Кожний шар системи клієнтський додаток, серверна логіка, мережеві протоколи, середовище виконання, системи моніторингу має містити власні засоби контролю, які компенсують або підсилюють один одного. Принцип zero trust передбачає, що жоден компонент, жоден запит і жодна сесія не вважаються довіреними за замовчуванням; кожна дія повинна бути підтверджена та перевірена незалежно від її походження. Secure-by-design, у свою чергу, визначає підхід, за якого безпека враховується не після виявлення проблем, а закладається в основу архітектури ще на етапі проєктування. Поєднання цих підходів забезпечує стійкість системи перед складними атаками та мінімізує можливість каскадної компрометації, коли експлуатація однієї вразливості спричиняє порушення роботи інших компонентів.

Створення узагальненої моделі інтеграції міжнародних стандартів у процеси розробки та експлуатації мобільних фінансових застосунків передбачає гармонійне поєднання технічних, процесних і організаційних елементів у єдину систему. Така модель має враховувати вимоги MASVS як основу для побудови архітектури та функціональних компонентів; рекомендації OWASP як практичний орієнтир щодо типових векторів атак і контрольних заходів; а також положення PCI DSS і NIST як фундамент для управління даними, ключами та середовищем виконання. Інтеграція цих стандартів повинна бути не формальною, а операційною вимоги мають бути перенесені у внутрішні політики, відображені в інструментах розробки та підтримані системою моніторингу, аудиту й реагування на інциденти.

Центральне місце в такій моделі посідає механізм зворотного зв’язку: результати тестування, аудиту або виявлені інциденти мають не тільки усуватися локально, а й повертатися до процесів планування, моделювання

загроз та оновлення архітектурних рішень. Це робить систему самокоригувальною, здатною адаптуватися до нових видів атак і технологічних змін. У поєднанні з автоматизованими перевірками безпеки, регулярною переоцінкою ризиків і стандартизованим підходом до документування рішень така модель забезпечує прозорість, керованість і повторюваність процесів, необхідних для тривалого підтримання високого рівня захищеності мобільних фінансових сервісів [40].

Підсумовуючи, інтеграція міжнародних стандартів у процеси розробки та експлуатації не є додатковим або факультативним елементом, а формує основу сучасної системи безпеки мобільного застосунку. Саме така інтегрована модель дозволяє забезпечити узгодженість архітектурних рішень, технічних засобів і процесних підходів, усуваючи розриви між окремими компонентами системи. Вона створює умови, за яких безпека стає не зовнішньою надбудовою, а природною властивістю продукту на всіх етапах його життєвого циклу. Таким чином, рекомендації, розглянуті у цьому підрозділі, доповнюють технічні та процесні заходи попередніх частин і формують цілісну структуру, що забезпечує довгострокову стійкість мобільних фінансових сервісів до сучасних загроз.

Висновки до розділу 3

У цьому розділі було сформовано комплекс рекомендацій, спрямованих на суттєве підвищення рівня захищеності мобільних застосунків фінансового призначення. Запропоновані технічні заходи передбачають усунення виявлених недоліків у механізмах автентифікації, керування сесіями, обробці транзакцій та захисті локальних даних, а також впровадження архітектурних рішень, які мінімізують імовірність повторного виникнення аналогічних вразливостей. Значну увагу приділено побудові процесної складової безпеки, що включає використання принципів Secure SDLC, стандартизацію підходів до тестування,

автоматизацію перевірок і формування внутрішніх політик, які забезпечують системність і передбачуваність у роботі над продуктом.

Окремий акцент зроблено на інтеграції міжнародних стандартів і галузевих практик, які становлять методологічну основу для побудови сучасних систем захисту мобільних фінансових сервісів. Масштабність і різноманітність цих практик дозволяють охопити всі критичні аспекти від архітектури та програмної реалізації до операційної підтримки й постійного моніторингу загроз. У поєднанні з внутрішніми технічними рішеннями та процесними підходами це формує багаторівневу модель безпеки, здатну забезпечити стійкість застосунку перед широким спектром атак.

Таким чином, рекомендації, розглянуті у цьому розділі, створюють узгоджену рамку для розробки, тестування та експлуатації мобільних фінансових застосунків у умовах постійного зростання кіберзагроз. Їхнє впровадження дозволяє не лише усунути наявні ризики, але й вибудувати довгострокову модель безпеки, що відповідає вимогам міжнародних стандартів і забезпечує високий рівень захисту користувацьких та фінансових даних.

ВИСНОВКИ

У межах магістерської кваліфікаційної роботи було проведено комплексне дослідження методів і інструментів оцінки захищеності мобільних застосунків фінансового сектору, що дозволило сформувати як теоретичні засади, так і практичні рекомендації щодо підвищення рівня їхньої безпеки.

У першому розділі розглянуто архітектурні особливості мобільних фінансових застосунків, включно з їхньою клієнтсько-серверною логікою, залежністю від можливостей платформи та специфічними вимогами до захисту конфіденційних даних і транзакційних операцій. Досліджено сучасні стандарти та підходи до оцінки безпеки, серед яких MASVS, OWASP Mobile та вимоги фінансових регуляторів, що визначають базові критерії надійності таких систем. Окрему увагу приділено методам дослідження статичному, динамічному та поведінковому аналізу і відповідному інструментарію, який дозволяє виявляти різнорівневі та різнотипні вразливості у реальних умовах.

Другий розділ роботи присвячено практичному аналізу безпеки мобільного застосунку Damn Vulnerable Bank (DVB). Було здійснено повний цикл тестування: від обґрунтування вибору об’єкта та постановки задач до детального статичного, динамічного та API-аналізу. У ході дослідження ідентифіковано низку критичних вразливостей, серед яких порушення механізмів автентифікації й керування сесіями, помилки бізнес-логіки фінансових операцій, небезпечне зберігання та передавання чутливих даних, відсутність контролю середовища виконання та недоліки у взаємодії з платформними компонентами Android. Подальша формалізація знайдених недоліків включала оцінювання їх критичності за CVSS v3.1, аналіз відповідності вимогам MASVS, OWASP та PCI DSS, що дозволило сформувати об’єктивну картину рівня захищеності досліджуваного застосунку та визначити системні причини його вразливості.

У третьому розділі сформовано комплекс рекомендацій, спрямованих на усунення виявлених недоліків та підвищення рівня безпеки мобільних застосунків фінансового призначення. Розглянуті технічні заходи включають відновлення коректної архітектури автентифікації та авторизації, захист бізнес-логіки, безпечне зберігання й обробку даних, посилення середовища виконання та впровадження стійких криптографічних механізмів. Процесні рекомендації охоплюють побудову Secure SDLC, автоматизацію перевірок безпеки, стандартизацію внутрішніх політик і систему контролю життєвого циклу ключів та токенів. Інтеграція міжнародних практик MASVS, OWASP, PCI DSS, NIST розглянута як необхідна умова створення узгодженої, довгостроково ефективної моделі безпеки, яка здатна протидіяти сучасним загрозам.

Підсумовуючи проведене дослідження, можна стверджувати, що безпека мобільних фінансових застосунків є багатокомпонентною проблемою, що потребує поєднання технічних рішень, процесних підходів та застосування міжнародних стандартів. Практичний аналіз DVB показав, що навіть базові порушення бізнес-логіки чи механізмів автентифікації здатні створити критичні ризики для користувачів і фінансової інфраструктури. Натомість системний підхід, заснований на комплексному тестуванні, стандартизованих вимогах і безпечній розробці, дозволяє суттєво підвищити стійкість таких застосунків.

Отримані результати мають як практичну, так і методологічну цінність: вони демонструють ефективність повного циклу оцінки безпеки мобільних застосунків, визначають ключові напрями модернізації та формують основу для подальших досліджень у галузі захисту фінансових мобільних сервісів.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Nagy M., Cseh T. Mobile Application Architecture and Security Challenges. International Journal of Computer Science and Information Security. 2019. Vol. 17, No. 3. P. 12–20. URL: https://www.ijcsis.org/papers/vol17no3/2.pdf

2. OWASP Mobile Application Security Verification Standard (MASVS). OWASP Foundation, 2024. URL: https://owasp.org/www-project-mobile-security-testing-guide/masvs/

3. OWASP Mobile Top 10 2023. OWASP Foundation. URL: https://owasp.org/www-project-mobile-top-10/

4. OWASP API Security Top 10 2023. OWASP Foundation. URL: https://owasp.org/www-project-api-security/

5. Mell P., Bergeron T., Henning D. *A Guide to Intrusion Detection and Analysis.* NIST Special Publication 800-31. Gaithersburg, MD: NIST, 2021. DOI: https://doi.org/10.6028/NIST.SP.800-31

6. PCI Security Standards Council. Payment Card Industry Data Security Standard. Requirements and Security Assessment Procedures. Version 4.0, 2022. URL: https://www.pcisecuritystandards.org/

7. Божко О. О. Оцінка ризиків інформаційної безпеки в фінансових інформаційних системах. Інформаційна безпека. 2021. Т. 27, № 1. С. 35–42. URL: https://journals.uran.ua/infsec/article/view/228627

8. Майстренко О. В. Методи аналізу захищеності програмного забезпечення: сучасний стан та перспективи. Сучасні інформаційні технології. 2019. № 3. С. 14–22. URL: https://sit-journal.org.ua/articles/2019-3/3.pdf

9. Egele M., Brumley D., Fratantonio Y., Kruegel C. *An Empirical Study of Mobile Application Security.* USENIX Security Symposium. 2017. P. 97–112. URL: https://www.usenix.org/conference/usenixsecurity17/

10. Faruki P. et al. *Android Security: A Survey of Issues, Malware Penetration, and Defenses.* IEEE Communications Surveys & Tutorials. 2015. Vol. 17, No. 2. P. 998–1022. DOI: https://doi.org/10.1109/COMST.2014.2364136

11. Чухліб Р. Р., Баран М. І. Методи аналізу уразливостей мобільних застосунків. Вісник НТУУ “КПІ”. Інформатика. 2020. № 4. С. 29–36. URL: https://visnyk-kpi.org.ua/inf/2020-4/4.pdf

12. Fahl S., Harbach M., Acar Y., Smith M. Why Eve and Mallory Love Android: An Analysis of Android SSL (In)Security. ACM CCS. 2012. P. 50–63. DOI: https://doi.org/10.1145/2382196.2382205

13. Enck W. *A Study of Android Application Security.* USENIX Security Symposium. 2011. URL: https://www.usenix.org/legacy/events/sec11/tech/full_papers/Enck.pdf

14. Mahindru R. Mobile App Security Testing: Techniques and Tools Overview. International Journal of Cyber Research. 2022. Vol. 8, No. 2. P. 41–55. URL: https://ijcr.org/articles/2022-2/6.pdf

15. F. Weiser, S. Gascon, J. T. Juen. *Foundations of Mobile App Reverse Engineering: Tools and Techniques.* Journal of Cybersecurity Research. 2021. № 5. С. 33–47. URL: https://jcr.org/fma-reverse-tools.pdf

16. Li L., Bissyandé T. F., Klein J. *DroidRA: Taming Reflection to Support Whole-Program Analysis of Android Apps.* Empirical Software Engineering. 2020. Vol. 25, No. 2. P. 1130–1160. DOI: https://doi.org/10.1007/s10664-019-09773-8

17. OWASP Mobile Security Testing Guide (MSTG). OWASP Foundation, 2024. URL: https://owasp.org/www-project-mobile-security-testing-guide/

18. Desnos A., Gueguen M. *Android Reverse Engineering Techniques.* REMSEC Security Conference. 2022. URL: https://remsec.org/2022/papers/desnos-android-re.pdf

19. Mubashshir S., Sharma A. *Dynamic Analysis of Mobile Applications: Approaches and Tools.* Journal of Network Security. 2023. Vol. 14, No. 1. P. 59–72. URL: https://jnsec.org/articles/2023-1/6.pdf

20. Frida Project Documentation. *Dynamic Instrumentation Toolkit for Developers, Reverse-Engineers, and Security Researchers.* Version 16.0.URL: https://frida.re/docs/

21. Drozer Framework. *Comprehensive Security Assessment Tool for Android.* Bishop Fox, 2023. URL: https://labs.bishopfox.com/tools/drozer

22. Burp Suite Professional Documentation. PortSwigger Ltd., 2024. URL: https://portswigger.net/burp/documentation

23. Enck W., Cox L., Gilbert P. *A Study of Android Application Security.* USENIX Security Symposium. 2011. P. 97–112. URL: https://www.usenix.org/legacy/events/sec11/tech/full_papers/Enck.pdf

24. Kaczmarczyk M., Bielova N. *Dynamic Taint Tracking in Android Systems.* ACM Computing Surveys. 2021. Vol. 54, No. 3. DOI: https://doi.org/10.1145/3446373

25. Davi L., Dmitrienko A., Sadeghi A. *Probabilistic Techniques for Mobile Malware Detection.* IEEE Security & Privacy. 2012. Vol. 10, No. 4. P. 62–69. DOI: https://doi.org/10.1109/MSP.2012.56

26. Manousakas S. *Practical API Penetration Testing for Mobile Applications.* Cybersecurity Engineering Journal. 2022. № 7. С. 18–29. URL: https://cejournal.org/articles/22-7-api.pdf

27. First.org. Common Vulnerability Scoring System (CVSS) v3.1 Specification. 2022. URL: https://www.first.org/cvss/specification-document

28. Latif S., Khan M. *Security Assessments of Mobile Banking Applications in Practice.* IEEE Access. 2021. Vol. 9. P. 55627–55640. DOI: https://doi.org/10.1109/ACCESS.2021.3070835

29. PCI Security Standards Council. *Mobile Payment Acceptance Security Guidelines.* Version 2023. URL: https://www.pcisecuritystandards.org/

30. Faruki P., Bharmal A., Laxmi V. *Android Security: Issues, Malware Penetration, and Defenses.* IEEE Communications Surveys & Tutorials. 2015. Vol. 17, No. 2. P. 998–1022. DOI: https://doi.org/10.1109/COMST.2014.2364136

31. OWASP Mobile Application Security Verification Standard (MASVS). OWASP Foundation, 2024. URL: https://owasp.org/www-project-mobile-security-testing-guide/masvs/

32. OWASP Mobile Security Testing Guide (MSTG). OWASP Foundation, 2024. URL: https://owasp.org/www-project-mobile-security-testing-guide/

33. NIST Special Publication 800-163 Revision 1. *Vetting the Security of Mobile Applications.* National Institute of Standards and Technology, 2022. DOI: https://doi.org/10.6028/NIST.SP.800-163r1

34. PCI Security Standards Council. *PCI DSS: Requirements and Security Assessment Procedures.* Version 4.0, 2022. URL: https://www.pcisecuritystandards.org/

35. Fitzgerald B., Stol K. *Continuous Software Engineering: Foundations and Issues.* Communications of the ACM. 2017. Vol. 59, No. 8. P. 64–70. DOI: https://doi.org/10.1145/2954331

36. McGraw G. *Software Security: Building Security In.* Addison-Wesley Professional, 2021. 448 p. URL: https://www.informit.com/articles/software-security

37. Howard M., Lipner S. *The Security Development Lifecycle.* Microsoft Press, 2019. URL: https://www.microsoft.com/sdl

38. Basile C., Gribaudo M. *Secure Mobile Software Development: Policies, Pitfalls, and Practices.* Journal of Systems and Software. 2020. Vol. 170. P. 110734. DOI: https://doi.org/10.1016/j.jss.2020.110734

39. Agrawal N., Gupta M. *Security Best Practices for Mobile Application Development in Financial Services.* International Journal of Information Management. 2021. Vol. 58. DOI: https://doi.org/10.1016/j.ijinfomgt.2020.102319

40. OWASP Software Assurance Maturity Model (SAMM). OWASP Foundation, 2023. URL: https://owaspsamm.org/