Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра кібербезпеки
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Виявлення шкідливих програм для операційної системи Android з використанням методу факторизації матриць

Виконав: студент VI курсу, групи СБм-61
спеціальності 125 Кібербезпека та захист інформації
(шифр і назва спеціальності)

(підпис) Ковальчук О.А. (прізвище та ініціали)

Керівник (підпис) Баран І.О. (прізвище та ініціали)

Нормоконтроль (підпис) (прізвище та ініціали)

Завідувач кафедри (підпис) Загородна Н.В. (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль - 2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра кібербезпеки
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Загородна Н.В.
(підпис) (прізвище та ініціали)
«__» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 125 Кібербезпека та захист інформації
(шифр і назва спеціальності)

Студенту Ковальчуку Олександру Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Виявлення шкідливих програм для операційної системи Android з використанням методу факторизації матриць

Керівник роботи Баран Ігор Олегович, к.т.н., доц., декан ФІС
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «20» 11 2024 року № 4/7-1112

2. Термін подання студентом завершеної роботи 26.12.2024 р.

3. Вихідні дані до роботи наукові літературні джерела

4. Зміст роботи (перелік питань, які потрібно розробити)
1. Аналіз предметної області.
2. Теоретична частина.
3. Практична частина.
4. Охорона праці та безпека в надзвичайних ситуаціях

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Тема роботи. 2. Актуальність. 3. Мета, задачі, об'єкт, предмет дослідження. 4. Наукова новизна, практичне значення роботи. 5. Типи (категорії) шкідливих програм під ОС Android. 6. Історичний розвиток теми виявлення шкідливих програм для Android 7. Характеристики безпеки застосунків. 8. Машини факторизації.. 9. Машини факторизації з урахуванням специфіки полів. 10. Порівняння моделей машин факторизації. 11. Схема обробки Android застосунків. 12. Фрагменти лістингу програмного коду 13. Точність класифікації навчених моделей. 14. Основні результати дослідження Обчислення результату комбінації

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Осухівська Г.М., зав. каф. КС		
Безпека в надзвичайних ситуаціях	Клепчик В.М., проректор з АГРБ		

7. Дата видачі завдання 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	20.11 – 22.11	*Виконано*
2.	Підбір джерел про виявлення шкідливих програм для		
	операційної системи Android	23.11 – 26.11	*Виконано*
3.	Опрацювання джерел про виявлення шкідливих		
	програм для операційної системи Android з		
	використанням методу факторизації матриць	27.11 – 30.11	*Виконано*
4.	Виконання дослідження щодо розробки системи		
	виявлення шкідливих програм для операційної системи		
	Android із використанням методу факторизації матриць	01.12 – 06.12	*Виконано*
5	Розробка алгоритмів функціонування системи	07.12 – 10.12	
6.	Оформлення розділу «Аналіз предметної області»	11.12 – 13.12	*Виконано*
7.	Оформлення розділу «Теоретична частина»	14.12 – 15.12	*Виконано*
8.	Оформлення розділу «Практична частина»	16.12 – 18.12	*Виконано*
9.	Виконання завдання до підрозділу «Охорона праці та		
	безпека в надзвичайних ситуаціях»	06.12 – 16.12	*Виконано*
10.	Оформлення кваліфікаційної роботи	14.12 – 19.12	*Виконано*
11.	Нормоконтроль	18.12 – 20.12	*Виконано*
12.	Перевірка на плагіат	16.12 – 19.12	*Виконано*
13.	Попередній захист кваліфікаційної роботи	17.12 – 20.12	*Виконано*
14.	Захист кваліфікаційної роботи	27.12	

Студент _____ Ковальчук О.А.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Баран І.О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Виявлення шкідливих програм для операційної системи Android з використанням методу факторизації матриць // Ковальчук Олександр Андрійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем та програмної інженерії, кафедра кібербезпеки, група СБм-61 // Тернопіль, 2024 // с. – 73, рис. – 16, табл. – 5, слайд. – 14, бібліогр. –35.

Досліджено проблему безпеки мобільних пристроїв під управлінням операційної системи Android. Є потреба у безперервному покращенні вже наявних та створенні нових безпекових систем. Проаналізовані методи виявлення шкідливих застосунків Android, наведена їх класифікація від Google.

Розглянуто характеристики безпеки Android- застосунків. Встановлено, що найзручнішим способом одержання атрибутів із Android застосунків є Python-модуль Androguard. Досліджено метод факторизації матриць і заснована на ньому модель машинного навчання - машини факторизації, а також її розширення - машини факторизації з урахуванням специфіки полів.

Реалізовані моделі машинного навчання на базі машин факторизації. Сформовано набори даних для навчання моделей та їх подальшого тестування. Перебором параметрів одержано графіки, котрі демонструють залежність точності виявлення шкідливих програм від значень параметрів. Здійснено оцінювання точності та часу роботи кожної реалізації машин факторизації. Встановлено, що реалізація стандартної моделі в модулі fastFM швидше здійснює навчання та тестування при великому ранзі факторизації. Реалізація ж xLearn відмінно проявила себе в точності класифікації за рахунок адаптивного навчання на кожній з епох і перевершує fastFM за будь-якого рангу факторизації.

Ключові слова: Android, виявлення шкідливих застосунків, машини факторизації, облік специфіки полів, безпека мобільних пристроїв

ANNOTATION

Detection of malware for Android operating system using the matrix factorization method // Kovalchuk Oleksandr // Ternopil Ivan Pul'uj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Cyber Security // Ternopil, 2024 // p. - 73, Fig. - 16, Table - 5, Slides - 14, References - 35.

The problem of the security of mobile devices under the control of the Android operating system has been studied. There is a need for continuous improvement of already existing and creation of new security systems. The methods of detecting malicious Android applications are analyzed, and their classification by Google is given.

The security characteristics of Android applications are considered. It has been established that the most convenient way to obtain attributes from Android applications is the Python module Androguard. The method of matrix factorization and the model of machine learning based on it - factorization machines, as well as its extension - factorization machines taking into account the specificity of fields, were studied.

Implemented machine learning models based on factorization machines. Data sets for model training and their further testing have been created. By sorting through the parameters, graphs were obtained that demonstrate the dependence of the accuracy of detection of malicious programs on the parameter values. The accuracy and running time of each implementation of the factorization machines were evaluated. It is established that the implementation of the standard model in the fastFM module performs training and testing faster with a large factorization rank. The xLearn implementation excelled in classification accuracy due to adaptive learning at each epoch and outperforms fastFM at any rank of factorization.

Keywords: Android, malware detection, factorization machines, field-aware, mobile security

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ СКОРОЧЕНЬ І ТЕРМІНІВ

API (application programming interface) - прикладниий програмний інтерфейс.

APK (Android Package Kit) – формат архівних файлів-застосунків для Android.

ART (Android RunTime) – нова, поки що експериментальна віртуальна машина.

Dalvik – віртуальна машина, частина мобільної платформи Android.

DEX (Dalvik Executable) - компактний виконуваний формат, призначений для систем з обмеженим обсягом пам'яті та швидкістю процесора.

kNN (k-nearest neighbor method) – метод k-найблиижчих сусідів.

SVM (support vector machine) – метод опорних векторів.

БД – база даних.

ІБ – інформаційна безпека.

МН – машинне навчання.

ОС – операційна система.

ПЗ – програмне забезпечення.

ШНМ – штучна нейронна мережа.

ЗМІСТ

ВСТУП

Актуальність теми. Мобільні пристрої, зокрема смартфони, зараз нерозривно пов’язані із життям людей. Для більшості користувачів власне у них міститься більшість приватної інформації, як то різні фото та відео, аккаутни банківських карток, паролі від різноманітних спеціалізованих застосунків.

Крім особистих даних на смартфонах може бути і корпоративна інформація, що становить комерційну таємницю. І тут, при її розкритті, людині, відповідальній за її зберігання в таємниці, або, якщо інформація сумлінно зберігалася в таємниці, то зловмиснику, котрий розкрив дані, може загрожувати відповідальність аж до кримінальної.

Незважаючи на всю значимість захисту інформації на мобільних пристроях, в Україні практично не існує нормативних документів, що містять приписи щодо забезпечення пристроїв захисним програмним, апаратним або програмно-апаратним забезпеченням при їх виробництві, про навіть, використанні в особистих або комерційних цілях, і відповідно, які зобов'язують і регламентують дотримуватися таких приписів, зважаючи на їх відсутність . Це призводить до того, що компанії, що виробляють, не мають зобов'язань перед встановлення захисних засобів і можуть зовсім не впроваджувати їх на вироблені мобільні пристрої. У зв'язку з чим зловмисники отримують можливість скористатися вразливістю мобільних пристроїв, у тому числі смартфонів, щоб отримати особисту або комерційну інформацію власника пристрою .

Одним із шляхів вирішення цієї проблеми може стати надійна система розпізнавання шкідливих застосунків, яка не дозволяє користувачеві встановити на смартфон шкідливу програму, а магазинам застосунків або іншим сервісам, що взаємодіють з застосунками - не допустити її розповсюдження.

Від якості та надійності такої системи залежатиме можливість зловмисника отримати конфіденційні дані, порушити цілісність чи доступність тієї чи іншої інформації. Тому проектована система повинна бути максимально точною, стійкою до відмови і не містить вразливості, що дозволяють обійти її.

Мета дослідження: виявлення шкідливих застосунків для ОС Android з використанням моделей, заснованих на факторизації матриць.

В роботі поставлено та розв’язано наступні задачі:

– аналіз існуючих робіт на тему виявлення шкідливих Android - застосунків;

– виділення значущих для безпеки характеристик Android - застосунків;

– створення навчальної вибірки шляхом обробки наявних наборів даних;

– розробка методу виявлення шкідливих Android -застосунків на основі факторизації матриць;

– програмна реалізація запропонованого методу як прототипів;

– навчання та порівняння результатів виявлення шкідливих Android - застосунків розроблених прототипів.

Об’єкт дослідження: програми для ОС Android.

Предмет дослідження: виявлення шкідливих Android- застосунків з використанням нейронних мереж, що містять у своїй основі метод факторизації матриць.

Методи дослідження: наукові праці закордонних та вітчизняних учених за тематикою дослідження, фундаментальні положення кібербезпеки та ШНМ.

Наукова новизна отриманих результатів:

– використання розширеного набору характеристик безпеки Android - застосунків у виявленні шкідливих застосунків за допомогою моделей, заснованих на факторизації матриць;

– застосування моделі ШНМ «Машина факторизації з урахуванням специфіки полів» для виявлення шкідливих Android -застосунків;

– порівняння практичних реалізацій моделі ШНМ «Машина факторизації» з точки зору виявлення шкідливих Android - застосунків.

Практичне значення одержаних результатів. Полягає у розробці та поданні результатів системи виявлення шкідливих застосунків із застосуванням машини факторизації з урахуванням специфіки полів. Також, практична значимість міститься у поданні результатів порівняння практичних реалізацій машин

факторизації.

Апробація. Результати дослідження апробовано на XII науково-технічній конференції «Інформаційні моделі, системи та технології» у вигляді опублікованих тез.

Структура роботи. Робота складається з пояснювальної записки та графічної частини. Пояснювальна записка складається з вступу, 4 розділів, висновків, списку використаної літератури та застосунків. Обсяг роботи: пояснювальна записка – 73 арк. формату А4, графічна частина – 14 слайдів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Питання безпеки мобільних пристроїв

Різного роду мобільне обладнання швидко стали загальновизнаним і доступним для усього світу, так як вона є зручним замінником стаціонарного. Відповідно до [1], число користувачів смартфонів за 5 років, з 2020 року по 2024, зросло з 5,92 мільярдів до 7,2 що практично дорівнює загальному числу людей, котрі проживають на Землі, а згідно із прогнозом на 2025 рік - сягне 7.43 мільярда користувачів. Кількість проданих за рік смартфонів, відповідно до [2], з 2015 по 2019 роки перебувала на рівні десь близько 1.5 мільярди, але вже починаючи з 2020 року ця цифра поступово зменшувалася аж до 1.33 мільярда штук. Це пов’язано, перш за все, із пандемією COVID-2019, хоча за прогнозами, вже у 2024 році ринок продажів мобільних пристроїв відновиться та повернеться на рівень до пандемії.

Мобільний пристрій є достатньо загальним поняттям, котре містить як смартфони, так і ноутбуки, а також інші сучасні пристрої, котрі можуть досить просто переміщати. У відповідності з ОС, функцій, котрі він виконує, апаратних і програмних засобів, будь-який з таких елементів володіє своїми особливостями щодо ІБ і має розглядатися самостійно, а не разом з іншими пристроями, так як є підсумком зв’язку значного числа визначених частин складного обчислювального приладу.

Проблеми забезпечення безпеки будь-якого виду мобільного пристрою мають досліджуватися та розв’язуватися упродовж усього його життєвого циклу – починаючи від створення прототипу, і аж до фінального пристрою, що вийшов у реліз. Окрім того, будь-яка апаратна та програмна частина пристрою мають бути досліджені окремо.

Інколи мобільний пристрій, із точки зору ІБ, ймовірно розглядати і комплексно, але це можливо лише при його представленні замовникам чи при створенні розгорнутого звіту щодо безпеки пристрою. У всіх решту моментах, коли складові частини пристрою є компонентами багатьох схожих пристроїв,

тоді їх необхідно розглядати без зв’язку один з одним. Саме тому у кваліфікаційній роботі і розглядатиметься окремий елементи мобільних пристроїв - це ОС Android, а зокрема - Android -застосунки.

Особлива зацікавленість до Android -пристроїв з боку ІБ в першу чергу викликаний найбільшим поширенням даної ОС серед мобільних пристроїв. Відповідно до [3], платформа Android утримує більше, ніж 50% ринку мобільних пристроїв вже навіть до 2014 року, а станом на 2024 рік має значну перевагу – близько 70% від загального ринку мобільних пристроїв – перед iOS.

Левова частка обох ОС, Android та iOS, це різного роду програми , починаючи від ігор і закінчуючи системними налаштуваннями. Проте, на противагу iOS, котра дає змогу інсталювати програми тільки із свого чітко керованого магазину App Store, ПЗ на платформу Android можна інсталювати як із стандартного місця - магазину Google Play, так і з яких-небудь інших варіантів, як то сайти, магазини, застосунки. При цьому перед інсталяцією користувач побачить попередження щодо небезпеки інсталяції програм із тих місць, котрі викликають недовіру, та рекомендацію дати можливість інсталювати з таких місць встановленням відповідного регулювання. Якщо таке налаштування буде виставлено, тоді будь-яка програма, котра взята з якого-небудь неконтрольованого місця, зможе бути інстальована без зайвих доказів .

Така власне "відкритість" ОС Android є не тільки у інсталяції застосунків, проте і в інших моментах її функціонування, для прикладу, у наданні дозволів застосункам. Програмна відкритість є іншим н’юансом платформи - усякий розробник має змогу змінити систему, таким чином з’являться нові проблеми разом із оновленнями. Окрім того, ОС Android співпрацює із виробниками пристроїв, котрі також можуть вносити помилки і вразливості разом зі змінюванням платформи.

Google Play має величезну кількість різних програм. Відповідно до [4], у вересні 2024 року їхня кількість десь 4 мільйони.

Для якісного фільтрування та контролю застосунків на безпеку необхідна високоточна система, котрою Google Play Protect не є через наявність шкідливих програм, котрі безперервно з’являються і детектуються у магазині Google Play.

Зокрема у 2024 році було виявлено близько 10 шкідливих застосунків, здатних обходити двофакторну авторизацію для банківських застосунків [5].

У 2013 році компанія Google повідомила [6], що питання безпеки ОС Android дуже прибільшена сторонами, які зацікавлені в цьому. Проте, навіть якщо в 2013 році це ймовірно і могло бути правдою через те, що Android мав тільки 40% ринку, а число користувачів смартфонів заледве було більше одного мільярду, тоді як число застосунків у Google Play було в районі одного мільйона, то в даний час картина абсолютно інша - перераховані показники збільшилися втричі, а в Google Play стабільно поступають нові шкідливі застосунки. Більше того, згідно статистичних даних [7], число шкідливих застосунків стало значно збільшуватися після 2013 року властиво через ріст популярності платформи Android.

1.2 Виявлення та класифікація шкідливих застосунків

Темі виявлення та класифікації шкідливих програм приділено велика кількість робіт, статей, доповідей. Системи виявлення шкідливих застосунків досить поширені, вони входять до складу антивірусних засобів, є частиною великих сервісів, що взаємодіють зі сторонніми програмами , наприклад, таких, як Google Play. Тому такі системи мають високу цінність як для дослідників, так і для компаній, що їх використовують.

Прийнято виділяти такі методи виявлення шкідливих застосунків :

- статичний;
- динамічний;
- гібридний.

Далі будуть поглиблено розглянуті перелічені методи та відповідні їм роботи для системи Android.

Статичний метод виявлення заснований на аналізі файлу застосунку і не враховує поведінку програми під час роботи. Шляхом аналізу вихідних кодів і маніфесту програми виділяються значущі характеристики безпеки, або

здійснюється пошук шкідливого коду і на основі отриманої інформації робиться висновок про безпеку програми.

Так, у роботах [8] і [9] розглядаються методи статичного виявлення, заснованого на характеристиках безпеки застосунків і графах передачі управління, а в [10] - конкретно на пошуку сигнатур. Однак дані системи практично не застосовні до реальних систем у зв'язку зі складністю їх автоматизації та не кращими результатами виявлення.

Динамічний метод заснований на аналізі поведінки програми. Застосунок запускається в пісочниці для того, щоб запобігти негативним наслідкам і на основі аналізу викликів застосунком сторонніх API, системних викликів, звернення до пам'яті та мережі, виявляє зловмисні дії програми.

Гібридний метод поєднує статичний та динамічний методи і робить рішення про безпеку застосунку як на основі статичних даних, так і динамічних. У більшості сучасних систем виявлення шкідливих застосунків використовується саме цей метод. Прикладом такого методу є [11], де до статичного та динамічного методів додається ще й кластеризація. Система показала досить високу ефективність.

В основному, системи виявлення та класифікації працюють на наступних принципах:

- сигнатурне виявлення;
- обчислення умовних ймовірностей;
- MH;
- SVM;
- Random Forest;
- kNN;
- Naïve Bayes.

Системи сигнатурного виявлення працюють на основі порівняння бази виявлених раніше сигнатур шкідливих застосунків та вихідних кодів аналізованого застосунку. Враховуються можливості зміни сигнатур, обфускування коду, проте дані системи не працюватимуть під час шифрування

вихідного коду програми. Через це нині дані системи мало використовуються і реалізуються.

Системи, що працюють із умовними ймовірностями, заздалегідь обчислюють умовні ймовірності появи характеристик безпеки у вихідних кодах та маніфесті для шкідливих та безпечних програм. При аналізі застосунку враховуються раніше обчислені ймовірності для кожної характеристики безпеки та вираховується ймовірність того, що аналізований застосунок є шкідливим. Динамічні системи, що працюють на даному принципі, обчислюють умовні ймовірності критичних дій програми та враховують їх під час аналізу.

Системи, що працюють на основі МН найбільш популярні в даний час завдяки їх точності виявлення та можливості класифікації шкідливих застосунків. Найбільшу складність у цьому випадку становлять:

– навчання систем;

– вибір оптимальних параметрів.

Складність навчання для різних систем наведена в табл. 1.1 [12], де n - кількість навчальних прикладів, d - розмірність вхідного вектора.

Таблиця 1.1 – Складність навчання систем виявлення та класифікації

Класифікатор	Складність
Random Forest	$O(nd \log n)$
SVM	$O(max(n, d), min((n, d)^2))$
kNN	$O(ndk)$
Naïve Bayes	$O(nd)$

Існує велика кількість реалізованих систем на основі МН, що працюють з різними методами виявлення шкідливих застосунків, одні з таких систем наведені в [13] та [14].

Крім того, DroidMat [15] виконує статичний аналіз файлу маніфесту та вихідного коду програми Android для отримання характеристик безпеки,

включаючи дозволи, апаратні ресурси та виклики API. Потім він використовує метод k-means і класифікацію k найближчих сусідів (k-NN) для виявлення шкідливих програм.

DREBIN [16] витягує аналогічні характеристики з файлу маніфесту та вихідного коду програми та використовує SVM для класифікації шкідливих програм на основі one-hot кодування відповідних характеристик.

У багатьох роботах досліджується та реалізується пошук шаблонів зловмисної поведінки за допомогою графів потоків управління або графів викликів. AppContext [17] класифікує програми, використовуючи МН на основі контекстів, які витягуються у разі виявлення підозрілої поведінки програми. Створюється граф викликів з вихідного коду програми та витягується контекст за допомогою аналізу потоків інформації всередині застосунку. Потім AppContext отримує особливі характеристики безпеки з потягненого контексту, які потім використовуються в алгоритмах МН. У статті [17] було проаналізовано 633 безпечні програми з магазину Google Play та 202 зразки шкідливих програм. AppContext вірно ідентифікував 192 шкідливі програми з точністю 87,7%.

У [18] так само використовували графи викликів для виявлення шкідливих програм. Після вилучення графів викликів із застосунків, у їх реалізації застосовується графічне ядро і за лінійний час із графіків викликів витягуються характеристики безпеки. Отримані характеристики подаються на вхід SVM, яка в свою чергу класифікує застосунок як шкідливий або безпечний. Було проведено експеримент із 136 тисячами безпечних та 12 тисячами шкідливих застосунків. У рамках експерименту система правильно виявила 89% шкідливих програм з 1% помилкових спрацьовувань. Недолік даного методу в тому, що він залежить від точності вилучення графа викликів.

Незважаючи на відносно позитивні результати перерахованих робіт, нині навіть такі системи, як FlowDroid [19] та IC3 [20], все ще не можуть повністю вирішити проблему побудови міжкомпонентних потокових графів управління (ICFG), особливо потоків, що створюються намірами та фільтрами намірів .

Значною перевагою даних систем є те, що класифікувати рідкісні застосунки вони можуть як за заздалегідь виділеним класифікуючим

властивостями , так і за тим, що вони самі виділять у процесі навчання.

Системи виявлення шкідливих застосунків, засновані на МН, маючи можливість працювати як на основі результатів статичного, так і динамічного аналізу, а отже, і гібридного, є потужним, розширюваним і найефективнішим інструментом на даний момент. У зв'язку з цим такі системи активно розвиваються і вдосконалюються, а дослідники намагаються досягти їх найбільшої точності виявлення, застосовуючи різні методи МН.

При розгляді питання безпеки мобільних пристроїв не можна не згадати про типи шкідливих програм. Google виділяє 14 категорій, що стосуються Android пристроїв [21]. Вони входять як стандартні типи шкідливих застосунків:

– використовують чорні ходи;
– викликають відмову в обслуговуванні;
– завантажувачі;
– здійснюють фішинг;
– підвищують привілеї;
– здирники;
– розсилають спам;
– шпигунські програми;
– троянські програми

Так і типи, що стосуються конкретно платформи або мобільних пристроїв загалом:

– програми-шахраї з виставленням рахунків - це тип шахрайства з відправкою платних SMS- повідомлень, здійсненням дорогих дзвінків та підпискою на платні телефонні послуги;

– комерційне шпигунське ПЗ - призначені для стеження за пристроєм, передачі особистої інформації з пристрою без повідомлення та згоди користувача;

– програми, що виконують рутинг, тобто отримання пристроєм прав суперкористувача root, без попередження;

- програми, що містять нестандартну для Android пристрою шкідливу функціональність, а також містять загрози для інших форм плат.

1.3 Стислий огляд літературних джерел

Як і говорилося раніше, тема виявлення шкідливих програм для системи Android є досить розвиненою і розглядається в багатьох роботах. У цьому розділі будуть описані найбільш примітні з праць, які певною мірою вплинули на хід подальшого розвитку теми досліджень без небезпеки Android. На рис. 1 наведено тимчасову пряму, що відображає розвиток теми виявлення шкідливих застосунків для системи Android

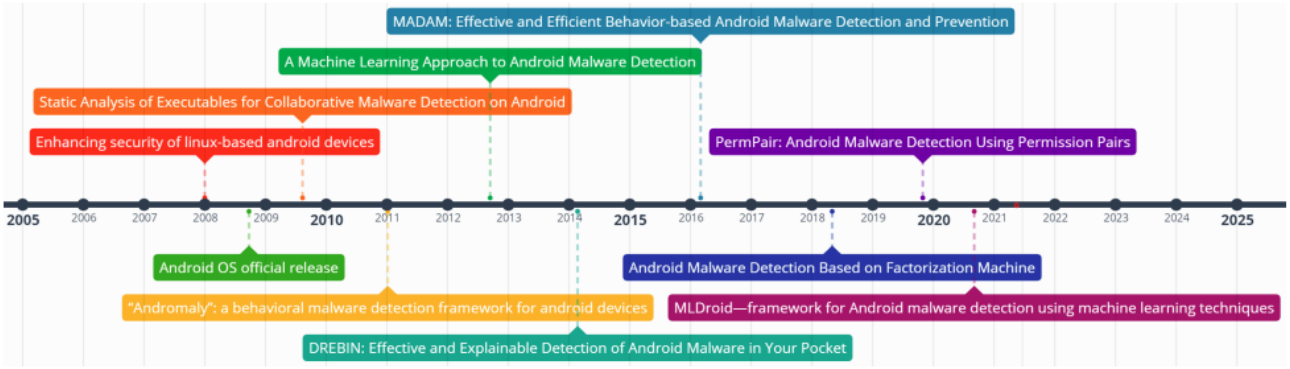

Рисунок 1.1 – Історичний розвиток теми виявлення шкідливих програм для Android

Дослідники перейнялися питаннями безпеки ОС Android і стали пропонувати різні рішення для її вдосконалення ще до її офіційного випуску. У січні 2008 року було представлено першу роботу [22] з виявлення шкідливих застосунків, розвиток якої було представлено в 2009 році [23], коли стало можливо оцінити перші офіційні версії платформи Android. Роботи були написані Обрі-Дерріком Шмідтом та компанією з 7 осіб з Берліна та Стамбула [22, 23]. Обидві роботи спираються на більш ранні праці, що стосуються інших мобільних систем та пристроїв, де пропонувалося використання сервера, що здійснює аналіз безпеки застосунків. Так, кожен застосунок, встановлений на пристрій, відповідно до робіт, повинен пересилатися на сервери компанії, що подає операційну систему і аналізуватись там. У роботах [22, 23] пропонується аналогічний підхід, але з удосконаленням системи Android так, що ще й усі

пристрої забезпечувалися модулем аналізу застосунків, який заснований на алгоритмах PART, Prism і Найближчих сусідів. Таким чином розумілася миттєва перевірка програми та розсилка всім пристроям інформації про дану програму.

Крім того, в роботі [23] для розширення можливостей забезпечення безпеки системи пропонується використання Linux -утиліт: антивіруса, fire wall, детекторів руткітів та інших інструментів, що використовуються для аналізу, виявлення та запобігання ризиків безпеки на системному рівні.

Мінусами даних підходів стали: використання сторонніх пристроїв, тобто серверів, які мають підтримуватися кимось; використання зайвих ресурсів мобільних пристроїв, у тому числі заряду батареї, що було особливо критично для того часу; розсилка інформації про застосунки повинна була здійснюватися через мобільні мережі, або мережі Wi-Fi, які ще не були поширені, для цього знадобився б додатковий протокол з постійною синхронізацією всіх пристроїв, що працюють під ОС Android, що знову викликало б додатковий витрата ресурсів; розширення вбудованої системи Linux та підтримка роботи сторонніх утиліт.

Хоча запропоновані покращення і не були реалізовані в Android, це був перший крок до створення систем виявлення шкідливих застосунків, який започаткував розвиток однієї з найпопулярніших тем для дослідження в даний час.

У 2010 році була опублікована робота Томаса Блізинга та інших представників німецького університету [24], де для динамічного аналізу було запропоновано використання пісочниць, в яких працюють Android -застосунки. У роботі запропоновано гібридний спосіб виявлення - під час статичного аналізу шукаються та логуються підозрілі та потенційно небезпечні функції та дозволи, під час динамічного аналізу підраховується та логується кількість системних викликів за час роботи програми.

Мінусами запропонованої системи є: додаткове навантаження на пристрій під час гібридного аналізу; тільки логування інформації, без реагування ; використання суб'єктивної інформації для логування - передбачається , що вона може вказувати на шкідливу поведінку, але все залежить від рішення людини,

яка оцінює логи.

Наприкінці роботи зазначено, що з даної системи доцільно використовувати МН виключення людини з оцінки логів докладання. Це було першим згадкою МН виявлення шкідливих застосунків і, безсумнівно, правильним.

До 2011 року було написано кілька робіт, які певною мірою пов'язані з МН. Так, у роботі Ікера Бургера та інших [25] було запропоновано фреймворк Crowdroid, який заснований на алгоритмі k-means. Фреймворк здійснює моніторинг системних викликів і створює вектор, що містить кількість відповідних системних викликів, задіяних у застосунку. Потім, використовуючи алгоритм k-means, фреймворк видає вердикт - чи є застосунок троянським конем.

Основним мінусом фреймворку є те, що алгоритм k-means не навчається заздалегідь, як це робиться в МН, він використовує результати , що отримуються від клієнтів. У зв'язку з цим, доки не буде утворено набір із достатньої кількості прикладів, алгоритм не буде точним. Через цю особливість випливають і інші проблеми: по-перше, результати з боку клієнтів можуть бути недостовірними, тоді система буде скопрометована; по-друге, алгоритм k-means завжди ділитиме вектори даних на k класів, у разі 2 - шкідливі і сумлінні докладання, навіть якщо у наборі даних алгоритму будуть лише сумлінні докладання.

Описані проблеми вирішуються використанням попереднього навчання системи, тобто використанням МН. Однак, у 2010-2011 роках ще не було достатньо даних про реальні шкідливі Android- застосунки, через що створення такої системи було неможливо.

У роботі Асафа Шабтая та інших [26] було вперше застосовано МН виявлення аномалій у системі Android. Було використано кілька алгоритмів для порівняльного аналізу. Даними для навчання та виявлення були системні метрики - споживання ЦП, використання мережі Wi-Fi та інші. Так як прикладів шкідливих застосунків не було, були створені 4 застосунки, що реалізують DoS атаки та крадіжки даних.

Це була перша робота, яка використовувала МН, але практична частина мала ряд мінусів - маленький, штучно створений набір навчальних прикладів,

зняття метрик, що відносяться не до конкретних застосунків, а до всього пристрою та інші мінуси. Але завдяки виявленому інтересу до МН вже в 2010-2011 роках, у наступні роки і до нашого часу стали активно з'являтися роботи, які використовують різні набори даних та алгоритми МН для досягнення найкращих результатів у виявленні шкідливих застосунків.

У 2012 році Яджин Чжоу і Сюйсянь Цзян [27] вперше описали шкідливі програми, знайдені протягом 2010-2011 років на платформі Android. Дані програми розташовувалися як в офіційних магазинах, так і в сторонніх. У роботі було перевірено 4 основні на той час антивіруси і всі вони показали досить слабкий результат - від 80% до 20% виявлених шкідливих застосунків. Логічним висновком з цієї роботи стало те, що необхідно створення надійної системи виявлення шкідливих застосунків, доступною для Android -пристроїв. Крім того, була представлена статистика використання шкідливими програмами Android - дозволів і зроблено висновок, що потрібно введення більш детальних дозволів, так як на той момент всі вони були досить спільними . Згодом система поступово забезпечувалася все більшою кількістю вужчих дозволів.

У роботі Джастіна Сахса і Латіфура Хана [28] використано найбільш підходящу для того часу модель МН One -Class SVM. Її перевага полягає в тому, що майже не маючи прикладів одного з класів, в даному випадку це шкідливі програми, систему можна навчити на прикладах другого класу, сумлінних застосункух, і при появі прикладу, що сильно відрізняється від навчальних, буде встановлено, що він є є доносним застосунком. Для такої системи потрібна велика кількість характеристик безпеки, тому в роботі пропонується використання всіх рішень, а також графа потоку управління.

У 2014 році Даніель Арп та інші [16] запропонували легковаговий фреймворк для Android -пристроїв на основі SVM, який показав досить хороші результати. Крім того, для навчання моделі був створений датасет Drebin, який став одним з найпопулярніших для використання в навчанні моделей для виявлення шкідливих Android -застосунків.

У 2016 році була представлена система виявлення та запобігання шкідливим програмам MADAM від Андреа Саранчино та інших [29]. Система

працює на Android -пристроях у вигляді програми і, що відрізняє її від інших систем, не тільки виявляє, але й запобігає роботі підозрілих програм на контрольованому пристрої. Для виявлення аномального поведінки система контролює системні виклики, відправлення SMS, користувальницьку активність та метадані застосунків. Відповідно, для такого контролю потрібні права рівня суперкористувача, тому контрольований пристрій має бути рутованим, що є великим мінусом і обмеженням для використання.

У 2018 році Ченлінь Лі та інші [30] опублікували роботу, в якій представлена система виявлення шкідливих застосунків, заснована на машині факторизації - однією з моделей МН, що розглядаються в даній роботі. Дослідження, представлені в даній випускній кваліфікаційній роботі мотивовані саме працею [30].

Аншул Арора та інші [31] у 2019 році запропонували систему, засновану на парах дозволів, які є найбільш вагомими для шкідливих програм. У роботі використовується графова структура, для якої поступово виконуються різні оптимізації з обчисленням ваги для кожної з пар. Підсумкові значення ваг використовуються для класифікації застосунків на часові і сумлінні.

Арвінд Махіндру та А. Л. Сангал [32] у 2020 році представили роботу, в якій описано веб-систему, що використовує 21 метод МН для отримання найкращих результатів виявлення шкідливих Android - застосунків. Крім того, були досліджені різні функції та ознаки, а також їхній вплив на точність виявлення.

1.4 Висновки до першого розділу

Досліджено, що популярність мобільних пристроїв, особливо тих, що функціонують під керуванням ОС Android, перебуває на максимальному розвитку в останній час. Число мобільних користувачів різко виросло упродовж останніх років та все-ще зростає в наш час. Число зловмисників, котрі розробляють різного роду шкідливі програми, відповідно також аналогічно зростатиме через збільшення числа потенційних жертв. Саме тому і необхідно

безперервне вдосконалення вже наявних і розроблення новітніх систем безпеки.

Також були проаналізовані методи виявлення шкідливих застосунків Android і наведена їх офіційна класифікація від Google. Класифікація включає як стандартні типи шкідливих застосунків, так і такі, що стосуються конкретно до мобільних пристроїв і Android зокрема.

Було досліджено розвиток теми виявлення шкідливих застосунків для системи Android. З кількості робіт і глибини опрацювання в них теми можна зробити висновок, що вона є досить розвиненою і продовжує вдосконалюватися до цього дня. З часом точність виявлення пропонованих систем збільшується, вони оптимізуються і можуть використовуватися на пристроях з малим обсягом ресурсів.

2 ТЕОРЕТИЧНА ЧАСТИНА

2.1 Характеристики безпеки в Android -застосунках

Властиво під характеристиками безпеки Android-застосунків будемо вважати такі компоненти програми, котрі торкаються аспектів безпеки Android пристроїв і можуть бути використані зловмисниками для порушення ІБ пристрою:

– крадіжки конфіденційної інформації;

– отримання віддаленого доступу до пристрою;

– заподіяння шкоди як самому пристрою, так і його програмним компонентам.

Загальна схема обробки Android програми для виділення характеристик безпеки складається з наступних кроків:

– розпакування та декомпіляція вихідних кодів програми;

– виділення характеристик безпеки з файлу маніфесту та декомпільованих кодів;

– кодування характеристик безпеки у вхідний вектор для навчання та тестування нейромережевої моделі.

2.2 Класифікація

Всі характеристики безпеки застосунків можна розділити на категорії відповідно до їхніх ролей:

– список компонентів програми. Компоненти декларуються у файлі маніфесту програми та оголошують різні інтерфейси для взаємодії як з кінцевим користувачем, так і з ОС Android. Імена даних компонентів використовуються для ідентифікації відомих шкідливих застосунків за його компонентами;

– використовувані апаратні засоби. Якщо програма потребує доступу до таких апаратних засобів пристрою, як GPS, камера, сенсори, то дані

характеристики повинні бути оголошені у файлі маніфеста програми. Надання прав доступу до деякого набору апаратних засобів може вказувати на можливість витоку чутливих даних, наприклад, задекларований доступ до GPS і мережі може вказувати на передачу зловмисникам розташування пристрою;

– дозволи програмам. Android використовує механізм дозволів для контролю та збереження конфіденційності користувача. Дозволи використовуються для надання прав застосунку до чутливих даних, компонентів апаратних засобів та доступу до сторонніх бібліотек та інтерфейсів взаємодії застосунків. Шкідливі програми використовують спеціальні набори дозволів за тим самим принципом, як і апаратні засоби. Крім того, є набір як звичайних дозволів, так і небезпечних. Їхня відмінність полягає в тому, що небезпечні дозволи запитуються у користувача безпосередньо під час роботи програми, при цьому попереджаючи його про небезпеку надання таких дозволів. Саме на основі великої кількості небезпечних дозволів, що прошиваються, будується більшість шкідливих програм, тому вони враховуються в більшості систем виявлення та класифікації шкідливих застосунків;

– фільтри намірів (intent filter). Intent - намір програми зв'язатися з іншою програмою або компонентом для отримання або передачі будь-якої інформації. Це основний спосіб "спілкування" різних застосунків, як системних, так і прикладних. Фільтр намірів використовується для того, щоб відкидати непотрібні для застосування наміри від інших застосунків і компонентів і приймати наміри, що проходять за критеріями запитаних дій, категорій намірів і даних, що передаються. Як правило, шкідливі програми роблять фільтрацію намірів для отримання тільки чутливої та дійсно потрібної інформації.

Крім стандартних категорій характеристик безпеки, які можна отримати з маніфесту програми, можна виділити особливі, які допоможуть точніше визначити програму як шкідливу та містяться безпосередньо в коді програми:

– повідомлення (broadcast receiver). Широкомовні повідомлення, аналогічно намірам, містять інформацію, що розсилається, застосункум і компонентам, проте, на відміну від намірів, спрямовані не на конкретний застосунок, а на всі, підписані на відповідні повідомлення. Якщо застосунок,

якому направлено широкомовне повідомлення не запущено, воно запускається, обробляє повідомлення і в залежності від результату обробки або завершується, або продовжує роботу, запускаючи відповідну активність або сервіс. Прикладом широкомовного повідомлення може бути повідомлення від контролера батареї про низький заряд батареї пристрою. Відповідно, шкідливий застосунок, підписуючись на широкомовні повідомлення може контролювати стан пристрою, застосунків та їх компонентів$

– постачальники контенту (content provider). Є загальними БД, що надаються програмами для інших програм. Їх прикладами можуть бути браузери, контакти, медіа-застосунки, налаштування. Відповідно, постачальники можуть зберігати чутливу інформацію, яка є метою для зловмисників, тому шкідливі програми не рідко запитують дані у постачальників;

– вбудовані API. Система Android надає список стандартних API для роботи з чутливими даними, доступ до яких надається за допомогою дозволів. По-перше, якщо виклик даних API здійснюється без запиту на це дозволу, то, можливо, ця програма є root експлойтом. По-друге, виклик функцій API, що працюють з чутливими даними, може також вказувати на шкідливість програми;

– сторонні API. При їх використанні деякі функції можуть виконуватися без надання відповідних дозволів з різних причин. Знаючи про існування таких, що представляють небезпеку API і виявляючи їх у застосункух, можна припускати, що такі застосунки є шкідливими;

– дозволи, які запитуються в коді програми. Виділяючи не тільки запити з файлу маніфесту, але і ті, що конкретно використовуються при виклику API та їх відповідних функцій, можна доповнити список раз рішень, так як надання прав може здійснюватися не тільки з файлу маніфесту, але і з коду програми;

– граф потоку управління. Виконання кожної програми відбувається послідовно - функція за функцією. Таким чином, можна створити граф, який відображатиме послідовність викликів функцій програми. При його візуалізації стає зрозуміло, як працює застосунок, яким є його алгоритм. Однак, в Android застосунках може бути не одна точка входу, а безліч - він може запускатися шляхом його відкриття користувачем, може виконувати завдання у вигляді

сервісу, може відкриватися і здійснювати дії при отриманні широкомовних повідомлень і так далі. Кількість точок входу залежатиме від кількості оголошених інтерфейсів взаємодії в застосунку. З точки зору виявлення шкідливих застосунків стає цікаво сформувати послідовність викликів API функцій, оскільки одні й ті ж API можуть однаково часто з'являтися і в шкідливих, і в сумлінних застосункух, однак порядок їх виклику може вказати на можливі шкідливі дії програми;

– динамічні показники. Витягуються під час динамічного аналізу, дозволяють виявити аномалії, відхилення і підозрілу або шкідливу поведінку при роботі застосунків. Такими характеристиками можуть бути споживані застосунком ресурси пристрою, частота та адреси звернення до мережі Інтернет, а також дані, що передаються при цьому, відправлення та прийняття SMS, виконання криптографічних операцій, підвантаження DEX модулів і безліч інших операцій, які можуть вказувати на шкідливість програми.

2.3 Методи пощуку

Інсталяційний APK файл програми - це стислий архів, який містить вихідний код програми, файли ресурсів, метадані, додаткові бібліотеки та файл маніфесту програми. Вихідний код закодований у файл з розширенням DEX, який інтерпретується і виконується віртуальною машиною Dalvik, яка в даний час замінена на ART, але виконує ті ж функції. Dalvik і ART - це середовища виконання застосунків , реалізовані в ОС Android. Файл маніфесту містить основну інформацію про застосунок, а також різні оголошення та специфікації, яких потребує застосунок. Файлами ресурсів є зображення, HTML файли, рядки, що використовуються в застосунку.

Оскільки файли DEX скомпільовані в виконуваний двійковий код, вони не призначені для читання або інтерпретації, вихідний код не може бути вилучений з них безпосередньо. Тому, DEX файли потрібно дизассемблювати і декомпілювати у вихідні формати, які можна буде вважати і інтерпретувати. Такі формати представляються Smali кодом (при дизассемблюванns) і Java кодом

(при декомпіляції). Smali код - байт-код машини Dalvik, який є такою ж низькорівневою мовою, як і асемблер (з цього процес перекладу DEX файлу в Smali код називають дизассемблюванням).

Для отримання дизассембльованого та декомпільованого коду є безліч утиліт. Один із сценаріїв декодування для ручного аналізу програми може представлятися наступним чином:

– розархівація APK файлу та декомпіляція вихідних кодів програм мій Apk Manager;

– конвертація DEX файлу у формат JAR Java коду;

– відображення отриманого Java коду утиліті jd-gui.

Крім цього, можна скористатися універсальною утилітою ApkTool, функціонал якої є досить широким і зручним для отримання вихідних кодів для подальшої роботи з ними за допомогою інших програм.

В описаних методах доведеться додатково використовувати сторонній парсер, або реалізовувати його самому, крім цього можуть виникнути проблеми з передачею результатів аналізу в програму для їх обробки.

Однак, існують пакети, написані мовою програмування Python для аналізу та декомпіляції файлів APK. Їх перевага полягає в тому, що результати аналізу можуть бути передані у вигляді структур даних безпосередньо в потрібну програму, модуль або функцію, яка вже буде працювати з результатами аналізу. Найбільш примітний пакет в даному випадку - Androguard, який надає широкий функціонал і примітний тим, що в ньому вбудований парсер, завдяки якому можна отримати потрібні компоненти з APK файлу програми, в тому числі перераховані раніше характеристики без небезпеки, для їх подальшого використання в оцінці безпеки застосунку .

2.4 Кодування

Так як характеристики безпеки є рядками, або структурами даних, а нейронні мережі працюють з числами і векторами, то після їх отримання шляхом аналізу APK з'являється завдання кодування характеристик для подальшої подачі

в систему виявлення для її навчання, тестування або перевірки програми.

Подання такої великої кількості даних, як характеристики безпеки, є розумним у вигляді N-вимірного бінарного вектора. Кожен застосунок представляється таким N-вимірним бінарним вектором, де N - кількість характеристик безпеки, що використовуються системою виявлення шкідливих застосунків.

Для конкретизації припустимо, що всі коли-небудь вилучені характеристики формують набір характеристик S розміру |S|, тоді можна представити кожен APK файл у вигляді двійкового вектора довжини |S|, елементи якого рівні 1 у випадку, якщо вони задіяні в цьому застосунку , і 0 - в іншому випадку.

Приклад такого кодування представлений на рис. 2.1, на ньому представлений і приклад формування набору характеристик S з характеристик застосунків A і B.

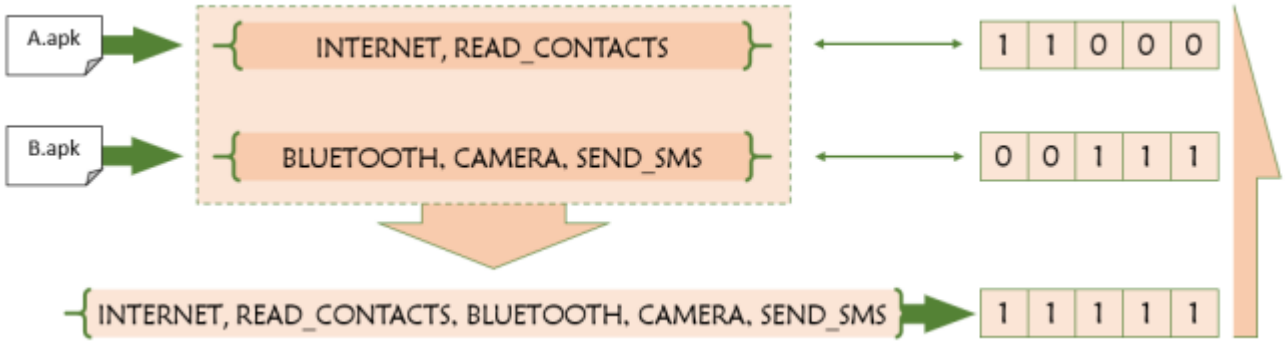

Рисунок 2.1 – Кодування характеристик безпеки

Даний метод кодування називається One-hot кодуванням, коли деякому набору елементів зіставляється бінарний вектор, розмір якого дорівнює кількості елементів і кожен елемент даного вектора однозначно пов'язаний з елементом набору. Тоді кожен об'єкт, який містить певний набір цих елементів, може бути представлений у вигляді такого бінарного вектора, що містить 1 на позиціях елементів, що належать даному об'єкту та 0 на інших.

2.5 Метод факторизації матриць

Факторизація (розкладання) матриці - це метод, який дозволяє спростити роботу з матрицею шляхом її розкладання на дві скорочені матриці. Даний метод широко застосовується в комп'ютерних обчислювальних системах, оскільки дозволяє спростити і прискорити роботу як над матрицями великого розміру, так і над сильно розрідженими матрицями, котрі містять більшість нульових елементів.

На практиці існує безліч методів розкладання матриць, які застосовуються як для розв'язання задач лінійної алгебри, так і для інших задач, наприклад, для побудови моделей МН.

У цій роботі найбільший інтерес представляє розкладання Холецького. Дане розкладання може застосовуватися до симетричної позитивно визначеної матриці, а результатом розкладання буде нижня трикутна матриця та матриця, що має вигляд її транспонування.

Приклад такого розкладання представлений на рис. 2.2. Однак, основна перевага даного методу полягає в тому, що при розкладанні можна сильно скоротити розмір результуючих матриць без втрати точності.

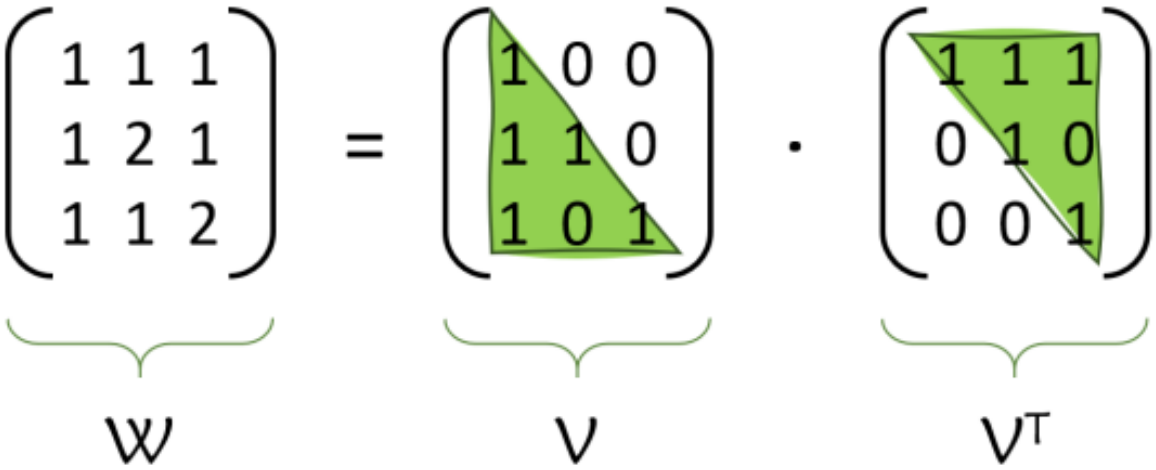

Рисунок 2.2 – Приклад розкладу Холецького симетричної додатньо визначеної матриці

2.5.1 Машини факторизації

Для стандартизованих моделей ШНМ детектування шкідливих застосунків за вектором атрибутів безпеки не береться до уваги "взаємодія" цих атрибутів. Є незаперечним той факт, що коли застосунок має декілька небезпечних атрибутів, поєднання якихось із них може недвозначно свідчити про шкідливість програми - ось це і є, фактично, сенсом врахування “взаємодії” атрибутів безпеки застосунків.

“Взаємодія” атрибутів показана їх попарним впливом один на одного за правилом кожен-з-кожним, як підсумок створюється матриця розміром n^2, тут n - чисельність атрибутів. Так як атрибутів є дуже багато, тому саме зберігання матриці розміром їх числа ще й в квадраті вимагатиме значних обсягів пам'яті, а беручи до уваги той факт, що на кожному етапі навчання ШНМ елементи матриці мають піддаватися зміні (іншими словами проходитиме постійна взаємодія із такою об’ємною конструкцією), навчання потребуватиме не лише значних затрат пам'яті, але й часових.

Вперше машини факторизації були презентовані Штеффеном Рендлом [33] у 2010 році. Вони є класом новітніх нейромережевих моделей, котрі містять плюси як SVM, так і факторизаційних моделей. Машини факторизації здатні вирішувати задачі регресії, класифікації та ранжування. Відмінна риса машин факторизації – це опрацювання яких-небудь вхідних векторів, котрі репрезентують об'єкти реального середовища, і оцінці парних взаємодій їх складових хоч і за значної розрідженості таких векторів, на противагу SVM, котрі не здатні діяти із сильно розрідженими даними чи здатні частково, проте із дуже низькою точністю.

Додатково до вже згаданих переваг машина факторизації має ще одну - лінійна складність $O(kn)$, тут k - гіперпараметр, котрий характеризує розмірність факторизації, а n - число параметрів об'єкта, котрий описують (властиво це розмір вхідного вектора).

Найкраще застосування машини факторизації отримали у т.зв. рекомендаційних системах, котрі мають враховувати власне парні взаємодії, разом з тим маючи властиво значно розріджені дані. Проте, більша частина

даних, котрі представляють об'єкти реального середовища, є дуже розрідженими через присутність множин можливих атрибутів, однак мають тільки незначну їх частину. Аналогічно і Android застосунки - усі володіють величезною множиною, залежно від вибору, параметрів безпеки, проте деякі застосунки, здебільшого, містять тільки декілька параметрів і зовсім рідко біля тисячі параметрів, хоча це все всього на всього мала частка їх загального числа.

Математична модель взаємодії складових частин x із вагами w представляється як поліноміальна регресії порядку два (2.1):

$$h(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} W_{ij} x_i x_j, \quad (2.1)$$

де W - матриця ваг взаємодії x_i та x_j.

Як було описано вище, робота з такою матрицею не є можливою, однак, матриця W може бути розкладена методом Холецького як (2.2):

$$W = V V^T. \quad (2.2)$$

Якщо представити v_i як i-й рядок V, ШНМ тренуватиме прихований вектор власне v_i для кожного x_i та ваги моделі парної взаємодії записати через скалярні добутки властиво відповідних прихованих векторів для x_i та x_j (2.3):

$$h(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle x_i x_j \quad (2.3)$$

де скалярний добуток векторів v розмірності k обчислюється за формулою (2.4)

$$\langle v_i, v_j \rangle = \sum_{m=1}^{k} v_{i,m} v_{j,m}. \quad (2.4)$$

Формула (2.3) і є математичним поданням моделі машини факторизації. Тут параметри w_0, w_i, v_i піддаються тренуванню та оновленню із застосуванням стохастичних методів, котрі забезпечують псевдовипадкові зміни вагових величин, зберігаючи при цьому ті зміни, котрі призводять до покращень.

Приховані вектори v_i створюються під час навчання ШНМ і переважно їх розмірність *k y* багато разів менше за *n* - розмір вхідного вектора *x,* через його власне розрідженість.

Машини факторизації можуть бути розширені і до поліноміальної регресії більшого порядку для дослідження взаємодії більш ніж двох ознак об'єкта за формулою (2.5)

$$h(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{l=2}^{d} \sum_{i_1=1}^{n} \dots \sum_{i_l=i_{l-1}+1}^{n} \left(\prod_{j=1}^{l} x_{i_j} \right) \left(\sum_{f=1}^{k_l} \prod_{j=1}^{l} v_{i_j,f}^{(l)} \right) \quad (2.5)$$

У такому випадку складність дорівнює *O* (κ_d n^d), але при математичному перетворенні рівняння (2.5) складність зводиться все до тієї ж лінійної *O(kn).* Доведення цього наведено у роботі [33].

2.5.2 Машини факторизації з урахуванням специфіки полів

Моделі, що працюють за принципом машин факторизації з урахуванням полів, є розширенням стандартної моделі машин факторизації. Вони були запропоновані Рендлом та Шмідом-Тімом [34] у 2010 році, а також Андреасом Тешером та іншими на конкурсі KDD Cup у 2012 році. Проте, конкретна концепція машин факторизації з урахуванням полів була сформована Ючин Цзюанем та іншими [35] у 2016 році.

Для машин факторизації з урахуванням специфіки полів крім

характеристик, що використовуються у стандартній моделі, вводиться додатковий параметр - поле. Поле - це клас, група, чи будь-яке інше поєднання показників. Так, наприклад, конкретні дозволи Android застосунків (READ_CONTACTS, INTERNET і т. д.) є характеристиками, а полем для них усіх може бути «Дозвіл». На відміну від досить конкретних характеристик поля можуть задаватися довільно, наприклад, дозволи можуть бути поділені на такі поля як «Компоненти пристрою» (CAMERA, BLUETOOTH тощо), «Мобільний зв'язок» (CALL_PHONE, READ_SMS тощо).), та інші, залежно від потрібної гранулярності.

На відміну від стандартної машини факторизації, де з кожною характеристикою був пов'язаний лише один прихований вектор v_i , у розширеній моделі з урахуванням специфіки полів, при навчанні, для кожної характеристики створюється стільки прихованих векторів, скільки введено полів, і поліноміальна регресія другого порядку буде мати вигляд (2.6):

$$h(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_{i,f_s}, v_{j,f_r} \rangle x_i x_j, \qquad (2.6)$$

де f_s - поле характеристики x_j, а f_r - поле характеристики x_i .

Для кращого розуміння цієї концепції можна розглянути приклад на основі наведених вище полів та характеристик. Припустимо, модель побудована на трьох характеристиках з відповідними полями, наведеними в табл. 2.1.

Таблиця 2.1 – Характеристики Android застосунків, поділені на відповідні поля

Компоненти пристрою (КУ)	Стільниковий зв'язок (МС)	Розташування (МП)
CAMERA (CAM)	CALL PHONE (CP)	ACCESS FINE LOCATION (AFL)

Тоді для стандартної моделі машини факторизації взаємодія характеристик для довільного застосування буде описуватися виразом (2.7):

$$\langle v_{CAM}, v_{CP} \rangle x_{CAM} x_{CP} + \langle v_{CAM}, v_{AFL} \rangle x_{CAM} x_{AFL} + \langle v_{AFL}, v_{CP} \rangle x_{AFL} x_{CP}, \quad (2.7)$$

де x n дорівнюватиме 1, якщо характеристика n міститься в застосунку, і 0 в в іншому випадку.

Відповідно, для машини факторизації з урахуванням специфіки полів взаємодія характеристик виглядатиме як (2.8):

$$\langle v_{CAM,\text{МС}}, v_{CP,\text{КУ}} \rangle x_{CAM} x_{CP} + \langle v_{CAM,\text{МП}}, v_{AFL,\text{КУ}} \rangle x_{CAM} x_{AFL} + \langle v_{AFL,\text{МС}}, v_{CP,\text{МП}} \rangle x_{AFL} x_{CP}. \quad (2.8)$$

Тут приховані вектори v зовсім інші - якщо в першому випадку для характеристики CAMERA прихований вектор v_{CAM} був одним для взаємодій x_{CAM} з x_{CP} і x_{CAM} з x_{AFL}, то в другому випадку, у зв'язку з тим, що x_{CP} і x_{AFL} належать до різних полів, для характеристики CAMERA створюється два приховані вектори – $v_{CAM,МС}$ і $v_{CAM,МП}$.

2.5.3 Порівняння моделей

Незважаючи на те, що модель машин факторизації з урахуванням специфіки полів відрізняється від стандартної моделі лише збільшеною кількістю прихованих векторів, що навчаються, вони будуть по-різному проявляти себе в залежності від вирішуваної задачі. Тоді вибір найбільш підходящої моделі для відповідного завдання гарантуватиме підвищення точності класифікації, регресії та ранжирування. Тому важливо виділити умови, за яких варто використовувати ту чи іншу модель машин факторизації.

Переваги та недоліки кожної моделі наведені у табл. 2.2.

Таблиця 2.2 – Порівняння моделей машин факторизації

		Стандартна модель	Модель з урахуванням специфіки полів
Час навчання та розпізнавання (при однаковому параметрі k)		Менше	Більше
Використовувана пам'ять (при однаковому параметрі k)		Менше	Більше
Розмірність k прихованих векторів, за якої досягається найбільша точність		Більше	Менше
Набір даних	Щільність	Будь-який	Сильно розріджений
	Роздільність	Будь-який	На велику кількість класів (полів)
	Тип даних	Будь-який	Категоріальні

Розглядаючи це питання з боку використовуваних ресурсів, тобто займаної пам'яті і часу, що витрачається на тренування і розпізнавання, машини факторизації з урахуванням специфіки полів поступаються стандартної моделі при однакових параметрах моделей у зв'язку з тим, що стандартна модель навчає і зберігає лише один прихований вектор для кожної характеристики, а розширена - по вектору на кожне поле.

З іншого боку, у зв'язку з тим, що машини факторизації з урахуванням специфіки полів навчають приховані вектори щодо окремих полів, а не на основі всіх характеристик, як у стандартній моделі, розмірність k прихованих векторів може бути в рази менша за тієї ж точності розпізнавання на відміну від звичайних машин факторизації.

Крім того, в оригінальній роботі [35] доведено, що машини факторизації з урахуванням специфіки полів будуть показувати кращі результати за достатньої розрідженості даних, в яких може бути виділено досить велику кількість полів. У цій роботі модель була протестована як на категоріальних, так і на числових

даних і зроблено висновок, що для числових даних результати роботи машин факторизації з урахуванням специфіки полів не краще, ніж стандартної моделі.

2.6 Висновки до другого розділу

Таким чином, в даному розділі було досліджено характеристики безпеки Android застосунків. Було виділено 11 груп показників відповідно до їх ролі в застосунку і на пристрої. Зроблено висновок про те, що найбільш зручним способом отримання характеристик з Android застосунків є Python модуль Androguard завдяки його широким можливостям. Описано метод One-hot кодування для формування вхідного вектора системі виявлення шкідливих застосунків з доступних у застосунку характеристик безпеки.

Також був досліджений метод факторизації матриць і заснована на ньому модель МН - машини факторизації, а також її розширення - машини факторизації з урахуванням специфіки полів. Математичною основою моделей, крім факторизації, є поліноміальна регресія другого порядку для дослідження парної взаємодії характеристик і вищого порядку для, відповідно, взаємодій вищого порядку. Було зроблено порівняння моделей і зроблено висновок, що розширена модель повинна застосовуватися у разі сильної розрідженості даних, які мають бути категоріальними, а також розділеними на достатню кількість полів (класів). Дані Android застосунків підходять під перелічені умови з чого можна зробити висновок, що при застосуванні моделі машин факторизації з урахуванням полів точність виявлення шкідливих застосунків повинна бути вищою, ніж у стандартної моделі.

3 ПРАКТИЧНА ЧАСТИНА

3.1 Розробка системи виявлення

Система виявлення шкідливих Android –застосунків, котра реалізується, є бінарними класифікаторами, так як проблема виявлення заснована тільки на двох класах - шкідливих та безпечних застосунків. Таким чином, не потрібно використання складних моделей, що виконують множинну класифікацію.

Весь код написаний мовою програмування Python через його простоту і велику кількість реалізованих модулів. Модулі, що включаються в реалізовані системи, не підтримують ОС Windows, тому створення систем виявлення відбувалося на ОС Ubuntu 18.04.

Для створення навчальної вибірки, а також самого навчання потрібно якомога більше оперативної пам'яті і доступних CPU, інакше процес буде знищений ядром. Тому було виділено віртуальну машину з ОС Ubuntu. Системі було виділено 32 Гб ОЗП та 8 ядер CPU від процесора Intel Xeon CPU E7-8860 з робочою частотою 2.27 ГГц.

Для розробки систем насамперед треба проаналізувати доступні набори даних і вже реалізовані бібліотеки МН для того, щоб уникнути виконання роботи, яка була зроблена раніше в інших роботах.

3.1.1 Датасети

Для систем виявлення шкідливих Android застосунків датасетами є самі програми. Найбільшою складністю є те, що більшість наборів є закритими через вміст у них шкідливих прикладів і до них складно отримати доступ. Крім того, якщо набір даних є не надійним, то потрібно додатково проводити аналіз застосунків на шкідливість або безпеку, що містяться в ньому, за допомогою додаткових засобів - антивірусів, систем виявлення та інших інструментів.

Одним з найпопулярніших наборів даних є DREBIN [16]. Цей набір містить зразки шкідливих програм, зібраних з 2010 до 2012 року. До нього включено 5560 застосунків із 179 сімейств шкідливих програм. DREBIN є

закритим набором даних і отримати доступ до нього можна тільки зв'язуючись з його власниками поштою.

Для ще двох популярних закритих датасетів, Genome [27] і AMD [36], зовсім були припинені видачі дозволів на доступ і отримати зараз їх неможливо. Однак це були великі та якісні набори. Genome містив понад 1200 застосунків, що покривають більшість існуючих на той момент сімейств шкідливих програм і зібраних з 2010 по 2011 роки. В AMD входило 24553 зразки, розділених на 135 різновидів серед 71 сімейства шкідливих програм, зібраних у період з 2010 по 2016 рік.

Найбільший на даний момент з відомих наборів даних – Andro-Zoo [37]. Він постійно розширюється і на середину 2024 року містить більше 24 мільйонів застосунків. AndroZoo збирає програми з кількох джерел, у тому числі, з офіційного магазину Android застосунків Google Play. Після попадання в БД AndroZoo застосунок аналізується десятками різних антивірусів для однозначного визначення того, чи є застосунок безпечним.

VirusShare - великий закритий репозиторій, що містить зразки шкідливих застосунків різних сімейств під основні системи. Виявлені програми під ОС Android розділені на роки згідно з їх публікацією. Загалом у репозиторії міститься понад 200 тисяч шкідливих Android застосунків. Доступ до них можна отримати шляхом надсилання запиту електронною поштою.

Набори даних Канадського інституту кібербезпеки UNB знаходяться у відкритому доступі та містять велику кількість репозиторіїв, створених на базі інституту. Репозиторії містять як самі застосунки - шкідливі та безпечні, так і сформовані CSV файли, що містять вектора характеристик застосунків і готові до подачі в системи МН.

Крім того, програми містяться і в магазинах Android застосунків - Google Play і на альтернативних платформах для розповсюдження застосунків. Однак при їх ручному зборі не можна бути впевненим у їхній безпеці, потрібно буде організовувати роботу щодо їх перевірки та формування баз безпечних та шкідливих застосунків. Такий спосіб найчастіше є зайвим, оскільки перераховані раніше репозиторії швидше за все вже зберігають усі зразки, які можна знайти у

відкритому доступі.

3.1.2 Модулі та бібліотеки

Завдяки розвитку проектів з відкритим вихідним кодом з'явилося безліч людей і компаній, які публікують свої результати розробок у репозиторіях версійного контролю, БД, на сайтах. Разом з цим з'явилася не лише можливість використання готових відкритих модулів та бібліотек, а й вибору між ними у зв'язку з їхньою різноманітністю. Для будь-яких реалізованих програмних комплексів пропадає потреба в написанні власної версії готових алгоритмів, що спрощує та прискорює розробку.

Для даної кваліфікаційної роботи потрібно досліджувати модулі та бібліотеки, що реалізують машини факторизації та їх розширення у вигляді обліку специфіки полів. У зв'язку з тим, що таких робіт багато, то після аналізу потрібно вибрати найкращі та найзручніші з них і порівняти на основі тестування побудованих на них систем виявлення.

В оригінальній роботі [38], через 2 роки представленої концепції машин факторизації була представлена їх реалізація libFM мовою C++. У ній доступна оптимізація методом стохастичного градієнтного спуску (SGD) і найменших квадратів (ALS), що чергуються, а також байєсовський висновок з використанням ланцюгів Маркова Монте-Карло (MCMC).

libFFM - реалізація машин факторизації з урахуванням специфіки полів, що включає можливість раннього припинення навчання, оскільки модель схильна до перенавчання. Бібліотека так само написана на C++, допускає розпаралелювання, але при цьому стає недетермінованою, і використовує модуль SSE для швидких обчислень.

Найбільш зручними є Python-реалізації через простоту мови. Такими модулями є:

– pyFM - це мінімалістична реалізація машин факторизації з використанням Cython. Попередня версія використовувала numpy. Інтерфейс бібліотеки схожий із популярним для МН модулем scikit-leam;

– fastFM - реалізація машин факторизації, що включає як

класифікацію, так і регуляризацію. Як і оригінальна бібліотека містить три вирішувача - SGD, ALS, MCMC. Пов'язана з модулем робота [39] визначає цю реалізацію. Вона так само побудована на Cyton у зв'язку з тим, що ядро моделі написано мовою програмування C. Також у роботі сказано, що забезпечуються швидкі операції з розрідженими матрицями та векторами, що також і спрощує роботу з поділом пам'яті між C і Python. При експериментах точність бібліотеки була такою самою, як і в libFM, проте за часом виконання перевершувала оригінальну бібліотеку;

– xLearn [40] - реалізація лінійної моделі логістичної регресії, машин факторизації та його розширення з урахуванням специфіки полів. У документації стверджується, що модуль працює в рази швидше за стандартні бібліотеки libFM і libFFM, а також забезпечує простоту і масштабованість. Крім того, надає оболонку для роботи через командний рядок та мову програмування R.

Крім перерахованих модулів є ще безліч реалізацій, але всі вони надають або ту ж, або меншу функціональність, тому не потрібні в описі. Також, усі перелічені модулі найбільш затребувані, згідно з оцінками GitHub, тому їх розгляд буде найбільш цікавим і для більшості розробників, які їх використовують.

3.1.3 Вибір наборів даних та їх обробка

Тепер потрібно сформувати досить великий набір даних для навчання та тестування. Для цього був поданий запит на доступ до репозиторій тимчасових застосунків VirusShare і через час отримані дані для авторизації та можливість скачування наборів.

Репозиторій VirusShare був обраний у зв'язку з його поширеністю у використанні, завдяки чому можна буде порівняти результати роботи реалізованих систем виявлення з іншими працями. Крім того, репозиторій надає широкий вибір наборів даних за роками їх публікації. Недоліком є те, що не всі набори є можливість завантажити через їх роздачу за допомогою технології Torrent та відсутності роздаючих. Таким чином було вибрано набір даних VirusShare_Android_APK_2016, що містить більше 12 тисяч шкідливих Android

застосунків.

Так як потрібно два набори застосунків - шкідливі та безпечні, було додатково взято набір даних CICMalAnal2017 [41] з репозиторію UNB. Він містить понад 10 тисяч зразків, 4000 з яких – шкідливі та 6500 – безпечні. Дані зібрані з різних джерел, у тому числі більшість із безпечних застосунків - з магазину Google Play, опубліковані з 2020 по 2022 роки. Крім того, авторами набору даних було встановлено 5065 безпечних програм на реальні пристрої для їх перевірки на відсутність шкідливих дій. У репозиторії набір безпечних застосунків поділено за роками, але для роботи було взято та об'єднано архіви всіх років.

Оскільки набори застосунків досить великі та його розбір займає багато часу, ще, можливі виняткові випадки, коли робота раптово переривається, потрібно організувати проміжне збереження результатів. У зв'язку з тим, що в Python зручно організована робота з форматом json за допомогою однойменного стандартного модуля, було вирішено зберігати дані саме в цьому форматі.

Формат та приклад заповнення JSON файлу представлений на рис. 3.1

Файл складається з 5 основних полів:

– all_features - список, що містить імена всіх характеристик безпеки розібраних застосунків;

– list_of_vectors - список, що містить списки, які є бінарними векторами, що відповідають списку all_features. Якщо n -а характеристика з all_features міститься в застосунку, то на n -ій позиції вектора буде знаходитися 1, інакше – 0;

– results - список, що показує, яким є застосунок - шкідливим або безпечним. Якщо n -ий вектор списку vectors представляє шкідливмй застосунок, то на n -ій позиції списку results буде -1, якщо безпечне, то 1;

– stage - етап, у якому було виконано останнє збереження (malware чи benign);

– num - кількість оброблених застосунків на відповідному етапі.

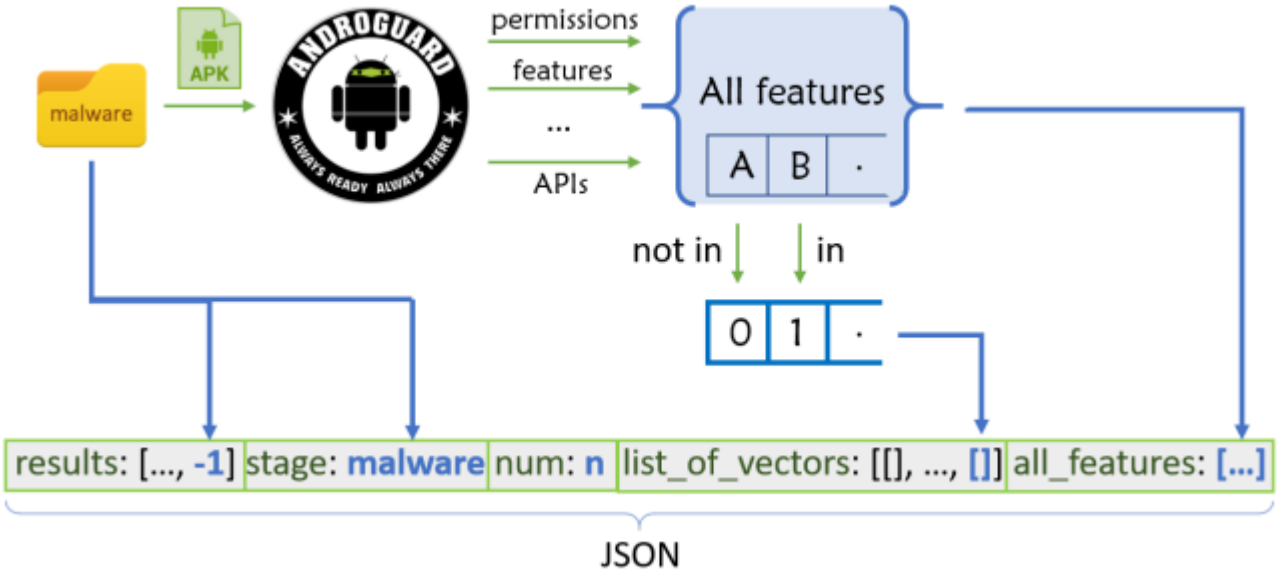

Рисунок 3.1 – Схема обробки Android застосунків

Був написаний код, як складова реалізованого класифікатора, який зчитує набір вхідних APK файлів і переводить їх характеристики до списку бінарних векторів. Для цього, насамперед, було сформовано два каталоги з безпечними та шкідливими програмами. Таким чином, спочатку зчитуються імена застосунків з каталогів, шляхи до яких записуються в коді змінні datasets_folder (загальна частина шляху), benign_folder і malware_folder, і формуються відповідні списки (рис. 3.2).

```
    trust = sorted([f for f in listdir(datasets_folder + be-
nign_folder) if isfile(join(datasets_folder + benign_folder, f))])
    malware = sorted([f for f in listdir(datasets_folder + mal-
ware_folder) if isfile(join(datasets_folder + malware_folder, f))])
```

Рисунок 3.2 – Фрагмент лістингу формування каталогів

Після цього, окремо для шкідливих та безпечних програм викликається функція, що відповідає за перебір файлів (рис. 3.3):

```
    get_features_from_files(trust,    datasets_folder    +    be-
nign_folder)
```

Рисунок 3.3 –Лістинг коду функції для перебору файлів

У ній запускається цикл за всіма отриманими раніше іменами файлів (рис. 3.4):

```
while i < len(listfiles):
        filename = listfiles[i]
        i += 1
        features = get_features(directory + '/' + filename)
        if not features:
            continue
        list_of_vectors.append(get_features_vector(features))
```

Рисунок 3.4 – Лістинг коду циклу за ознаками

Тут listfiles - переданий список імен застосунків, який, утворюючи повний шлях, передається у функцію get_features.

Ця функція за допомогою засобів модуля Androguard декодує APK файл, дизасемблює вихідні коди і повертає потрібні характеристики програми шляхом виклику відповідних функцій.

Крім того, перед обробкою програми файл APK перевіряється на те, що він є ZIP файлом і організується обробка винятків для коректного продовження роботи навіть у разі проблем з APK файлом (рис. 3.5):

```
def get_features(apk_name):
    features = []
    if zipfile.is_zipfile(apk_name):
        try:
            apk, dalvik, analysis = AnalyzeAPK(apk_name)
            features += apk.get_permissions()
            features += apk.get_declared_permissions()
            features += apk.get_features()
            features += apk.get_libraries()
            features += apk.get_providers()
            features += apk.get_receivers()
            features += apk.get_requested_aosp_permissions()
            features += apk.get_uses_implied_permission_list()
            del apk
            del dalvik
            del analysis
        except Exception:
```

Рисунок 3.5 – Фрагмент лістингу функції декодування

Таким чином, за допомогою модуля Androguard формується список наступних характеристик:

– дозволи, оголошені у маніфесті застосунку;
– дозволи, що запитуються під час роботи програми;
– використовувані апаратні засоби;
– використовувані бібліотеки;
– постачальники контенту, що використовуються ;
– підписки на широкомовні повідомлення;
– привілейовані дозволи системи Android;
– дозволи, що видаються залежно від інших дозволів або вбудовані дозволи використовуваної версії SDK.

Якщо програма була вдало оброблена, то результат роботи функції get_features подається на вхід функції get_features_vector, яка, у свою чергу, кодує отримані характеристики в бінарний вектор, що відповідає конкретному

застосунку.

Насамперед формується список feature_vector, що містить нулі на всіх позиціях. Розмір цього списку дорівнює розміру списку all_features, який містить усі назви раніше лічених характеристик.

feature_vector = [0] * len(all_features)

Далі кожну характеристику потрібно розібрати так, щоб отримати назву характеристики, яка залежить від конкретного пакета програми.

feature = rawfeature[rawfeature.rfind('.')+1:]

Далі, якщо отримане ім'я характеристики міститься у списку all_features, то відповідної позиції feature_vector встановлюється 1.

if feature in all_features:

feature vector[all features.index(feature)] = 1

Якщо характеристика не міститься в all_features, то вона додається до all_features, а в кінець кожного вектора, відповідного застосунку, закодованому раніше, додається елемент, що містить 0.

Результатом функції є закодований вектор характеристик конкретного Android -застосунку, який додається до результуючого списку векторів list_of_vectors.

list_of_vectors.append(get_features_vector(features))

В результаті роботи коду, що відповідає за формування навчальної вибірки, отримуємо список list_of_vectors, що містить навчальні вектори, а також список all_features, що містить імена всіх отриманих раніше характеристик навчальних застосунків. Крім того, формується список результатів, в якому зберігаються результати еталонної класифікації. Також, зважаючи на те, що обробка застосунків для формування навчальної вибірки - найбільш трудомістка задача в контексті реалізованої системи (деякі програми можуть оброблятися близько 10 хвилин і цей час пропорційно розміру програми), щоб не обробляти вже закодовані раніше файли знову, як було описано раніше, результати записуються у файл result.json кожні dump_every_n оброблених програм. Мною було виставлено значення у 5 застосунків.

Загальна схема кодування застосунків у векторі та збереження їх у вигляді

JSON файлу представлена на рис. 3.1.

Результати обробки застосунків наборів даних VirnsShare та CICMalAnal2017 - відповідні їм JSON файли були викладені у відкритий репозиторій [42]. Набори даних мали велику кількість програм, які не змогли обробитися Andrguard'ом. Було пропущено близько тисячі безпечних програм і близько третини шкідливих, в результаті файли JSON містять вектора 5642 безпечних і 8744 шкідливих програм.

Формат JSON не є стандартним для наборів даних у МН, оскільки найчастіше використовується формат CSV. Однак, для деяких реалізацій алгоритмів формат JSON є більш зручним, тому може бути корисним для дослідників, які ведуть роботи з такими реалізації.

3.1.4 Вибір бібліотек для реалізації систем

Тепер, коли навчальна вибірка сформована, потрібно вибрати модулі для виявлення шкідливих Android застосунків і використовувати їх для реалізації систем виявлення та їх подальшого порівняння.

Було обрано два модулі - fastFM і xLearn у зв'язку з тим, що їх розробники у відповідних джерелах стверджують, що ці бібліотеки працюють швидше, ніж libFM, але наскільки вони відрізняються за швидкістю один від одного - невідомо. Крім того, обидва багатофункціональні модулі - можуть виконувати різні завдання, а xLeam надає не тільки машини факторизації, але і інші моделі і можливості. У зв'язку з цим дані бібліотеки є найбільш популярними як пакети для Python.

FastFM. Надає процедури SGD, CD, а також MCMC. Модуль можна використовувати для завдань регресії, класифікації та ранжування.

Теоретично MCMC має краще справлятися із завданням класифікації шкідливих застосунків, проте, практично, більшість завдань MCMC і SGD справляються однаково ефективно. Тому для даної роботи був вибраний метод стохастичного градієнтного спуску, а крім того, він має найбільш швидке прогнозування і ефективно працює з великими наборами даних , що важливо для розв'язуваного завдання.

Реалізація системи виявлення шкідливих застосунків Android за допомогою fastFM починається зі зчитування файлу results.json, формат якого був описаний в пункті 3.1.3. Даний файл містить результати обробки застосунків - бінарні вектори для навчання (рис. 3.6).

```
if isfile('results.json'):
        with open('results.json') as json_file:
            json_res = json.load(json_file)
            all_features = json_res['all_features']
            list_of_vectors = json_res['vectors']
            results = json_res['results']
            del json_res
```

Рисунок 3.6 – Лістинг коду файлу з результатами обробки застосунків

FastFM для навчання приймає бінарні вектори, що відповідають за вміст у застосункух відповідних характеристик безпеки, у вигляді стиснутої розрідженої матриці пакета scipy. Для цього список векторів перетворюється на цей формат за допомогою функції sparse.csc_matrix.

matrix = scipy.sparse.csc_matrix(list_of_vectors)

Вектор, що описує клас, до якого відноситься відповідний застосунок, повинен мати вигляд масиву numpy, тому список результатів перетворюється на масив функцією np.asarray.

result_array = np.asarray(results)

Після цього викликається функція training, в якій почала створюється об'єкт для моделі машини факторизації з відповідними параметрами, а потім навчається на створеному наборі даних.

fm = sgd.FMClassification(n_iter=1000, init_stdev=0. 1, l2_reg_w=0, l2_reg_V=0, rank=2, step_size=0.1)

fm.fit(matrix, result_array)

Допускається наступний набір параметрів машини факторизації:

– n_iter – кількість ітерацій алгоритму;

– init_stdev – ініціалізація параметра stdev;

– l2_reg_w - штрафна вага в L2 регуляризації для лінійних коефіцієнтів;

– l2_reg_V - штрафна вага в L2 регуляризації для парних коефіцієнтів ;

– rank – ранг факторизації;

– step_size – розмір кроку для SGD.

Після навчання мережі можна отримати всі ваги, що містяться в ній:

– w0_ - Байєсовський терм;

– w_ - лінійні коефіцієнти;

– V_ - коефіцієнти матриці факторизації другого порядку.

Сукупність даних параметрів визначає навчену модель машини факторизації. Таким чином, зберігаючи дані параметри, зберігається навчена модель, ці параметри можна буде використовувати для подальшого відновлення моделі без потреби навчання заново.

Після навчання мережі її можна використовувати для класифікації. Метод predict використовується для прогнозування класифікації. Отримуючи на вхід матрицю, що містить вектор конкретного досліджуваного застосунку, описаного раніше виду, метод повертає 1, якщо застосунок безпечно, -1, якщо застосунок часу.

Метод predict_proba отримує на вхід все ту ж матрицю, а повертає імовірність того, що класифікація була виконана правильно.

Для тестування моделі було винесено частину застосунків з вихідних наборів окрему папку. При тестуванні моделі ці застосунки обробляються так само, як і при формуванні навчальної вибірки, за винятком того, що якщо виявляються нові характеристики безпеки, то вони жодним чином не впливають на бінарний вектор програми і результат класифікації.

xLearn. Підтримує оптимізацію SGD, адаптивним градієнтним методом (Adagrad) та методом наступного за регулюючим лідером (FTRL).

У порівнянні з традиційним методом SGD, Adagrad адаптує швидкість навчання до характеристик, виконуючи більше оновлень для тих характеристик, котрі рідко з'являються, і менше оновлень для характеристик, що часто

з'являються. Тому він добре підходить для роботи з розрідженими даними.

FTRL (Follow-the-Regularized-Leader) - метод, який широко використовується для великомасштабних розріджених завдань. Однак для використання FTRL необхідно налаштовувати більшу кількість гіперпараметрів у порівнянні з SGD та Adagrad.

В даному випадку був обраний метод Adagrad у зв'язку з тим, що він краще проявляє себе при роботі з розрідженими даними, ніж SGD, і при цьому не вимагає великої кількості гіперпараметрів, як FTRL.

xLearn використовує формати libsvm або CSV для стандартної моделі машин факторизації та libffm для моделі з урахуванням специфіки полів. У зв'язку з цим сформований набір даних був переформатований з формату JSON в libsvm. Крім того, для машин факторизації з урахуванням специфіки полів були виділені поля характеристик оброблених раніше застосунків, що відповідають типам виведених Androguard характеристик, і на основі цього сформовані файли для навчання та тестування, що містять ті ж дані, що і для модуля fastFM.

xLearn надає простіший інтерфейс створення та перевикористання моделей. Так, модель стандартної машини факторизації створюється і тренується відповідно до наступному коду (рис. 3.7), згідно зі схемою на рис. 3.8:

```
fm_model = xl.create_fm()
fm_model.setTrain("./results.libsvm")
fm_model.setValidate("./testing.libsvm")
param = {'task':'binary', 'lr':0.2, 'lambda':0.002, 'k':4,
'metric': 'acc'}
fm_model.fit(param, "./model.out")
```

Рисунок 3.7 – Лістинг коду файлу для тренування моделі

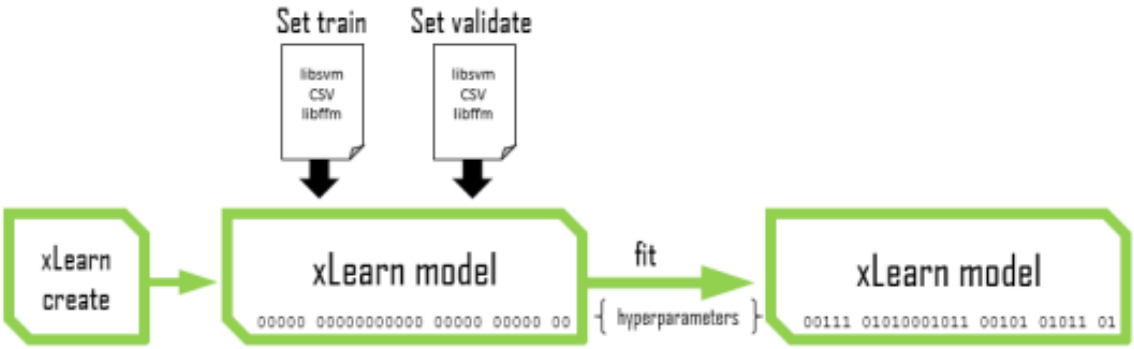

Рисунок 3.8 – Схема роботи моделі, побудованої на основі xLearn

Для моделі може налаштовуватися більша кількість параметрів, але найцікавіші - це:

– lr – швидкість навчання;

– lambda – параметр L2 регуляризації;

– k – ранг факторизації.

Для розширеної моделі з урахуванням специфіки полів змінюються лише вхідні файли, а під час створення моделі використовується метод create_ffm замість create_fm.

3.1.5 Вибір оптимальних параметрів моделей

Показники, щодо яких оцінюються та порівнюються моделі МН, називаються метриками. Усі вони розраховуються щодо складових матриці помилок, яка представлена у табл. 3.1, де $\tilde{y}$ - результат класифікації від моделі, y - справжня класифікація об'єкта

Таблиця 3.1 – Матриця помилок

	$y = 1$	$y = 0$
$\tilde{y} = 1$	True Positive (TP)	False Positive (FP)
$\tilde{y} = 0$	False Negative (FN)	True Negative (TN)

Основні використовувані метрики та їх обчислення:

– частка вірно класифікованих моделлю об'єктів $accuracy = \frac{TP+TN}{TP+TN+FP+FN}$

– частка істинно позитивних об'єктів від їх позитивно класифікованої загальної кількості $precision = \frac{TP}{TP+FP}$

– частка вірно класифікованих позитивних об'єктів від їх загальної кількості $recall = \frac{TP}{TP+FN}$

– середнє гармонійне precision та recall $F1 = 2 * \frac{precision*recall}{precision+recall}$

– AUC - площа під кривою помилок $TPR = \frac{TP}{TP+FN}$ и $FPR = \frac{FP}{FP+TN}$.

Для системи виявлення шкідливих застосунків найбільш важлива точність у зв'язку з використанням у магазинах застосунків та в антивірусах, які повинні, у тому числі для своєї репутації, з однаковою точністю пропускати безпечні та блокувати шкідливі програми. Крім того, дана метрика відмінно підходить для систем виявлення у зв'язку з можливістю її тестування на великій кількості даних, що містять класи безпечних та шкідливих прикладів однакового обсягу.

Для досягнення найвищих показників метрик, у даному випадку - точності, для моделей мають бути підібрані відповідні параметри. Для цього був використаний метод послідовного перебору параметрів з подальшим узагальненням шляхом побудови графіків для кожної з моделей з метою максимізації точності визначення класу прикладів. Перед цим було розділено вихідний набір даних на набори для навчання та для тестування, і після цього навчено модель та отримано результати перебору параметрів даних для тестування.

В результаті було отримано наступні набори параметрів:

1. FastFM – машина факторизації.
   – $rank$ = 150000;
   – n_iter = 3000;
   – step_size = *0.007*.

2. xLearn – машина факторизації.
   - $k = 5$;
   - $lambda = 0.00001$;
   - lr = 2.
3. xLearn – машина факторизації з урахуванням специфіки полів.
   - $k = 5$;
   - $lambda = 0.00001$;
   - lr = 2.

На рис. 3.9 – 3.11 представлені графіки залежності точності моделей МН від відповідних параметрів, що використовуються в них.

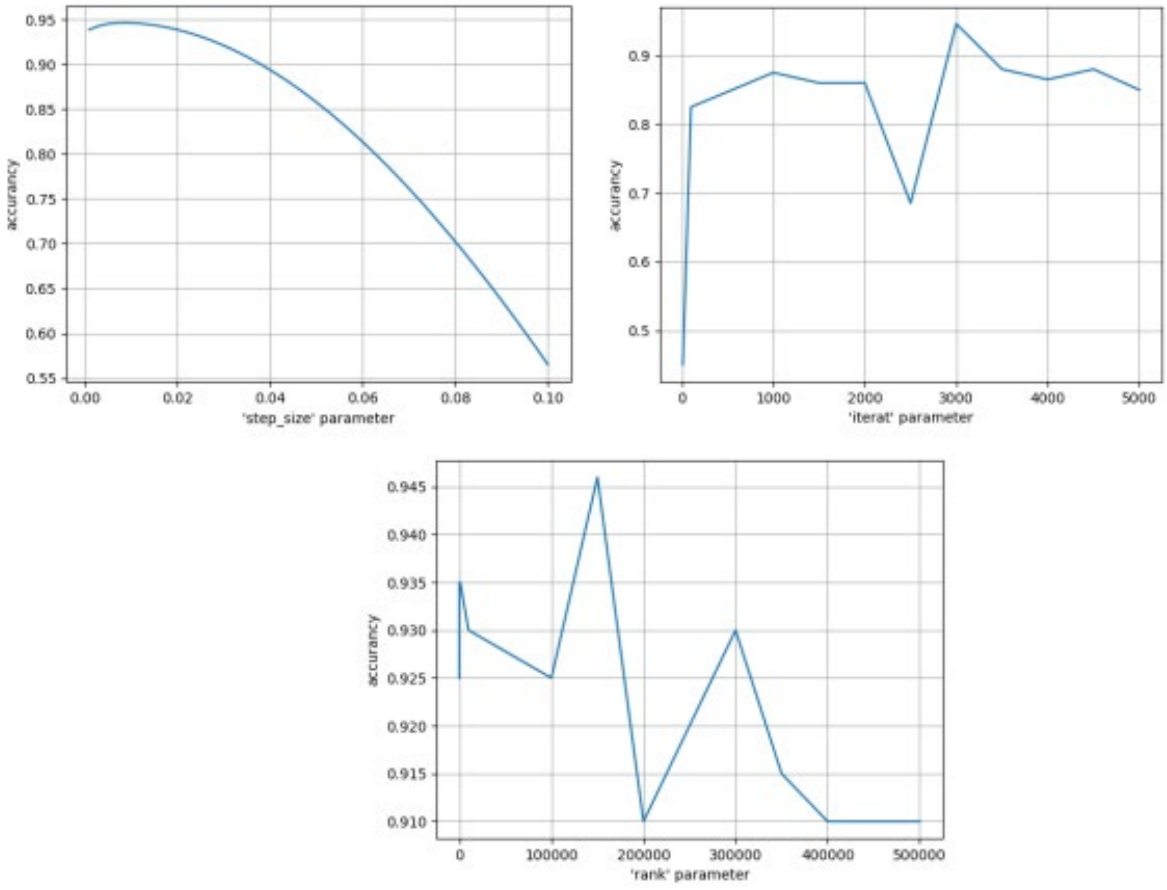

Рисунок 3.9 – Залежність точності моделі fastFM від значень її параметрів

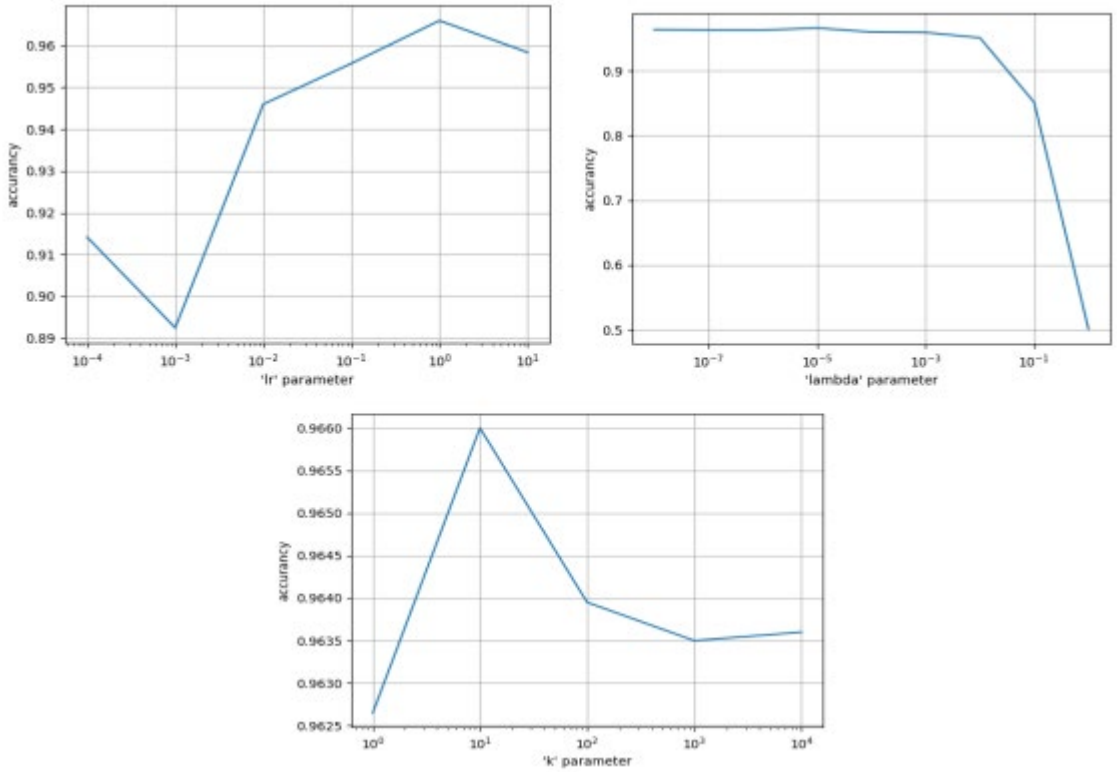

Рисунок 3.10 – Залежність точності моделі машин факторизації xLearn від значень її параметрів

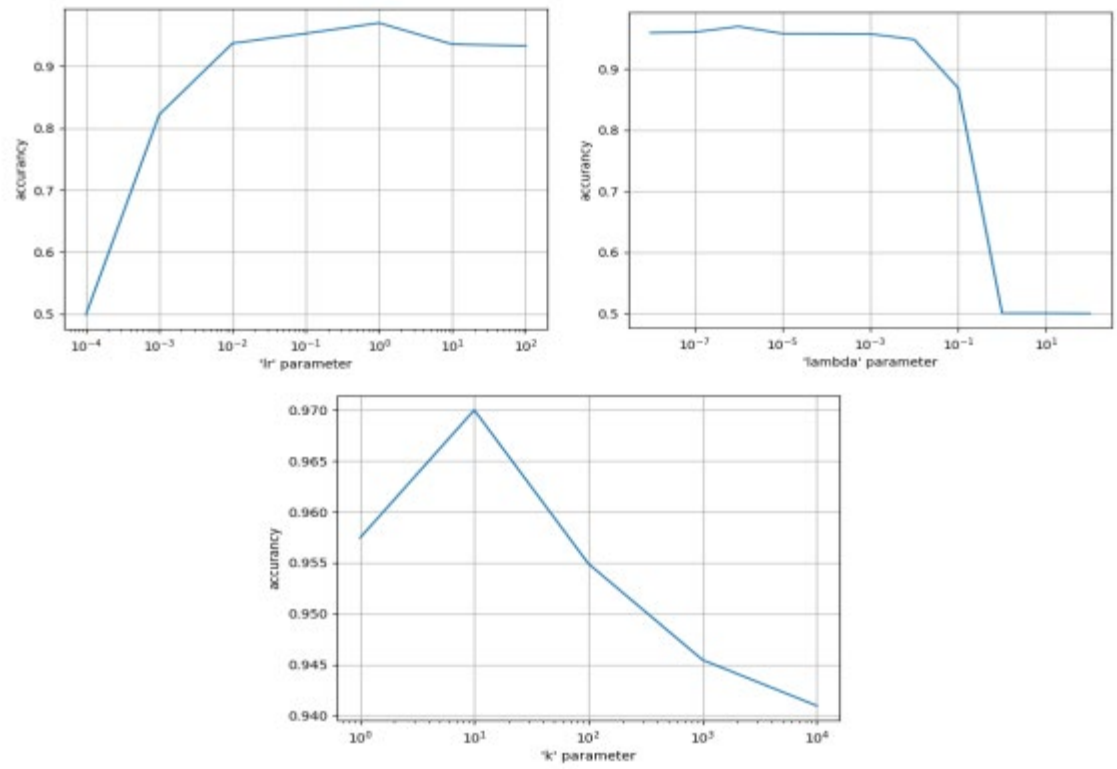

Рисунок 3.11 – Залежність точності моделі машин факторизації з урахуванням специфіки полів xLearn від значень її параметрів

3.2 Порівняння і оцінка результатів

Два критерії, за якими можна порівняти реалізовані моделі – це швидкість навчання та точність класифікації. Крім того, у зв'язку з розповсюдженістю набору даних VirusShare, є можливість порівняти точність класифікації отриманих моделей з іншими моделями, що так само використовують цей набір даних.

У підпункті 3.1.5 розділу 3 цієї роботи було визначено найкращі параметри для моделей fastFM та xLearn щодо точності класифікації. У табл. 3.2 показані максимально досягнуті точності класифікації кожної з моделей.

Таблиця 3.2 – Точність класифікації навчених моделей

	fastFM	xLearn (машина факторизації)	xLearn (з урахуванням специфіки полів)
Точність	94,6%	96,6%	97%

Отримана точність є досить високою і перевершує більшість робіт, спрямованих на вирішення задачі виявлення шкідливих застосунків Android і використовують набір даних VirusShare. Так, у роботах [43 45] авторам вдалося досягти лише від 92% до 95,5% точності класифікації. У раніше розглянутій роботі [29] була отримана порівняна з даною роботою точність - 96,9%. Але є і перевищують отриману точність класифікації - наприклад, робота [46], у якій точність становила 97,66%. Порівняльна діаграма точності різних робіт наведена на рис. 3.12.

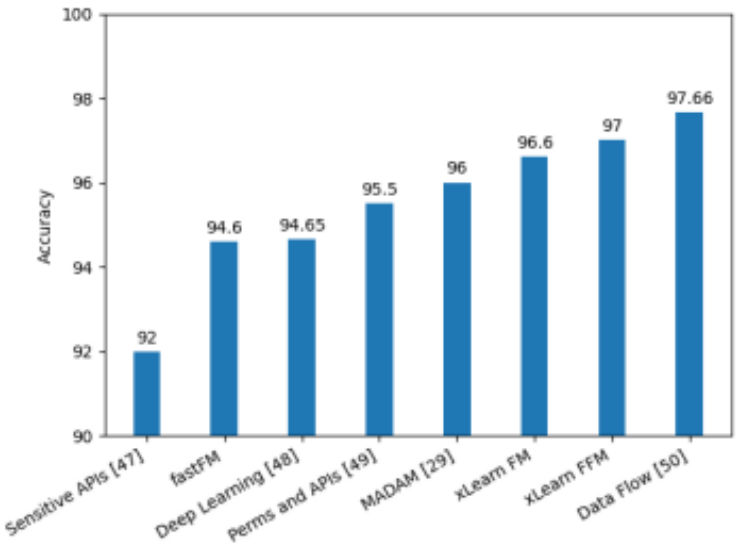

Рисунок 3.12 – Порівняльна діаграма точності класифікації

У документації до обох пакетів - fastFM і xLearn, зазначено, що швидкість їх роботи вища, ніж швидкість роботи традиційної моделі libFM, тому було цікаво порівняти обидва ці модулі один з одним для визначення найбільш швидкого з них.

Час роботи моделей включає час навчання і тестування. Для того, щоб моделі були в рівних умовах, для них були виставлені параметри, що постачаються. Моделі навчалися 10 епох для отримання усереднених результатів однієї епохи. Залежність часу роботи моделей від рангу факторизації представлена на рис. 3.13.

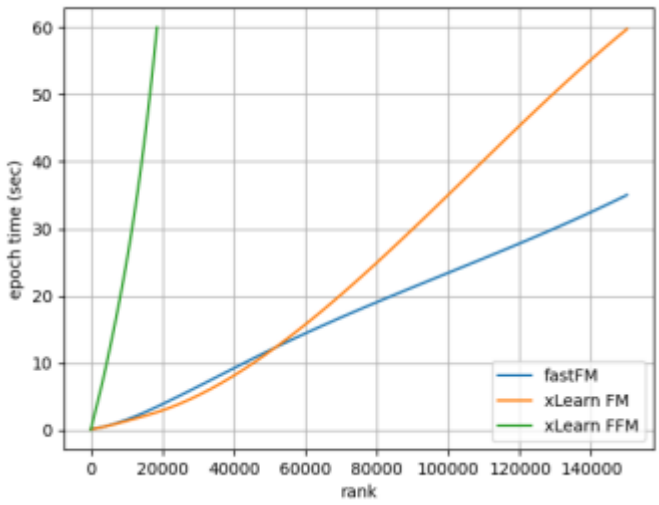

Рисунок 3.13 – Залежність часу роботи моделей від рангу факторизації

Таким чином, xLearn на основі машин факторизації працює трохи швидше з невеликим рангом факторизації, проте при великому ранзі fastFM обробляє модель помітно швидше.

Окремо слід розглядати xLearn, засновану на обліку специфіки полів. У зв'язку з тим, що модель навчає більше прихованих векторів, час її роботи помітно більше за будь-якого рангу факторизації.

3.5 Висновки до третього розділу

Таким чином, у цьому розділі були реалізовані моделі МН, основою для яких стали машини факторизації та їх розширення із врахуванням специфіки полів. Було сформовано набори даних для навчання моделей та їх подальшого тестування. Перебором параметрів було отримано графіки, що показують залежність точності виявлення шкідливих програм від значень параметрів. Максимуми на графіках показують найвищу точність моделей. Для fastFM це 94,6%, для xLearn, що базується на машині факторизації - 96,6%, заснованої на розширенні з урахуванням специфіки полів - 97%.

Також була проведена оцінка точності та часу роботи кожної реалізації машин факторизації. Було встановлено, що реалізація стандартної моделі в модулі fastFM швидше здійснює навчання та тестування моделі при великому ранзі факторизації (помітна різниця є при ранзі рівному 100 000 і зростає зі збільшенням рангу). Реалізація ж xLearn відмінно проявила себе в точності класифікації за рахунок адаптивного навчання на кожній з епох і перевершує fastFM за будь-якого рангу факторизації. Мінусом xLearn є те, що вона недетермінована за рахунок паралелізації під час навчання. Паралелізацію можна вимкнути для моделей, але в цьому випадку вона стане ще повільнішою.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Охорона праці

Метою кваліфікаційної роботи магістра є виявлення шкідливих застосунків для ОС Android з використанням моделей, заснованих на факторизації матриць. Оскільки, проведення робіт з розробки та використання такого ПЗ передбачає використання комп'ютерної техніки, зокрема ПК та периферійних пристроїв, то обов'язковим є дотримання вимог з охорони праці і техніки безпеки.

Для ефективної і безпечної роботи колективу працівників виявлення шкідливих застосунків для ОС Android з використанням моделей, заснованих на факторизації матриць, необхідно організувати безпечні умови праці. При цьому керівник організації несе безпосередню відповідальність за порушення нормативно-правових актів з охорони праці [47]. Окрім цього, на робочих місцях працівників необхідно забезпечити дотримання вимог, затверджених Наказом Мінсоцполітики від 14.02.2018 за № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». Згідно Вимог приміщення, де розміщені робочі місця операторів, крім приміщень, у яких розміщені робочі місця операторів великих ЕОМ загального призначення (сервер), мають бути оснащені системою автоматичної пожежної сигналізації відповідно до цих вимог;

– переліку однотипних за призначенням об'єктів, які підлягають обладнанню автоматичними установками пожежогасіння та пожежної сигналізації, затвердженого наказом Міністерства України з питань надзвичайних ситуацій та у справах захисту населення від наслідків Чорнобильської катастрофи від 22.08.2005 N 161, зареєстрованого в Міністерстві юстиції України 05.09.2005 за N 990/11270 (НАПБ Б.06.004-2005);

– Державних будівельних норм "Інженерне обладнання будинків і споруд. Пожежна автоматика будинків і споруд", затверджених наказом Держбуду України від 28.10.98 N 247 (далі - ДБН В.2.5-56:2014, з димовими пожежними сповіщувачами та переносними вуглекислотними вогнегасниками.

В інших приміщеннях допускається встановлювати теплові пожежні сповіщувачі. Приміщення, де розміщені робочі місця операторів, мають бути оснащені вогнегасниками, кількість яких визначається згідно з вимогами ДСТУ 4297:2004 «Пожежна техніка. Технічне обслуговування вогнегасників». Загальні технічні вимоги і з урахуванням граничнодопустимих концентрацій вогнегасної рідини відповідно до вимог НАПБ А.01.001-2014. Приміщення, в яких розміщуються робочі місця операторів сервера загального призначення, обладнуються системою автоматичної пожежної сигналізації та засобами пожежогасіння відповідно до вимог ДБН В.2.5-56:2014, ДБН В.2.5-56:2010, НАПБ А.01.001-2014 і вимог нормативно-технічної та експлуатаційної документації виробника. Проходи до засобів пожежогасіння мають бути вільними.

Лінія електромережі для живлення комп'ютера та периферійних пристроїв повинні бути виконаними як окрема групова трипровідна мережа шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів. Не допускається використовувати нульовий робочий провідник як нульовий захисний провідник. Нульовий захисний провідник прокладається від стійки групового розподільного щита, розподільного пункту до розеток електроживлення. Не допускається підключати на щиті до одного контактного затискача нульовий робочий та нульовий захисний провідники.

Площа перерізу нульового робочого та нульового захисного провідника в груповій трипровідній мережі має бути не менше площі перерізу фазового провідника. Усі провідники мають відповідати номінальним параметрам мережі та навантаження, умовам навколишнього середовища, умовам розподілу провідників, температурному режиму та типам апаратури захисту, вимогам НПАОП 40.1-1.01-97.

У приміщенні, де одночасно експлуатуються понад п'ять комп'ютерів, на помітному, доступному місці встановлюється аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення. Комп'ютери повинні підключатися до електромережі тільки за допомогою справних штепсельних з'єднань і електророзеток заводського

виготовлення.

У штепсельних з'єднаннях та електророзетках, крім контактів фазового та нульового робочого провідників, мають бути спеціальні контакти для підключення нульового захисного провідника. Їхня конструкція має бути такою, щоб приєднання нульового захисного провідника відбувалося раніше, ніж приєднання фазового та нульового робочого провідників. Порядок роз'єднання при відключенні має бути зворотним. Не допускається підключати комп'ютери до звичайної двопровідної електромережі, в тому числі – з використанням перехідних пристроїв. Електромережі штепсельних з'єднань та електророзеток для живлення комп'ютерної техніки повинні бути виконаними за магістральною схемою, по 3-6 з'єднань або електророзеток в одному колі. Штепсельні з'єднання та електророзетки для напруги 12 В та 42 В за своєю конструкцією мають відрізнятися від штепсельних з'єднань для напруги 127 В та 220 В. Штепсельні з'єднання та електророзетки, розраховані на напругу 12 В та 42 В, мають візуально (за кольором) відрізнятися від кольору штепсельних з'єднань, розрахованих на напругу 127 В та 220 В.

При підвищенні ефективності контролю доступу в приміщення, де для забезпечення безпеки мешканців, співробітників і збереження майна використовуються ДС, важливим, з точки зору охорони праці, є забезпечення достатньої величини природного та штучного освітлення, які визначені у НПАОП 0.00-7.15-18. Організація робочого місця фахівця із дослідження методів та програмно-апаратних засобів оптимізаційних процесів на основі ГА повинна забезпечувати відповідність усіх елементів робочого місця та їх розташування ергономічним вимогам ДСТУ 8604:2015 «Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги». Відстань від екрана до ока фахівців, які працюють за комп'ютером визначається згідно з вимогами ДСанПіН 3.3.2.007-98.

Розміщення принтера або іншого пристрою введення-виведення інформації на робочому місці має забезпечувати добру видимість екрана комп'ютера, зручність ручного керування пристроєм введення-виведення інформації в зоні досяжності моторного поля згідно з вимогами ДСанПіН

3.3.2.007-98.

Таким чином, у результаті аналізу вимог щодо охорони праці користувачів комп'ютерів, визначено особливості організації робочих місць, вимог з електробезпеки, природного та штучного освітлення для ефективної і безпечної роботи фахівців з виявлення шкідливих застосунків для ОС Android з використанням моделей, заснованих на факторизації матриць.

4.2. Функціонування державної системи спостереження, збирання, обробки та аналізу інформації про стан довкілля під час надзвичайних ситуацій мирного та воєнного часу

Моніторинг довкілля – це система спостереження, збирання та аналізу інформації про ситуацію, що може скластись під час надзвичайних ситуацій мирного та воєнного часу. Також це система спостереження за визначеними об'єктами, явищами та процесами з метою оперативного оцінювання їх стану, виявлення результатів впливу на них зовнішніх чинників та прийняття відповідних управлінських рішень (ДСТУ 3891:2013) (див. ДСТУ 7295:2013).

Моніторинг потенційно небезпечних об'єктів це спостереження, контролювання за зміною параметрів технологічних режимів з метою збирання, збереження, передавання та аналізування інформації щодо поточного стану потенційно небезпечних об'єктів, наявності та кількості порушень вимог безпеки, відпрацювання рекомендацій щодо проведення 98 робіт із запобігання та ліквідування техногенних надзвичайних ситуацій та їх наслідків (ДСТУ 7295:2013).

Моніторинг джерел надзвичайних ситуацій це система спостереження за об'єктами, які можуть бути джерелами надзвичайних ситуацій, що має на меті виявлення небезпеки, збирання, узагальнення та аналізування оперативної інформації стосовно стану об'єктів моніторингу та розроблення науково-обґрунтованих рекомендацій щодо проведення заходів із запобігання та ліквідування надзвичайних ситуацій (ДСТУ 7295:2013).

Моніторинг довкілля – це систематичні спостереження і контролювання,

які проводять регулярно, за єдиною програмою для оцінювання стану довкілля, аналізування процесів, які відбуваються в ньому і своєчасне виявлення тенденцій його змінювання (ДСТУ 7295:2013).

Моніторинг надзвичайних ситуацій (НС) – система спостереження за об'єктами, які можуть бути джерелами надзвичайних ситуацій, що має на меті виявлення небезпеки, збирання, узагальнення та аналізування оперативної інформації щодо об'єктів моніторингу та розроблення науково обґрунтованих рекомендацій щодо проведення заходів із запобігання та ліквідування НС [48].

Моніторинг небезпечних явищ та процесів це система спостереження та контролювання за розвитком небезпечних та стихійних природних явищ і процесів, чинниками, які спричинюють їх формування та розвиток, аналізування, збереження та передавання інформації щодо виявлення тенденцій їх змінювання, розроблення комплексу заходів щодо запобігання природним надзвичайним ситуаціям та ліквідування їх наслідків. Небезпечні природні явища і процеси підрозділяють на геофізичні, геологічні, гідрологічні, метеорологічні, медико-біологічні та пожежі в природних екосистемах (ДСТУ 7295:2013).

Моніторинг пожеж в екосистемах це спостереження, контролювання, збирання, аналізування, збереження та передавання інформації щодо 99 пожежної небезпеки в природних екосистемах (умов погоди, стану горючих матеріалів, інших пожежонебезпечних чинників), з метою своєчасного планування та здійснення заходів щодо запобігання виникненню і ліквідування пожеж та їх наслідків (ДСТУ 7295:2013).

Моніторинг радіаційної безпеки це спостереження і контролювання рівня радіоактивного забруднення місцевості, повітря, води, продовольства, об'єктів господарювання, дозових навантажень на населення з метою прийняття оперативних рішень щодо запобігання виникненню та ліквідування надзвичайних ситуацій та їх наслідків (ДСТУ 7295:2013).

Моніторинг хімічної небезпеки це спостереження, контролювання, збирання, аналізування, збереження та передавання інформації щодо визначення ступеня і характеру хімічного забруднення довкілля, санітарногігієнічний нагляд за дотриманням установлених нормативів з метою виявлення джерела

надходження небезпечних хімічних речовин, запобігання виникненню та ліквідування надзвичайних ситуацій та їх наслідків (ДСТУ 7295:2013).

Збір та аналіз інформації про стан довкілля під час мирного та воєнного стану дає можливість приймати оперативні рішення для адекватного реагування на ситуацію [48].

4.3. Висновки до розділу

В цьому розділі проаналізовано важливі питання охорони праці та безпеки в надзвичайних ситуаціях, висвітлено питання функціонування державної системи спостереження, збирання, оброблення та аналізу інформації про стан довкілля під час надзвичайних ситуацій мирного та воєнного час.

ВИСНОВКИ

У кваліфікаційній роботі були досліджені методи МН, побудовані на факторизації матриць - машини факторизації та їх розширення, що враховує специфіку полів. Були проаналізовані існуючі роботи, реалізації та набори даних для побудови систем виявлення шкідливих Android застосунків.

У практичній частині були оброблені датасети VirusShare і CICMalAnal2017, для них у форматі JSON сформовані набори векторів, що представляють Android застосунки, Були реалізовані, навчені та протестовані моделі МН на основі пакетів fastFM та xLearn для мови Python.

При тестуванні моделей було отримано високі показники точності класифікації застосунків Android . Для машин факторизації, побудованих на основі пакету fastFM, точність досягла 94,6%, на основі xLearn - 96,6%. Для розширеної моделі з урахуванням специфіки полів, побудованої на основі xLearn, точність досягла 97%. Таким чином, отримані практично результати підтверджують теоретичне підвищення точності шляхом застосування алгоритму машин факторизації з урахуванням специфіки полів виявлення шкідливих Android застосунків.

Крім того, було проаналізовано швидкість роботи fastFM та xLearn. Згідно з результатами для стандартної моделі машин факторизації, xLearn працював трохи швидше при невеликому ранзі факторизації і значно поступався за досить великого (приблизно від рангу рівного 100 000). xLearn на основі розширеної моделі працював набагато довше у зв'язку з великою кількістю прихованих векторів.

Подальшим розвитком можуть бути дослідження залежності точності класифікації застосунків від кількості виділених полів і характеристик, що містяться в них. Також можна застосовувати алгоритми оптимізації гіперпараметрів для отримання найкращих результатів без потреби повного перебору параметрів моделі МН. Також варто розглянути залежність точності класифікації від навчального набору, тобто його розміру та застосунків, що до нього входять.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Google Play Store: number of apps 2024 | Statista. Statista. URL: https://www.statista.com/statistics/266210/number-of-available-applications-in-thegoogle-play-store/ (date of access: 17.11.2024).
2. Mobile network subscriptions worldwide 2028 | Statista. Statista. URL: https://www.statista.com/statistics/330695/number-ofsmartphone-users-worldwide/ (date of access: 18.11.2024).
3. Mobile OS market share worldwide 2009-2024 | Statista. Statista. URL: https://www.statista.com/statistics/272698/global-market-share-held-by-mobileoperating-systems-since-2009 (date of access: 18.11.2024).
4. Smartphone sales worldwide 2007-2023 | Statista. Statista. URL: https://www.statista.com/statistics/263437/globalsmartphone-sales-to-end-users-since-2007 (date of access: 20.11.2024).
5. ZAGORODNA, N., STADNYK, M., LYPA, B., GAVRYLOV, M., & KOZAK, R. (2022). Network Attack Detection Using Machine Learning Methods. Challenges to national defence in contemporary geopolitical situation, 2022(1), 55-61.
6. Google: Android Malware Threat is Vastly Exaggerated. Infosecurity Magazine. URL: https://www.infosecurity-magazine.com/news/google-android-malware-threat-is-vastly/ (date of access: 21.11.2024).
7. Tymoshchuk, V., Mykhailovskyi, O., Dolinskyi, A., Orlovska, A., & Tymoshchuk, D. (2024). OPTIMISING IPS RULES FOR EFFECTIVE DETECTION OF MULTI-VECTOR DDOS ATTACKS. Матеріали конференцій МЦНД, (22.11. 2024; Біла Церква, Україна), 295-300.
8. Babu Rajesh V, Phaninder Reddy, Himanshu P and Mahesh U Patil. Droidswan: detecting malicious android applications based on static feature analysis. URL: https://www.researchgate.net/publication/307797420_droidswan_detecting_malicious_android_applications_based_on_static_ feature_analysis (date of access 21.11.2024).

9. Lypa, B., Horyn, I., Zagorodna, N., Tymoshchuk, D., Lechachenko T., (2024). Comparison of feature extraction tools for network traffic data. CEUR Workshop Proceedings, 3896, pp. 1-11.

10. Venugopal D., Hu G. Efficient Signature Based Malware Detection on Mobile Devices. Mobile Information Systems. 2008. Vol. 4, no. 1. P. 33–49. URL: https://doi.org/10.1155/2008/712353 (date of access: 23.11.2024).

11. Tymoshchuk, D., Yasniy, O., Mytnyk, M., Zagorodna, N., Tymoshchuk, V., (2024). Detection and classification of DDoS flooding attacks by machine learning methods. CEUR Workshop Proceedings, 3842, pp. 184 - 195.

12. TinyDroid: A Lightweight and Efficient Model for Android Malware Detection and Classification / T. Chen et al. Mobile Information Systems. 2018. Vol. 2018. P. 1–9. URL: https://doi.org/10.1155/2018/4157156 (date of access: 14.12.2024).

13. ТИМОЩУК, Д., ЯЦКІВ, В., ТИМОЩУК, В., & ЯЦКІВ, Н. (2024). INTERACTIVE CYBERSECURITY TRAINING SYSTEM BASED ON SIMULATION ENVIRONMENTS. MEASURING AND COMPUTING DEVICES IN TECHNOLOGICAL PROCESSES, (4), 215-220.

14. The Android Platform Security Model / R. Mayrhofer et al. ACM Transactions on Privacy and Security. 2021. Vol. 24, no. 3. P. 1–35. URL: https://doi.org/10.1145/3448609 (date of access: 14.12.2024).

15. ТИМОЩУК, Д., & ЯЦКІВ, В. (2024). USING HYPERVISORS TO CREATE A CYBER POLYGON. MEASURING AND COMPUTING DEVICES IN TECHNOLOGICAL PROCESSES, (3), 52-56.

16. Palma C., Ferreira A., Figueiredo M. Explainable Machine Learning for Malware Detection on Android Applications. Information. 2024. Vol. 15, no. 1. P. 25. URL: https://doi.org/10.3390/info15010025 (date of access: 14.12.2024).

17. Tymoshchuk, V., Vantsa, V., Karnaukhov, A., Orlovska, A., & Tymoshchuk, D. (2024). COMPARATIVE ANALYSIS OF INTRUSION DETECTION APPROACHES BASED ON SIGNATURES AND ANOMALIES. Матеріали конференцій МЦНД, (29.11. 2024; Житомир, Україна), 328-332.

18. Structural detection of android malware using embedded call graphs / H. Gascon et al. CCS'13: 2013 ACM SIGSAC Conference on Computer and Communications Security, Berlin Germany. New York, NY, USA, 2013. URL: https://doi.org/10.1145/2517312.2517315 (date of access: 14.12.2024).

19. FlowDroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps: ACM SIGPLAN Notices: Vol 49, No 6. ACM SIGPLAN Notices. URL: https://dl.acm.org/doi/10.1145/2666356.2594299 (date of access: 14.12.2024).

20. Tymoshchuk, V., Vorona, M., Dolinskyi, A., Shymanska, V., & Tymoshchuk, D. (2024). SECURITY ONION PLATFORM AS A TOOL FOR DETECTING AND ANALYSING CYBER THREATS. Collection of scientific papers «ΛΌГОΣ», (December 13, 2024; Zurich, Switzerland), 232-237.

21. Ковальчук О.Я. Типи шкідливих програм для Android // Інформаційні моделі, системи та технології: Праці XII наук.-техн. конф. (Тернопіль, ТНТУ ім. І. Пулюя, 18-19 грудня 2024 р.) С. 126.

22. Тимощук, В., Долінський, А., & Тимощук, Д. (2024). СИСТЕМА ЗМЕНШЕННЯ ВПЛИВУ DOS-АТАК НА ОСНОВІ MIKROTIK. Матеріали конференцій МЦНД, (17.05. 2024; Ужгород, Україна), 198-200.

23. Enhancing security of linux-based android devices / Schmidt A. D. et al. International Linux Kongress. Lehmann, 2008. pp. 1-16.

24. Tymoshchuk, D., Yasniy, O., Maruschak, P., Iasnii, V., & Didych, I. (2024). Loading Frequency Classification in Shape Memory Alloys: A Machine Learning Approach. Computers, 13(12), 339.

25. Crowdroid | Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices. ACM Conferences. URL: https://dl.acm.org/doi/10.1145/2046614.2046619 (date of access: 14.12.2024)..

26. Бекер, І., Тимощук, В., Маслянка, Т., & Тимощук, Д. (2023). МЕТОДИКА ЗАХИСТУ ВІД ПОВІЛЬНИХ ТА ШВИДКИХ BRUTE-FORCE АТАК НА IMAP СЕРВЕР. Матеріали конференцій МНЛ, (17 листопада 2023 р., м. Львів), 275-276.

27. Zhou Y., Jiang X. Dissecting Android Malware: Characterization and Evolution. 2012 IEEE Symposium on Security and Privacy (SP) Conference dates subject to change, San Francisco, CA, USA, 20–23 May 2012. 2012. URL: https://doi.org/10.1109/sp.2012.16 (date of access: 14.12.2024).

28. Sahs J., Khan L. A Machine Learning Approach to Android Malware Detection. 2012 European Intelligence and Security Informatics Conference (EISIC), Odense, Denmark, 22–24 August 2012. 2012. URL: https://doi.org/10.1109/eisic.2012.34 (date of access: 14.12.2024).

29. Ванца, В., Тимощук, В., Стебельський, М., & Тимощук, Д. (2023). МЕТОДИ МІНІМІЗАЦІЇ ВПЛИВУ SLOWLORIS АТАК НА ВЕБСЕРВЕР. Матеріали конференцій МЦНД, (03.11. 2023; Суми, Україна), 119-120.

30. Tymoshchuk, V., Dolinskyi, A., & Tymoshchuk, D. (2024). MESSENGER BOTS IN SMART HOMES: COGNITIVE AGENTS AT THE FOREFRONT OF THE INTEGRATION OF CYBER-PHYSICAL SYSTEMS AND THE INTERNET OF THINGS. Матеріали конференцій МЦНД, (07.06. 2024; Луцьк, Україна), 266-267.

31. PermPair: Android Malware Detection Using Permission Pairs. IEEE Xplore. URL: https://ieeexplore.ieee.org/abstract/document/8886364 (date of access: 14.12.2024).

32. Mahindru A., Sangal A. L. MLDroid–framework for Android malware detection using machine learning techniques. Neural Computing and Applications. 2020. URL: https://doi.org/10.1007/s00521-020-05309-4 (date of access: 14.12.2024).

33. Tymoshchuk, D., & Yatskiv, V. (2024). Slowloris ddos detection and prevention in real-time. Collection of scientific papers «ΛΌГОΣ», (August 16, 2024; Oxford, UK), 171-176.

34. Rendle S., Schmidt-Thieme L. Pairwise interaction tensor factorization for personalized tag recommendation. the third ACM international conference, New York, New York, USA, 4–6 February 2010. New York, New York, USA, 2010. URL: https://doi.org/10.1145/1718487.1718498 (date of access: 14.12.2024).

35. Field-aware Factorization Machines for CTR Prediction / Y. Juan et al. RecSys '16: Tenth ACM Conference on Recommender Systems, Boston

Massachusetts USA. New York, NY, USA, 2016. URL: https://doi.org/10.1145/2959100.2959134 (date of access: 14.12.2024).

36. Тимощук, В., & Тимощук, Д. (2022). Віртуалізація в центрах обробки даних-аспекти відмовостійкості. Матеріали X науково-технічної конференції „Інформаційні моделі, системи та технології “Тернопільського національного технічного університету імені Івана Пулюя, 95-95.

37. AndroZoo | Proceedings of the 13th International Conference on Mining Software Repositories. ACM Conferences. URL: https://dl.acm.org/doi/10.1145/2901739.2903508 (date of access: 14.12.2024).

38. Stefanyshyn, I., Pastukh, O., Stefanyshyn, V., Baran, I., & Boyko, I. (2024). Robustness of AI algorithms for neurocomputer interfaces based on software and hardware technologies. In CEUR Workshop Proceedings (Vol. 3742, pp. 137-149).

39. Bayer I. fastfm: A library for factorization machines. Journal of Machine Learning Research. URL: https://www.jmlr.org/papers/volume17/15-355/15-355.pdf (дата звернення: 14.12.2024).

40. Get Started with xLearn ! – xLearn 0.4.0 documentation. Get Started with xLearn ! – xLearn 0.4.0 documentation. URL: https://xlearn-doc.readthedocs.io/en/latest/ (date of access: 14.12.2024).

41. Zagorodna, N., Skorenkyy, Y., Kunanets, N., Baran, I., & Stadnyk, M. (2022). Augmented Reality Enhanced Learning Tools Development for Cybersecurity Major. In ITTAP (pp. 25-32).

42. GitHub - maxherobrine/android_apk_datasets. GitHub. URL: https://github.com/maxherobrine/android_apk_datasets (date of access: 14.12.2024).

43. Баран, І. О. (2003). Високоточні обчислювальні алгоритми та система автоматизованого розрахунку дифузійних процесів в багатокомпонентних середовищах (Doctoral dissertation, Тернопіль, ТДТУ).

44. Sandeep H. R. Static Analysis of Android Malware Detection using Deep Learning. 2019 International Conference on Intelligent Computing and Control Systems (ICCS), Madurai, India, 15–17 May 2019. 2019. URL: https://doi.org/10.1109/iccs45141.2019.9065765 (date of access: 14.12.2024).

45. Quick and Accurate Android Malware Detection Based on Sensitive APIs / C. Zhao et al. 2018 IEEE International Conference on Smart Internet of Things (SmartIoT), Xi'an, 17–19 August 2018. 2018. URL: https://doi.org/10.1109/smartiot.2018.00034 (date of access: 14.12.2024).

46. Баран, І. О., & Воронін, В. С. (2020). До питання розробки системи збереження даних для хмарної платформи openstack. Збірник тез доповідей IX Міжнародної науково-технічної конференції молодих учених та студентів „Актуальні задачі сучасних технологій “, 2, 6-6.

47. Заікіна Д., Глива В. Основи охорони праці та безпека життєдіяльності. 2019. URL: https://doi.org/10.31435/rsglobal/001 (дата звернення: 14.12.2024).

48. Кучма М.М. Цивільна оборона (цивільний захист). Навчальний посібник. Львів : Магнолія 2006 . 2024. 296 с.

ДОДАТОК А

Тези конференції