НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

«На правах рукопису»

УДК 004.056

«До захисту допущено»
В.о. завідувача кафедри
_____ М.В.Грайворонський
“___”_____2018 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності: 125 Кібербезпека

на тему: Запобігання витоку таємних ключів використовуючи машинне навчання

Виконав (-ла): студент (-ка) 2 курсу, групи ФБ-71мп
(шифр групи)

Сніговий Дмитро Сергійович
(прізвище, ім’я, по батькові)

Науковий керівник к.ф.-м.н., доц. Грайворонський Микола Владленович _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент к.т.н., доцента ФІОТ Пасько В.П. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою
Спеціальність (спеціалізація) – 125 Кібербезпека («Системи і технології кібербезпеки)

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ М.В.Грайворонський
(підпис)

«___»_____2018 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Сніговому Дмитру Сергійовичу

1. Тема дисертації: Запобігання витоку таємних ключів використовуючи машинне навчання
науковий керівник дисертації к.ф.-м.н., доц. Грайворонський Микола Владленович,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «15» листопада 2018 р. № 4171-с

2. Термін подання студентом дисертації 12.12.2018 р.

3. Об'єкт дослідження _____
_____

4. Вихідні дані _____
_____
_____
_____

5. Перелік завдань, які потрібно розробити _____
_____
_____
_____
_____
_____

6. Орієнтовний перелік ілюстративного матеріалу _____
_____
_____

7. Орієнтовний перелік публікацій _____
_____

8. Консультанти розділів дисертації*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка

Студент _____ (підпис) _____ (ініціали, прізвище)

Науковий керівник дисертації _____ (підпис) _____ (ініціали, прізвище)

* Консультантом не може бути зазначено наукового керівника магістерської дисертації.

Реферат

Робота обсягом 86 сторінок, містить 42 ілюстрації, 22 таблиці та 10 літературних джерел. Мета дослідження – довести можливість автоматизації пошуку та знешкодження витоку таємних ключів за допомогою машинного навчання. Область дослідження даної роботи – операційна система Android. В ході виконання даної дипломної роботи відбувається аналіз витоку памяті, витоку таємних ключів, алгоритмів машинного навчання, що могли б бути застосовані для автоматизації процесу пошуку витоку таємних ключів. В практичній частині відбувається як ручне так і автоматичне(використовуючи машинне навчання) виявлення витоків таємних ключів того чи іншого типу. Дослідження проведені у даній роботі можуть слугувати або бути розціненими як один з варіантів поліпшення процесу пошуку да видалення витоку таємних ключів усіх категорій. Також процес аналізу відкритого коду для групування даних з метою їх подальшого використання у машинному навчанні може сприяти розвитку повноцінної та автоматизованої системи пошуку та видалення витоку таємних ключів.

Результатом роботи є практичне та теоретичне підтвердження існування можливості поліпшення та прискорення процесу пошуку та вирішення проблеми витоку секретних ключів. Можливості використання машинного навчання для вирішення проблеми витоку секретних ключів були продемонстровані у практичній частині даної роботи.

ANDROID, ВИТОКИ ПАМЯТІ, ТАЄМНІ КЛЮЧІ, МАШИННЕ НАВЧАННЯ.

Abstract

Work capacity - 86 pages, contains 42 illustrations, 22 tables and 10 literary sources. The purpose of the study is to make it possible to automate the search and disposal of secret keys leakage through machine learning. The study area of this work is the Android operating system. During the execution of this thesis there is an analysis of leakage of memory, leakage of secret keys, algorithms of machine learning that could be applied to automate the process of finding leakage of secret keys. In the practical part there is both manual and automatic (using machine learning) detection of the sources of secret keys of one type or another. Studies conducted in this paper may serve or be regarded as one of the options for improving the search process and removing the leakage of secret keys of all categories. Also, an open-source analysis process for grouping data for further use in machine learning can facilitate the development of a fully-fledged and automated search engine and eliminate the leakage of secret keys.

The result of the work is the practical and theoretical confirmation of the possibility of improving and accelerating the process of finding and solving the problem of leakage of secret keys. The possibilities of using machine learning to solve the problem of leakage of secret keys have been demonstrated in the practical part of this work.

ANDROID, MEMORY LEAKS, SECURE MEMORY LEAKS, MACHINE LEARNING.

Перелік умовних позначень, символів, одиниць, скорочень і термінів

CC	Common Criteria
MDFPP	Protection Profile for Mobile Defensive Fundamentals
TOE	Target of Evaluation
AOSP	Android Open Source Project
IPC	Inter Process Communication
BT/BLE	Bluetooth/Bluetooth Low Energy

Вступ

Актуальність роботи. Android сьогодні є найпопулярнішою операційною системою, що інсталюється в мобільні телефони, планшети, телевізори, комп'ютери тощо. Тому, питання безпеки та здатність цієї системи протистояти різного родам атакам є вкрай важлива. Всі найновіші версії Android використовують велику кількість технологій захисту. Разом з великою кількістю компонент та механізмів пов'язана велика кількість таємних ключів, їх основні види: ті які використовуються в криптографічних операціях шифрування, генерації інших ключів, також велика кількість паролів та тимчасових ключів. Кожний таємний ключ є об'єктом який не повинен потрапити до рук зловмисника. З кожною новою версією Android кількість ключів або даних, що залежать від таких ключів зростає. Проблема витоку пам'яті була відома дуже давно, як і проблема витоку таємних ключів. Основна проблема полягає в складності пошуку та видаленню таких витоків. На сьогодні не існує автоматизованого механізму виявлення місця витоку ключів. Тому, враховуючи всі фактори, спроба автоматизації, нехай і часткової є вкрай корисна і має перспективи для подальшого розвитку.

Мета і завдання дослідження. Мета дослідження – довести існування можливості автоматизації пошуку та знешкодження витоку таємних ключів. Завдання дослідження – на основі проаналізованих алгоритмів машинного навчання та зібраних даних з відкритого коду операційної системи Android розробити автоматизований застосунок для аналізу та виявлення витоку секретних ключів.

Об'єкт дослідження – операційна система Android (AOSP збірка).

Предмет дослідження – витоки пам'яті таємних ключів.

Методи дослідження. В даній роботі було використано два методи дослідження: емпіричний і експериментальний. Тобто, проводяться дослідження існуючих технологій, що з'явилися в результаті багатолітніх досліджень різних професіоналів, їх аналіз на основі досвіду та знань автора даної роботи. Також,

вагомою частиною досліджень були практичні, тобто такі, які базувалися на деякій кількості практичних спроб, встановлення істинності чи хибності, різних досліджуваних процесів.

Наукова новизна одержаних результатів. На сьогоднішній день не існує застосунків або підходів для автоматичного виявлення витоку секретних ключів. Тому запропоновані підходи та застосунки в даній роботі є унікальні та перспективні для мільйонів людей.

Практичне значення одержаних результатів. Наведені техніки для автоматизації пошуку витоків таємних ключів можуть бути використані будь-яким вендором, що розробляє свою операційну систему, її модулі чи будь-які інші компоненти.

Апробація результатів роботи. Всі отримані результати даної роботи підтверджені практично та теоретично, були перевірені на працездатність в ході написання автоматизованого застосунку у середовищі операційної системи Linux(Ubuntu 14.04).

ЗМІСТ

1 ОГЛЯД ОПЕРАЦІЙНОЇ СИСТЕМИ ANDROID, ОСНОВНІ ВИЗНАЧЕННЯ, СТАТИСТИКА

1.1 Операційна система Android

Android — операційна система для мобільних телефонів та планшетних комп'ютерів, створена компанією Google на базі ядра Linux. Підтримується альянсом Open Handset Alliance (OHA).

Хоча Android базується на ядрі Linux, він стоїть дещо осторонь Linux-спільноти та Linux-інфраструктури. Базовим елементом цієї операційної системи є реалізація Dalvik віртуальної машини Java, і все програмне забезпечення і застосування спираються на цю реалізацію Java.

Лі́нукс (англ. Linux, повна назва — GNU/Linux) — загальна назва UNIX-подібних операційних систем на основі однойменного ядра. Це один із найвидатніших прикладів розробки вільного (free) та відкритого (з відкритим кодом, open source) програмного забезпечення (software). На відміну від власницьких операційних систем (на кшталт Microsoft Windows та MacOS X), їх вихідні коди доступні усім для використання, зміни та розповсюдження абсолютно вільно (в тому числі безкоштовно).

Ядро Лінукс спочатку проектувалося для мікропроцесорів Intel 80386, однак, наразі підтримує чималу кількість комп'ютерних архітектур. Лінукс входить до списку операційних систем, котрі працюють на найбільшій кількості архітектур — від кишенькових комп'ютерів iPAQ на основі ARM до мейнфреймів, на кшталт IBM System z9.

Ядро Linux — ядро UNIX-подібної операційної системи. Розповсюджується під ліцензією GNU General Public License (GPL), і розробляється людьми з усього світу, що дозволило йому стати одним із найвидатніших прикладів відкритого програмного забезпечення та увійти до числа наймасштабніших проектів з розробки програмного забезпечення: версія 4.5 мала 21 млн рядків вихідного коду, а за 2015 рік до роботи над ним долучилось близько чотирьох тисяч розробників та понад 440 різних організацій.

Ядро́ (англ. Kernel) — центральна частина операційної системи, що реалізує інтерфейс між прикладними процесами та обладнанням комп'ютера. Завантажується в оперативну пам'ять комп'ютера і безпосередньо взаємодіє з апаратурою, забезпечуючи керування апаратними засобами (при цьому використовуються драйвери (модулі ядра) підключеного в систему обладнання), підтримку одночасної роботи багатьох користувачів (багатокористувацький режим), підтримку паралельного виконання багатьох процесів в системі (багатозадачність). Зазвичай ядро робить ці об'єкти доступними для прикладних процесів через механізми міжпроцесної взаємодії і системних викликів (рисунок 1.1).

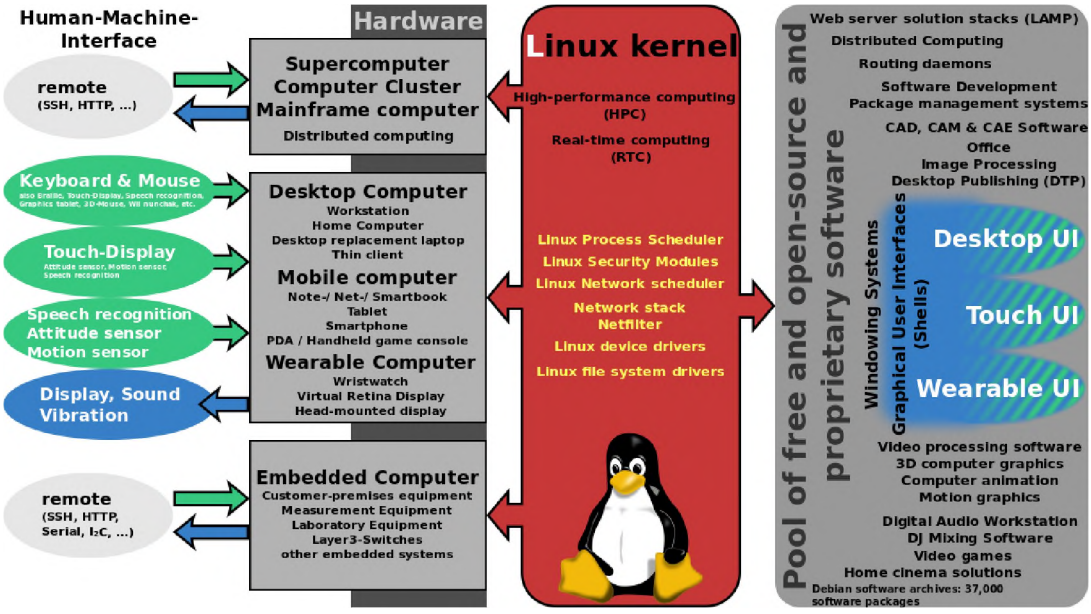

Рисунок 1.1 – Схема основних функцій ядра Linux

1.2 Android та Linux. Архітектура та основні компоненти.

Android використовує ядро Linux. Оскільки Linux має відкритий вихідний код, розробники Android з компанії Google мали можливість модифікувати ядро Linux під свої потреби. Linux надає розробникам Android для початку попередньо зібране і вже підтримуване ядро операційної системи з тим, щоб їм не потрібно було писати своє власне ядро. Це той метод, за допомогою якого було побудовано багато різних пристроїв, наприклад, в PlayStation 4 використовується ядро

FreeBSD з відкритим вихідним кодом, тоді як в Xbox 1 використовує ядро Windows NT, яке можна знайти в сучасних версіях Windows.

Ви навіть побачите версію ядра Linux, яка працює на вашому пристрої, в пункті меню About phone (Про телефоні) або About tablet (Про планшет) в меню Android's Settings (Рисунок 1.2).

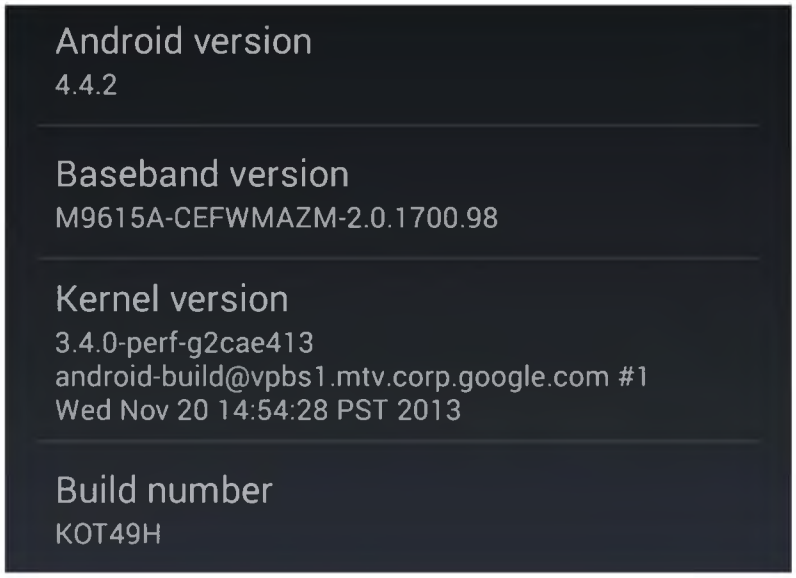

Рисунок 1.2 – Версія ядра Linux відображена в ОС Android

Коли ви включаєте пристрій з Android, ядро Linux завантажується так само, як це було б в дистрибутиві. Проте, велика частина решти програмного забезпечення різниться. В Android не входить бібліотека GNU C Library (glibc), яка використовується в стандартних дистрибутивах Linux, а також не входять всі ті бібліотеки GNU, які є в типовому дистрибутиві Linux.

Замість того, щоб запускати типові програми Linux, Android використовує віртуальну машину Dalvik виключно для того, щоб запускати застосунки, написані на мові Java. Ці застосунки орієнтовані на пристрої Android і інтерфейси прикладного програмування (API), які представлені в Android, а не на Linux в цілому.

Структура операційної системи Android або Android стек зображені на Рисунку 1.3.

Рисунок 1.3 – Android стек

Linux Kernel. Основою платформи Android є ядро Linux. Наприклад, Runtime Android (ART) покладається на ядро Linux для базових функціональних можливостей, таких як керування потоками і управління низькими рівнями пам'яті.

Використання ядра Linux дозволяє Android скористатися перевагами ключових функцій безпеки і дозволяє виробникам пристроїв розробляти драйвери для драйверів для відомого ядра.

Hardware Abstraction Layer (HAL). Апаратний абстрактний рівень (HAL) забезпечує стандартні інтерфейси, які виявляють можливості апаратного забезпечення апаратного забезпечення на рівні Java API. HAL складається з декількох бібліотечних модулів, кожен з яких реалізує інтерфейс для певного типу апаратного компонента, такого як камера або модуль Bluetooth. Коли API-структура робить виклик для доступу до апаратного забезпечення пристрою, система Android завантажує модуль бібліотеки для цього апаратного компонента.

Android Runtime. Для пристроїв під керуванням Android версії 5.0 (API-рівень 21) або новішої версії кожна програма працює у власному процесі та з власним екземпляром Android Runtime (ART). APT написаний для запуску декількох віртуальних машин на пристроях з малою пам'яттю, виконуючи файли DEX, формат байт-коду, розроблений спеціально для Android, оптимізований для мінімального відстані пам'яті. Побудовані інструментальні ланцюжки, такі як Джек, складають джерела Java у байт-код DEX, який може працювати на платформі Android.

Деякі з основних особливостей APT включають в себе наступне:

Попередня (AOT) і просто-в-час (JIT) компіляція

Оптимальне збирання сміття (GC)

У Android 9 (API-версії 28) і вище конвертація файлів формату Executable (DEX) для пакету додатків до більш компактного машинного коду.

Покращена підтримка налагодження, включаючи спеціальний профілі відбору проб, детальні діагностичні винятки та звіти про аварійне завершення роботи, а також можливість встановлювати точки спостереження для відстеження певних полів.

До Android версії 5.0 (рівень API 21), Dalvik був робочою версією Android. Якщо ваш додаток добре працює на APT, тоді він повинен працювати також і на Дальвіку, але навпаки, може бути неправда.

Android також включає в себе набір основних бібліотек виконання, які забезпечують більшість функціональних можливостей мови програмування Java, включаючи деякі мовні функції Java 8, які використовує рамки Java API.

Native C/C++ Libraries. Багато основних компонентів і сервісів Android, таких як ART та HAL, будуються з власного коду, що вимагає власних бібліотек, написаних у C і C ++. Платформа Android підтримує Java API-інтерфейси для виявлення функціональності деяких з цих нативних бібліотек для додатків. Наприклад, ви можете отримати доступ до OpenGL ES через API Java OpenGL для Android, щоб додати підтримку для малювання та маніпулювання 2D та 3D-графікою у вашому додатку.

Якщо ви розробляєте додаток, для якого потрібен код C або C ++, ви можете використовувати Android NDK, щоб отримати доступ до деяких цих бібліотек нативної платформи безпосередньо з вашого власного коду.

Java API Framework. Весь набір функцій ОС Android доступний вам за допомогою API, написаних на мові Java. Ці API являють собою будівельні блоки, необхідні для створення додатків для Android, шляхом спрощення повторного використання основних, модульних компонентів системи та служб, до яких належать наступні:

Багата та розширювана система перегляду, яку ви можете використовувати для створення інтерфейсу додатка, включаючи списки, сітки, текстові поля, кнопки та навіть вставлений веб-браузер

Менеджер ресурсів, що забезпечує доступ до ресурсів без кодів, таких як локалізовані рядки, графічні файли та файли макета

Менеджер сповіщень, який дозволяє всім програмам відображати власні сповіщення в рядку стану

Менеджер дій, який керує життєвим циклом додатків і забезпечує загальний стек навигації назад

Постачальники вмісту, які дозволяють додаткам отримувати доступ до даних з інших програм, таких як програма "Контакти", або для обміну власними даними

Розробники мають повний доступ до тих самих базових API, які використовують системні застосунки Android.

System Apps. Android постачається з набором основних програм для електронної пошти, SMS-повідомлень, календарів, перегляду через Інтернет, контактів тощо. Додатки, включені до платформи, не мають спеціального статусу серед програм, які користувач вибирає для встановлення. Таким чином сторонні програми можуть стати веб-браузером за замовчуванням, SMS-повідомленнями або навіть клавіатурою за замовчуванням (застосовуються деякі винятки, наприклад, додаток "Налаштування системи").

Системні застосунки функціонують як додатки для користувачів, так і для забезпечення ключових можливостей, доступних розробникам за допомогою власного додатка. Наприклад, якщо ваш додаток хотів би доставити SMS-повідомлення, вам не потрібно самостійно будувати цю функцію - замість цього ви можете запустити будь-яку SMS-програму, яка вже встановлена, щоб доставити повідомлення одержувачу, який ви вказали.

1.3 CC

Загальні критерії оцінки безпеки інформаційних технологій (далі - Common Criteria або CC) є міжнародним стандартом (ISO / IEC 15408) для сертифікації комп'ютерної безпеки.

Загальні критерії є основою, в якій користувачі комп'ютерної системи можуть визначати свої функціональні вимоги щодо безпеки та вимоги забезпечення (відповідно SFRs та SARs) у цільовому забезпеченні безпеки (ST) і можуть бути взяті з профілів захисту (PP). Постачальники можуть потім реалізувати або подавати заяви про атрибути безпеки своєї продукції, а тестові лабораторії можуть оцінити продукти, щоб визначити, чи дійсно вони відповідають вимогам. Іншими словами, Загальні критерії забезпечують переконання, що процес специфікації, впровадження та оцінки продукту комп'ютерної безпеки був проведений суворо, стандартно та повторюваним

чином на рівні, відповідному цільовій середовищі для використання. Категорії продуктів та кількість зображена на Рисунку 1.4.

2499 Certified Products by Category *		
Category	**Products**	**Archived**
Access Control Devices and Systems	69	60
Biometric Systems and Devices	3	0
Boundary Protection Devices and Systems	77	124
Data Protection	70	92
Databases	30	53
Detection Devices and Systems	12	57
ICs, Smart Cards and Smart Card-Related Devices and Systems	1194	33
Key Management Systems	22	28
Mobility	31	19
Multi-Function Devices	193	182
Network and Network-Related Devices and Systems	257	234
Operating Systems	104	74
Other Devices and Systems	298	316
Products for Digital Signatures	102	8
Trusted Computing	37	0
Totals:	**2499**	**1280**
Grand Total:		**3779**

** A Certified Product may have multiple Categories associated with it.*

Рисунок 1.4 – Типи сертифікованих продуктів

Ключові поняття:

Оцінки загальних критеріїв виконуються на продуктах та системах комп'ютерної безпеки.

Цільова оцінка (TOE) - продукт або система, що є предметом оцінки. Оцінка служить для підтвердження претензій, поданих щодо цілі. Для практичного використання, оцінка повинна перевірити функції безпеки цілі. Це робиться за допомогою наступного:

- Профіль захисту (PP) - це документ, який зазвичай створюється користувачем або спільнотою користувачів, який визначає вимоги до безпеки для класу пристроїв безпеки (наприклад, смарт-картки, що використовуються для забезпечення цифрових підписів або мережевих брандмауерів), що мають відношення до цього користувача для особлива

мета. Виробники продуктів можуть вибрати для реалізації продукти, що відповідають одному або декількох ПП, і оцінити їх продукцію щодо цих ПП. У такому випадку PP може слугувати шаблоном ST для продукту (Security Target, як визначено нижче), або автори ST будуть принаймні забезпечувати, щоб усі вимоги у відповідних PP також містились у документі ST цілі. Клієнти, які шукають певні типи продуктів, можуть зосередитись на тих, хто сертифікується за ПП, що відповідає їх вимогам.

- Цільова ціль (ST) - документ, який визначає властивості безпеки цілі оцінки. ST може претендувати на відповідність одному або декількох ПП. Оцінка TOE оцінюється відповідно до SFR (Функціональні вимоги до безпеки, знову ж таки, див. Нижче), встановлені в його ST, не більше і не менше. Це дозволяє постачальникам адаптувати оцінку до точного відповідності передбачуваним можливостям їх продукту. Це означає, що мережевий брандмауер не повинен відповідати однаковим функціональним вимогам, подібним до системи керування базами даних, і що різні брандмауери насправді можуть бути оцінені зовсім іншими списками вимог. Звичайно публікується ST, щоб потенційні клієнти могли визначати специфічні функції безпеки, які були сертифіковані за оцінкою.
- Функціональні вимоги безпеки (SFR) - вкажіть окремі функції безпеки, які можуть бути надані продуктом. Загальні критерії представляють стандартний каталог таких функцій. Наприклад, SFR може вказати, як користувач може виконати автентифікацію. Список СФР може варіюватися від однієї оцінки до наступної, навіть якщо дві цілі є одним і тим же видом продукту. Незважаючи на те, що загальні критерії не передбачають, що будь-які SFR будуть включені в ST, він визначає залежності, в яких правильна робота однієї функції (наприклад, можливість обмежувати доступ відповідно до ролей) залежить від іншої (наприклад, можливість ідентифікувати окремі ролі)

Процес оцінки також намагається встановити рівень довіри, який може бути поміщений у ознаки безпеки продукту через процеси забезпечення якості:

- Вимоги щодо забезпечення безпеки (SAR) - опис заходів, які вживаються під час розробки та оцінки продукту, щоб забезпечити відповідність заявленим функціям безпеки. Наприклад, оцінка може вимагати, щоб весь вихідний код зберігався в системі управління змінами або що повне функціональне тестування виконувалося. Загальні критерії наводять каталог цих, і вимоги можуть відрізнятися від однієї оцінки до наступної. Вимоги щодо конкретних цілей або видів продукції задокументовані відповідно в ST та PP.
- Рівень забезпечення оцінки (EAL) - числовий рейтинг, що описує глибину та суворість оцінки. Кожен EAL відповідає пакету вимог забезпечення безпеки (SARs, див. Вище), який охоплює повний розвиток продукту з певним рівнем строгості. Загальні критерії наводяться на семи рівнях, причому EAL 1 є найбільш базовим (і тому найдешевшим для впровадження та оцінки), а EAL 7 є найсуворішим (і найдорожчим). Зазвичай, ST або PP автор не буде вибирати вимоги до страхування індивідуально, але вибирати один з цих пакетів, можливо, "збільшувати" вимоги в декількох областях з вимогами з більш високого рівня. Вищі стандарти EAL не обов'язково означають "кращу безпеку", вони лише означають, що заявлена гарантія безпеки OE була більш повно перевірена.

Статистика сертифікації за рівнем EAL зображена на рисунку 1.5.

Certified Products by Assurance Level and Certification Date																				
EAL	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	Total
EAL1	0	0	0	0	0	0	1	1	6	3	1	0	1	10	2	2	3	3	8	6	47
EAL1+	1	0	0	0	0	0	0	0	17	0	2	11	2	0	1	2	1	0	0	1	38
EAL2	0	0	0	0	0	0	1	0	8	1	7	2	3	1	10	12	18	15	23	10	111
EAL2+	0	0	0	1	1	1	2	2	8	8	8	4	5	10	8	27	59	76	66	39	325
EAL3	0	0	0	0	0	0	0	0	10	3	1	9	5	1	7	12	9	2	3	2	64
EAL3+	0	0	0	0	0	2	1	1	37	10	12	11	12	19	5	23	17	19	10	9	188
EAL4	0	1	0	1	0	0	0	0	28	5	9	4	6	2	7	2	0	5	2	8	80
EAL4+	0	1	1	2	2	3	3	2	142	58	66	56	60	87	62	51	56	56	52	36	796
EAL5	0	0	0	0	0	0	0	0	6	3	2	0	1	0	0	0	0	3	1	3	19
EAL5+	0	0	0	0	0	0	3	0	50	27	31	43	35	27	56	51	43	69	68	47	550
EAL6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
EAL6+	0	0	0	0	0	0	0	0	0	0	2	3	0	4	5	6	10	8	12	20	70
EAL7	0	0	0	0	0	0	0	0	0	0	1	0	0	0	4	0	0	0	0	0	5
EAL7+	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	2
Basic	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Medium	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
US Standard	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
None	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	8	13	20	79	80	204
Totals:	**1**	**2**	**1**	**4**	**3**	**6**	**11**	**6**	**312**	**118**	**142**	**144**	**130**	**161**	**171**	**196**	**229**	**276**	**324**	**262**	**2499**

Рисунок 1.5 – Сертифікація за рівнями забезпечення оцінки

Поки що більшість PP та найбільш оцінюваних сертифікованих продуктів / сертифікованих продуктів застосовуються для IT-компонентів (наприклад, міжмережевих екранів, операційних систем, смарт-карт). Інтегрована сертифікація Common Criteria іноді визначена для закупівлі IT. Інші стандарти, що містять, наприклад, взаємодію, керування системою, навчання користувачам, доповнення CC та інших стандартів продукту. Приклади включають в себе ISO / IEC 17799 (або більш належним чином BS 7799-1, який зараз є ISO / IEC 27002) або німецьким IT-Grundschutzhandbuch.

Деталі криптографічного впровадження в рамках ОУ виходять за межі СК. Натомість, національні стандарти, такі як FIPS 140-2, дають специфікації для криптографічних модулів, а різні стандарти визначають використовувані криптографічні алгоритми.

Зовсім недавно автори ПР включають в себе криптографічні вимоги до оцінок CC, які, як правило, охоплюються оцінками FIPS 140-2, розширюючи межі ЦК за допомогою специфічних інтерпретацій схем.

Деякі національні схеми оцінки поступово скасовують оцінку на базі EAL і приймають лише продукти для оцінки, що вимагають суворого відповідності затвердженому ПП. Сполучені Штати наразі дозволяють проводити оцінку лише на основі ПП. Канада намагається поступово скасувати оцінки на базі EAL.

Організації, що проводять тестування:

Всі випробувальні лабораторії повинні відповідати вимогам стандарту ISO 17025, а органи сертифікації, як правило, повинні бути затверджені відповідно до Довідника ISO / IEC 65 або BS EN 45011.

Відповідність стандарту ISO 17025, як правило, демонструється національному органу, що підтверджує:

- У Канаді Рада стандартів Канади (SCC) в рамках Програми акредитації лабораторій (PALCAN) акредитує об'єкти оцінки загальних критеріїв (CCEF)
- У Франції Comité français d'accréditation (COFRAC) акредитує об'єкти оцінки загальних критеріїв, часто називають Центром оцінювання інформаційних технологій інформації (CESTI). Оцінки проводяться відповідно до норм та стандартів, визначених Національною інформаційною системою (ANSSI).
- У Великобританії Служба акредитації Великобританії (UKAS) акредитує Комерційну цільову оцінку (CLEF)
- У США Національна програма акредитації добровільних лабораторій (HICT) Національного інституту стандартів і технологій (HICT) акредитує лабораторії випробувань спільних критеріїв (CCTL)
- У Німеччині, Бундесштамп für Sicherheit in der Informationstechnik (BSI)
- У Іспанії Національний криптологічний центр (CCN) акредитує лабораторії спільних критеріїв тестування, які працюють на іспанській схемі.
- У Нідерландах схема сертифікації Нідерландів в галузі безпеки IT (NSCIB) акредитує засоби оцінювання безпеки інформаційних технологій (ITSEF).

Характеристики цих організацій були розглянуті та представлені на ICCC 10.

Кількість сертифікованих продуктів за рівнями EAL та країною, що здійснює сертифікацію зображено на рисунку 1.6

Certified Products by Scheme and Assurance Level																		
Scheme	**EAL1**	**EAL1+**	**EAL2**	**EAL2+**	**EAL3**	**EAL3+**	**EAL4**	**EAL4+**	**EAL5**	**EAL5+**	**EAL6**	**EAL6+**	**EAL7**	**EAL7+**	**B**	**M**	**S**	**N**	**Total**
Australia	2	1	10	9	2	3	5	12	0	0	0	0	1	0	0	0	0	22	67
Canada	3	0	9	118	0	0	0	3	0	0	0	0	0	0	0	0	0	41	174
Germany	10	4	13	27	14	63	17	331	8	184	0	40	0	0	0	0	0	8	719
Spain	9	8	7	14	4	13	0	36	0	8	0	0	0	0	0	0	0	5	104
France	1	18	1	15	0	41	5	292	5	315	0	18	4	0	0	0	0	0	715
India	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	3
Italy	5	7	1	1	2	0	1	19	0	0	0	0	0	0	0	0	0	0	36
Japan	0	0	7	87	19	30	0	0	0	0	0	0	0	0	0	0	0	9	152
Republic of Korea	7	0	5	9	8	14	24	15	0	15	0	0	0	0	0	0	0	4	101
Malaysia	6	0	24	10	0	4	1	3	0	0	0	0	0	0	0	0	0	0	48
Netherlands	0	0	5	1	1	1	1	21	0	14	0	12	0	2	0	0	0	1	59
Norway	0	0	2	25	2	12	19	21	3	11	0	0	0	0	0	0	0	0	95
New Zealand	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Sweden	4	0	12	2	7	4	6	5	3	0	0	0	0	0	0	0	0	2	45
Turkey	0	0	10	1	3	0	1	11	0	0	0	0	0	0	0	0	0	0	26
United Kingdom	0	0	4	6	1	3	0	26	0	3	0	0	0	0	0	0	0	2	45
United States	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	110	110
Totals:	**47**	**38**	**111**	**325**	**64**	**188**	**80**	**796**	**19**	**550**	**0**	**70**	**5**	**2**	**0**	**0**	**0**	**204**	**2499**

Рисунок 1.6 – Сертифікація по країнах світу

1.4 MDFPP

Оглял TOE. Цей стандарт забезпечення безпеки визначає вимоги до інформаційної безпеки для мобільних пристроїв для використовувати на підприємстві. Мобільний пристрій в контексті цього стандарту забезпечення - це пристрій який складається з апаратної платформи та її системного програмного забезпечення. Пристрій зазвичай забезпечує бездротовий зв'язок і може включати в себе програмне забезпечення для таких функцій, як захищений обмін повідомленнями, електронна пошта, Інтернет, VPN-зв'язок та VoIP (Voice over IP) для доступу до захищеного підприємства мережеві, корпоративні дані та програми, а також для спілкування з іншими мобільними пристроями. Мережа мобільного пристрою та його оточення зображено на рисунку 1.7

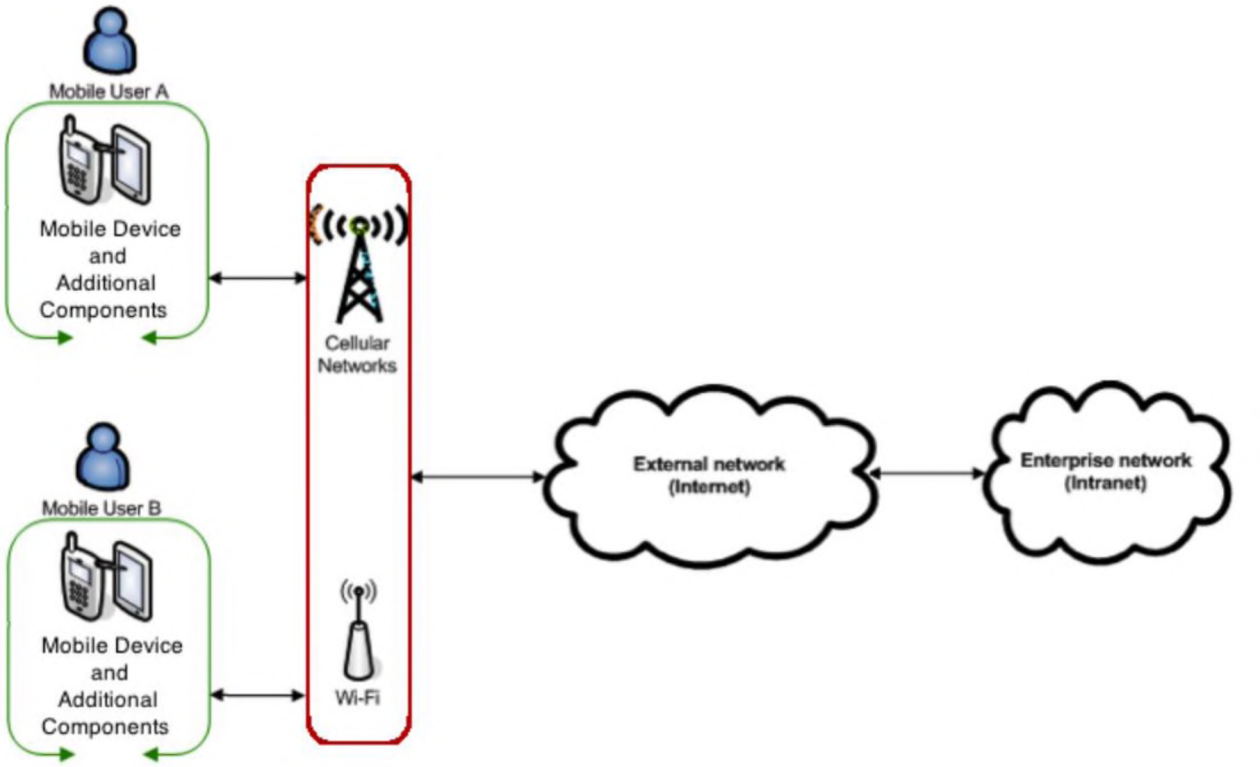

Рисунок 1.7 – Мережа мобільного пристрою

Приклади "мобільного пристрою", який повинен вимагати відповідність цьому профілю захисту, включають смартфони, планшетні комп'ютери та інші мобільні пристрої з аналогічними можливостями.

Мобільний пристрій надає основні послуги, такі як криптографічні послуги, захист від передачі даних та послуги зберігання ключів для підтримки безпечної роботи додатків на пристрої. Додаткові функції безпеки, такі як дотримання політики безпеки, обов'язкового контролю доступу до програми, функції антизапровадження, автентифікація користувачів та захист цілісності програм, використовуються для усунення загроз.

Цей стандарт забезпечення описує ці важливі служби безпеки, що надається Мобільним пристроєм, і слугує основою для безпечної мобільної архітектури. Як показано на рисунку 1.8, очікується, що типовий розгортання також включатиме як сторонні або комплектуючі компоненти, які забезпечують:

- Захист даних в транзиті (наприклад, клієнт VPN, клієнт VoIP, веб-браузер)
- Управління політикою безпеки (наприклад, система MDM)

Незалежно від того, що ці компоненти входять до складу мобільного пристрою виробником або розроблені сторонніми особами, вони повинні бути окремо перевірені відповідно до відповідної гарантії стандарти. Додаткові застосунки, які можуть бути попередньо встановлені на мобільному пристрої та не перевірені, вважаються потенційно помилковими, але не зловмисними. Приклади включають в себе VoIP-клієнта, поштового клієнта та веб-браузера.

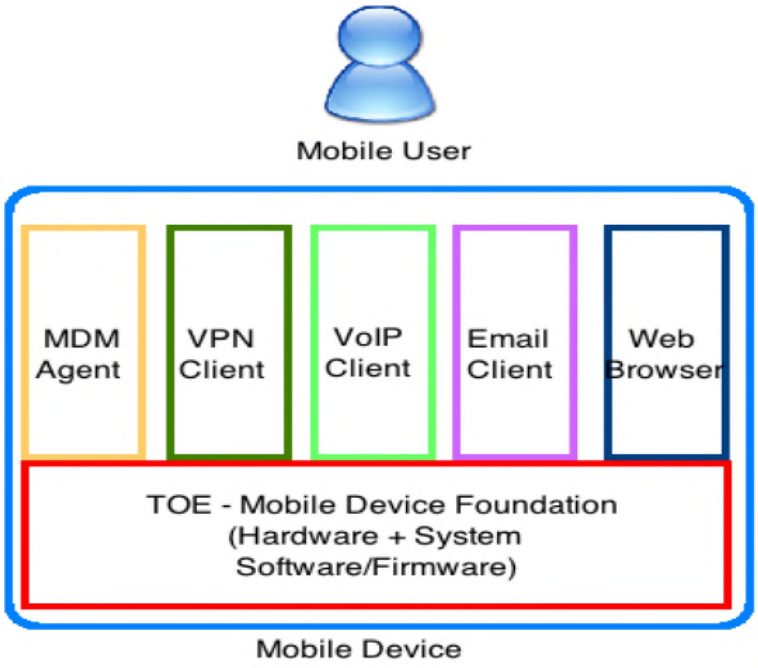

Рисунок 1.8 – Модель мобільного пристрою

TOE використання. Мобільний пристрій може працювати в ряді випадків використання. На додаток до надання важливих служб безпеки, мобільний пристрій містить необхідні функції безпеки для підтримки конфігурації для різних випадків використання. У кожному випадку використання може знадобитися додаткова конфігурація та додатки для досягнення бажаної безпеки. Вибір цих випадків використання наведено нижче.

1. Приватне підприємство, яке використовується для загального користування та обмежене для особистого користування

Пристрій, що належить підприємству для загального користування, передбачає значну ступінь контролю підприємства над конфігурацією та, можливо, інвентаризації програмного забезпечення. Підприємство вибирає надання

користувачам мобільних пристроїв та додаткових додатків (таких як VPN або поштові клієнти), щоб зберегти контроль над їхніми підприємствами та безпекою їхніх мереж.

Користувачі можуть використовувати Інтернет-з'єднання для перегляду веб-сторінок або доступу до корпоративної пошти або для запуску корпоративних програм, але це з'єднання може значно контролюватися підприємством.

2. Підприємство, що належить компанії, призначене для спеціалізованого, високобезпечного використання

Пристрій, що належить підприємству, з обмеженим наміром мережевим з'єднанням, жорстко контрольованою конфігурацією та обмеженим інвентарем програмного забезпечення, підходить для спеціалізованих, високобезпечних використання випадків Наприклад, пристрій може не допускати підключення до будь-яких зовнішніх периферійних пристроїв. Він може мати можливість спілкуватися лише через свої WiFi або стільникові радіоприймачі з мережею, що працює під керуванням підприємства, яка навіть не може дозволити підключення до Інтернету. Використання пристрою може спричинити дотримання політики, яка не буде вважатись реалістичною у будь-якому випадку загального призначення, але може пом'якшити ризики надзвичайно конфіденційної інформації. Як і в попередньому випадку, підприємство буде шукати додаткові програми, що забезпечують підключення до корпоративної мережі та послуги, що мають такий же рівень впевненості, що і платформа.

3. Пристрій для особистого та корпоративного використання

Пристрій, що використовується особисто і для особистої діяльності, і для корпоративних даних, зазвичай називається Bring Your Own Device (BYOD). На відміну від справ, що належать підприємствам, підприємство обмежене тим, до яких політик безпеки він може застосувати, оскільки користувач придбав пристрій в першу чергу для особистого користування і навряд чи прийматиме політику, що обмежує функціональність пристрою. Однак, оскільки підприємство дозволяє користувачеві повний (або майже повний) доступ до корпоративної

мережі, підприємству буде потрібна певна політика безпеки, наприклад, політика щодо пароля або екрану, і може вимагати гарантованого корпоративного програмного забезпечення, наприклад клієнта VPN, до надання доступу. Пристрій може бути забезпечений доступом до ресурсів підприємства після значного особистого використання.

4. Пристрій для особистого і обмеженого використання

Особистому пристрою також може надаватися доступ до обмежених служб підприємства, таких як корпоративне електронне повідомлення. Оскільки користувач не має повного доступу до даних підприємства або підприємства, підприємству, можливо, не потрібно застосовувати будь-які політики безпеки на пристрої. Однак підприємство може захотіти захищувати електронну пошту та веб-переглядач із запевненням, що послуги, що надаються цим клієнтам мобільним пристроєм, не порушені.

1.5 Визначення проблеми безпеки.

Загрози. Мобільні пристрої піддаються загрозі традиційних комп'ютерних систем разом із тими, що пов'язані з їх мобільністю. Загрози, розглянуті в цьому профілі захисту, - це підслухування мережі, мережеві атаки, фізичний доступ і шкідливі або неякісні програми, як це детально описано в наступних розділах.

- Network Eavesdropping (Підслуховування мережі)

Зловмисник розташовується на бездротовому каналі зв'язку або іншому місці в інфраструктурі мережі. Зловмисники можуть контролювати та отримувати доступ до даних, обмінюваних між Мобільним пристроєм та іншими кінцевими точками.

- Network Attack (Мережева атака)

Зловмисник розташовується на бездротовому каналі зв'язку або в іншому місці на мережевій інфраструктурі. Зловмисники можуть ініціювати зв'язок із мобільним пристроєм або змінювати комунікації між мобільним пристроєм та іншими кінцевими точками, щоб поставити під сумнів мобільний пристрій. Ці атаки включають в себе оновлення шкідливих програм будь-яких програм або системного програмного забезпечення на пристрої. Ці атаки також включають

шкідливі веб-сторінки або вкладення електронної пошти, які зазвичай надходять на пристрої через мережу.

- Physical Access (Фізичний доступ)

Втрата або крадіжка мобільного пристрою може призвести до втрати конфіденційності користувацьких даних, включаючи облікові дані. Ці загрози фізичного доступу можуть включати в себе атаки, які намагаються це зробити

доступ до пристрою через зовнішні апаратні порти, через його інтерфейс користувача, а також через прямий і, можливо, деструктивний доступ до його носія. Мета таких атак - це доступ до даних із втраченого або викраденого пристрою, який, як очікується, не повернеться користувачеві.

- Malicious or Flawed Application

Програми, завантажені на мобільний пристрій, можуть включати в себе шкідливий чи програмний код. Цей код може бути навмисно включений його розробником або невідомого розробником, можливо, як частина бібліотеки програмного забезпечення. Шкідливі програми можуть намагатися відфільтрувати дані, до яких вони мають доступ. Вони також можуть здійснювати атаки на системне програмне забезпечення платформи що надасть їм додаткові привілеї та можливість проводити подальшу шкідливу діяльність. Шкідливі програми можуть керувати датчиками пристрою (GPS, камерою, мікрофоном) для збору інформації про оточення користувача, навіть якщо ці дії не включають дані, резидентні або передані з пристрою. Неправильні програми може дозволити злодію отримати доступ до мережевих або фізичних атак, які інакше могли б бути запобігти.

- Persistent Access (Постійний доступ)

Постійний доступ до пристрою зловмисником означає, що пристрій втратив цілісність і не зможе повернути його. Можливо, цей пристрій втратив цю цілісність через інший вектор загрози постійний доступ зловмисника представляє собою постійну загрозу сама по собі. У цьому випадку пристрій та його дані можуть контролюватися як супротивником, так і його законним власником.

1.6 Цілі безпеки

- Protected Communications (Захищені комунікації)

Для подолання мережевого підслуховування та загрози мережевої атаки, пов'язаної з бездротовою передачею даних Enterprise та даних користувачів та даних конфігурації між TOE та об'єктами віддаленої мережі, відповідні TOE використовуватимуть надійний шлях зв'язку. TOE буде здатний здійснювати зв'язок за допомогою одного (або більше) цих стандартних протоколів: IPsec, DTLS, TLS або HTTPS. Протоколи визначаються RFC, що пропонують різні варіанти реалізації. Вимоги були накладені на деякі з них вибір (особливо для криптографічних примітивів) для забезпечення оперативної сумісності та стійкості до криптографічної атаки.

Хоча відповідні TOE повинні підтримувати всі варіанти, вказані в ST, вони можуть підтримувати додаткові алгоритми та протоколи. Якщо такі додаткові механізми не оцінюються, адміністратору слід надати керівництво, щоб чітко вказати на те, що вони не були оцінені.

- Protected Storage

Щоб вирішити проблему втрати конфіденційності користувацьких даних у разі втрати мобільного пристрою (T.PHYSICAL), відповідні TOE використовуватимуть захист «data-at-rest». TOE буде здатний шифрувати дані та ключі, що зберігаються на пристрої, і запобігатимуть несанкціонований доступ до зашифрованих даних.

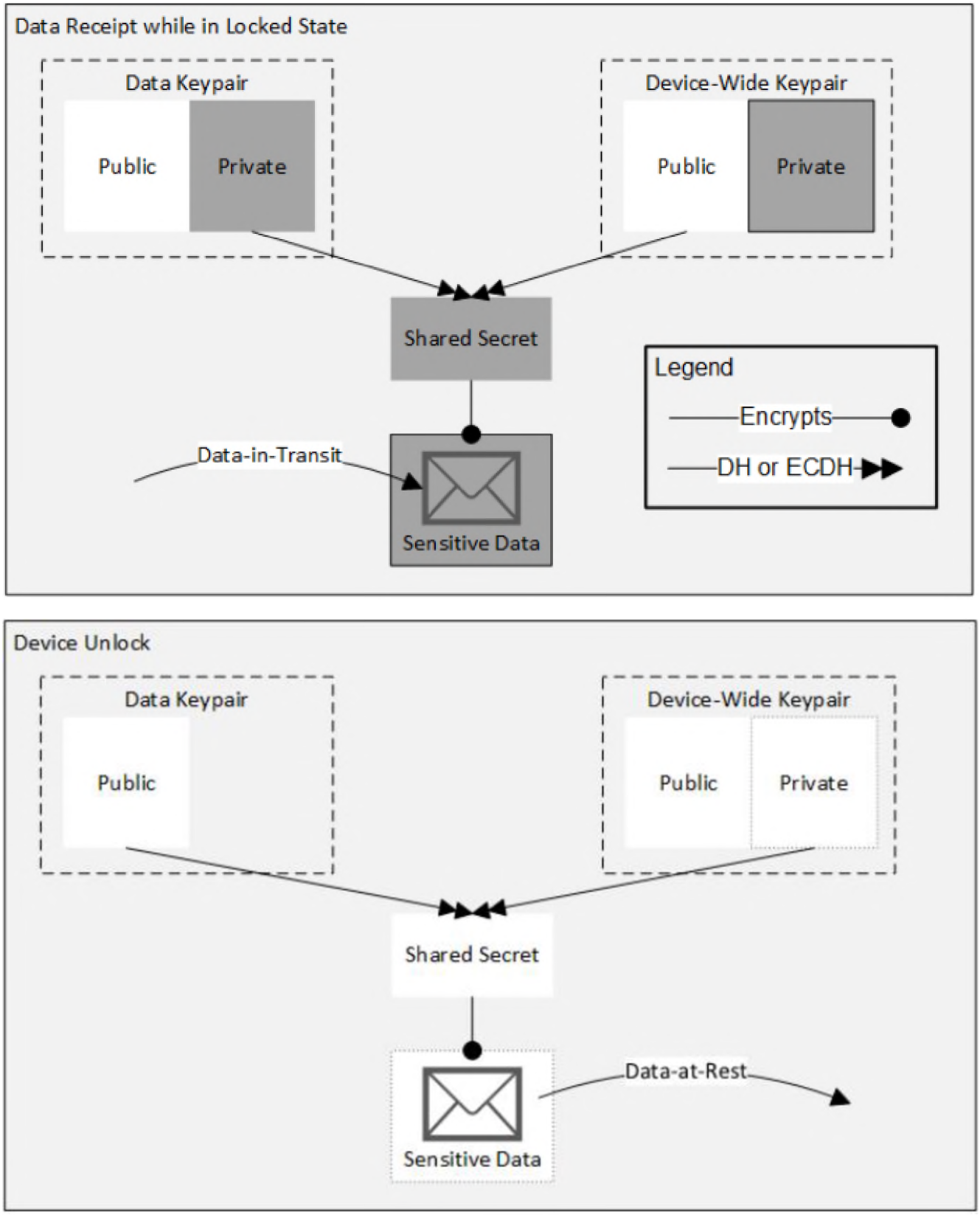

Рисунок 1.9 - Схема основної угоди для шифрування отриманих конфіденційних даних у заблокованому стані

- Mobile Device Configuration (Конфігурація мобільного пристрою)

Щоб мобільний пристрій захищав дані користувача та підприємства, які він може зберігати чи обробляти, відповідні TOE забезпечать можливість конфігурування та застосування політики безпеки, визначеної користувачем та адміністратором підприємства. Якщо політика безпеки корпорації налаштовується, вони мають застосовуватися у пріоритеті до правил, встановлених користувачем.

- Authorization and Authentication (Авторизація та автентифікація)

Щоб вирішити проблему втрати конфіденційності користувацьких даних у разі втрати мобільного пристрою (T.PHYSICAL), користувачі повинні ввести коефіцієнт автентифікації на пристрій перед

для доступу до захищених функцій та даних. Перед введенням коефіцієнта автентифікації можна отримати доступ до деяких нечутливих функцій (наприклад, екстрених викликів, текстових повідомлень). Пристрій буде автоматично блокуватись після певного періоду бездіяльності, намагаючись забезпечити авторизацію у разі втрати або крадіжки пристрою.

Аутентифікація кінцевих точок надійного шляху зв'язку потрібна для доступу до мережі, щоб атаки не могли встановити неавторизовані мережні підключення до

підірвати цілісність пристрою.

Повторні спроби користувача авторизувати TSF будуть обмежені або обмежені, щоб забезпечити затримку між невдалими спробами.

- Mobile Device Integrity (Цілісність мобільного пристрою)

Для забезпечення цілісності мобільного пристрою підтримується відповідність TOE виконувати самотести для забезпечення цілісності критичних функціональних можливостей, програмного забезпечення / вбудованого програмного забезпечення та даних

підтримується Користувач повинен бути повідомлений про будь-які збої цих самотестів. (Це захистить від загрози T.PERSISTENT.)

Щоб вирішити проблему програми з шкідливим або недоліком коду (T.FLAWAPP), цілісність завантажених оновлень програмного забезпечення /

вбудованого програмного забезпечення буде перевірено до встановлення / виконання об'єкта на ОУ. Крім того, операційна система буде обмежувати застосування додатків лише для доступу до системних служб та даних, яким дозволено взаємодіяти з ними. Операційна система надалі захищатиме від шкідливих програм від доступу до даних, яким вони не мають права доступу, шляхом рандомізації макета пам'яті.

Висновки до розділу 1

В даному розділі були розглянуті основні принципи будови операційної системи Android, основні компоненти та визначення ключових понять для даної роботи. Було проаналізовано ту частину загальної інформації про взаємозв'язок Android та Linux, Android kernel та Linux kernel, сертифікаційний процес мобільних пристроїв, його основні визначення та властивості.

Було встановлено, що Android є прямим спадкоємцем Linux, а саме ядро Android представляє собою ядро Linux з деякими змінами, тут варто зазначити, чим довше розвивається Android, тим більше його ядро відрізняється від свого пращура ядра Linux. З такого тісного взаємозв'язка випливає й те, що сертифікація пристроїв на Android може бути частково застосована для мобільних пристроїв під керуванням ОС Linux.

Також була проаналізована статистика сертифікації пристроїв серед країн світу, що показує стрімкий ріст кількості сертифікованих пристроїв та зацікавленість серед країн світу. Даний факт є потужним доказом того, що вимоги СС є вкрай важливі та актуальні.

2 ТАЄМНІ КЛЮЧІ, ВИТОКИ ТА ЇХ АНАЛІЗ

2.1 Таємні ключі

Для детального ознайомлення з терміном таємний ключ, розглянемо реальний випадок.

Модуль Android – Bluetooth(stack), ключ – Link Key.

Link key генерація. Два пристрої, що спілкуються вперше, пройдуть етап ініціалізації; в цуй момент вони будуть "пов'язані". Створення Link key починається, коли користувачі вводять ідентичні PIN-коди на обох пристроях, які використовуються пристроями для створення своїх секретних Link key. Однією з сильних сторін безпеки Bluetooth є те, що в подальших комунікаціях між пристроями Link key ніколи не передається за межі пристрою; Link key просто використовується в криптографічних алгоритмах для створення відповідних послідовностей.

Автентифікація. У Bluetooth, автентифікація досягається за допомогою схеми реагування на challenge-response, метою якої є перевірка того, що пристрій, що запитує доступ, знає Link key ключ. Запитуючий пристрій спершу надсилає свою унікальну адресу до перевіряючого пристрою. Після цього перевіряючий пристрій надсилає 128-бітове випадкове число згенероване генератором випадкових чисел, обидва пристрої використовують алгоритм E1 над, адресами, Link key ключем та випадковим числом, що дає змогу перевіряючому пристрою переконатися, що результати співпадають рисунок .

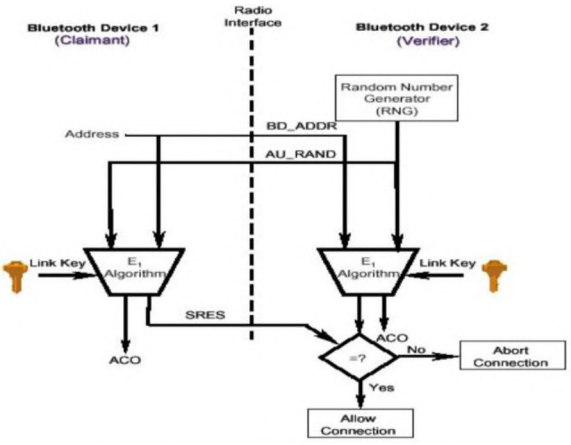

Рисунок 2.1 – Автентифікація в Bluetooth

Конфіденційність. Bluetooth шифрує свої дані за допомогою потокового шифру, що називається E0. Використовуваний ключовий потік створюється за допомогою алгоритму, який приймає такі значення як: адресу пристрою, випадкове число, номер слота та ключ шифрування. Ключ шифрування виробляється від генератора внутрішніх ключів, який приймає на вхід: Link key, випадкове число і значення з процедури автентифікації рисунок 2.2.

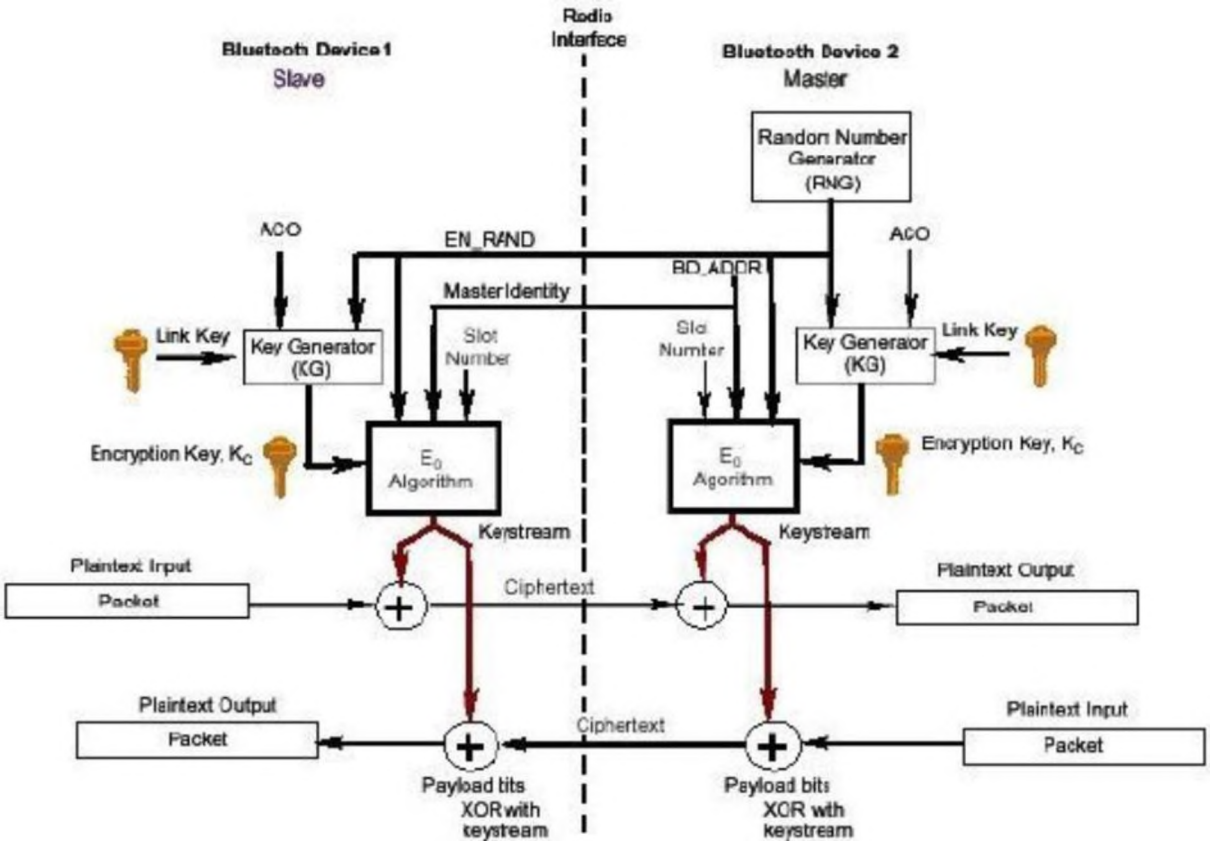

Рисунок 2.2 – Шифрування даних в Bluetooth

Повна схема ієрархії Bluetooth ключів зображена на рисунку 2.3.

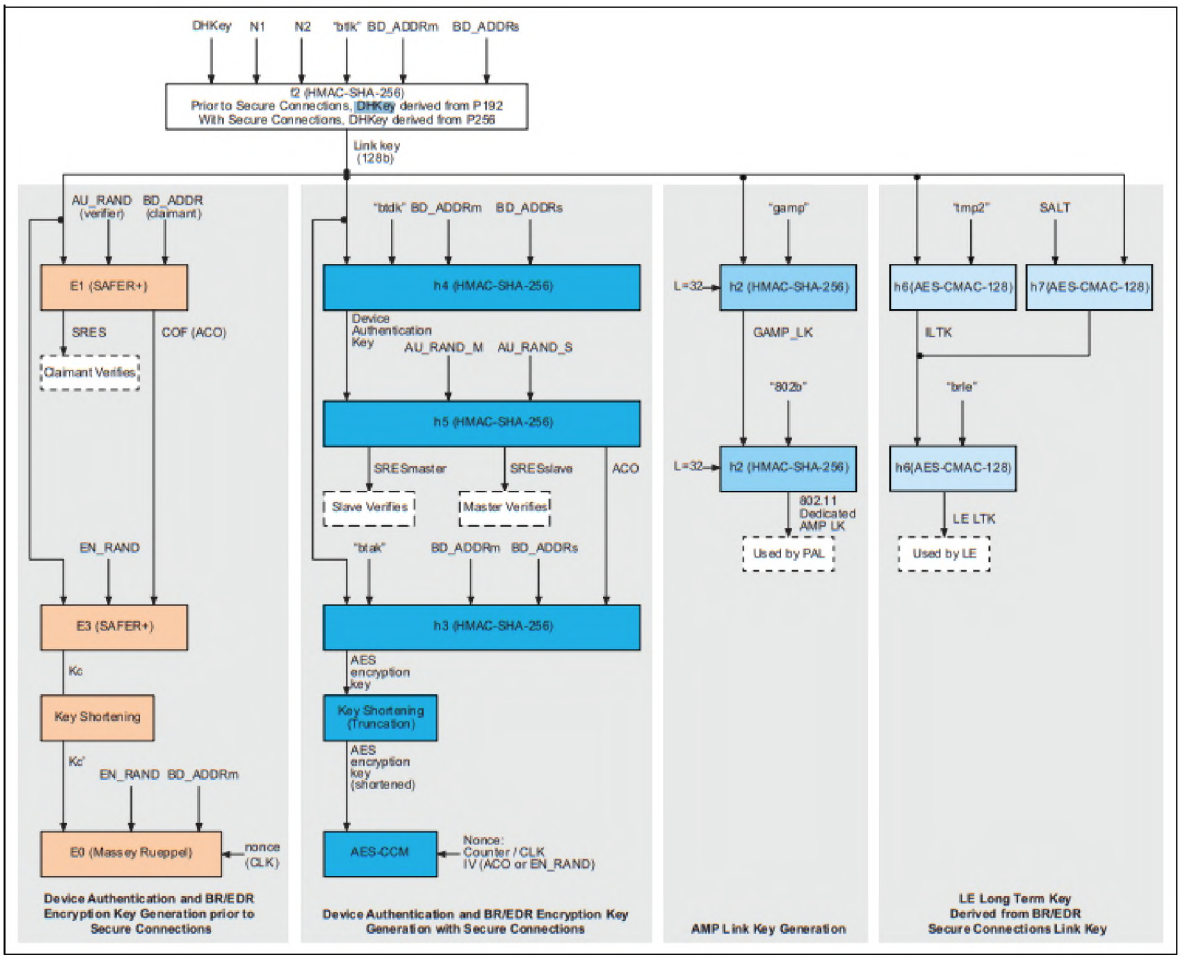

Рисунок 2.3 – Ієрархія ключів Bluetooth

2.2 Витік пам'яті та техніки запобігання

Витік пам'яті. У комп'ютерній техніці витік пам'яті - це тип витоку ресурсу, який виникає, коли комп'ютерна програма неправильно керує розподілом пам'яті таким чином, що пам'ять, яка більше не потрібна, не звільнюється. Витік пам'яті також може статися, коли об'єкт зберігається в пам'яті, але недоступний робочий код. Витік пам'яті має симптоми, подібні ряду інших проблем, і, як правило, можна діагностувати лише програміст, який має доступ до вихідного коду програми.

Проблема з пробілом виникає, коли комп'ютерна програма використовує більше пам'яті, ніж це потрібно. На відміну від витоків пам'яті, де витік пам'яті ніколи не звільняється, пам'ять, споживана витік простору, звільняється, але пізніше, ніж очікувалося.

Оскільки вони можуть вичерпати доступну системну пам'ять при запуску програми, витоки пам'яті найчастіше є причиною або сприяють старінню програмного забезпечення.

Витік пам'яті знижує продуктивність комп'ютера, зменшуючи кількість доступної пам'яті. Зрештою, у гіршому випадку, надто більша кількість доступної пам'яті може стати розподіленою, і вся система або пристрій або його частина припиняють працювати, програма не працює або система значно сповільниться через збиття.

Витоки пам'яті можуть не бути серйозними або навіть виявляти за допомогою звичайних засобів. У сучасних операційних системах звичайна пам'ять, яка використовується додатком, випускається після закінчення програми. Це означає, що витік пам'яті в програмі, що працює лише протягом короткого часу, може не помітитись і рідко буває серйозним.

Набагато більш серйозні витоки включають такі:

- де програма працює протягом тривалого часу і споживає додаткову пам'ять у часі, наприклад, фонових завдань на серверах, але особливо на вбудованих пристроях, які можуть залишатися працювати протягом багатьох років
- де часто використовується нова пам'ять для одноразових завдань, таких як при рендерингу кадрів комп'ютерної гри або анімаційного відео
- де програма може запитати пам'ять - наприклад, загальну пам'ять -, що не відпускається, навіть коли програма закінчується
- де пам'ять дуже обмежена, наприклад, у вбудованій системі або портативному пристрої
- де витік відбувається в операційній системі або диспетчері пам'яті
- коли драйвер системного пристрою викликає витік
- працює в операційній системі, яка автоматично не випускає пам'ять при завершенні програми

Витоки памяті притаманні всім або майже всім мовам програмування, а отже будь-якому виду програмного забезпечення. Нижче наведений приклад витоку памяті рисунок 2.4

```
/* Function with memory leak */
#include <stdlib.h>

void f()
{
   int *ptr = (int *) malloc(sizeof(int));

   /* Do some work */

   return; /* Return without freeing ptr*/
}
```

Рисунок 2.4 – Витік памяті

Подолати дану проблему досить легко. Під час розробки інженери та програмісти мають ретельно слідкувати за написаним кодом та менеджментом памяті. На рисунку 2.5 наведено приклад усунення витоку памяті.

```
/* Function without memory leak */
#include <stdlib.h>;

void f()
{
   int *ptr = (int *) malloc(sizeof(int));

   /* Do some work */

   free(ptr);
   return;
}
```

Рисунок 2.5 – Усунення витоку памяті

До витоків памяті вразливі навіть такі мови програмування як ті, що базуються і виконують свій кінцевий байт код у віртуальній машині. Наприклад Java.

Стандартне визначення витоку пам'яті - це сценарій, який виникає, коли програма більше не використовує об'єкти, однак прибиральник сміття не може видалити їх із робочої пам'яті, оскільки вони все ще мають посилання. В

результаті, програма витрачає все більше і більше ресурсів - що, зрештою, призводить до фатального OutOfMemoryError.

Для кращого розуміння концепції, просте візуальне подання зображене на рисунку 2.6.

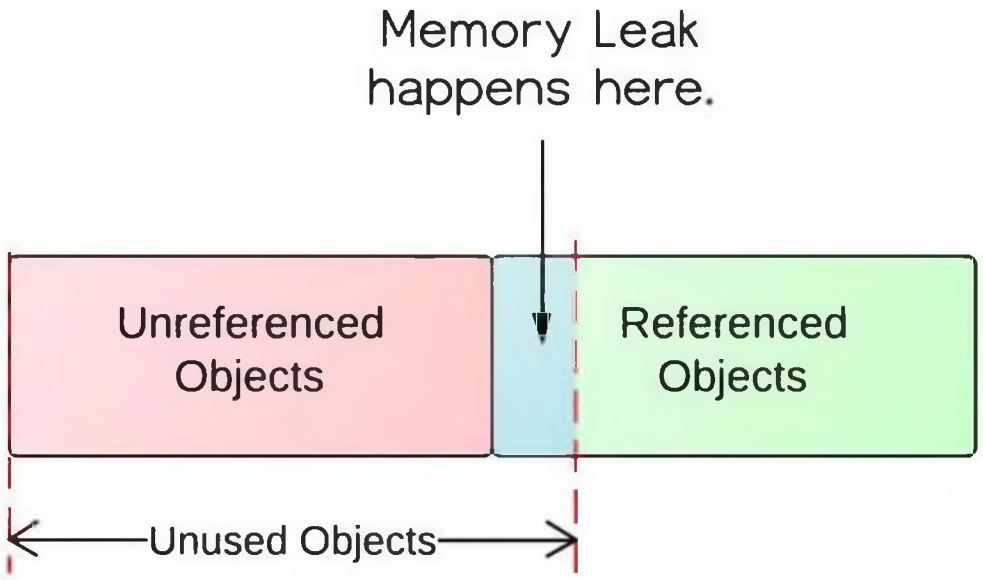

Рисунок 2.6 – Місце витоку памяті в програмі написаній на Java

Як ми бачимо, у нас є два типи об'єктів - посилання та посилання; Смітник може видалити об'єкти, які не мають відношення. Об'єкти з посиланням не будуть збиратися, навіть якщо вони насправді не використовуються додатком.

Виявлення витоків пам'яті може бути складним. Ряд інструментів виконує статичний аналіз для визначення потенційних витоків, але ці методи не є досконалими, оскільки найважливішим аспектом є фактична поведінка робочої системи під час виконання.

Отже, давайте розглянемо деякі стандартні методи попередження витоків пам'яті, аналізуючи деякі загальні сценарії.

Витоки Java в купі(heap). У цьому початковому розділі ми зосередимось на класичному сценарії витоку пам'яті - де об'єкти Java постійно створюються без випуску.

Вигідним методом для розуміння цих ситуацій є полегшення відтворення витоку пам'яті, встановивши нижчий розмір для купи. Ось чому при запуску

нашої програми ми можемо налаштувати JVM відповідно до наших потреб пам'яті:

```
-Xms<size>
```

```
-Xmx<size>
```

Рисунок 2.7 – Опції, що встановлюють розмір купи

Ці параметри визначають початковий розмір Java Heap, а також максимальний розмір Heap.

Перший сценарій, який може спричинити витік пам'яті у Java, - посилання на важкий об'єкт із статичним полем.

```
private Random random = new Random();
public static final ArrayList<Double> list = new ArrayList<Double>(1000000);

@Test
public void givenStaticField_whenLotsOfOperations_thenMemoryLeak() throws InterruptedE
xception {
    for (int i = 0; i < 1000000; i++) {
        list.add(random.nextDouble());
    }

    System.gc();
    Thread.sleep(10000); // to allow GC do its job
}
```

Рисунок 2.8 – Практична реалізація витоку памяті в Java Heap

Ми створили наш ArrayList як статичне поле, яке ніколи не збиратиметься JVM Garbage Collector протягом всього періоду роботи JVM, навіть після того, як були проведені розрахунки для його використання. Ми також викликали thread.sleep (10000), щоб дозволити GC виконати повне зібрання та спробувати відновити все, що може бути відновлено.

Запустимо тест і проаналізуємо JVM з нашим профілером рисунок 2.9:

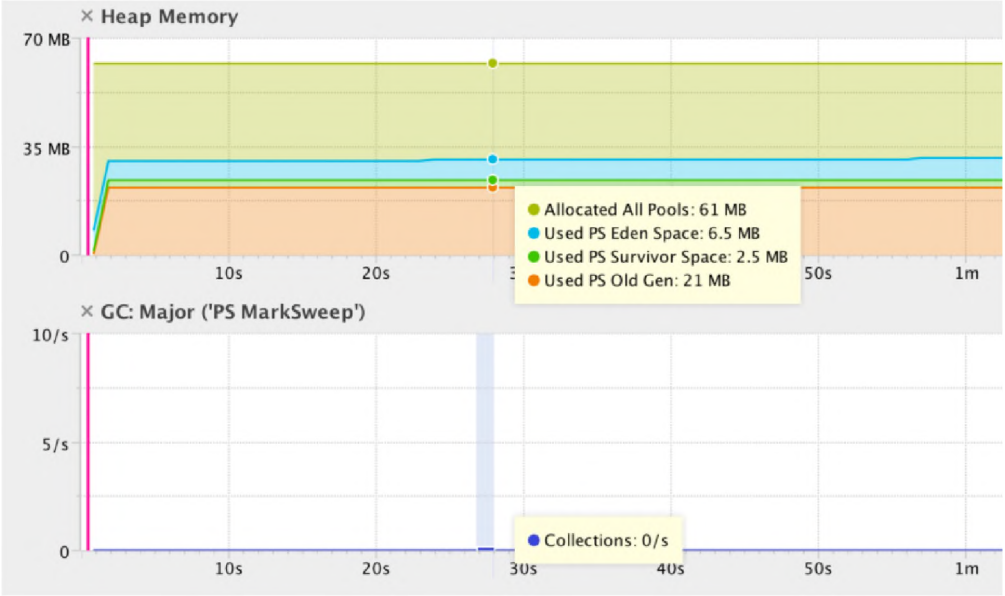

Рисунок 2.9 - Java Heap, стан при виконанні та завершенні

Зверніть увагу, як спочатку вся пам'ять, звичайно, вільна. Потім, за 2 секунди, процес ітерації запускається і завершується - завантажуючи все в список (природно, це буде залежати від машини, на якій ви використовуєте тест).

Після цього запускається повний цикл збору сміття, і тест продовжує виконуватись, щоб цей цикл було запущено та завершено. Як видно, цей список не відновлюється, а споживання пам'яті не зменшується.

Давайте тепер побачимо точний той самий приклад, тільки цього разу, ArrayList не посилається статичною змінною. Замість цього це локальна змінна, яка створюється, використовується, а потім відкидається рисунок 2.10.

```java
@Test
public void givenNormalField_whenLotsOfOperations_thenGCWorksFine() throws Interrupted
Exception {
    addElementsToTheList();
    System.gc();
    Thread.sleep(10000); // to allow GC do its job
}

private void addElementsToTheList(){
    ArrayList<Double> list = new ArrayList<Double>(1000000);
    for (int i = 0; i < 1000000; i++) {
        list.add(random.nextDouble());
    }
}
```

Рисунок 2.10 – Дублювання функціоналу на Рисунку 2.5, але використовуючи локальну змінну замість статичної

Як тільки метод завершить свою роботу, ми будемо спостерігати за основною колекцією GC, рисунок 2.11.

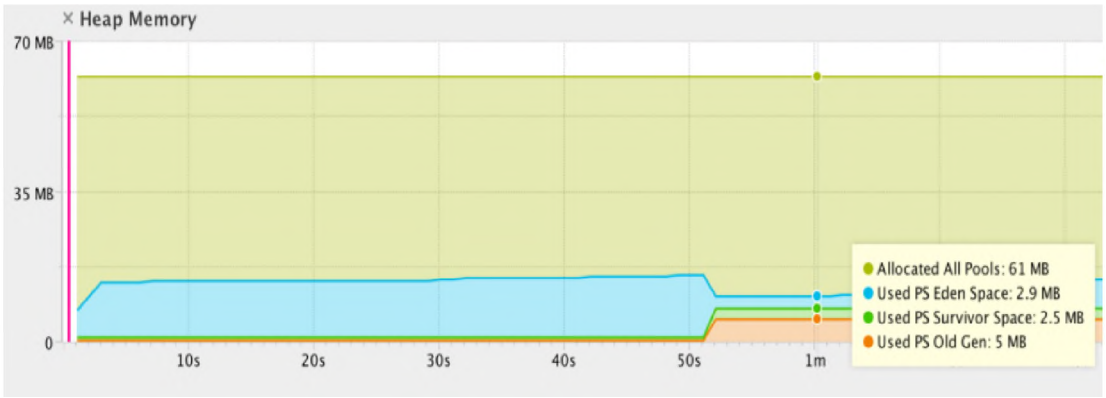

Рисунок 2.11 - Java Heap

Як запобігти таким витокам. По-перше, ми повинні приділяти пильну увагу наше використання статичного; декларуючи будь-яку колекцію або важкий об'єкт, як статичні зв'язку його життєвий цикл до життєвого циклу самого JVM, і робить весь об'єкт графіка неможливо зібрати.

Нам також слід знати про колекції в цілому - це найпоширеніший спосіб випадкового тримання довідок довше, ніж ми потребуємо.

2.3 Машинне навчання

Машинне навчання (ML) - це вивчення алгоритмів та математичних моделей, які комп'ютерні системи використовують для поступового підвищення їх продуктивності на конкретне завдання. Алгоритми машинного навчання створюють математичну модель зразків даних, відомі як "дані тренування", для того, щоб робити прогнози або рішення, не будучи явно запрограмованими для виконання завдання. Алгоритми навчання машини використовуються в програмах фільтрації електронної пошти, виявлення мережевих вторників, і комп'ютерне бачення, де неможливо розробити алгоритм конкретних інструкцій для виконання завдання. Машинне навчання тісно пов'язане з обчислювальною статистикою, яка зосереджується на створенні прогнозів з використанням комп'ютерів. Вивчення математичної оптимізації забезпечує методи, теорію та області застосування в галузі машинного навчання. Видобуток даних - це область вивчення в рамках машинного навчання, і основна увага приділяється аналізу дослідницьких даних через безконтрольне навчання. У своєму застосуванні через проблеми бізнесу машинне навчання також називається прогнозною аналітикою.

Застосування машинного навчання.

- У класифікації входи поділяються на два або більше класів, і система-учень мусить породити модель, яка відносить небачені входи до одного або більше з цих класів. Це, як правило, намагаються розв'язувати керованим чином. Прикладом класифікації є фільтри спаму, в яких входами є повідомлення електронної пошти (або чогось іншого), а класами є «спам» та «не спам».
- У регресії (англ. regression), також керованій задачі, виходи є безперервними, а не дискретними.
- У кластеруванні (англ. clustering) набір входів повинно бути поділено на групи. На відміну від класифікації, групи не відомі заздалегідь, що зазвичай робить це завданням для спонтанного навчання.
- Оцінка густини знаходить розподіл входів у деякому просторі.

- Зниження розмірності спрощує входи шляхом відображення їх на простір меншої розмірності. Пов'язаною задачею є тематичне моделювання, в якому програмі надають перелік документів людською мовою, і ставлять задачу з'ясувати, які документи охоплюють подібні теми.

В даній дипломній роботі, для пошуку та прогнозування витоку таємного ключа, тобто для класифікації певного набору даних, на предмет позитивного чи негативного рузультату витоку буде використано алгоритми машинного навчання, що вирішують питання класифікації.

Дерево рішень. Тип контрольованого алгоритму навчання, який в основному використовується для класифікації проблем. Як не дивно, він працює як для категоричних, так і для постійних залежних змінних. У цьому алгоритмі ми розбиваємо популяцію на два або більше однорідних множини. Це робиться на основі самих значущих атрибутів / незалежних змінних, щоб зробити їх якомога більш різними групами рисунок 2.12.

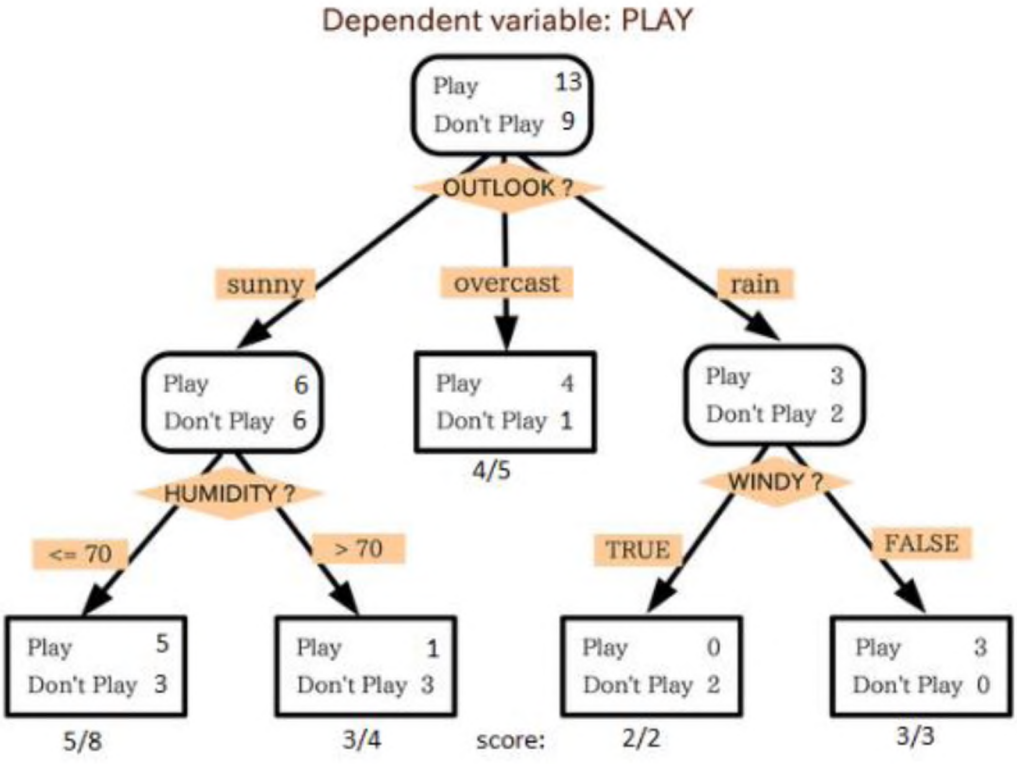

Рисунок 2.12 – Візуалізація дерева рішень

SVM (підтримка векторної машини). Це класифікаційний метод. У цьому алгоритмі ми зображаємо кожен елемент даних як точку в n-мірному просторі (де

n - це кількість параметрів, що є в наявності), причому значення кожної функції є значенням певної координати.

Наприклад, якщо у нас були тільки два параметри, такі як висота людини і довжина волосся, ми спочатку розбиваємо ці дві змінні у двомірному просторі, де кожна точка має дві координати (ці координати називаються векторами підтримки) рисунок 2.13.

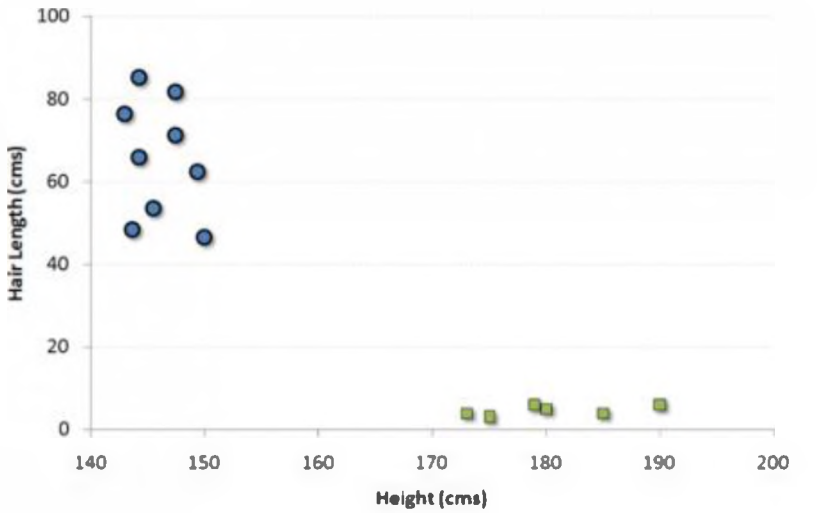

Рисунок 2.13 – Розбивання параметрів у двомірному просторі

Тепер ми знайдемо деяку лінію, яка розподіляє дані між двома різними класифікованими групами даних. Це буде така лінія, що відстані від найближчої точки в кожній з двох груп будуть віддалені далеко рисунок 2.14

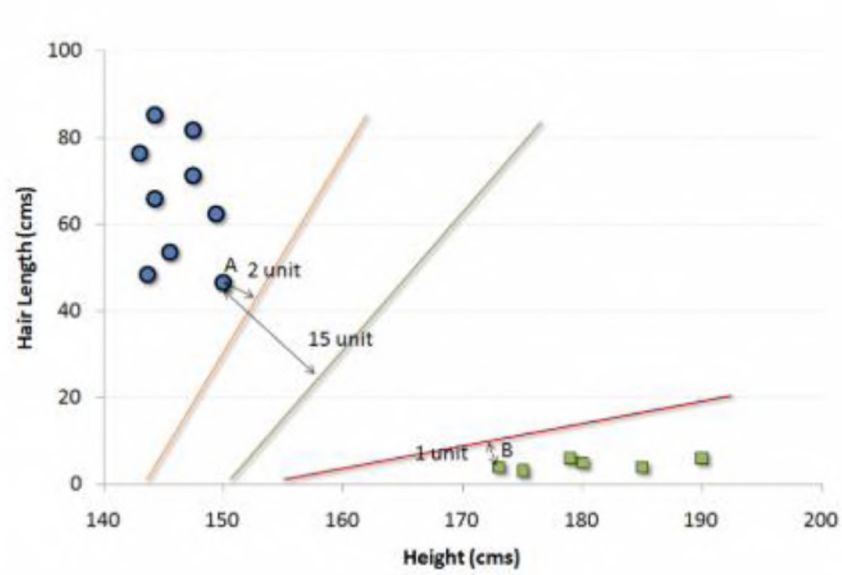

Рисунок 2.14 – Класифікуюча чорна лінія

У прикладі, показаному вище, лінія, яка розбиває дані на дві класично відомі групи, є чорною лінією, оскільки дві найближчі точки є найвіддаленіші, від цієї лінії. Ця лінія - наш класифікатор. Тоді, залежно від того, з якого боку дані будуть відносно класифікатора, алгоритм і визначає приналежність даних до того, чи іншого класу.

Naive Bayes. Це класифікаційна техніка, заснована на теоремі Байєса з припущенням про незалежність між предикторами. У простих термінах Naive Bayes класифікатор припускає, що наявність певної функції в класі не має відношення до наявності будь-якої іншої функції. Наприклад, фрукт можна вважати яблуком, якщо він червоний, круглий і близько 3 дюймів у діаметрі. Навіть якщо ці функції залежать один від одного або від наявності інших функцій, Naive Bayes класифікатор розгляне всі ці властивості, щоб вони самостійно сприяли ймовірності, що цей фрукт - це яблуко.

Naive Bayes модель легко побудувати і особливо корисна для дуже великих наборів даних. Нарівні з простотою, Naive Bayes, як відомо, перевершує навіть дуже складні методи класифікації.

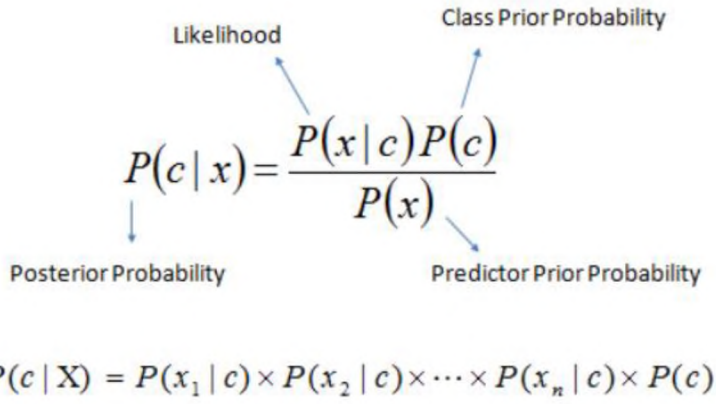

Рисунок 2.15 – Теорема Баєса

kNN (к- Найближчих сусідів). Його можна використовувати як для класифікації, так і для регресії. Проте, більш широко використовується для класифікації. К найближчих сусідів - це простий алгоритм, який зберігає всі доступні випадки та класифікує нові випадки більшістю голосів своїх сусідів.

Данні часного випадку будуть призначений для класу, що найбільш поширений серед своїх найближчих K сусідів, виміряних за допомогою функції дистанції.

Ці відстані можуть бути Евклідова, Манхеттен, Мінковскі та Хаммін. Перші три функції використовуються для безперервної функції, а четверте - для категоричних змінних (Хеммінга). Якщо K = 1, тоді справа просто призначається класу свого найближчого сусіда. Іноді вибір K виявляється складним завданням при виконанні моделювання kNN рисунок 2.16.

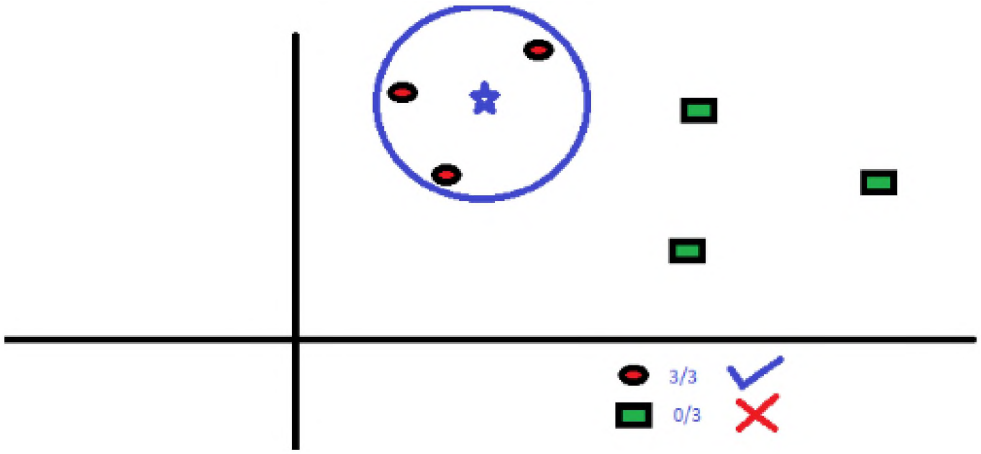

Рисунок 2.16 – Візуалізація алгоритму kNN

K-Means. Це тип алгоритму без нагляду, який вирішує проблему кластеризації. Його процедура має простий і легкий спосіб класифікації заданих даних через певну кількість кластерів (припустимо, k кластерів). Дані точки всередині кластера є однорідними та неоднорідними групам однолітків рисунок 2.17..

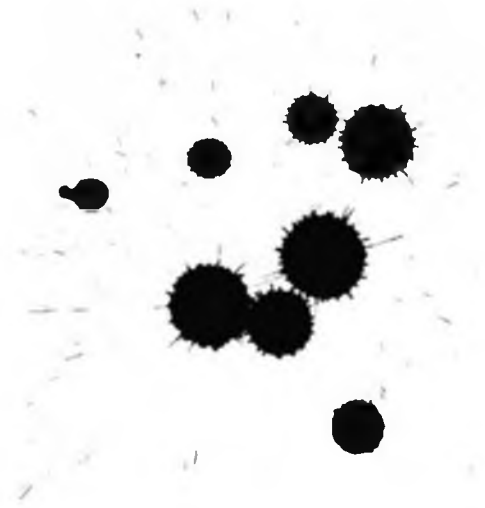

Рисунок 2.17 – Візуалізація K-Means

K- means утворюють кластер:

- K-means збирає k кількість точок для кожного кластера, відомий як centroids.
- Кожна точка даних утворює кластер з найближчими центроїдами, тобто k кластерами.
- Знаходить центроїд кожного кластера на основі існуючих членів кластера. Тут ми маємо нові центроїди.
- Оскільки ми маємо нові центроїди, повторіть кроки 2 і 3. Знайдіть найближчий відстань для кожної точки від нових центроїдів і зв'яжіться з новими k-кластери. Повторіть цей процес, доки не відбудеться конвергенція, тобто центроїди не змінюються.

Висновки до розділу 2

В даному розділі було дано визначення таємним ключам та їх витокам. Проаналізовано природу звичайних витоків памяті та порівняно звичайні витоки памяті з витоками таємних ключей, показана різниця.

Було проаналізовано декілька основних алгоритмів машинного навчання, що будуть використовуватись в практичній частині даної роботи, задля порівняння результатів та вибору оптимального алгоритму для поставленої задачі в рамках цієї роботи.

3 УСУНЕННЯ ВИТОКІВ ТАЄМНИХ КЛЮЧІВ

3.1 Налаштування середовища

Завантаження дерева Андроід. Дерево джерел Android знаходиться в сховищі Git, розміщеному компанією Google. Репозиторій Git містить метадані для джерела Android, зокрема ті, що стосуються змін до джерела та дати їх створення.

- Ініціалізація репозиторію

Repo - це інструмент, який полегшує роботу з Git у контексті Android. Створюємо папку та додаємо її в шлях системи:

mkdir ~/bin

PATH=~/bin:$PATH

- Завантажуємо застосунок та переконаємося, що він має права на виконання:

curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo

chmod a+x ~/bin/repo

- Ініціалізуємо Репо клієнт

Створюємо робочу папку, що буде містити всі необхідні файли для роботи:

mkdir WORKING_DIRECTORY

cd WORKING_DIRECTORY

- Тепер сконфігуруємо git для використання в системі:

git config --global user.name "Your Name"

git config --global user.email "you@example.com"

- Тепер оновимо застосунок Репо та вкажемо необхідну гілку Андроіду з якою і будемо працювати:

repo init -u https://android.googlesource.com/platform/manifest

repo init -u https://android.googlesource.com/platform/manifest -b android-7.0.15_r1

Завантаження обраної гілки Андроід

repo sync

- Підготовка до зборки Андроід

AOSP не може бути використаний лише з чистого вихідного коду і вимагає запуску додаткових бібліотек, пов'язаних з обладнанням, наприклад, для прискорення апаратної графіки. Тож перш за все необхідно завантажити та встановити необхідні бібліотеки. Пілся чого можна продовжити. Ініціалізація середовища за допомогою envsetup.sh:

source build/envsetup.sh

- Вибір платформи та версії прошивки яку будемо збирати

В залежності, яку конфігурацію слід побудувати з використанням застосунку lunch . Точна конфігурація може бути передана як аргумент. Наприклад, наступна команда посилається на повне зібрання для емулятора, при увімкненому налагодженні:

lunch aosp_arm-eng

Усі конфігурації побудови приймаються за формою BUILD-BUILDTYPE, де BUILD - це кодова назва, що відноситься до певної комбінації властивостей. BUILDTYPE є одним з таких, як у таблиці:

Рисунок 3.1 – Види можливих версій бінарів

Тип конфігурації	Опис
user	обмежений доступ
userdebug	як користувач, але з root доступом і debuggability; краще для налагодження
eng	конфігурація розробки з додатковими засобами налагодження

Рисунок 3.1 - Види можливих версій бінарів

- Побудова Андроід

Побудова здійснюється за допомогою утиліти make. GNU make може обробляти паралельні завдання за допомогою аргументу -jN, і загальним є використання ряду завдань N, що в межах від 1 до 2 разів перевищує кількість апаратних потоків на комп'ютері, який використовується для збирання. Наприклад, на

комп'ютері з подвійним E5520 (2 процесори, 4 ядра на ЦП, 2 нитки на ядро) найшвидший збір виконується за допомогою команд між make-j16 і make-j32.

make -j16

3.2 Пошук та усунення Bluetooth Link Key leak

Link key – приватний аутентифікаційний ключ, 128-бітові випадкові числа, що використовуються для цілей аутентифікації. Парні пристрої поділяють спільний Link Key.

- Для початку аналізу витоків даного ключа, необхідно провести попередню підготовку та визначити необхідну процедуру для генерації данного ключа, а також визначити процедуру після якої даний ключ має бути видалено з памяті.

Сценарій генерації ключа – проведення процедури спарення двох мобільних пристроїв та отримання необхідного значення ключа. Почнемо логування у функції, що викликається при процедурі пейрінгу рисунок 3.1.

```
bt_status_t btif_storage_add_bonded_device(RawAddress* remote_bd_addr,
                                           LINK_KEY link_key, uint8_t key_type,
                                           uint8_t pin_length) {
  std::string bdstr = remote_bd_addr->ToString();
  int ret = btif_config_set_int(bdstr, "LinkKeyType", (int)key_type);
  ret &= btif_config_set_int(bdstr, "PinLength", (int)pin_length);
  ret &= btif_config_set_bin(bdstr, "LinkKey", link_key, sizeof(LINK_KEY));

  if (is_restricted_mode()) {
    BTIF_TRACE_WARNING("%s: '%s' pairing will be removed if unrestricted",
                       __func__, bdstr.c_str());
    btif_config_set_int(bdstr, "Restricted", 1);
  }

  /* write bonded info immediately */
  btif_config_flush();
  return ret ? BT_STATUS_SUCCESS : BT_STATUS_FAIL;
}
```

Рисунок 3.1 – Генерація Link Key та місце для його логування

Після встановлення логування у потрібне місце, необхідно перезібрати Андроід, згідно до дій описаних у першому підрозділі даного розділу.

Після побудови Андроід все готово для прошивання телефону та початку сценарію для генерації та отримання значення шуканого ключа.

Для Bluetooth pairing процедури потрібно два пристрої, що мають підтриму, як апаратну так і програмні модуля Bluetooth. Перш за все вмикаємо Bluetooth на обох пристроях та ініціюємо процедуру від імені пристрою, що знаходиться під тестом(Мастер). Рисунок 3.

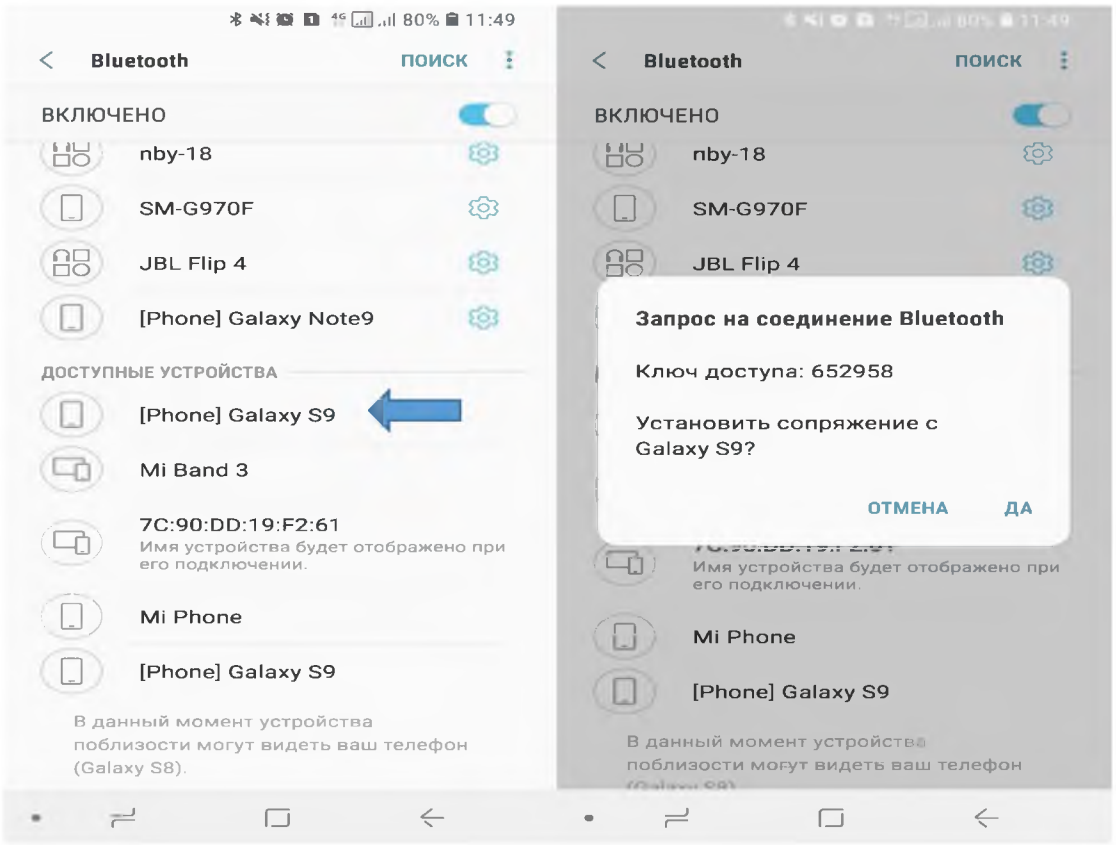

Рисунок 3.2 - Bluetooth pairing

Після успішної процедури переконаємось, що два пристрої дійсно успішно завершили процедуру. Рисунок 3.3

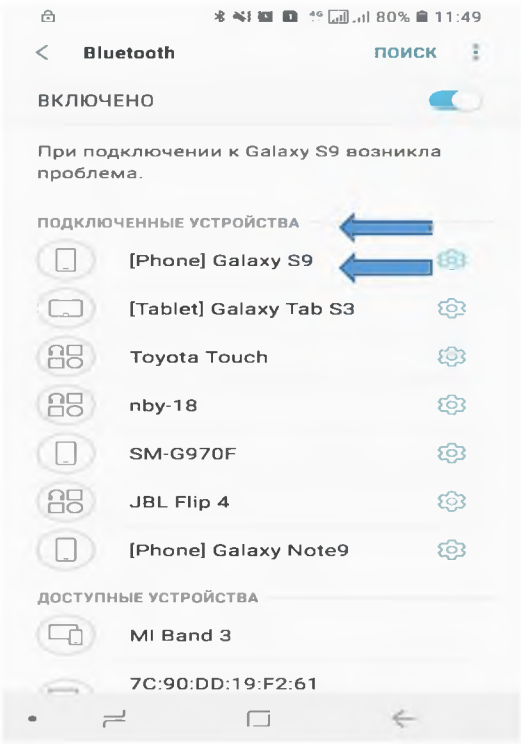

Рисунок 3.3 – Пристрій додан до числа підключених

В той час, як користувач натиснув кнопку підключитися до іншого пристрою, відбулась процедура пейрінгу, а отже в LogCat-і ми мали б побачити новий, згенерований ключ, логування якого ми додавали. Перевіремо цей факт рисунок 3.4

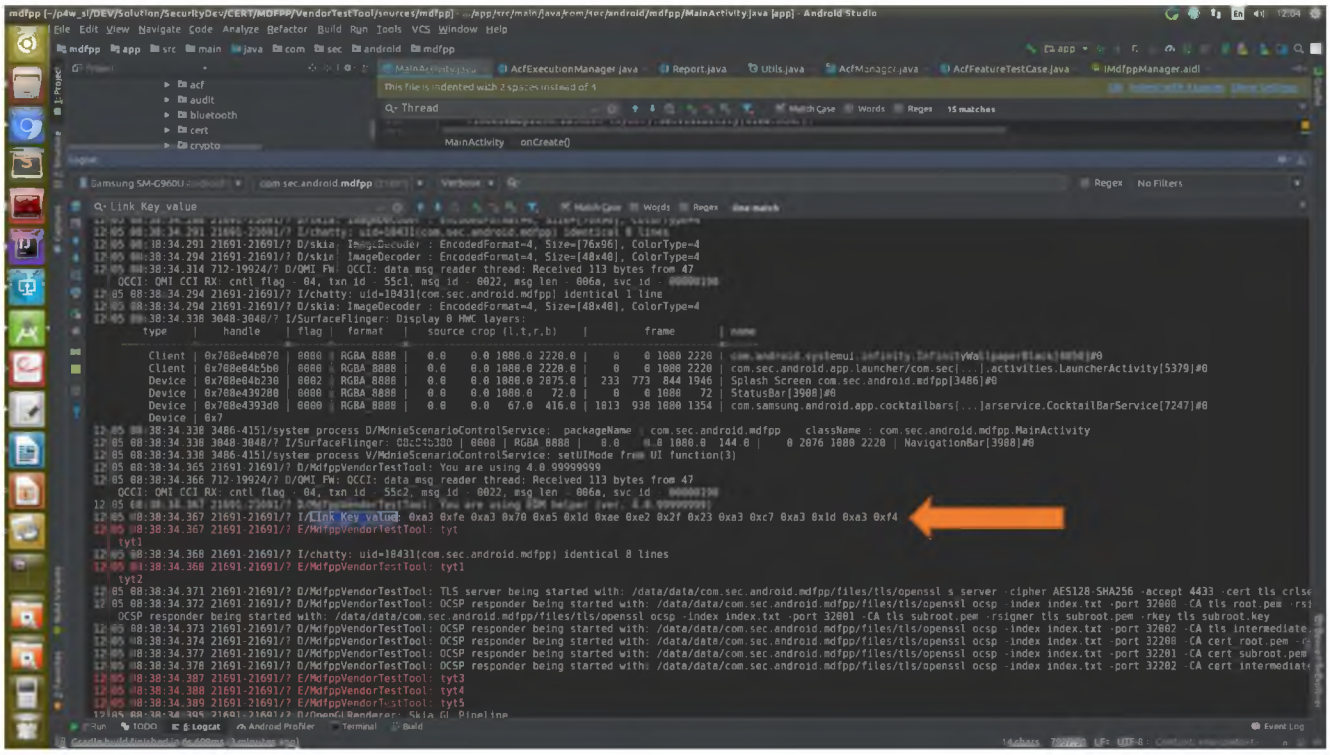

Рисунок 3.5 – Логування шуканого ключа

Отримане значення ключа - 0xa3 0xfe 0xa3 0x70 0xa5 0x1d 0xae 0xe2 0x2f 0x23 0xa3 0xc7 0xa3 0x1d 0xa3 0xf4. Після цього, наступає процедура після якої даний ключ має бути затертий у памяті телефону. Для Блютусу, ця дія – або вимкнення Блютусу, або процедура відвяки і забування збереженого пристрою. Тож ініціюємо дану процедуру. Після чого девайс має зникнути зі списку збережених. Тепер для повної і чесної перевірки усієї памяті пристрою на предмет наявності в ньому витоку ключа у відкритому вигляді, необхідно зняти дамп фізичної памяті телефону. Для зняття дампу будемо використовувати утиліту rdx. Переводимо наш пристрій в спеціальний режим та перевіряємо чи бачить його наша утиліта рисунок 3.6

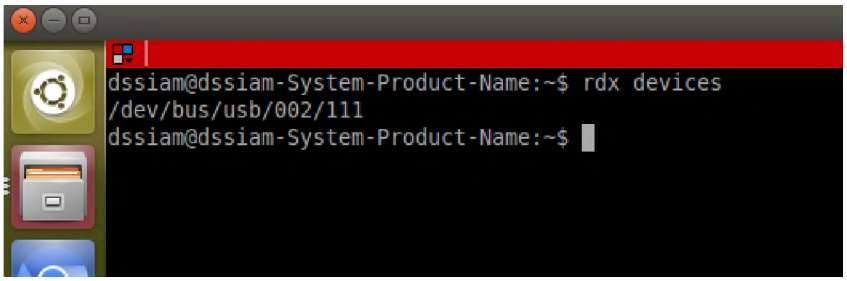

Рисунок 3.7 – Перевірка наявності пристрою

Пристрій знайдено тому тепер є можливість отримати дамп памяті рисунок 3.8

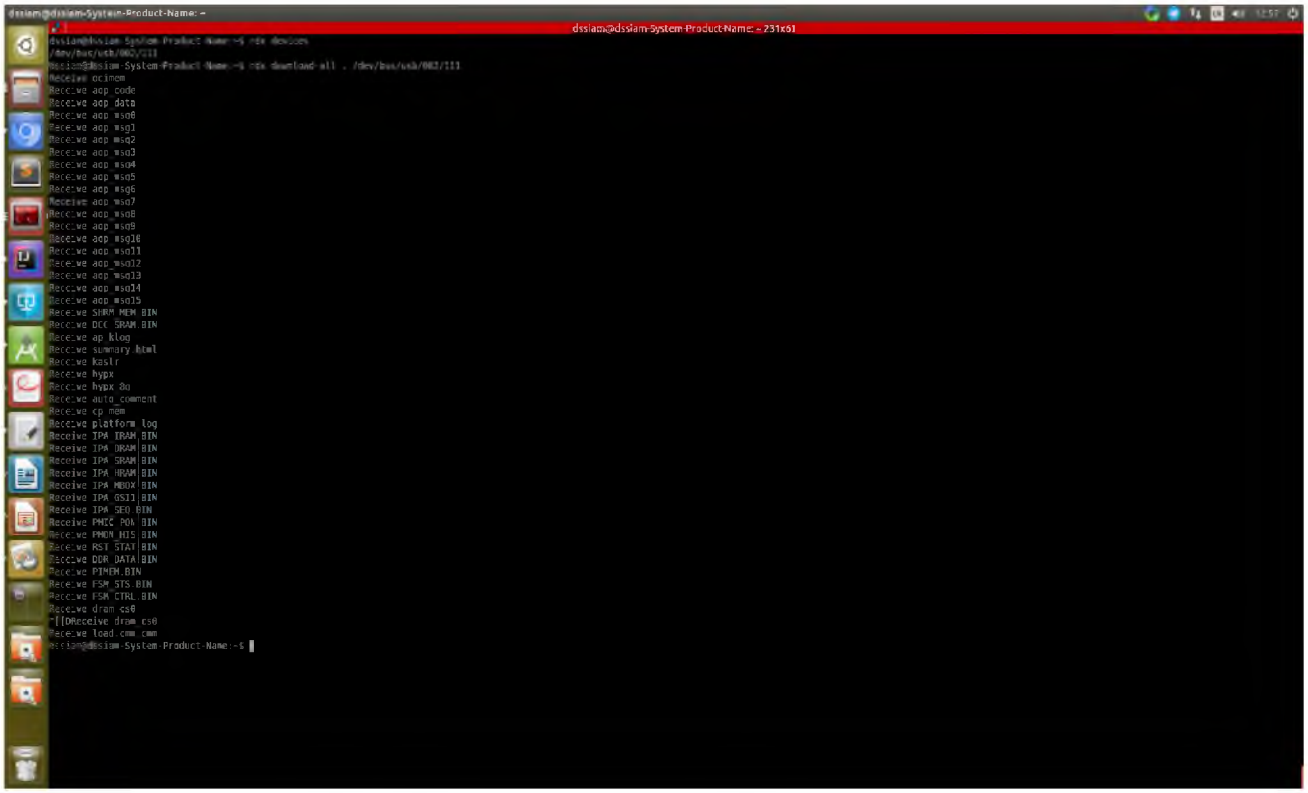

Рисунок 3.8 – Зняття дампу памяті з пристрою

Отримавши дамп можна приступити до пошуку необхідних даних, а саме – це значення нашого ключа. Зазначимо, що так-як була проведена процедура анпейрінгу, ми не маємо знайти значення шуканого ключа в дампах Рисунок 3.9.

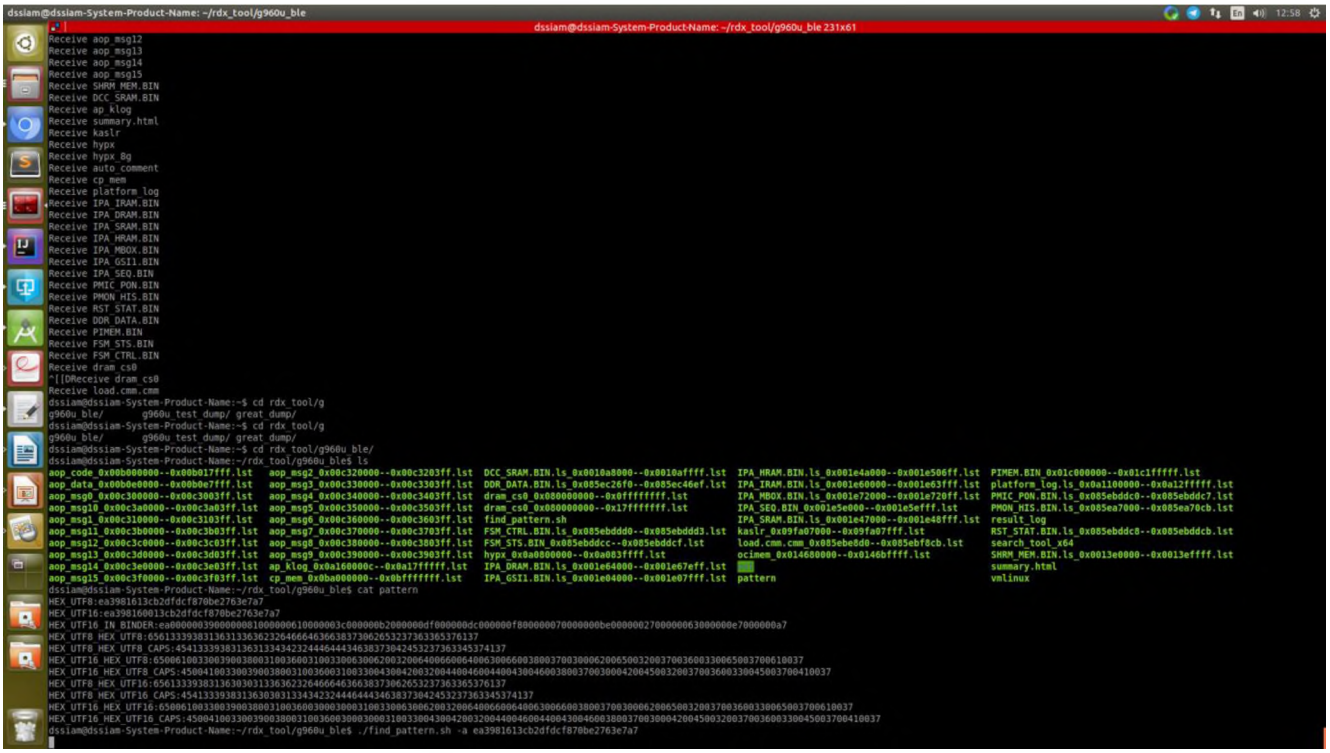

Рисунок 3.9 - Пошук витоку

Для пошуку данних в дампах можна використовувати будь-яку утиліту, я використав графічну bless рисунок 3.10.

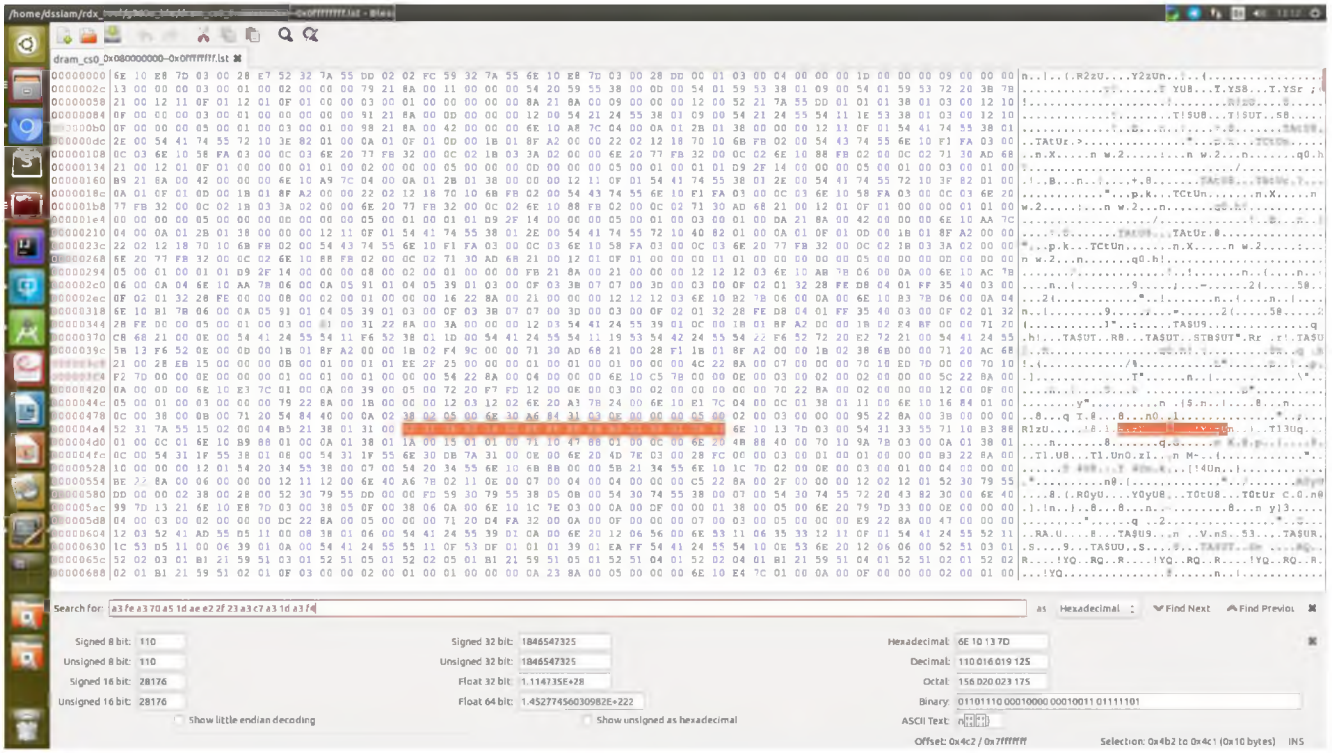

Рисунок 3.10 – Утиліта bless, пошук необхідних даних

За отриманим результатом, в дампах було знайдено 2 витоки тестового ключа. Отже можна зробити висновок, що необхідний пошук місця, де можна було б подолати данну проблему. Для цього будемо аналізувати код. Ми знаємо, що початкове місце відправлення – це процедура видалення пристрою зі списку збережених.

Отже стартовий метод – unpair(). Спробуємо знайти його на http://androidxref.com. Рисунок 3.11

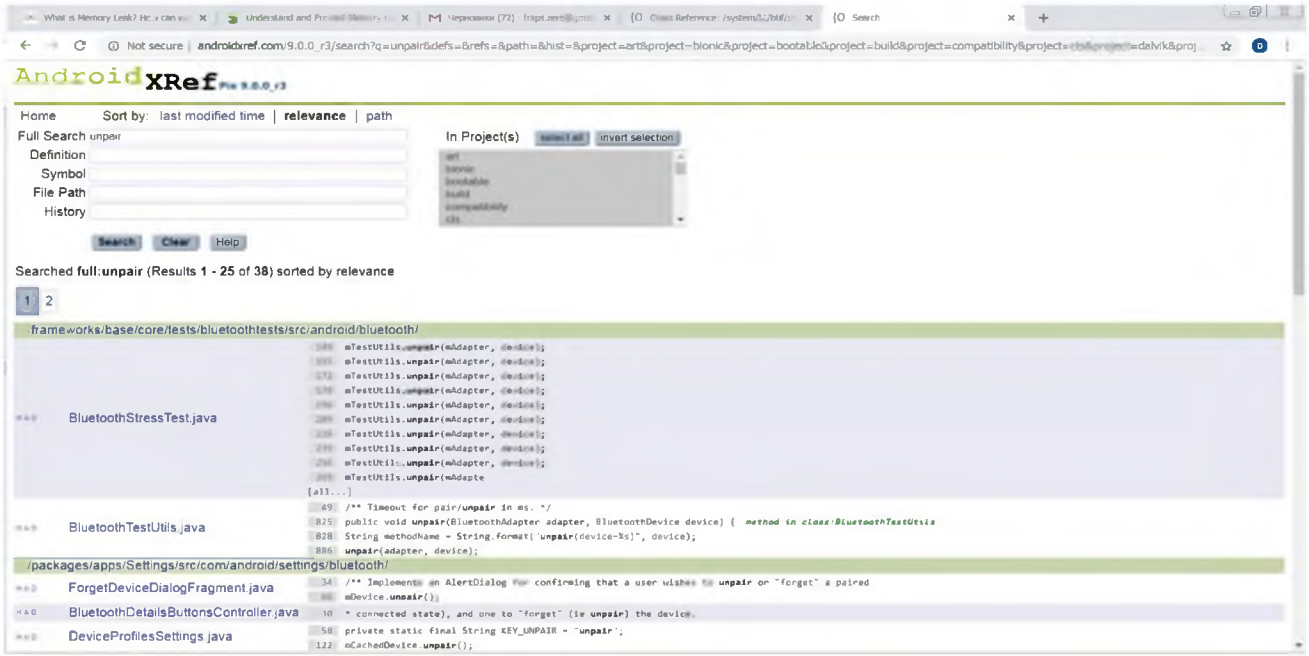

Рисунок 3.11 – Використання http://androidxref.com

Можна побачити, що даний метод використовується в багатьох класах. Найбільш цікавий для нас є ForgetDeviceDialogFragment.java атже це компонент(Фрагмент), що відповідає за графічний інтерфес для дії – «Забути девайс». Розглянемо даний клас більш детально рисунок 3.12

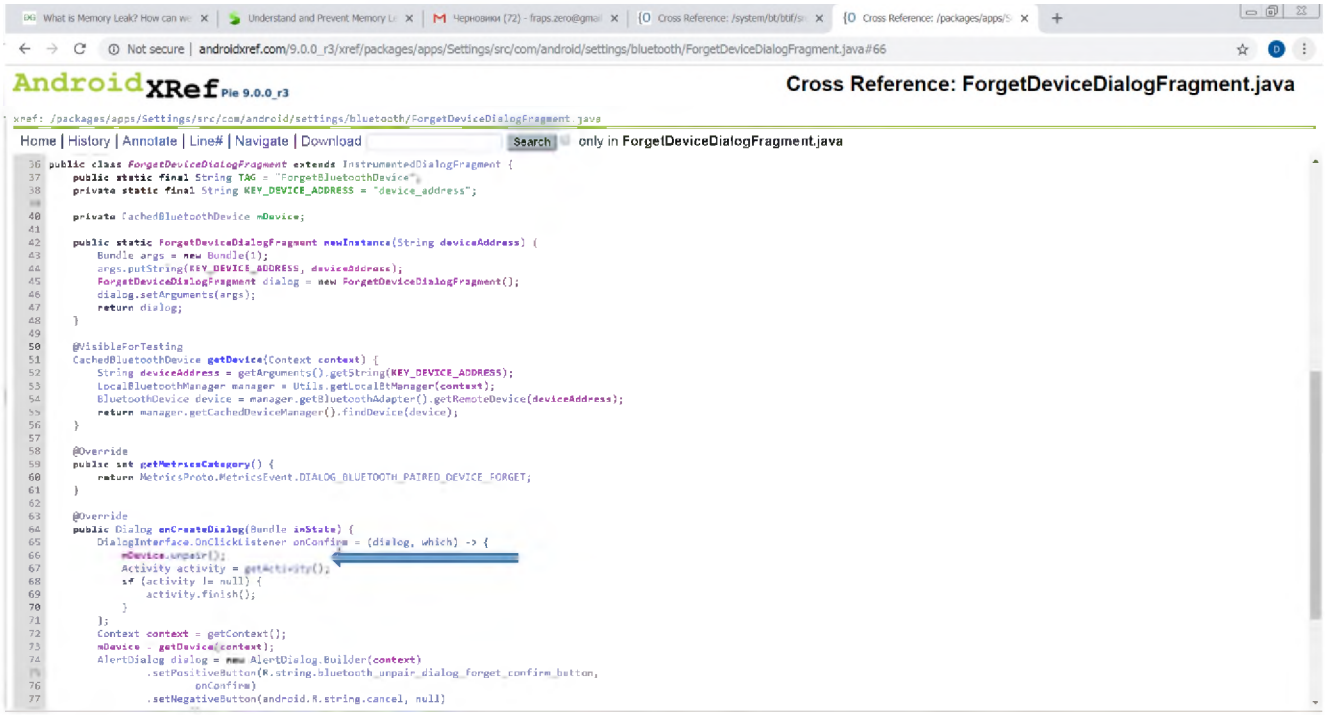

Рисунок 3.13 – Виклик функції unpair

Як можна бачити unpair є методом класу CachedBluetoothDevice, тому його реалізація знаходиться саме там. Перейдемо до даного класу задля наступного аналізу рисунок 3.14 – клас CachedBluetoothDevice та реалізація методу unpair.

```
public void unpair() {
    int state = getBondState();

    if (state == BluetoothDevice.BOND_BONDING) {
        mDevice.cancelBondProcess();
    }

    if (state != BluetoothDevice.BOND_NONE) {
        final BluetoothDevice dev = mDevice;
        if (dev != null) {
            final boolean successful = dev.removeBond();
            if (successful) {
                if (Utils.D) {
                    Log.d(TAG, "Command sent successfully:REMOVE_BOND " + describe(null));
                }
            } else if (Utils.V) {
                Log.v(TAG, "Framework rejected command immediately:REMOVE_BOND " +
                    describe(null));
            }
        }
    }
}
```

Рисунок 3.14 – Реалізація методу unpair

Далі по аналогії перейдемо до класу `BluetoothDevice.java` та проаналізуємо реалізацію наступного методу.

```
/**
 * Remove bond (pairing) with the remote device.
 * <p>Delete the link key associated with the remote device, and
 * immediately terminate connections to that device that require
 * authentication and encryption.
 * <p>Requires {@link android.Manifest.permission#BLUETOOTH_ADMIN}.
 *
 * @return true on success, false on error
 * @hide
 */
@SystemApi
@RequiresPermission(android.Manifest.permission.BLUETOOTH_ADMIN)
public boolean removeBond() {
    final IBluetooth service = sService;
    if (service == null) {
        Log.e(TAG, "BT not enabled. Cannot remove Remote Device bond");
        return false;
    }
    try {
        Log.i(TAG, "removeBond() for device " + getAddress()
                + " called by pid: " + Process.myPid()
                + " tid: " + Process.myTid());
        return service.removeBond(this);
    } catch (RemoteException e) {
        Log.e(TAG, "", e);
    }
    return false;
}
```

Рисунок 3.15 - Реалізація методу removeBond

Далі виклик передається іншому процесу, в Андроід міжпроцесорна взаємодія IPC відбувається за допомогою Binder. Розглянемо цей механізм більш детально.

Базовий клас для об'єкта, що відновлюється, основна частина легкого механізму видалення процедур, визначена IBinder. Цей клас - це реалізація IBinder, яка забезпечує стандартну локальну реалізацію такого об'єкта.

Більшість розробників не впроваджують цей клас безпосередньо, замість цього використовуючи інструмент helpl для опису потрібного інтерфейсу, він повинен генерувати відповідний підклас Binder. Однак, ви можете отримати безпосередньо від Binder для реалізації власного протоколу RPC або просто інстанціїти raw Binder безпосередньо для використання як токен, який можна розподілити між процесами.

Цей клас є лише базовим прикладом IPC; це не впливає на життєвий цикл додатка і діє лише тоді, коли процес, який його створив, продовжує працювати.

Щоб правильно використовувати це, ви повинні робити це в контексті компоненту програми верхнього рівня (служби, активності або контент-постачальника), яка дає змогу системі знати, що ваш процес повинен працювати.

Ви повинні мати на увазі ситуації, коли ваш процес може піти, і, отже, вимагати, щоб ви пізніше знову створили новий Binder і знову приєднаєте його, коли процес починається знов. Наприклад, якщо ви використовуєте це в межах однієї діяльності, процес вашої активності може бути вбитий в будь-який час, коли діяльність не розпочата; якщо діяльність пізніше буде знову створено, вам доведеться створити нову Binder і повернути її знову до потрібного місця; ви повинні знати, що ваш процес може бути запущений з іншої причини (наприклад, для отримання трансляції), що не вимагатиме повторного створення активності, і таким чином запустити його код, щоб створити новий Binder.

Binder передасть виконання видалення пристрою в Bluetooth stack, тобто нативний код. Буде викликано низькорівневі функції рисунок 3.16

```
/*******************************************************************************
 *
 * Function         bta_dm_remove_sec_dev_entry
 *
 * Description      Removes device entry from Security device DB if ACL
connection with
 *                  remtoe device does not exist, else schedule for dev entry
removal upon
                     ACL close
 *
 * Returns          void
 *
 ******************************************************************************/
static void bta_dm_remove_sec_dev_entry(const RawAddress& remote_bd_addr) {
  if (BTM_IsAclConnectionUp(remote_bd_addr, BT_TRANSPORT_LE) ||
      BTM_IsAclConnectionUp(remote_bd_addr, BT_TRANSPORT_BR_EDR)) {
    APPL_TRACE_DEBUG(
        "%s ACL is not down. Schedule for  Dev Removal when ACL closes",
        __func__);
    BTM_SecClearSecurityFlags(remote_bd_addr);
    for (int i = 0; i < bta_dm_cb.device_list.count; i++) {
      if (bta_dm_cb.device_list.peer_device[i].peer_bdaddr == remote_bd_addr) {
        bta_dm_cb.device_list.peer_device[i].remove_dev_pending = TRUE;
        break;
      }
    }
  } else {
    BTM_SecDeleteDevice(remote_bd_addr);
    /* need to remove all pending background connection */
    BTA_GATTC_CancelOpen(0, remote_bd_addr, false);
    /* remove all cached GATT information */
    BTA_GATTC_Refresh(remote_bd_addr);
  }
}
```

Рисунок 3.16 – Продовження ланцюгу виклику функцій після Binder

У подальшому виклику нас цікавить лише дві функції рисунок 3.17 та 3.18

```
/*******************************************************************************
 *
 * Function         BTM_SecDeleteDevice
 *
 * Description      Free resources associated with the device.
 *
 * Parameters:      bd_addr          - BD address of the peer
 *
 * Returns          true if removed OK, false if not found or ACL link is active
 *
 ******************************************************************************/
bool BTM_SecDeleteDevice(const RawAddress& bd_addr) {
  if (BTM_IsAclConnectionUp(bd_addr, BT_TRANSPORT_LE) ||
      BTM_IsAclConnectionUp(bd_addr, BT_TRANSPORT_BR_EDR)) {
    BTM_TRACE_WARNING("%s FAILED: Cannot Delete when connection is active",
                      __func__);
    return false;
  }

  tBTM_SEC_DEV_REC* p_dev_rec = btm_find_dev(bd_addr);
  if (p_dev_rec != NULL) {
    btm_sec_free_dev(p_dev_rec);
    /* Tell controller to get rid of the link key, if it has one stored */
    BTM_DeleteStoredLinkKey(&p_dev_rec->bd_addr, NULL);
  }

  return true;
}
```

Рисунок 3.17 – Реалізація функції SecDeleteDevice()

```
/*******************************************************************************
 *
 * Function         btm_sec_free_dev
 *
 * Description      Mark device record as not used
 *
 ******************************************************************************/
void btm_sec_free_dev(tBTM_SEC_DEV_REC* p_dev_rec) {
  /* Clear out any saved BLE keys */
  btm_sec_clear_ble_keys(p_dev_rec);
  list_remove(btm_cb.sec_dev_rec, p_dev_rec);
}
```

Рисунок 3.18 – Реалізація функції btm_sec_free_dev()

Можна бачити, що в функції btm_sec_free_dev відбувається видалення BLE ключів, проте занулення наого ключа не відбувається. Тому справедливо припустити, що занулення можна спробувати реалізувати саме в цьому місці. Для цього додамо такий код у функцію btm_sec_free_dev:

```
memset(p_dev_rec->link_key, 0, sizeof(LINK_KEY));
```

Після чого перезберемо Android та повторимо процедури та сценарії, виконані в цьому розділі, а саме: перепрошити пристрій, провести процедуру пейрінгу двох пристроїв, провести процедуру анпейрінгу двох пристроїв, зняти дамп памяті та переконатися, що витоку даного ключа немає.

Після повторення всіх перерахованих дій, дійсно ми не змогли знайти витоку шуканого ключа в дампі памяті телефону, а отже ми змогли побороти даний витік.

3.3 Машинне навчання для автоматизації процесу пошуку витоків

В даній роботі було дано визначення таємним ключам, їх природі, а також було практично пок4азано механізм ручного пошуку та знешкодження витоку, варто зазначити, що наведений в цій роботі випадок витоку є простим та легким для відслідковування. Зазвичай, більшість подібних витоків не піддається такому швидкому аналізу.

В даномі розділі відбудеться спроба автоматизації процесу пошуку витоків таємних ключів, використовуючи алгоритми машинного навчання, що були проаналізовані в другому розділі даної роботи.

Перш за все необхідно визначити область даних, що буде охопленна для аналіз.

- Аналіз буде відбуватися статично, тобто по суті – парсинг коду Android OS.
- Аналізуватись буде лише C частина коду, в силу того, що включаючи в аналіз інші мови програмування, особливо Java, виникають проблемні ситуації для статичного аналіз коду.
- Далі для машинного навчання необхідно виокремити важливі дані, тут треба більш детально все проаналізувати

Парсинг ми будемо починати з вихідної точки. Вихідною точкою в даній роботі ми будемо називати функцію, що викликається і після виклику якої, шуканий таємний ключ має бути видалений з памяті пристрою. Наприклад для Bluetooth LinkKey такою функцією є unpair() у класу Device.java. Але так як

аналіз з машинним навчанням буде проводитись лише у мові С, то ланцюг виклику функцій до С ми будем ігнорувати.

Далі виокремимо важливі дані в функції, їх можна поділити на дві частини. Перша частина – це дані, що спричиняють копіювання шуканого ключа:

- Функції що в своїй основі використовують mmap(), аргументом даної функції має бути шуканий ключ, в противному випадку пропускаємо цю функцію
- Функції що в своїй основі використовують memcpy (), аргументом даної функції має бути шуканий ключ, в противному випадку пропускаємо цю функцію. Також, тут варто відслідковувати порядко аргументів функції, бо копі
- Також в дану групу будемо заносити дані про присвоєння адресу шуканого ключа іншим змінним

Друга група – це дані, що мають спричиняти очищення памяті:

- free() подібні функції
- memset() подібні функції

Зібрані дані будемо групувати відповідно до цих двух груп, як зображено на рисунку 3.19

key	func_1_in	funct_1_out	funct_2_out	.	.	.	.	funct_n_out	sum_in/sum_out
irk	3	4	0	.	.	.	.	2	0.3234

Рисунок 3.19 – Формат даних для машинного навчання

Для збільшення кількості даних, будемо використовувати допоміжні ключі, що знаходяться в структурах, що зберігають шукані таємні ключі рисунок 3.20.

```
keys = ['irk', 'ltk','csrk','stk','tk','ediv_bt','erand_bt','peer_irk',
'peer_ltk','peer_csrk','peer_stk','peer_tk','password','pin_bt','pin_password',
'wpa_psk','pmk','eap_tls', 'dhk', 'link_key', 'bd_addr', 'passkey']
# bta_api.h, wpa.h, dpp.h,
helpfull_keys = ['ptk', 'dhk', 'ir', 'bd_name', 'rssi', 'ble_primary_phy',
'ble_secondary_phy', 'num_uuids', 'er', 'akmp', 'spa', 'cui', 'wpa_ptk_rekey',
'p2p', 'network_ctx', 'aes_siv_key_len', 'jwk_crv', 'pubkey', 'pubkey_hash',
'peer_bootstrap_key', 'passphrase', 'psk', 'own_protocol_key', 'peer_protocol_key',
'net_access_key', 'c_sign_key', 'peer_disc_mac_addr']
```

Рисунок 3.20 – Таємні ключі

Також, будемо аналізувати всі версії Android та динаміку зміни стану витоків у тій чи іншій версії рисунок 3.21

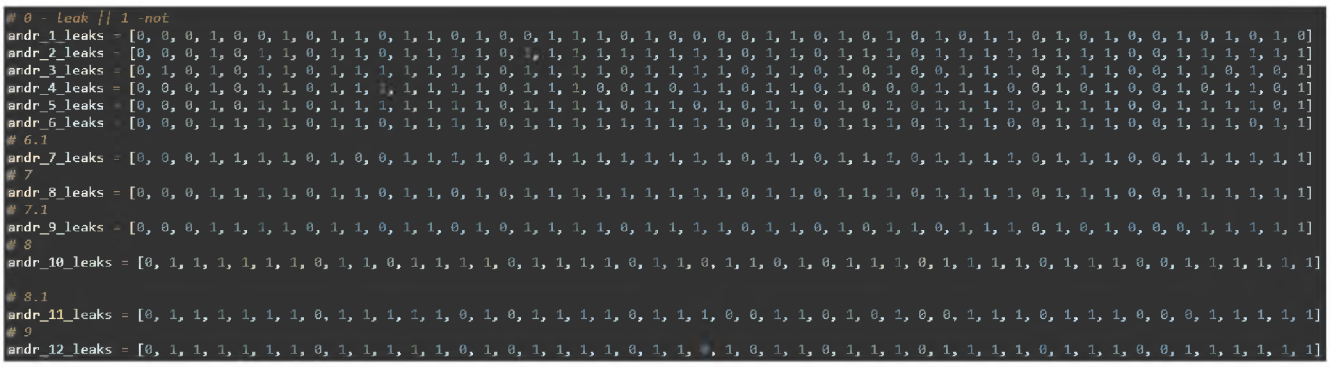

Рисунок 3.21 – Відомості про витік ключів за версіями Android

Для збіру даних з відкритого коду будемо використовувати застосунок, що побудований на використанні REST API OpenGrock. Ресурс де розвернутий OpenGrock на коді Android – AndroidXRef.

Отже визначивши необхідні дані та зібравши їх використовуючи наявні застосунки та ресурси можна приступити до апробації алгоритмів машинного навчання.

Навчати будемо на всіх отриманих даних для всіх версій Android окрім 8.1 та 9.

Дерево рішень. Шаблон коду python для навчання рисунок 3.22

```
#Import Library
from sklearn.linear_model import LogisticRegression
#Assumed you have, X (predictor) and Y (target) for training data set and x_test(predictor) of test_da
taset
# Create logistic regression object
model = LogisticRegression()
# Train the model using the training sets and check score
model.fit(X, y)
model.score(X, y)
#Equation coefficient and Intercept
print('Coefficient: \n', model.coef_)
print('Intercept: \n', model.intercept_)
#Predict Output
predicted= model.predict(x_test)
```

Рисунок 3.22 – Реалізція алгоритму дерева рішень

Код навчання використовуючи алгоритм дерева рішень знаходиться в додатках даної роботи.

Аналіз отриманих даних для 8.1:

```
8.1: 0 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 1 0 1 1 1 0 0 1 1 0 1 0 1 0 0 1
     1 1 0 1 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1
8.1: 1 1 0 1 1 1 0 0 0 1 1 1 1 1
     1 1 1 1 1 1 1 0 1 1 1 1 1 1
```

33 з 49 правильних прогноза, що становить 67% правильних відповідей.

Аналіз отриманих даних для 9:

```
9: 0 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 1 0 1 1 0 1 0 1 1 0 1 1 1 0 1 1 1
   1 1 1 1 0 0 1 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0
9: 1 0 1 1 1 0 0 1 1 1 1 1 1
   1 1 1 0 1 1 0 1 1 1 1 1 1
```

32 з 49 правильних прогноза, що становить 65% правильних відповідей.

SVM. Шаблон коду python для навчання рисунок 3.23

```
#Import Library
from sklearn import svm
#Assumed you have, X (predictor) and Y (target) for training data set and x_test(predictor) of test_d
ataset
# Create SVM classification object
model = svm.svc() # there is various option associated with it, this is simple for classification. Yo
u can refer link, for mo# re detail.
# Train the model using the training sets and check score
model.fit(X, y)
model.score(X, y)
#Predict Output
predicted= model.predict(x_test)
```

Рисунок 3.23 – Реалізція алгоритму SVM

Код навчання використовуючи алгоритм дерева рішень знаходиться в додатках даної роботи.

Аналіз отриманих даних для 8.1:

```
8.1: 0 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 1 0 1 1 1 0 0 1 1 0 1 0 1 0 0 1
     0 1 1 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 0 0 0 1 1 0 0 0 0 0 1 0
8.1: 1 1 0 1 1 1 0 0 0 1 1 1 1 1
     1 1 1 0 1 0 0 0 0 1 1 1 0 1
```

32 з 49 правильних прогноза, що становить 65% правильних відповідей.

Аналіз отриманих даних для 9:

```
9: 0 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 1 0 1 1 0 1 0 1 1 0 1 1 1 0 1 1 1
   1 1 0 1 1 1 0 0 0 0 1 1 0 0 0 0 1 1 0 1 0 1 0 1 1 0 0 1 1 1 0 0 1 1 0 1
9: 1 0 1 1 1 0 0 1 1 1 1 1 1
   1 0 1 1 1 1 0 0 1 1 1 0 1
```

29 з 49 правильних прогноза, що становить 59% правильних відповідей.

Naive Bayes. Шаблон коду python для навчання рисунок 3.24

```
#Import Library
from sklearn.naive_bayes import GaussianNB
#Assumed you have, X (predictor) and Y (target) for training data set and x_test(predictor) of test_d
ataset
# Create SVM classification object model = GaussianNB() # there is other distribution for multinomial
 classes like Bernoulli Naive Bayes, Refer link
# Train the model using the training sets and check score
model.fit(X, y)
#Predict Output
predicted= model.predict(x_test)
```

Рисунок 3.24 – Реалізція алгоритму Naive Bayes

Код навчання використовуючи алгоритм дерева рішень знаходиться в додатках даної роботи.

Аналіз отриманих даних для 8.1:

```
8.1: 0 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 1 0 1 1 1 0 0 1 1 0 1 0 1 0 0 1
     1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 0 1 0 1 1 1
8.1: 1 1 0 1 1 1 0 0 0 1 1 1 1 1
     1 0 1 0 1 1 1 1 1 1 1 1 1 1
```

26 з 49 правильних прогноза, що становить 53% правильних відповідей.

Аналіз отриманих даних для 9:

```
9: 0 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 1 0 1 1 0 1 0 1 1 0 1 1 1 0 1 1 1
   1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1
9: 1 0 1 1 1 0 0 1 1 1 1 1 1
   1 1 0 1 1 1 1 1 1 1 1 1 1
```

34 з 49 правильних прогноза, що становить 69% правильних відповідей.

k- Nearest Neighbors. Шаблон коду python для навчання рисунок 3.25

```
#Import Library
from sklearn.neighbors import KNeighborsClassifier
#Assumed you have, X (predictor) and Y (target) for training data set and x_test(predictor) of test_d
ataset
# Create KNeighbors classifier object model
KNeighborsClassifier(n_neighbors=6) # default value for n_neighbors is 5
# Train the model using the training sets and check score
model.fit(X, y)
#Predict Output
predicted= model.predict(x_test)
```

Рисунок 3.25 – Реалізція алгоритму k- Nearest Neighbors

Код навчання використовуючи алгоритм дерева рішень знаходиться в додатках даної роботи.

Аналіз отриманих даних для 8.1:

```
8.1: 0 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 1 0 1 1 1 0 0 1 1 0 1 0 1 0 0 1
     1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 0 1 1 0 0 1 1
8.1: 1 1 0 1 1 1 0 0 0 1 1 1 1 1
     1 1 0 1 1 0 1 1 1 1 1 0 1 1
```

32 з 49 правильних прогноза, що становить 65% правильних відповідей.

Аналіз отриманих даних для 9:

```
9: 0 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 1 0 1 1 0 1 0 1 1 0 1 1 1 0 1 1 1
   0 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 0 0 1 1 1
9: 1 0 1 1 1 0 0 1 1 1 1 1 1
   1 0 1 1 1 1 1 1 0 0 0 1 1
```

36 з 49 правильних прогноза, що становить 73% правильних відповідей.

K-Means. Шаблон коду python для навчання рисунок 3.26

```
#Import Library
from sklearn.cluster import KMeans
#Assumed you have, X (attributes) for training data set and x_test(attributes) of test_dataset
# Create KNeighbors classifier object model
k_means = KMeans(n_clusters=3, random_state=0)
# Train the model using the training sets and check score
model.fit(X)
#Predict Output
predicted= model.predict(x_test)
```

Рисунок 3.26 – Реалізція алгоритму K-Means

Код навчання використовуючи алгоритм дерева рішень знаходиться в додатках даної роботи.

Аналіз отриманих даних для 8.1:

```
8.1: 0 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 1 0 1 1 1 0 0 1 1 0 1 0 1 0 0 1
     0 0 1 1 0 1 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 1 0 1 0 1 1
8.1: 1 1 0 1 1 1 0 0 0 1 1 1 1 1
     0 1 1 0 0 0 0 1 0 0 0 0 0 0
```

21 з 49 правильних прогноза, що становить 42% правильних відповідей.

Аналіз отриманих даних для 9:

```
9: 0 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 1 0 1 1 0 1 0 1 1 0 1 1 1 0 1 1 1
   0 0 1 1 0 1 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 0 1 0 0 1 0
9: 1 0 1 1 1 0 0 1 1 1 1 1 1
   1 1 0 0 0 0 1 0 0 0 0 0 0
```

22 з 49 правильних прогноза, що становить 44% правильних відповідей.

Отже результуюча картина зображена на рисунку 3.27

Алгоритм	Середній результат
Decision tree	66%
SVM	62%
Naive Bayes	61%
kNN	69%
K-Means	43%

Рисунок 3.27 – Результуючий відсоток правильних прогнозів для кожного алгоритму

Результати показали, що найбільш ефективними алгоритмами є Дерево Рішень та К найближчих сусідів. Найгірші прогнози дав алгоритм K-Means.

Висновки до розділу 3

В ході написання даної практичної частини було практично усунено витік таємного ключа – без використання машинного навчання, задля того щоб усвідомити природу витоків такого роду.

В отсанньому пункті даної роботи було проведено декілька спроб машинного навчання, використовуючи основні алгоритми машинного навчання, що вирушують задачу класифікації, що показали досить непогані результати по точності прогнозуваппня витоку ключа для коржної з тестових версій Android.

Проаналізувавши результуючу точність прогнозів кожного з алгоритмів, зясувалося, що найбільш ефективним для вирішення даної задачі є kNN, а найгіршим K-Means.

Отримані результати в даному розділі говорять про те, що аналіз, збір даних має відбуватись більш ретельно та на більшій вибірці даних, для збільшення точності прогнозів, застосовуючи алгоритм що найкраще проявив себе для поставленої задачі - kNN. Отже можна зробити висновок, що вирішення задачі автоматизації пошуку витокув секретних ключів, використовуючи машинне навчання є можливим і може застосовуватись на практиці.

4 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

4.1 Опис ідї проекту

Таблиця 4.1 - Опис ідеї стартап-проекту

Зміст ідеї	*Напрямки застосування*	*Вигоди для користувача*
Автоматизація пошуку витоку таємних ключів та їх усунення	Компанії, що сертифікують свої пристрої на відповідність до вимог MDFPP	Зменшення часу, що витрачається кожним співробітником для пошуку та знешкодження витоків
	Вендори, що пропонують свої послуги по сертифікації пристроїв	Зменшення часу, що витрачається кожним співробітником для пошуку та знешкодження витоків
	Лабораторії, що перевіряють пристрої на їх відповідність вимогам MDFPP	Зменшення часу, що витрачається кожним співробітником для пошуку та знешкодження витоків

Таблиця 4.2 - Визначення сильних, слабких та нейтральних характеристик ідеї проекту

1	*2*	*3*		*4*	*5*	*6*
№ п/п	*Техніко-економічні характеристики ідеї*	*(потенційні) товари/концепції конкурентів*		*W (слабка сторона)*	*N (нейтральна сторона)*	*S (сильна сторона)*
		Мій проект	*Вендори, що підтримують MDFPP*			
1.	економічні	Витрати на розробку рішення, – 3000 $	Витрати на розробку рішення, закупку ліцензій, сервіс - 250,000 $	Недостатній рівень витрат на маркетинг та рекламу	Схожий функціонал	Дешевша реалізація проекту. Часові переваги у пошуку та знешкодженні витоку таємних ключів
2.	технічні	Використання сучасних технологій в індустрії мобільної безпеки – машинне навчання	Ручний, не автоматизований алгоритми аналізу витоків	Складність у розробці програмного забезпечення	немає	Часові переваги у пошуку та знешкодженні витоку таємних ключів ої безпеки

Продовження таблиці 4.2.

1	2	3		5	6	7
3.	Техно-логічні	Використання механізмів, підходів та алгоритмів, що надають можливість автоматизції та пришвидшення пошуку витокуів теємних ключів	Відсутність таких компонентів	немає	Більші видатки на експлуатацію рішення	Підвищення якості та швидкості роботи компаній, що займаються сертифікацією мобільних пристроїв
4.	Ергоно-мічні	Можливість налаштування системи на будь-який таємний ключ	немає	Додаткова складність при налаштуванні	немає	Різноманітність способів використання
5	Органо-лептич-ні	Швидкість роботи залежить майєе миттєва	Швидкість роботи залежить лише від кваліфікації інженера	немає	Потребує оновлення бази сигнатур	Потребує постійного навчання

4.2Технологічний аудит ідеї проекту

Таблиця 4.3 - Технологічна здійсненність ідеї проекту

№ п/п	*Ідея проекту*	*Технології її реалізації*	*Наявність технологій*	*Доступність технологій*
1	Автоматизований пошук витоків таємних ключів	Використання алгоритмів класифікації машинного навчання	Усі необхідні технології є в наявності	Усі можливості по використанню - відкриті
2	Автоматизований пошук конкретного місця витоків таємних ключів	Використання алгоритмів класифікації машинного навчання	Усі необхідні технології є в наявності	Так, ця технологія є доступною
3	Усунення витоків памяті	Використання алгоритмів класифікації машинного навчання	Усі необхідні технології є в наявності	Усі можливості по використанню - відкриті
Обрана технологія реалізації ідеї проекту: так як для реалізації ідеї проекту, всі технології є наявними та доступними, тому обираються всі вище описані технології.				

Аналіз ринкових можливостей запуску стартап-проекту

Таблиця 4.4 - Попередня характеристика потенційного ринку стартап-проекту

1	*2*	*3*
№ п/п	*Показники стану ринку сертифікації*	*Характеристика*
1	Кількість головних гравців, од	NIST, NAIP, CC, MDFPP, APPLE, SAMSUNG
2	Загальний обсяг продаж, грн/ум.од	5 млн ум.од
3	Динаміка ринку (якісна оцінка)	Зростає

Продовження таблиці 4.4.

1	2	3
4	Наявність обмежень для входу	Потреба у кадрах із високим рівнем компетентності у сфері інформаційної безпеки
5	Специфічні вимоги до стандартизації та сертифікації	немає
6	Середня норма рентабельності в галузі, %	Не менше 100

Таблиця 4.5 - Характеристика потенційних клієнтів стартап-проекту

№ п/п	*Потреба, що формує ринок*	*Цільова аудиторія (цільові сегменти ринку)*	*Відмінності у поведінці різних потенційних цільових груп клієнтів*	*Вимоги споживачів до товару*
1	Виток таємних ключів на мобільних пристроях може призводити до втрати конфіденційних даних користувачів	Виробники мобільних пристроїв, а також клієнти, що купляють дані пристрої. Лабораторії та інститути, що приймають участь в сертифікації мобільних пристроїв	Для кожної категорій існують окремі цінності пропонованого рішення, наприклад виробникам мобільних пристроїв важливо швидко та якісно сертифікувати свої пристрої, лабораторіям, що перевіряють пристрої також.	- Захищений пристрій, який не дає змог зловмисника м викрасти інформацію

Таблиця 4.6 - Фактори загроз

№ п/п	*Фактор*	*Зміст загрози*	*Можлива реакція компанії*
1	Затрачений час на розробку може не відповідати поставленим цілям	Складність реалізації	Зниження вартості продукту

Таблиця 4.7 - Фактори можливостей

№ п/п	*Фактор*	*Зміст можливості*	*Можлива реакція компанії*
1	Поява нових технологій	Розширення спектру надаваних послуг для клієнтів, збільшення ефективності роботи сервісу,	Швидке впровадження нових технологій, створення нових сервісів
2	Залучення нових відомих компаній до співробітництва	Розширення використання пропонованого продукту	Розробка та впровадження нових засобів та програмних комплексів.

Таблиця 4.8 - Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	*В чому проявляється дана характеристика*	*Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)*
1. Олігополістична конкуренція	Динаміка цін, яка майже не залежить від рівню попиту на продукцію;	Автоматизація процесів усунення витокув для підвищення якості сервісу

2. За рівнем конкурентної боротьби - національний	Надання послуг для різних груп та типів клієнтів по всьому світу	Застосування технологій машинного навчання та підходів для вдосконалення рішення
3. За галузевою ознакою - міжгалузева	Рішення може використовуватися користувачами з галузей інформаційної безпеки	Впровадження нових функцій сервісу
4. Конкуренція за видами товарів: - товарно-видова	Рішення, що використовується для задоволення потреб клієнтів, та технологічно нове, в конкурентів такого немає	Надання інструментів для автоматичного пошуку витоків таємних ключів
5. За характером конкурентних переваг - цінова	Цінова	Надання нового інструменту
6. За інтенсивністю - марочна	Сукупність характеристик та властивостей рішення	Підвищення якості роботи інструменту

Таблиця 4.9 - Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	*Потенційні конкуренти*	*Постачальники*	*Клієнти*	*Товари-замінники*
Складові аналізу	APPLE, Huawei, LG, Google, Samsung	Розмір капіталовкладень; доступ до ресурсів у конкурентів; наявність патентів та товарних знаків.	немає	Розміри закупівель державних підприємств та органів влади;	Ціна товару та змінні витрати
Висновки:	Наявна досить інтенсивна конкурентна боротьба з боку прямих конкурентів	Є можливості входу на ринок, але існує чимало серйозних конкурентів, що вже працюють на цьому ринку.	немає	Клієнти диктують умови на ринку; при організації закупівель товарів та послуг, відчутний та його якості.	немає.

Даний проект має принципові можливості для роботи на ринку з огляду на конкурентну ситуацію. Серед сильних сторін, можна виділити використання нових технологій в мобільній сфері, надання інструменти із широким функціонал; залучення великих відомих компаній в якості постачальників чи клієнтів.

Таблиця 4.10 - Обґрунтування факторів конкурентоспроможності

№ п/п	*Фактор конкурентоспроможності*	*Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)*
1	Ціна	Ціноутворення, яке є однаковим для різних типів клієнтів
2	Гнучкість цін на продукт	Є можливість зменшення цін на продукт
3	Різномантіний фнкціонал	В порівнянні з конкурентами - функціонал є новим та іноваційним

Таблиця 4.11 - Порівняльний аналіз сильних та слабких сторін «Довірений монітор»

№ п/п	*Фактор конкурентоспроможності*	*Бали 1-20*	*Рейтинг товарів-конкурентів у порівнянні з довірений монітор*						
			–3	*–2*	*–1*	*0*	*+1*	*+2*	*+3*
1	Ціна	15			+				
4	Гнучкість цін на послуги	12		+					
5	Різномантіний фнкціонал	17		+					

Таблиця 4.12 - SWOT-аналіз стартап-проекту

Сильні сторони: нова технологія автоматизації, що може бути корисна для всіх клієнтів галузі	*Слабкі сторони*: низький рівень маркетингу;
Можливості: швидке впровадження нових технологій залучення великих відомих компаній у співробітництві	*Загрози*: цінова конкуренція;зниження доходів потенційних споживачів;

Таблиця 4.13 - Альтернативи ринкового впровадження стартап-проекту

№ п/п	*Альтернатива (орієнтовний комплекс заходів) ринкової поведінки*	*Ймовірність отримання ресурсів*	*Строки реалізації*
1	Побудова автоматизованої системи	80%	4 місяці

З означених альтернатив ринкового впровадження даного стартап-проекту було вирішено обрати побудову автоматизованої системи пошуку витоку таємного ключа, при цьому строки реалізації становитимуть приблизно 4 місяці.

Розроблення ринкової стратегії проекту.

Таблиця 4.14 - Вибір цільових груп потенційних споживачів

1	*2*	*3*	*4*	*5*	*5*
№ п / п	*Опис профілю цільової групи потенційних клієнтів*	*Готовність споживачів сприйняти продукт*	*Орієнтовний попит в межах цільової групи (сегменту)*	*Інтенсивність конкуренції в сегменті*	*Простота входу у сегмент*
1	Виробники мобільних пристроїв	Клієнти потребують продукт такого типу та готові ним користуватись	Високий попит, пов’язаний із необхідністю автоматизувати вузькі місця в процесі впровадження сертифікаційних вимог	Середній рівень конкуренції	немає
2	Лабораторії та центри сертифікації	Клієнти зацікавлені продуктом	Середній рівень попиту	Середній рівень конкуренції	немає
3	Інститути	Клієнти зацікавлені продуктом	Середній рівень попиту	Середній рівень конкуренції	немає

На підставі ринкової стратегії обрано використання стратегії диференційованого маркетингу.

Таблиця 4.15 - Визначення базової стратегії розвитку

№ п/п	*Обрана альтернатива розвитку проекту*	*Стратегія охоплення ринку*	*Ключові конкурентоспроможні позиції відповідно до обраної альтернативи*	*Базова стратегія розвитку*
1	Флангова атака	Стратегія диференційованого маркетингу	Сильні сторони та можливості рішення	Стратегія диференціації

Таблиця 4.16 - Визначення базової стратегії конкурентної поведінки

№ п/п	*Чи є проект «першопрохідцем» на ринку?*	*Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?*	*Чи буде компанія копіювати основні характеристики товару конкурента, і які?*	*Стратегія конкурентної поведінки*
1	Так	Так	Ні	Стратегія виклику лідера

Таблиця 4.17 - Визначення стратегії позиціонування

№ п/п	*Вимоги до товару цільової аудиторії*	*Базова стратегія розвитку*	*Ключові конкурентоспроможні позиції власного стартап-проекту*	*Вибір асоціацій, які мають сформувати комплексну позицію власного проекту*
1	Отримання системи, що дозволяэ автоматизовувати процеси пошуку втоку таэмного ключа в операційній системі Android, а також прогнозування витоків в нових версіях операційної системи	Стратегія диференціації	Новизна та унікальність. Автоматизація затратного по часу процесу.	Автоматизація процесу пошуку та знешкодження витоків таємних ключів

Розроблення маркетингової програми стартап-проекту

Таблиця 4.18 - Визначення ключових переваг концепції потенційного товару

1	*2*	*3*	*4*
№ п/п	*Потреба*	*Вигода, яку пропонує товар*	*Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити*
1	Автоматизація процесу пошуку витоків таємних ключів	Забезпечення захищеності даних користувача	Отримання єдиної системи, що автоматизує затратну по часу проблему пошуку та усунення витоків таємних ключів

Таблиця 4.19 - Опис трьох рівнів моделі товару

Рівні товару	*Сутність та складові*		
I. Товар за задумом	Автоматизація процесу пошуку витоків таємних ключів та їх усунення, а також прогнозування їх витоків		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Використання Machine Learning	М	Тх/Тл/Е
	2. Оновлення та виправлення для програмного забезпечення	Нм	Вр/Тл/Е
	4. Можливість налаштування логування	М	Тх/Е/Ор
	5. Швидкість роботи рішення для користувача	Нм	Тл/Е/Ор
	Якість: Забезпечення захищеності даних, що зберігаються на мобільниому пристрої		
	Надання користувачу можливості автоматизації та економії часу		
	Марка: автоматизація пошуку витокув		
III. Товар із підкріпленням	До продажу: для стимулювання попиту на продукт можна розробити програму надання тимчасового доступу до рішення.		
	Після продажу: Розробка та проведення рекламної кампанії		
За рахунок чого потенційний товар буде захищено від копіювання: захист інтелектуальної власності			

Таблиця 4.20 - Визначення меж встановлення ціни

№ п/п	*Рівень цін на товари-замінники*	*Рівень цін на товари-аналоги*	*Рівень доходів цільової групи споживачів*	*Верхня та нижня межі встановлення ціни на товар/послугу*
1	30000 ум.од	700 ум.од	Високий	Ціна на ліцензію– 700-700 ум.од.;

Таблиця 4.21 - Формування системи збуту

№ п/п	*Специфіка закупівельної поведінки цільових клієнтів*	*Функції збуту, які має виконувати постачальник товару*	*Глибина каналу збуту*	*Оптимальна система збуту*
1	прийняття рішення про необхідність автоматизовувати процеси, що потребують багато часу та людських ресурсів	пристосування збутової мережі до запитів споживачів; пошук перспективних засобів просування товарів; розробка та вдосконалення маркетингової політики; вибір посередників	Дворівневий канал збуту (із залученням посередника та дистриб'ютора)	Залучення компаній посередників та партнерів для формування системи збуту

Таблиця 4.22 - Концепція маркетингових комунікацій

№ п/п	*Специфіка поведінки цільових клієнтів*	*Канали комунікацій, якими користуються цільові клієнти*	*Ключові позиції, обрані для позиціонування*	*Завдання рекламного повідомлення*	*Концепція рекламного звернення*
1	Клієнти даного стартапу зацікавлені в отриманні систем по автоматизації	Канали інтегрованих маркетингових комунікацій	Використання сильних сторін рішення: машинне навчання; надання широкого функціоналу;швидке впровадження нових технологій	повідомлення, пов'язані з особистою вигодою аудиторії, що показують, як товар може задовольнят и потреби покупця	реклама, що демонструє якість товару, його економічність, цінність або можливості експлуатації

Висновки до розділу 4

Результатом проведеного аналізу та оцінки ризиків, було виявлено, що даний проект має можливість для ринкової комерціалізації. Для представленого рішення є високий рівень попиту, що пов'язаний із станом інформаційної безпеки користувачів, тому робота над проектом має гарну рентабельність на ринку послуг у сфері кібербезпеки. Для цього стартап-проекту є непогані перспективи входження в ринок, але наявні певні бар'єри, подолання яких підвищую конкурентоспроможність представленого рішення. Щодо альтернативи впровадження стартап-проекту та його ринкової реалізації, доцільно обрати механізми для побудови рішення з використанням компонентів однієї компанії постачальника.

ВИСНОВКИ

В даній дипломній роботі було висвітлено проблему витоку таємних ключів в операційній системі Android. Було проаналізовано потребу в сертифікації мобільних пристроїв на предмет їх відповідності MDFPP. Була проаналізована та детально обстежена вимога щодо зачистки таємних ключів.

Було проведено детальний аналіз природи витоку таємного ключа, його особливості та основні ознаки були виокремлені та виділені в групи даних, що використовувалися в подальших дослідженнях.

Була сформована та поставлена задача автоматизації процесу пошуку та знешкодження витоку таємного ключа в памяті пристрою.

Практична реалізація поставленої задачі в даній дипломній роботі була досягнута та протестована на 5 основних алгоритмах машинного навчання. Результати роботи кожного з алгоритмів є позитивні та такі, що можуть допомогти прискорити пошук та усунення таємних витоків ключів, а при збільшенні вибірки даних та точності прогнозування, є перспективи автоматичного попередження подібних витоків памяті.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЬ

1 NIAP, Protection Profile for Mobile Device Fundamentals [Электронный ресурс] – Режим доступу: https://www.commoncriteriaportal.org/files/ppfiles/PP_MD_V1.0.pdf

2 CC, Common criteria [Электронный ресурс] – Режим доступу: https://www.commoncriteriaportal.org/

3 Sunil Ray, Essentials of Machine Learning Algorithms [Электронный ресурс] – Режим доступу: https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/

4 Eugen Paraschiv, How Memory Leaks Happen in a Java Application [Электронный ресурс] – Режим доступу: https://stackify.com/memory-leaks-java/

5 Bluetooth Specification 5.0, [Электронный ресурс] – Режим доступу: https://www.bluetooth.com/specifications/bluetooth-core-specification

Додатки

```
import json
import numpy
import operator
from sklearn.preprocessing
import Normalizer
from sklearn.neighbors
import KNeighborsClassifier
from sklearn.naive_bayes
import GaussianNB
from sklearn.ensemble
import RandomForestClassifier
from sklearn.cluster
import KMeans
from sklearn
import decomposition
from sklearn
import svm
from sklearn.preprocessing
import StandardScaler
from sklearn
import tree
from sklearn.linear_model
import LogisticRegression

# 0 - leak || 1 - not
andr_1_leaks = [0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0]
andr_2_leaks = [0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1]
andr_3_leaks = [0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1]
andr_4_leaks = [0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1]
andr_5_leaks = [0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1]
andr_6_leaks = [0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1]# 6.1
andr_7_leaks = [0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1]# 7
andr_8_leaks = [0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1]# 7.1
andr_9_leaks = [0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1]# 8
andr_10_leaks = [0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1]

# 8.1
andr_11_leaks = [0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1]
# 9
andr_12_leaks = [0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1]

andr_leaks = [andr_1_leaks, andr_2_leaks, andr_3_leaks, andr_4_leaks, andr_5_leaks, andr_6_leaks, andr_7_leaks, andr_8_leaks, andr_9_leaks,
andr_10_leaks]

all_keys = keys + helpfull_keys
current_data_set = numpy.ndarray(shape = (len(all_keys), 31), dtype = float, order = 'F')

# Train on different algorithms

# model = tree.DecisionTreeClassifier(criterion = 'gini')
# model = svm.SVC(kernel = 'linear', C = 1, gamma = 1)
# model = GaussianNB()
# model = KNeighborsClassifier(n_neighbors = 2)
# model = KMeans(n_clusters = 2, random_state = 0)
idx_1 = 0
```

```
for andr in andr_leaks:
    idx = idx + 1
    idx_2 = 0

    with open('call_statistic.json') as f:
        data = json.load(f)
        for key, value in data.iteritems():
            current_data_set[idx_1][idx_2] = value
            idx_2 = idx_2 + 1

    with open('leaks_info.json') as f:
        data = json.load(f)

    # model.score(current_data_set, andr)
    model.fit(current_data_set, andr)

# predict for latest Android
andr_leaks = [andr_11_leaks, andr_12_leaks]
idx_1 = 0
for andr in andr_leaks:
    idx = idx + 1
    idx_2 = 0

    with open('call_statistic.json') as f:
        data = json.load(f)
        for key, value in data.iteritems():
            current_data_set[idx_1][idx_2] = value
            idx_2 = idx_2 + 1

    predicted = model.predict(current_data_set)
    print(predicted)
```