

硕士学位论文

**基于移动API的恶意软件应用检测与加固方法研究**

学位申请人：Masroor Qadir

指导教师：Xiaozhi Du教授

合作导师：Masroor 教授

学科名称：工商管理 *or* 管理科学与工程

2024年02月

**Research on Mobile API-Based Malware Application Detection and Hardening Method**

A thesis submitted to

Xi’an Jiaotong University

In partial fulfillment of the requirements

For the degree of

Master of Computer Science

By

Masroor Qadir

Supervisor: (Associate) Prof. Xiaozhi Du

Associate Supervisor: Prof. Name Name

School of Software Computer Science

February 2024

**硕士学位论文答辩委员会**

**基于移动API的恶意软件应用检测与加固方法研究**

答辩人：Masroor Qadir

答辩委员会委员：

University Name大学Tittle of teacher: \_teacher’s name

(Examples)西安交通大学教授: 王树国\_\_\_\_\_\_\_\_\_\_\_\_

(Examples)西安交通大学教授: 王树国\_\_\_\_\_\_\_\_\_\_\_\_

(Examples)西安交通大学教授: 王树国\_\_\_\_\_\_\_\_\_\_\_\_

(Examples)西安交通大学教授: 王树国\_\_\_\_\_\_\_\_\_\_\_\_

答辩时间：2019年05月07日

答辩地点：西安交通大学where？

摘 要

在之前的研究中，利用API的特征与通过机器学习的权限静态特征共存，但本论文研究提供了利用机器学习和深度学习的共存的动态特征。这是研究和发现恶意软件或良性应用程序的深度。研究的目的是防止恶意软件的侵害。和检测总是有搜索空间来证明这一点，我们从 Drebin 和 Malgenome 驱动了我们的数据集，包括这些数据集的多种组合。我们检测恶意软件的方法基于新颖的动态共存方法，具有权限和 API 调用功能。使用动态共存级别 1 至级别 5 的多个阈值来寻找具有负频繁项的 FP-growth 算法 (FPM-IN) 和具有近似算法的 FP-Max (FPMAA)，这两种算法 FPM- IN 基于规则的关联挖掘方法应用了修改和 FPMAA 关联基于样本的技术。新数据集是根据 Drebin 和 Malgenome 的数据集构建的。这些数据集非常大，可用于恶意软件领域。

我们在本论文中的研究结果证明了利用动态共存特征进行恶意软件检测的实时性，结果表明Android恶意软件的识别具有很高的准确性，具有良好的性能。我们观察到随机分类器方法在 API 和权限特征的第 4 级共存组合下获得了 99% 的最大准确率。此外，本研究中使用的方法击败了最先进的模型，在 Malgenome 和 Drebin 数据集上获得了 99.4% 和 98.49% 的准确率，相应地，与最先进的 99% 和 99% 的准确率相当。 98%

**关 键 词**：深度学习；机器学习；共存；动态功能、API 权限

**论文类型**：研究报告

.

ABSTRACT

In the previous study the co-existence of features utilizing APIs and permissions static features through machine learning, but this thesis research offers dynamic features of co-existence using machine learning and deep learning. It is the depth of research and finding malware or benign apps. The purpose of research to secure from the malware. and detection always room for search to prove this, we have driven our dataset from the Drebin and Malgenome, including multiple combinations of these datasets. Our approach to detect malware novel based approach of dynamic co-existence features permissions and API Calls. Using multiple threshold for the level 1 to level 5 of dynamic co-existence for the finding the FP-growth algorithm with negative frequent items (FPM-IN) items and FP-Max with Approximate Algorithm (FPMAA), these two-algorithm FPM-IN rule-based association mining approach applied with modifications and FPMAA associate the sample-based technique. The new datasets were constructed implementing datasets from Drebin, and Malgenome. These datasets are very large and useable in the field of malware.

Our findings in this thesis to prove the real-time using dynamic co-existence features of malware detection and results to show that Android malware is identified with high accuracy, with performance. We observed Random Classifier method got the greatest accuracy 99% at the 4th level co-coexistence combination of API and permission features. Moreover, the method used in this study beats the state-of-the-art model, obtaining an accuracy of 99.4% and 98.49% on the Malgenome and Drebin datasets, accordingly, comparable to the state-of-the-art's 99% and 98%

**KEY WORDS**: deep-learning; machine-learning; co-existence; dynamic-features, API-permission;

**TYPE OF THESIS**: Research report

# 

目 录

1 摘要（英文 II

[1 摘要（中文 1](#_Toc160431786)II

[1 前言 1](#_Toc160431786)

[1.1 意义和背景 1](#_Toc160431787)

[1.2 研究现状概览 2](#_Toc160431788)

[1.2.1 Drebin研究现状 2](#_Toc160431789)

[1.2.2 Malgenome 2研究现状 2](#_Toc160431790)

[1.3 研究贡献 2](#_Toc160431791)

[1.4 研究目的及意义 3](#_Toc160431792)

[1.5 论文3主要研究内容 3](#_Toc160431793)

[1.5.1 所需平台 3](#_Toc160431794)

[1.5.2 加固平台 4](#_Toc160431795)

[1.5.3 数据收集 4](#_Toc160431796)

[1.5.4 处理不平衡 5](#_Toc160431797)

[1.5.5 热腾码 5](#_Toc160431798)

[1.5.6 阈值 5](#_Toc160431799)

[1.5.7 随机森林 6](#_Toc160431800)

[1.5.8 决策树 6](#_Toc160431801)

[1.5.9 逻辑回归 6](#_Toc160431802)

[1.5.10 卷积神经网络（CNN) 6](#_Toc160431803)

[1.5.11 递归神经网络 (RNNs) 7](#_Toc160431804)

[1.5.12 人工神经网络 (ANNs) 7](#_Toc160431805)

[1.5.13 求解矩阵 8](#_Toc160431806)

[1.6 论文的组织 9](#_Toc160431807)

[2 相关恶意软件检测分析 11](#_Toc160431808)

[2.1 简介 11](#_Toc160431809)

[2.2 恶意软件分析背景 11](#_Toc160431810)

[2.3 恶意软件检测理论分析 11](#_Toc160431811)

[2.3.1 不断变化的威胁格局 12](#_Toc160431812)

[2.4 恶意软件检测技术 12](#_Toc160431813)

[2.4.1 静态分析 12](#_Toc160431814)

[2.4.2 动态分析 12](#_Toc160431815)

[2.4.3 基于行为的分析 13](#_Toc160431816)

[2.4.4 机器学习 13](#_Toc160431817)

[2.4.5 混合分析 13](#_Toc160431818)

[2.5 技术评估与比较 13](#_Toc160431819)

[2.6 恶意软件检测方法 14](#_Toc160431820)

[2.6.1 静态方法 14](#_Toc160431821)

[2.6.2动态方法 14](#_Toc160431822)

[2.6.3基于行为的方法 14](#_Toc160431823)

[2.6.4 机器学习方法 14](#_Toc160431824)

[2.6.5 混合方法 14](#_Toc160431825)

[2.7 恶意软件检测的实证文献综述. 15](#_Toc160431826)

[2.8 简要总结 18](#_Toc160431827)

[3 动态共存和FPM-IN算法 19](#_Toc160431828)

[3.1 简介 19](#_Toc160431829)

[3.2 问题陈述 19](#_Toc160431830)

[3.3 动态共存和FPM-IN算法提出的模型 20](#_Toc160431831)

[3.3.1 负项频繁模式挖掘（FPM-IN）算法 20](#_Toc160431832)

[3.3.2负面项目的挑战 20](#_Toc160431833)

[3.3.3 应对挑战的方法 21](#_Toc160431834)

[3.3.4 负关联规则挖掘（NARM) 21](#_Toc160431835)

[3.3.5 FPM-IN中的频繁模式 21](#_Toc160431836)

[3.3.6 动态共存特征生成的新数据集 22](#_Toc160431837)

[3.4 动态共存结果实验及分析 22](#_Toc160431838)

[3.4.1 Drebin 动态共存特征结果 23](#_Toc160431839)

[3.4.2 马尔基因组动态共存特征结果 31](#_Toc160431840)

[3.5 简要总结 40](#_Toc160431841)

[4 动态共存特征和 FPMA 算法 41](#_Toc160431842)

[4.1 简介 41](#_Toc160431843)

[4.2 问题陈述 41](#_Toc160431844)

[4.3 动态共存和FPMA算法提出的模型 42](#_Toc160431845)

[4.3.1 频繁模式挖掘近似算法 (FPMAA) 42](#_Toc160431846)

[4.3.2 FPMA 算法面临的挑战 43](#_Toc160431847)

[4.3.3 使用 FPAMAA 时应对挑战的方法 43](#_Toc160431848)

[4.3.4 频繁模式 FPMAA 44](#_Toc160431849)

[4.3.5 新数据集生成的动态共存特征 44](#_Toc160431850)

[4.4 动态共存结果实验及分析 44](#_Toc160431851)

[4.4.1 Drebin 动态共存特性结果 45](#_Toc160431852)

[4.5 Melgenome数据集结果 51](#_Toc160431853)

[4.6 简要总结 57](#_Toc160431854)

[5 结论与建议 59](#_Toc160431855)

[5.1 1 结论 59](#_Toc160431856)

[5.2 建议 59](#_Toc160431857)

[致谢 61](#_Toc160431858)

[References 62](#_Toc160431859)

[附录 64](#_Toc160431860)

CONTENTS

[ABSTRACT(ENGLISH) II](#_Toc160416525)

[ABSTRACT(Chinese) III](#_Toc160416526)

[1 Preface 1](#_Toc160416528)

[1.1 Significance and Background 1](#_Toc160416529)

[1.2 Glance Of Current Research Status 2](#_Toc160416530)

[1.2.1 Research Status of Drebin 2](#_Toc160416531)

[1.2.2 Research Status of Malgenome 2](#_Toc160416532)

[1.3 Contribution Of Research 2](#_Toc160416533)

[1.4 Research Objectives And Significance 3](#_Toc160416534)

[1.5 Main Research Content Of Thesis 3](#_Toc160416535)

[1.5.1 Platform required 3](#_Toc160416536)

[1.5.2 Hardening platform 4](#_Toc160416537)

[1.5.3 Collect the data 4](#_Toc160416538)

[1.5.4 Handling imbalanced 5](#_Toc160416539)

[1.5.5 Hotencode 5](#_Toc160416540)

[1.5.6 Thresholds 5](#_Toc160416541)

[1.5.7 Random Forest 6](#_Toc160416542)

[1.5.8 Decision Tree 6](#_Toc160416543)

[1.5.9 Logistic Regression 6](#_Toc160416544)

[1.5.10 Convolutional neural networks (CNNs) 6](#_Toc160416545)

[1.5.11 Recurrent Neural Networks (RNNs) 7](#_Toc160416546)

[1.5.12 Artificial Neural Networks (ANNs) 7](#_Toc160416547)

[1.5.13 Solving Matrices 8](#_Toc160416548)

[1.6 Organization Of Thesis 9](#_Toc160416549)

[2 Related Malware Detection Analysis 11](#_Toc160416550)

[2.1 Introduction 11](#_Toc160416551)

[2.2 Background Of Malware Analysis 11](#_Toc160416552)

[2.3 Malware Detection Theoretical Analysis 11](#_Toc160416553)

[2.3.1 The Evolving Threat Landscape 12](#_Toc160416554)

[2.4 Malware Detection Techniques 12](#_Toc160416555)

[2.4.1 Static analysis 12](#_Toc160416556)

[2.4.2 Dynamic analysis 12](#_Toc160416557)

[2.4.3 Behavior-based analysis 13](#_Toc160416558)

[2.4.4 Machine Learning 13](#_Toc160416559)

[2.4.5 Hybrid Analysis 13](#_Toc160416560)

[2.5 Evaluation and Comparison of Techniques 13](#_Toc160416561)

[2.6 Approach Of Malware Detection 14](#_Toc160416562)

[2.6.1 Static Approach 14](#_Toc160416563)

[2.6.2 Dynamic Approach 14](#_Toc160416564)

[2.6.3 Behavior Based Approach 14](#_Toc160416565)

[2.6.4 Machine Learning Approach 14](#_Toc160416566)

[2.6.5 Hybrid Approach 14](#_Toc160416567)

[2.7 Empirical Literature Review Of Malware Detection. 15](#_Toc160416568)

[2.8 Brief Summary 18](#_Toc160416569)

[3 Dynamic Co-existence And The FPM-IN Algorithm 19](#_Toc160416570)

[3.1 Introduction 19](#_Toc160416571)

[3.2 Problem Statement 19](#_Toc160416572)

[3.3 Dynamic Co-existence And FPM-IN Algorithm Proposed Model 20](#_Toc160416573)

[3.3.1 Frequent Pattern Mining with Negative Items (FPM-IN) Algorithm 20](#_Toc160416574)

[3.3.2 Challenges with Negative Items 20](#_Toc160416575)

[3.3.3 Approaches for Addressing the Challenges 21](#_Toc160416576)

[3.3.4 Negative Association Rule Mining (NARM) 21](#_Toc160416577)

[3.3.5 Frequent Pattern In FPM-IN 21](#_Toc160416578)

[3.3.6 New Datasets Generated By The Dynamic Co-existence Features 22](#_Toc160416579)

[3.4 Experimental of Dynamic Co-existence Result and Analysis 22](#_Toc160416580)

[3.4.1 Drebin Dynamic Co-existence Features Results 23](#_Toc160416581)

[3.4.2 Malgenome Dynamic Co-existence Features Results 31](#_Toc160416582)

[3.5 Brief Summary 40](#_Toc160416583)

[4 Dynamic Co-existence Features And FPMA Algorithm 41](#_Toc160416584)

[4.1 Introduction 41](#_Toc160416585)

[4.2 Problem Statement 41](#_Toc160416586)

[4.3 Dynamic Co-existence and FPMA Algorithm Proposed Model 42](#_Toc160416587)

[4.3.1 Frequent Pattern Mining Approximate Algorithm (FPMAA) 42](#_Toc160416588)

[4.3.2 Challenges with FPMA Algorithm 43](#_Toc160416589)

[4.3.3 Approaches for Addressing the Challenges while using FPAMAA 43](#_Toc160416590)

[4.3.4 Frequent pattern FPMAA 44](#_Toc160416591)

[4.3.5 New Datasets Generated Dynamic Co-Existence Features 44](#_Toc160416592)

[4.4 Experimental of Dynamic Co-existence Result and Analysis 44](#_Toc160416593)

[4.4.1 Drebin Dynamic Co-existence Features Results 45](#_Toc160416594)

[4.4.2 Malgenome, Dynamic Co-existence Features Results 51](#_Toc160416595)

[4.5 Brief Summary 57](#_Toc160416596)

[5 Conclusions and Suggestions 59](#_Toc160416597)

[5.1 Conclusion 59](#_Toc160416598)

[5.2 Suggestions 59](#_Toc160416599)

[Acknowledgements 60](#_Toc160416600)

[References 61](#_Toc160416601)

[Appendix 63](#_Toc160416602)

[Decision of Defense Committee 64](#_Toc160416603)

[General Reviewers List 65](#_Toc160416604)

Declarations

LIST OF ABBREVIATIONS

|  |  |
| --- | --- |
| ANN | Artificial Neural Network |
| RNN | Recurrent Neural Network |
| CNN | Convolutional Neural Network |
| FPM-NI | Frequent Pattern Mining with Negative Items |
| FPMAA | Frequent Pattern Mining Approximation Algorithm. |
| API | Application Programming Interface |
| IDS | Intrusion Detection System |
| IPS | Intrusion Prevention System |
| ANN | Artificial Neural Network |
| RNN | Recurrent Neural Network |
| CNN | Convolutional Neural Network |
| FPM-NI | Frequent Pattern Mining with Negative Items |
|  |  |

章的MathType的章标记（打印前将其字体颜色变为白色，在打印预览中看不见即可）：

# Preface

Malicious malware is becoming more prevalent in the world of technology and this trend is not slowing down. In 2023, the internet had 4.5 billion malware violence which was a 2% increase from the previous year [24]. These attacks was continuously highlighting the critical requirements of strong and agile defenses.

Traditional methods and static methods based on malware detection, while effective in identifying known threats, are increasingly rendered obsolete by sophistication and the rapid proliferation of new malware strains. These are the new requirements for exploring novel and comprehensive approaches that will protect us from unseen malware. The previous researchers used static features, which was not very explained, and it’s using static methods and the most relevant features they skipped.

The previous research depended on static features, dynamic analysis, and a dynamic approach, while our research completely relied on dynamic features, dynamic analysis, and a hybrid approach. For the purpose of the novel-based approach is to achieve new or improved results that will compare with the other methods.

In this, our aim is to use novel-based dynamic features of co-existence to detect the malware through an API, which we have constructed, as well as our dynamic analysis to detect the malware on the platform as software in the real world. The API we construct not only proves by the result but actually in our system that can be visible how to detect malware. This is our experiment in the real world. In our thesis the chapter 1 is associate with introduction, chapter 2 define the literature review, chapter 3 define FPM-IN its result. Chapter 4 defined our FPMAA and its results, and chapter 5 conclusion and suggestions.

## Significance and Background

In the background of work for malware detection An Efficient Android Malware Detection Method Based on Hybrid Analysis [20], an Android malware detection method, combines static and dynamic analysis for improved accuracy. It first extracts features like permissions from the static analysis and then monitors the app's behavior in a controlled environment, capturing the sequence of system calls dynamic analysis. This raw call sequence is change into a function call graph (FCG) to reduce data size while preserving call order. Finally, features from both static and dynamic analysis are combined and fed into a machine learning model to differentiate between benign and malicious apps. In [32] research The problem was to detect the malware Android operating system have provided a decent approach based on static analysis of the android apk for feature extraction like API Calls, Intents, Permissions and Command signature using the Drebin and Malgenome datasets by the static analysis. In [2] the facing of issue to detect malware from these datasets Drebin, Malgenome and CIC Maldriod 2020, finding the co-existence malware detection using static co-existence features to detect malware they have made different level of combination static analysis. Contain the dataset for the research Drebin, Malgenome, and CIC\_Maldriod2020 API and Permision to analyze the co-existence features it accuracy 98% in Malgenome. In [31] Mobile malware is a growing threat, and there is a need for effective detection techniques. There are a variety of mobile malware detection techniques available, but no single technique is perfect. A combination of techniques is often needed to provide effective malware detection. The challenges of mobile malware detection include the ever-changing nature of malware and the need to protect user privacy [31]. Discussing the type of analysis to detect the malware and our malware detection strategy.

## Glance Of Current Research Status

In this section we explained about research status. malware remains a significant threats to individuals and organizations causing financial losses, data breaches, and system disruptions. Effective methods are critical for maintaining the cybersecurity.

Recent of years the scientist choose the hybrid analysis and experiment of malware detection. As [31, 32, 33, 45, 22] these are researches static and hybrid analysis. We conclude these malware authors continuously develop new techniques to evade detection, demanding constant adaptation and improvement of detection algorithms. While deep learning models offer advantages, their decision-making processes remain challenging to interpret, hindering trust and explain ability.

### Research Status of Drebin

This study researcher uses the Drebin dataset, which covering the period from 2010 to 2022 [45]. The dataset includes both benign and malicious software samples, with the number of goodware applications significantly exceeding the number of malware (123,453 vs. 5,560 in August 2010 - October 2012, and 175,327 vs. 15,892 in 2019). The dataset also contains a large number of features (1,230,854 in 2019) extracted from the samples using various techniques and tools. However, a previous study identified a significant portion of these features as redundant and ineffective for malware detection, suggesting a need for feature selection or reduction techniques.

### Research Status of Malgenome

The Malgenome dataset, also called to as the Genome Project a review paper [46], that shows 1260 samples of malware collected in 2012. While some concerns exist regarding the potential privacy impacts of including data from specific companies, the dataset's unique features are pertaining to malware features make it a valuable resource for research and analysis. However, it is crucial to acknowledge potential privacy issues associated with using this dataset, and appropriate moral guidelines and best practices.

## Contribution Of Research

Previous studies on malware detection have investigated several approaches. This thesis introduces a new strategy that differs from previous methods by using dynamic co-existence properties. We present and assess two algorithms, FPM-IN and FPMAA, designed to enhance current datasets (Drebin and Malgenome) by integrating dynamic co-existence features from levels 1 to 5. The enhanced datasets are examined and used to build machine and deep learning models for efficient malware detection

## Research Objectives And Significance

The thesis aims to deliberate about these specific points the problem which we defined we are going to describe these. The new approach to make co-existence dynamic features

1. Using the dynamic analysis, hybrid approach and dynamic features of co-existence
2. All the frequent items within the datasets
3. In-depth analysis and validation of assumptions about the co-existence features
4. Limitation we will describe its only API calls and permissions. And computational cost
5. Level will be correct defined with different thresholds 50%, 25%, 10%, 5% and 1%
6. API for detect malware using this novel based malware approach on a real-time
7. Dynamic co-existence features with deep learning models accuracy to detect malware
8. Sample-based approach with FP-Max algorithm to know about large dataset can be worked well and what will new innovation with malware detection

We also would like to know how dynamic feature approaches can be applied to better detect hidden patterns of malware threats for Android using machine learning. Additionally, we shall probe into how dynamic co-existence characteristics affect the accuracy and performance of models employed in recognizing Android malware. These are aim to investigate.

In this thesis of dynamic co-existence features it can be help to detect malware API calls and permissions. This is new idea about within the co-existence which we will using level 1 to level 5 based on thresholds it will create most potential features of detection malware.

## Main Research Content Of Thesis

In this thesis we are explain the core of functionalities and which contents we need to explain these are the general content among them to prove our novel based dynamic features detection of malware.

### Platform required

In the thesis required laptop specification processor core i7 5th generation, processor x86 64-bit CPU (Intel or AMD architecture), minimum 8 GB (8 GB recommended), disk storage minimum 5 GB free space, and operating system will be more than windows 7 or higher. For software need to use Python version 3.6 or later and tensorflow version 2.0. additional need which GPU highly recommended for faster training. IDE for write the code visual studio code or google colab.

### Hardening platform

Python proposes a powerful toolkit for implemented systems and application hardening techniques, improving security opposed to a varied type of attacks. Critically integrating those strategies greatly raises resilience and secures sensitized data [1]. We utilize those methods, and we build our atmosphere to secure our platform/program.

### Collect the data

All around this stage, we are going to provide more information about our data collection method. Using established datasets in the Android malware detection zone, the study boosts an exact structure to analyze the model of work. Those features are huge and diversified APK collections from services like Drebin, and Malgenome, allowing researchers a well-developed splash for examination of program. Establishing upon past study using Drebin and Malgenome databases [2], that provides a variety of splash for malware samples, the study found the Drebin-215 dataset extremely beneficial. That set, spanning 215 features over permissions, APIs, intents, and command signatures, proving enlightening in understanding malware behavior. Likewise, the Malgenome-215 dataset, with 215 attributes, puts more light on the features of malicious programs. Also, the CIC maldroid2020 dataset is a more modern and understandable collection of both malware and benign APKs. That dataset, received from the Canadian Institute for Cybersecurity, supply a more in-depth picture of the Android app landscape. This study authorized a rigorous and understandable examination of the appropriate technique by engaging the obtain knowledge and experience from past investigations and applying these stable datasets

#### Drebin-215

Throughout the time of our study investing Android malware, the Drebin dataset was a wonderful resource. Its features 15,031 diverse apps, containing 5,550 from 179 different malware families and 9,477 benign software. The data was attentively collated between 2010 and 2012, showing a detailed historical view on the development of Android malware. Utilizing this broad dataset, the researchers recovered 181 distinct features, which consist of 109 permissions and 72 APIs. Such attributes are critical indicators of an application's behavior and possibly unfavorable purpose [2]. This is a popular dataset in the domain of malware detection.

#### Malgenome-215

The Malgenome dataset is an understandable collection of Android malware programs covering the years 2012 to 2015. That includes 3,798 individual apps, 1,260 of which belong to 49 different malware families and 2,538 of that are benign. This dataset illustrates the evolution of Android malware over time. Researchers show that they were able to extract 181 unique properties, including 109 permissions and 72 APIs, by imposing the huge data offered by the Malgenome dataset [2]. These attributes are necessary indicators of an application's behavior and potential unfavorable intent. The development and testing of Android malware detection algorithms have malgenome dataset instrument. Its diverse set of applications and derived features enables researchers to properly evaluate the effectiveness of their proposed solutions [2].

### Handling imbalanced

Our research work used the random under-sampling method before generating the combinations and during the single HotEconded procedure this process to build a reliable dataset, the combinations must rely on roughly an equal amount of malware and benign samples, and it must be constructed with authentic data that can distinguish malware from benign programs. The random under-sampling methodology was chosen because it is a simple method for removing samples without imposing any constraints on the data. Furthermore, several studies have shown that when using random under-sampling, the accuracy does not differ considerably between tests [1]. Furthermore, there are enough instances to train and test machine learning and deep learning models, and each program is decompiled individually. As a result, deleting some apps does not result in significant data loss. Our thesis research discovered and removed single extracted skills from the all further during data processing because follow to prevent data from being unnecessary, copy data were detected and removed. Types and bad format led to unfaithful authorization data, prompting their transfer. Permissions requested by a little number of applications, generally for user groups or specific markets (e.g., Huawei), appear to be less informative and were prohibited. This purifying technique reduces the initial 1633 retrieved permissions to 825, emerges in a group of correct, crucial and often employed qualities for more investigation.

### Hotencode

In the digital age of machine learning, computers require statistics and absolute data that the sklearn's One Hot-Encoder begins in, serving as a converter. It takes such features and puts those into a network of binary attributes, where every attribute represents a single categorization and contains either a 1 (it's present) or a 0 (it's not) [4]. Building training data additional useful One-hot encoding enables the data to be employed for training more understandable and usable, One-hot encoding permits algorithms requiring numerical inputs to structured variables. We transform the data into numerical figures to make it untroubled for machine learning algorithms with no concern. We worked simply on a single column that we are targeting the foundation of qualities.

### Thresholds

In the ever-evolving battlefield of malware detection, thresholds form a delicate frontline against unseen threats. At the fulcrum sits 50%, the pivotal point where suspicion solidifies into action [7]. Signatures matching half the criteria scream "malware!" triggering immediate containment. Descending, 25% and 10% mark caution zones, where suspicious behaviors, though not definitive, warrant deeper investigation. Sandboxing, behavioral analysis, and threat intelligence come into play, dissecting anomalies with a scrutinizing lens. 5% and 1% then become whispers in the noise, statistically unlikely blips on the radar [2]. Yet, even these faint echoes warrant a listen, for in the intricate tapestry of malware, a single odd thread can unravel the entire web. These thresholds, constantly adapting to the shifting landscape of cyber threats, remind us that in the face of hidden danger, vigilance never sleeps, ever watchful for the telltale signs, no matter how faint, that reveal the lurking shadows within [3]. We are using threshold using 50%, 25%, 10%, 5% and 1% these each percent we create a file of co-existence. These going to make sure about the features of each a file and we are using 8 files. Each file have a level of the threshold to generate co-existence features.

### Random Forest

In the constantly changing field of cybersecurity operational categorization models are important for properly recognizing and reducing malware or benign app.  The random forest classification system stands out for its integrated machine learning approach it associate with decision tree classifier which will achieve the great accuracy [13]. in our conclusion of this model from 2020 to 2024 proves that it's helpful in different security scenarios [13]. it shows effectiveness in detecting malware for Android platforms and demonstrating the capacity to manage unbalanced datasets and tolerate adversarial attacks. In our analysis emphasizes the success in recognizing network intrusion attempts hailing to recognize the different patterns of malware. the result of this model most notably their ability to adapt to overfitting, capacity to handle highly dimensional input, and comprehension. However, it's crucial to identify possible restrictions. Configuring hyperparameters for optimal efficiency may be complex, and feature selection is critical for accurate measurements [14].

### Decision Tree

The decision tree classifier algorithm is the best model against malware due to its data structure shape and this model works as leaf leaf-based node to explore feature-based queries until reaches the final leaf node [15]. Another study [15] showed their success in categorizing Android apps as dangerous or benign, emphasizing their interpretability as a useful tool for understanding the reasons contributing to their classifications.

### Logistic Regression

Logistic Regression stands as a useful technique for assessing the likelihood underlying destructive activities in the difficult realm of malware detection. the statistical method evaluates numerous attributes of programs determining the likelihood of them belonging to the malware category. as it compared decision trees with their distinct options Logistic Regression estimates a probability delivering a deeper insight on probable viruses. study by [16, 17] proves its ability in categorizing Android applications based on their API and permission requests features delivering useful insights into the malware and benign app aspects connected with distinct app behaviors.

### Convolutional neural networks (CNNs)

In Convolutional neural networks (CNNs) are strong tools in the field of identifying malware. They may analyze complicated data pictures, such as imagery and patterns and extract distinctive characteristics and patterns that can assist uncover hazardous data buried inside pictures, network information, and app coding. CNNs replicate the framework of the natural visual structure, making them appropriate for this kind of job. Recent research from 2020 to 2024 has revealed the incredible potential of CNNs. The authors [18] prove their efficacy in identifying malware-infected Android applications by assessing app icons and images, achieving excellent accuracy levels despite inadequate training data. additionally exhibit their capacity to spot dangerous network traffic patterns by time-series analysis, delivering full defense against shifting malware.

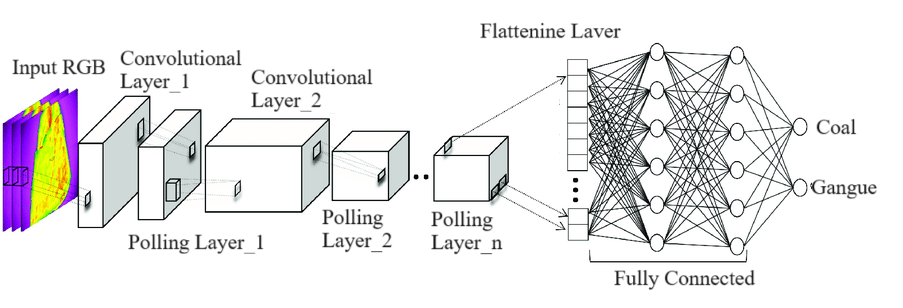


Figure 3-2 Applying algorithm and process to generate co-existence features

### Recurrent Neural Networks (RNNs)

In spotting bad software, Recurrent Neural Networks (RNNs) give a new way by looking at step-by-step data, finding secret patterns in app actions, internet traffic, or code use. While fixed ways just see data alone, RNNs see the order and time links among parts, making them good at finding harmful sets that simple ways might miss [19].

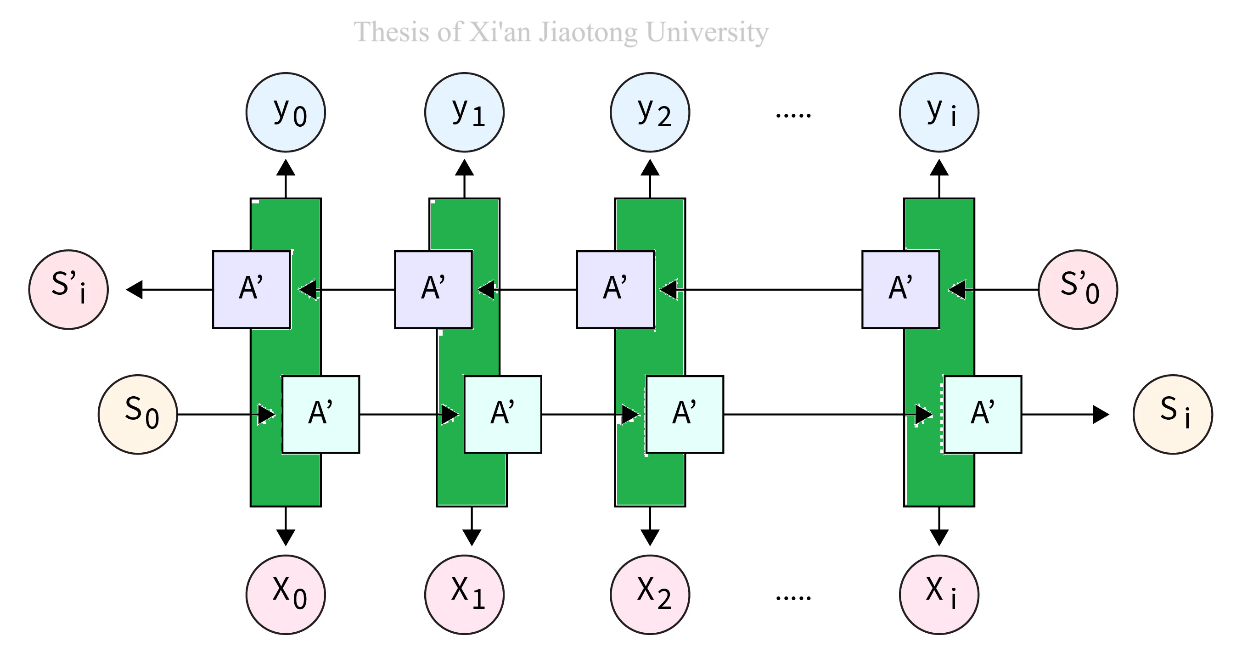


Figure 3-3 RNN model which used in the thesis, bidirectional to identify malware and benign app

### Artificial Neural Networks (ANNs)

In the fast world of finding bad software, Artificial Neural Networks (ANNs) could be very good at learning and sorting complicated patterns in data. ANNs are like spider webs, copying the human brain, they have lots of connected parts (neurons) working together to find detailed features and connections. They can change to fit different data, which makes them good for looking at different parts of possible bad software, like app code, network use, and how a system works [21, 22].

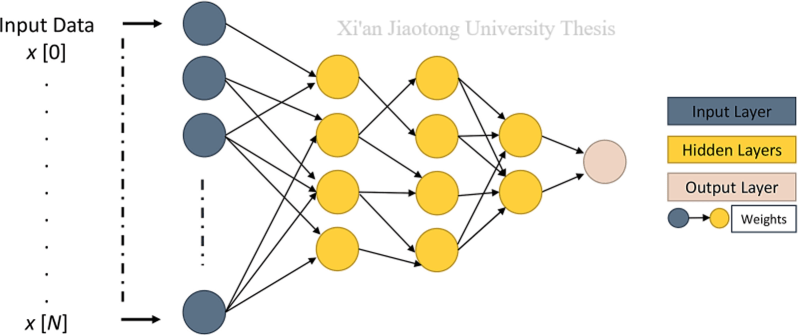


Figure 3-4 ANN model which inspire from the human brain.

### Solving Matrices

This section only for model score, how we can find the accuracy, precision, recall, F1, hamming loss, and in deep learning models we are using average precision. We used Confusion matrix to evaluate our model’s performance. It can be helpful to show the errors of model and visualize our model. About [44] article, it can be calculated by the mathematical calculation.

The FPR is the percentage of benign samples that the model incorrectly classifies as malware. The following equation can be used to calculate the FPR.

* FP denotes the number of false positives (benign samples that are incorrectly classified as malware)
* TN stands for the number of true negatives (benign samples that are correctly classified as benign)

The FNR is the percentage of malware samples that the model incorrectly classifies as benign. You can use the following equation to calculate the FNR:

The number of false negatives is denoted by FN (malware samples that are incorrectly classified as benign) The number of true positives is denoted by TP (malware samples that are correctly classified as malware)

The F1 score it used to measure accuracy. Its weight average of the precision and recall, 1 is for the best score and 0 is called as worst.

## Organization Of Thesis

1.1 Preface

* Background of the Study
* Current Status of Research.
* Main Contributions
* General Functionalities of Research
* Research Questions/ Aim.

1.2 Literature Review

* Critically analyze relevant research
* Malware Detection Techniques
* Malware approaches
* Empirical analysis
* Targeted aim of the thesis

1.3 FPM-IN Algorithm and Create Dynamic Features of Co-existence

* Ensure clarity and provide sufficient detail for replication.
* Problem Statement
* Critical Thinking and Solutions
* Proposed model
* Experimental Analysis and Results
* Summary of Chapter

1.4 FPMAA Algorithm and Create Dynamic Features of Co-existence

* Ensure clarity and provide sufficient detail for replication.
* Problem Statement
* Critical Thinking and Solutions
* Proposed model
* Experimental Analysis and Results
* Summary of Chapter

1.6 Conclusion and Future works

* Summarize our keys and reiterate the significance of our research.
* Briefly suggest any recommendations or future research directions relevant to the topic.

1.7 References

* List all sources cited in your thesis using a consistent referencing style.

1.8 Appendices (code)

.

# Related Malware Detection Analysis

## Introduction

The digital landscape thrives on connectivity and the constant exchange of information. However, this interconnectedness also creates fertile ground for malicious actors who develop and deploy malware – software designed to cause harm or disruption. Malware can take many forms, from simple data-stealing programs to sophisticated ransomware that can cripple entire systems. The consequences of a successful malware attack can be devastating, leading to financial losses, compromised sensitive information, and disrupted critical operations.

In the face of this ever-present threat, effective malware detection techniques are crucial for safeguarding our digital lives. These techniques act as a shield, identifying and neutralizing malicious software before it can wreak havoc. By understanding the various methods of malware detection, we can develop robust defenses and contribute to a more secure digital environment.

This chapter delves into the world of Android malware detection, novel approach dynamic features of co-existence. We will examine the underlying principles of each methods, analyze their strengths and limitations, and discuss their suitability in different scenarios. These are feature based techniques to detect malware. in our thesis we are using dynamic analysis a novel based malware detection using dynamic features of co-existence. It is novel approach to detect malware and real-time analysis about the malicious and benign app. The sandbox of our system and using API feature to detect malware

## Background Of Malware Analysis

The ever-growing risk posed by malware for Android needs fresh approaches. Recent study examines combining several learning algorithms, assessing varied sets of features generated from static, dynamic, and hybrid statistical approaches, to effectively detect dangerous programs. While previous studies (described in future parts and compared in Table 1) study individual aspects, this work provides a fresh technique. Our method utilizes the relationship between co-existence features across different levels to boost the precision of detection. This is the first effort to leverage characteristic co-existence at various levels, perhaps leading to a more robust and holistic strategy for Android malware detection.

## Malware Detection Theoretical Analysis

Our thesis we are explaining the has some areas to explore in the thesis of malware detection in the literature analysis the author must know about the previous researches, that researches has theoretical analysis we are going to discuss each point of malware detection.

In this thesis we are discussing the three-section static, dynamic and hybrid analysis these are major analysis of these parts.

### **The Evolving Threat Landscape**

As we conclude our literature it is the term most of researches used [2, 6, 8]. In a book section [21] its written the landscape of cyber threats particularly malware is constantly evolving, posing, a significant challenge for individual application or organization in the dynamic environment.

#### **Continuous Innovation by Attackers**

Attackers continuously develop new types of malware with novel functionalities [8], making it difficult for traditional detection methods to keep pace. malware designed as specific purposes like stealing financial information [23], disrupting critical infrastructure, or launching coordinated attacks Malware authors to bypass existing security measures to employ ever-sophisticated techniques. Obfuscation techniques rely on masking malicious code, exploiting vulnerabilities in software and hardware, and leveraging social engineering tactics to improve their own security. constantly developing new ways to evade malicious detection by authors. This includes techniques like polymorphism code injection , and living-off-the-land.

#### **Challenges for Defenders**

It difficult for defenders to keep up and implement effective preventive measures in the rapid pace of innovation by attackers. tools to stay ahead of the evolving threat landscape need to continuously update their knowledge by the security professionals [28]. become increasing complex, with interconnected devices and software applications in the modern systems world. complexity creates more potential entry points for attackers and makes more harder to identify and mitigate vulnerabilities. cybersecurity, making it challenging to implement comprehensive security providing often have limited resources to Organizations and invest in the latest tools and technologies to combat evolving threats. need raising awareness among users about cyber threats and best practices for secure online behavior, including being cautious about suspicious emails, attachments, and website links

## **Malware Detection Techniques**

In this literature review we are explaining the first how to detect the detection techniques malware change it pattern or hacker make the malware to using these technique

### Static analysis

The static analysis is the analysis of malware without program running code, it can be suspicious features specific kind of string functions, code, patterns associate with known malware [2, 31, 32].

### Dynamic analysis

The dynamic known by the running program on the any sandbox, it can be computer, or environment-controlled program that can be monitoring the behavior. That can observe how program interacts the system, file, or network to find the potential of malicious activities [2 4, 7, 31]

### Behavior-based analysis

Taking anomaly detection a step further, this technique investigates the actual activities and behavior of programs after execution. In dynamic analysis of behavior analysis sandboxes provide an isolated setting for monitoring program action, indicating suspicious behaviors such as attempts to access sensitive data or it could be edit system files. these strategies help us to identify the most of complex malware or benign app that might bypass signature-based approaches [26]. It's used in the advanced threats of  detection systems of malware analysis platforms, and endpoint of the security services.

### Machine Learning

This fast-developing sector plays a vital role in current malware detection. By analyzing huge amounts of data and recognizing the patterns. the machine learning algorithms can recognize small irregularities and even forecast approaching attacks by malware or benign apps. Many software algorithms develop gradually inform tp the system that is continually responding to the evolving threats of the environment [25]. It used by the advanced settings of malware identification networks, computer security solutions, and network security analytics systems.

### Hybrid Analysis

In the fast developing detection sector to analysis about the hybrid analysis, static and dynamic, include with machine learning and behave analysis the mixture of combination of analysis it’s called hybrid analysis. In the malware have lot prospective to analyze the behavior of malware in our security sector [2, 25, 27].

## Evaluation and Comparison of Techniques

Evaluating and comparing different malware detection techniques is crucial for choosing the most effective approach for a specific situation. **Accuracy, False Positives, False Negatives, Detection Rate, Efficiency and Resource Consumption these are also techniques to detect the malware. explain with the table in this chapter table 2-1**

|  |  |  |
| --- | --- | --- |
| Technique | Strengths | Weaknesses |
| Static Analysis | Fast, lightweight, identifies known malware variants | Don’t detect novel malware, vulnerable to obfuscation techniques |
| Dynamic Analysis | Can detect novel malware and evasion techniques | Slower, resource-captures, may not capture all malicious behavior |
| Signature-based Detection | Fast, effective against known malware using database | Ineffective against novel malware, requires frequent updates |
| Machine Learning and Deep Learning | Can learn complex patterns and improve over time, adaptable to evolving threats | Requires large datasets for training, potential for bias and lack of explain ability |

**Table 2-1 Comparison of techniques analysis of different way to malware**

## Approach Of Malware Detection

### Static Approach

In malware detection, the static approach refers to methods that analyze potential threats without actually executing them [2]. This analysis is primarily based on examining the structure, properties, and characteristics of the suspected malware. Analyzing code or files without execution is inherently faster and requires less computational resources compared to dynamic analysis. This makes static methods ideal for initial screening and scanning large numbers of files [25]. known malware by comparing characteristics like file size, code patterns, or resource usage with pre-defined signatures or databases of malicious code by static analysis which can be effectively identify . By examining code structure and patterns, static analysis can sometimes identify potential vulnerabilities in software that attackers might exploit to introduce malware [28]. This can serve as a preventive measure to improve software security. It called as unique code characteristics to detect malware.

### Dynamic Approach

Basically, for dynamic appoarch malware detection on the sandbox, run the program and observe the behavior of malware. This actual harm to computers from malware, as we have experienced with the Windows file, corrupts or damages the operating system, while we also analyze different autogenerated files created that harm our computers. The attacker invented new techniques to get the data; these are called dynamic approaches. the disadvantage is a slow processing and is time consuming [25].

### Behavior Based Approach

This approach focuses on **monitoring the behavior of a program** while it's running to detect malicious activity. As described previously in the context of dynamic analysis. Running the program in a controlled space to prevent harm. Tracking the program's interactions with the system, network, and files. Identifying suspicious behavior patterns that indicate malicious intent.

### Machine Learning Approach

This approach utilizes **machine learning algorithms** to analyze various features of a file or program to **classify it as malicious or benign**. These features can include Size, file type, origin, Function calls, API usage, code patterns, System calls, network activity, file access attempts **Machine learning algorithms are trained on a large dataset** of labeled malware and benign files. Based on this training, they learn to identify patterns and relationships between features and the likelihood of them being malicious

### Hybrid Approach

This approach combines the strengths of both **behavior-based and machine learning approaches.** It leverages the **detailed analysis of program behavior** with the **efficiency and adaptability of machine learning** Machine learning algorithms perform a**fast and efficient check**on the file using features like file type, size, and basic code structure. If the initial screening raises suspicion, the file is sent for**deeper analysis**using the behavior-based approach in a sandbox environment The final verdict on the file's maliciousness is made by**combining the results**from both stages.

## Empirical Literature Review Of Malware Detection.

The authors [29] employed a range of artificial intelligence addresses to train their models, such as random forest training, support vector machine learning, and neural networks as well. Researchers tested the efficacy of their models using a held-out test set with data. The researchers additionally offered two novel learning methods to increase the long-term viability of these models gradual learning and transfer training. Progressive learning allows the models to be upgraded with new information without retraining the whole model. The transfer learning method employs a model that has been trained on an associated issue as the basis for training the model on the malware detection job. the greatest accuracy of 97.8% and an F1-score of 98.8%. The findings imply that their strategies help preserve the performance of computer learning-based Android malware detection systems over time. Overall, the study delivers a noteworthy contribution to the field regarding machine learning-based Android malware detection. However, additional study is needed to test the proposed approaches on a bigger and more diversified dataset and contrast them to other state-of-the-art methods.

In [30] this study analyzes feature selection techniques for boosting the use of machine learning for the detection of Android malware. Detecting Android malware efficiently requires picking which are the most relevant data points (features) for machine learning algorithms. However, there's no consensus on which traits are most effective or which feature selection approaches provide the greatest outcomes. The study investigates three key feature types: Permissions, Intents, and API Calls. They utilize 11 distinct feature selection methods on different combinations of these feature groupings and test their effectiveness across many machine learning classifiers. this research employs static analysis for Android malware detection. The static analysis method assesses the code and its structure without running it, concentrating on elements like permissions, intents, and API calls used in the program.

In [2] The simultaneous presence of specific permissions, APIs, and other features might indicate malicious software. For example, an app that seeks both the right to access the user's contacts and the authorization to send SMS messages is more likely to be a dangerous app than an app that simply asks for one of these permissions. Feature selection is the procedure for picking the most relevant characteristics from a dataset. In the case of Android malware detection, feature selection is crucial because it can help increase the precision of a machine learning classifier. The paper only evaluates the proposed approach on three datasets. It would be interesting to see how the approach performs on other datasets, such as the Contagio dataset, which is a larger and more diverse dataset. The approach is based on the co-existence of features, which means that it is vulnerable to obfuscation techniques that can break the co-existence of features.The paper does not discuss the limitations of the proposed approach in detail. For example, the paper does not discuss how the approach would scale to large datasets or how it would handle new and emerging types of malware.

In "Android Malware Identification Using Machine Learning Methods," Raghuvanshi and Singh (2022) present a static evaluation method to detect malicious Android applications. They extract elements like API calls, permissions, and intent data from Android APK files without executing them. Using machine learning models like Random Forest and Support Vector Machines with dimensionality reduction, they obtain high accuracy of 98.87%, and 97.56% for determining malware and benign programs on both the Drebin and Malgenome datasets. This technique presents a potential option for quick malware detection on Android devices but may miss more complex attacks that rely on dynamic behavior. Further study might examine merging both static and dynamic analytic approaches and evaluating more diversified artificial intelligence models for even greater detection skills.

In [31] The EAODroid approach is technically solid because it is built on widely recognized ideas of API incorporation, grouping, and deep learning. The algorithm has also been tested empirically on a vast and broadened dataset of benign and malicious Android apps, where it scored a success rate of over 99% for identifying malware.  API embedding is an approach that turns every API into a dense vectors representation, and which preserves the meaning of the links between the APIs. This permits the EAODroid approach to collect patterns in the order in which APIs are applied, even if the particular APIs that are used by the malware shift. Grouping is a method that combines related data points together. The EAODroid approach employs grouping to gather the API embeddings into a set of prototypes. This allows the EAODroid approach to discover the most frequent sequence of APIs that are utilized by malware. Machine learning is a technology that allows computers to learn from data. The EAODroid technique employs machine learning to build a classifier to anticipate the class of each application sequence based on a matching series of API embed prototype. In addition, the authors do not discuss the potential drawbacks of their approach. For example, incremental learning can be more computationally expensive than traditional machine learning techniques, and transfer learning can be less effective if the pre-trained model is not well-matched to the target task. Overall, the researcher provides a valuable contribution to the field of machine learning-based Android malware detection. However, further research is needed to validate the proposed techniques on a larger and more diverse dataset, and to compare them to other state-of-the-art methods.   
This research by Omer Faruk Turan Cavli and Sevil Sen [33] analyzes the classification of Android malware pairs using a hybrid analytic technique combined with machine learning. The study utilizes both static analysis (viewing code without execution) and dynamic analysis (observing runtime behavior) to obtain an exhaustive set of features. Features include network-related data and activity bigrams (sequences of two consecutive events) for gaining deeper insights into virus behavior. Different machine learning algorithms are employed to examine the retrieved data and categorize samples of malware into their families 96% of correctness in both datasets

This section deals with a few associated investigations regarding leveraging dynamic characteristics in identifying malware for Android. Afonso et al [32] established a methodology centered around dynamic analysis of applications for Android. The suggested remedy dynamically identifies malware utilizing Mobile API calls and system calls logs. The researchers utilized distinct data sets from Malgenome project [40] and Virusshare [36] with a total of 7520 applications. The suggested approach evaluated multiple methods of classification and attained an accuracy of 96.66%. Nevertheless, their technique managed to track the harmful conduct under multiple circumstances.

Static analysis is used to extract features from Android apps such as permissions, API calls, and system calls. The KNN model is then trained using these features. Once trained, the model can be used to classify new Android apps as benign or malicious. The paper assesses the proposed system's performance on a dataset of over 10,000 Android apps. The system achieves an accuracy of 93 percent, a precision of 95 percent, a recall of 90 percent, and an F1 score of 92 percent, according to the results. These results are comparable to, if not superior to, those of other cutting-edge malware detection systems. [2] is our targeted paper, Novel based- machine algorithm using co-existence feature, this is new approach about the malware detection. It shows about 98% of accuracy using co-existence feature in binary detection of malware moreover we can write the table of our literature

The literature review we are display by the table 1 which can easy to understand our research related

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **work** | **Year** | **Analysis** | **Features** | **Datasets** | **Algorithm** | **Accuracy** |
| [34] | 2020 | Static | Permission and Intents | Derbin[35], VirusShare[36],  Androzoo[37] | E.R Native, Rondom Forest, KNN and SVM | Drebin, 93%  VirusShare 94%  Androzoo 97.3%  Mixed 92.5% |
| [38] | 2020 | Static | API calls, intent, and permissions | Derbin[35], Contagio[39],  Malgenome[40] | Hamming, RNN, ANN, WANN, KMNN | Variation between 90% to 99% depends on features |
| [41] | 2021 | Static | Raw opcodes, permission and API calls | Drebin[35] | Discriminative Adversarial Network | 97.5 |
| [31] | 2023 | Hybrid | API calls and permissions | Google10,000 benign and malicious Android applications | CNN and RNN models | 99.5% accuracy, and model score 99.5% |
| [42] | 2021 | Dynamic | Methods calls and intercomponent communication ICC | Genome[43] Drebin[35], VirusShare[36], Androzoo[37] | DriodCat | 97% |
| [29] | 2022 | Static | API Calls | 126,000 apps Kaggle | SVM, and Random Forest, | 97.8% and an F1-score of 98.8%. |
| [2] | 2023 | Static | API calls, Permission, combination of API and Permission | Genome[43] Drebin[35], Malgenome[36], CIC\_Maldriod[43] | Machine learning algorithms | Melgenome 98%  Drebin 97% |

Table 2-2 Related with the literature major researches

## Brief Summary

In this chapter we study about the research work about malware detection This paper [18] proposes a novel method, called IMCL Net, for classifying malware using deep learning. IMCL Net tackles the challenge of balancing accuracy and efficiency in malware detection by converting malware samples into images and classifying them using a lightweight convolutional neural network. Traditional methods struggle to keep up with the increasing number and complexity of malware. The model incorporates Coordinate Attention, Depthwise Separable Convolution, and Global Context Embedding to achieve high accuracy with low resource consumption. The co-existence of certain permissions, APIs, and other features can be indicative of a malicious app. For example, an app that requests both the permission to access the user's contacts and the permission to send SMS messages is more likely to be a malicious app than an app that only requests one of these permissions [2]

Feature selection is the process of selecting the most relevant features from a dataset. In the context of Android malware detection, feature selection is important because it can help to improve the accuracy of the machine learning classifier.[2] This [44] presents a system for detecting malicious Android applications using machine learning and dynamic analysis. Here are the key points. This paper has only detect malware proposed as dynamically. Dynamically detection malware but 2020, to 2023 these years researchers used hybrid analysis of malware, its increase by day malware changes it behavior we need to detect malware

# Dynamic Co-existence And The FPM-IN Algorithm

## Introduction

In this chapter we are going to describe dynamic features of co-existence using Frequent Pattern Mining with Negative Items (FPM-IN). Before we start our new approach, we can discuss about the pervious research. In previous research that belongs to features based approach to detect the malware but in the detection always a room for the research to make it good finding about the malware. Previous method is not good to make novel based approach however novel based detection still investigation in the process. Traditional frequent pattern mining (FP-Growth) focuses on identifying itemsets that appear frequently within a dataset. However, incorporating negative items presents challenges and requires modifications to the standard techniques. in this algorithm we focused on negative association rules mining (NARM). First we discuss these terms separately in our proposed model. In this chapter we are explaining our work, start with problem statement and than proposed model FPM-IN algorithms.

This chapter about the tool techniques, what we are using model, language, programming APIs and user interfaces we need to explore for malware detection. We are using Dynamic analysis and implement into the real-world for malware detection. Our limitation is we are only applying these work into permissions and API calls

## Problem Statement

The previous research depended on static features, dynamic analysis, and a dynamic approach the features not correctly defined even the static 150 features are less to investigate. The previous research does not mention about the deep learning if applied. No limitations describe by the author means it can be works on cloud, the answer is no. the other problems in this previous research we have identified multiple problems, we are going to mention every problem which authors has done below

1）Lacks in-depth analysis and validation of assumptions about the co-existence features

2）The author of malware does not define all frequent elements they were using

3）Limitation has not been described and computational cost of feature generation

4）Only machine learning analysis about the co-existence features not deep learning.

5）Categorically analysis of features

6）Not using hyperparameters.

7）They focus on static datasets instead of dynamic features that will enhance the accuracy

In the Current Android malware detection approaches, only those employing the FP-growth algorithm within Frequent Pattern Mining (FPM), generally concentrate on identifying frequent itemsets. These represent combinations of features that frequent item within a dataset. However, this FP-growth approach has limitations. By the analyzing frequent patterns, crucial information may be ignored frequent itemsets. Anti-frequent itemsets are represent sets of features that rarely co-exist, potentially signifying abnormal or malicious behavior

## Dynamic Co-existence And FPM-IN Algorithm Proposed Model

Our research proposed model based on two models one model is dynamic co-existence and second features-based detection malware. figure 3-1 shows our research working diagram. Our research purpose to detect the malware with high accuracy and dynamic feature-based malware detection. We are going to explain our research approach that utilizes dynamic co-existence features the feature-based malware detection using malware 1 and benign app 0. Our research figure 3-2, its explain in next section 3.3.1.

In figure 3-1-chapter 1 section 1.5 these introductions were defined there and these are our two innovation of malware detection to prove dynamic co-existence of features.

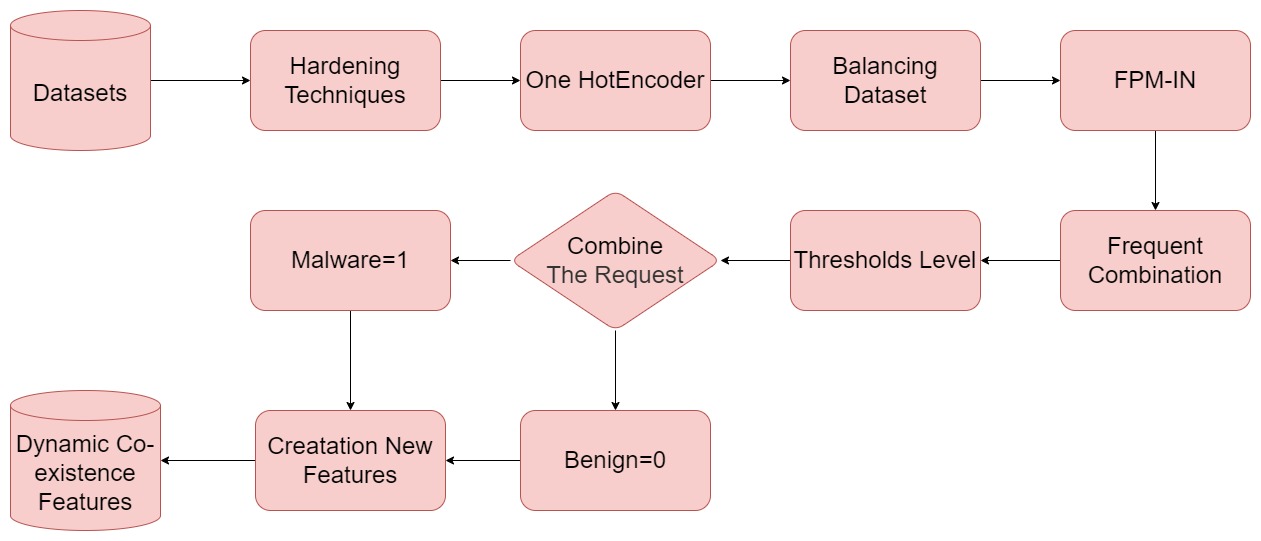


Figure 3-1 This is research working model of malware detection of our proposal model

### Frequent Pattern Mining with Negative Items (FPM-IN) Algorithm

The introduction of this algorithm, Traditional frequent pattern mining (FPM) focuses on identifying itemsets that appear frequently within a dataset. However, incorporating negative items presents challenges and requires modifications to the standard techniques. can uncover patterns where certain sets of items frequently appear together while specific items do not appear with them. This can be particularly useful in anomaly detection were identifying abnormal behavior often involves looking for deviations from expected patterns by considering negative items, FPM-IN can potentially improve the discrimination power of the mined patterns, making them more informative and potentially leading to more accurate models when used for tasks like fraud detection or malware detection

### Challenges with Negative Items

In this challenge, we are going to describe high support for negative items. in the algorithm of FPM-IN, as we can apply the FP-growth algorithm to get the frequent items positive. while we missed the negative items. to find the negative items of frequent items. We have two approaches and one approach of high support items to assume. The second approach is to provide support with thresholds. One approach is the highly supported algorithm of the native association of rule of mining. A high threshold might miss genuine patterns with slightly lower support but potentially higher significance. A low threshold might lead to an overwhelming number of potentially irrelevant patterns due to the high support of many negative itemsets. How to solve these challenges to make a good algorithm for malware detection Let’s have the address to challenge.

### Approaches for Addressing the Challenges

In this thesis, we are facing the above challenges. We are finding dynamic co-existence with a level-based approach; we are already using the existing approach. We find frequent items by varying the threshold. while, if we use a highly negative approach, we have huge datasets. We are using a negative symbol for a negative item, which will generate two types of frequent items. These depend on the threshold for negative items, which becomes complex. After these addressing the issues we need to solve using Negation Association Rule of Mining (NARM). Pseudo-Items commonly used in this approach. To introduce these approach the original dataset is then transformed to include both regular items and their pseudo-items in each transaction. In the context of NARM, these itemsets might include regular items alongside negative pseudo-items like (A, ~A). While the approach identifies itemsets involving pseudo-items, it doesn't directly capture the strength or significance of the negative relationships between the actual items.

In this thesis we are using this algorithm in both approaches because we already mentioned we are using dynamic co-existence of malware detection and we also worked from level 1 to level 5 behalf of thresholds. In the experiment section its defined here is just introduction of the terminology and figures how to deal these issues.

### Negative Association Rule Mining (NARM)

This approach called Pseudo-Items representing the negation of existing items. These pseudo-items are then used within the traditional FPM framework to discover negative association rules. However, the high support of negative items can still lead to misleading results. In dynamic co-existence features malware detection. This is a novel based approach while we used different level of thresholds it will affect the negative frequent items but when we use the less thresholds it can be highly negative frequent item.

### Frequent Pattern In FPM-IN

In the frequent pattern when will use FPM-IN algorithm. It will say about the specific set which will excluded but what we did we executed the loop towards from the datasets all the features. We convert all datasets into NARM than we used this FPM-IN algorithm. Another side we were using thresholds, while using thresholds when deceases the values the negative items are highly supports for negative. These itemsets where highly rated about the negative items and positive with the different level of thresholds. After the settings of name and save into the new files as per level 1 to 5, it means if a file API calls, it will dissociate into 5 level. And each level has detection malware of dynamic co-existence into 6 algorithms.

### New Datasets Generated By The Dynamic Co-existence Features

Our methodology is going to reveal about dynamic co-existence of features that we need some libraries in Python and make an environment in Python where all the libraries are installed next. Download the main datasets, Drebin or Malgenome. Check if these files are suitable or not; if not, take the data into pieces and merge the data. After the data, use the pre-processing technique of one hotencode operation: if the value of the cell is greater than 1, use 1; if the data is less than 0, or in double format, consider 0, in the representation of malware 1, and 0 is a benign app. Next step. Balance the data and apply the FPM-IN algorithm within a loop. A loop has certain values. [50, 25, 10, 5, 1]—these are the thresholds of our method. Then request that each variable be checked and then dumped into a csv file. But make sure one variable is a dependent variable. We have malware classes of variables to classify malware and benign apps

## Experimental of Dynamic Co-existence Result and Analysis

We are going to discuss about experiment first, we took the datasets from Malgenome, Drebin and only two datasets about API and permissions of datasets these are very popular now a days in research of malware In this experiment about dynamic co-existence in a dataset of permissions and call APIs for Android, Our main objective is to identify frequent elements in these datasets. For the initial step, we install Python and its libraries, like Numpy, Pandas, and Scikit-Learn, and for the pattern analysis, we use Mxltend These are useful interfaces for our research work.

Next, we load the CSV file, which contains data about permissions or API calls. These files have a specific variable whose name is class. that label denotes an app category. Check the name to ensure the class variable exists or not. and also some files as permission contain class variables, so we name them by the programming and make them correct for our needs. The class variable employs one-hot encoding to transform the categorical into a numerical representation to make it easy for analysis by the machine and deep learning, then we fetch the data and remove the duplication for redundancy.

Now, we have reached the core of our aim: to generate frequent pattern mining using the transform the data from negatively charged like A, ~A, than applied FP-Growth algorithm, a powerful tool for discovering frequent patterns and negative frequent items pattern. It contains combinations of items within a dataset. In these cases, if using permission, a combination of permission and API, or only API datasets, it will focus on exactly two columns, We continuously changes from 50% to 1% of thresholds so that only one pattern appears. In this study, we consider 50% to 1% or more frequent elements to be mineable. The most interesting part of our experiment to varying threshold. It is depend upon how to we have fixed the values and remember this directly effect our research and finding the level of dynamic features of co-existence.

Our exploration continues to generate dynamic co-existence features. It loops through the frequently created patterns, and for each permission or API call within a pattern, it adds a new column to a dedicated dataframe. This new column holds the related information of permission or API call value from the original data and generates a binary representation of whether an app contains a specific information permission combination.

To better understand insights, co-existence is categorized into five levels. It defines the five thresholds that determine varying levels of co-existence. For each level threshold, the code meticulously examines the average value for each co-existence feature. If the average value surpasses the threshold, it shows that the individual permission or API call combination is frequently associated, and the corresponding feature name is added to a list to show that level.

Finally, our experimental model, executed meticulously, saves the data for further analysis. loop iterates through each level, showing the list of co-existence features. For each level, it creates a new dataframe containing only those features deemed significant based on the chosen threshold, along with the original class label for context. These dataframes were saved as separate CSV files, each with a descriptive filename indicating the level and the FPM-IN algorithm used, providing a clear organization for further exploration and potential use in understanding app behavior or security implications

### Drebin Dynamic Co-existence Features Results

In this section we are collecting the result of Drebin dataset. These datasets most famous in the malware detection world. Are basically try each model and get the result and analysis about the result.

#### Drebin, API-Permission Combination Dynamic Co-existence Features Result

In this section, we took a dataset file API and Permission together interact about the malware. we need to check this dataset. We are applying our six model with each datasets. The threshold of the dynamic features.

##### Decision tree

In the Table3-1 the model we used the decision classifier model to detect malware using dynamic co-existence features. The accuracy, precision, recall, and F1 score of the model increase as the level increases as per thresholds. this model result shows the ability of malware detection increase as per level, and also shows it can be solved complex patterns of malware, overfitting training data is also improved. The accuracy of the model is highest at level 4, with a score of 97.51%. The precision of the model is highest at level 4, with a score of 97.59%. The recall of the model is highest at level 4, with a score of 96.09%. The F1 score of the model is highest at level 4, with a score of 96.59%. The cross-validation of the model is highest at level 4, with a score of 93.83%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Precision | Recall | Cross-validation | F1-score |
| Level 1 | 50% | 93.11 | 94.46 | 91.45 | 91.37 | 93.72 |
| Level 2 | 25% | 96.48 | 97.57 | 95.27 | 93.54 | 96.63 |
| Level 3 | 10% | 96.17 | 97.29 | 94.95 | 93.72 | 96.08 |
| Level 4 | 5% | 97.51 | 97.59 | 96.09 | 93.83 | 96.59 |
| Level 5 | 1% | 96.08 | 96.59 | 95.45 | 93.81 | 96.53 |

Table3-1 Decision tree model result of dynamic co-existence features using API-permissions

##### Random Forest

In this Table3-2 work explores the efficacy of a random classifier model incorporating dynamic co-existence features for malware detection. The model's performance across five distinct levels of complexity is evaluated using various metrics: accuracy, precision, recall, F1 score, and cross-validation F1 score. The analysis reveals a promising trend, with most metrics exhibiting improvement as the model level increases, suggesting enhanced differentiation between malware and benign samples. Notably, level 4 achieves the highest scores for accuracy (98.42%), precision (99.44%), recall (97.37%), and F1 score (97.96%), showcasing the potential of incorporating dynamic co-existence features. However, the cross-validation F1 score, though peaking at level 4 (97.96%), demonstrates a smaller advantage, indicating a potential trade-off between complexity and generalizability. While higher complexity enhances performance on training data, it might lead to overfitting, hindering real-world effectiveness. Therefore, while the current results are encouraging, further investigation is crucial to optimize the model's complexity for practical malware detection in my application.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Precision | Recall | Cross-validation | F1-score |
| Level 1 | 50% | 94.15 | 95.5 | 92.54 | 92.24 | 94.06 |
| Level 2 | 25% | 98.06 | 99 | 97.09 | 95.08 | 98.28 |
| Level 3 | 10% | 98.15 | 99 | 97.18 | 95.03 | 98.01 |
| Level 4 | 5% | 98.42 | 99.44 | 97.37 | 94.55 | 97.96 |
| Level 5 | 1% | 98.19 | 99.25 | 97.09 | 94.75 | 97.96 |

Table 3-2 Random Forest model result of dynamic co-existence features using API-permissions

##### Logistic regression

This model we examine in the Table3-3 the effectiveness of a logistic regression model utilizing dynamic co-existence features for malware detection. The model's performance across five levels of complexity is assessed using various metrics: threshold, accuracy, precision, recall, F1 score, and cross-validation. The analysis reveals a promising trend with most metrics improving as the model's complexity increases, suggesting enhanced capability to distinguish malware from benign samples. Notably, levels 4 and 5 achieve peaks in accuracy and precision (90.95% and 90.82%, respectively), while level 3 takes the lead in recall and F1 score (91.18% and 89.47%, respectively). However, the cross-validation F1 score paints a different picture. While level 5 remains strong, the highest score of 90.59% is achieved at level 2, indicating a potential trade-off between complexity and generalizability. Increased complexity leads to better performance on training data but might introduce overfitting, hindering real-world effectiveness.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Precision | Recall | Cross-validation | F1-score |
| Level 1 | 50% | 86.04 | 84.83 | 87.45 | 87.3 | 86.13 |
| Level 2 | 25% | 90.5 | 89.86 | 91.09 | 90.45 | 90.3 |
| Level 3 | 10% | 90.95 | 90.6 | 91.18 | 90.15 | 89.47 |
| Level 4 | 5% | 90.09 | 89.14 | 91.09 | 90.18 | 90.46 |
| Level 5 | 1% | 90.95 | 90.82 | 90.9 | 90.21 | 90.59 |

Table3-3 Logistic regression model result of dynamic co-existence features using API-permissions

##### CNN

Table3-4 shows the performance of a CNN model on dynamic features of co-existence. The accuracy of the model increases from 91.8% at level 1 to 95.63% at level 4. The precision also increases from 96.13% to 98.25%, and the F1 score increases from 91.74% to 98.05%. However, the Hamming loss decreases from 0.081 to 0.043. This model shows the ability to detect malware and learn the most effective features. However, it depends on the task that we have performed on it. these APIs and permissions about this result.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Average Precision | F1-score | Hamming Loss |
| Level 1 | 50% | 91.8 | 96.13 | 91.74 | 0.081 |
| Level 2 | 25% | 95.39 | 98.21 | 95.39 | 0.046 |
| Level 3 | 10% | 95.42 | 98.4 | 95.41 | 0.045 |
| Level 4 | 5% | 95.63 | 98.25 | 95.61 | 0.043 |
| Level 5 | 1% | 95.02 | 98.13 | 98.05 | 0.049 |

Table3-4 CNN model result of dynamic co-existence features using API-permissions

##### RNN

In Table3–5, the results of the RNN model show the accuracy, average precision, F1 weight, and Hamming loss of the RNN model at five different levels of dynamic co-existence features. As we observe, the RNN model performs well on API permissions, with an accuracy of over 91% at all levels. This means that the model is able to correctly classify over 91% of the malware and benign samples in the dataset. The average precision is also high, ranging from 96% to 98%, indicating that the model is good at identifying true-positive cases of malware. The F1 weight, which takes into account both precision and recall, is also above 91% for all levels, suggesting the model has a good balance between these two metrics. Finally, the Hamming loss is relatively low, ranging from 0.051 to 0.058, which means that the model is making few mistakes in its predictions.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Average Precision | F1-score | Hamming Loss |
| Level 1 | 50% | 91.29 | 96 | 91.27 | 0.087 |
| Level 2 | 25% | 94.86 | 98.21 | 94.84 | 0.051 |
| Level 3 | 10% | 94.62 | 97.41 | 94.6 | 0.053 |
| Level 4 | 5% | 95.01 | 98.17 | 95.01 | 0.051 |
| Level 5 | 1% | 94.5 | 97.47 | 94.08 | 0.058 |

Table3-5 RNN model result of dynamic co-existence features using API-permissions

##### ANN

In the result of the ANN model using dynamic features of co-existence, Table3-6 displayed the accuracy, average precision, F1 weight, and Hamming loss of the ANN model at different levels with thresholds. These metrics are all used to measure the performance of a deep learning model. The accuracy of the model ranges from 91% to 95.29%, the average precision is 96% to 98%, and the F1 weight is 91% to 95%. the highest accuracy, precision, and F1 score in level 2. F1 weight is a harmonic mean of precision and recall. Hamming loss is a measure of the difference between the predicted labels and the true labels.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Average Precision | F1-score | Hamming Loss |
| Level 1 | 50% | 91.85 | 96.05 | 91.79 | 0.081 |
| Level 2 | 25% | 95.29 | 98.2 | 95.27 | 0.047 |
| Level 3 | 10% | 94.83 | 98.03 | 94.82 | 0.051 |
| Level 4 | 5% | 95.01 | 97.17 | 92.5 | 0.074 |
| Level 5 | 1% | 94.82 | 98.03 | 94.82 | 0.058 |

Table3-6 ANN model result of dynamic co-existence features using API-permissions

#### Drebin, Only API Calls Dynamic Co-existence Features Result

In this paragraph we are showing the value of Call API dynamic co-existence features malware detection results and analysis

##### Decision Tree

In the Table 3-7 the model we used the decision classifier model to detect malware using dynamic co-existence features. The accuracy, precision, recall, and F1 score of the model increase as the level increases as per thresholds. this model result shows the ability of malware detection increase as per level, and also shows it can be solved complex patterns of malware, overfitting training data is also improved. The accuracy of the model is highest at level3, with a score of 95.27%. The precision of the model is highest at level 3, with a score of 98.44%. The recall of the model is highest at level3 and level4, with a score of 91.9%. The F1 score of the model is highest at level3, with a score of 94.59%. The cross-validation of the model is highest at level3, with a score of 92.52%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Precision | Recall | Cross-validation | F1-score |
| Level 1 | 50% | 88.56 | 93.16 | 83 | 88.13 | 84.93 |
| Level 2 | 25% | 94.59 | 96.93 | 92 | 92.05 | 94.91 |
| Level 3 | 10% | 95.27 | 98.44 | 91.9 | 92.52 | 94.59 |
| Level 4 | 5% | 94.82 | 97.49 | 91.9 | 92.1 | 94.53 |
| Level 5 | 1% | 94.19 | 96.45 | 91.63 | 92.19 | 94.7 |

Table 3-7 Decision tree model result of dynamic co-existence features

##### Random Forest

In the Table 3-8 the model we used the Random classifier model to detect malware using dynamic co-existence features. The accuracy, precision, recall, and F1 score of the model increase as the level increases as per thresholds. this model result demonstrates a smaller advantage, indicating a potential trade-off between complexity and generalizability. While higher complexity enhances performance on training data, it might lead to overfitting, hindering real-world effectiveness. The accuracy of the model is highest at level3, with a score of 95.81%. The precision of the model is highest at level 3, with a score of 98.74%. The recall of the model is highest at level4, with a score of 93.18%. The F1 score of the model is highest at level3, with a score of 95.39%. The cross-validation of the model is highest at level4, with a score of 93.17%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Precision | Recall | Cross-validation | F1-score |
| Level 1 | 50% | 89.01 | 93.31 | 83.81 | 88.41 | 89.69 |
| Level 2 | 25% | 95.49 | 97.7 | 93.09 | 92.97 | 95.17 |
| Level 3 | 10% | 95.81 | 98.74 | 92.72 | 93.03 | 95.39 |
| Level 4 | 5% | 95.67 | 98 | 93.18 | 93.17 | 95.23 |
| Level 5 | 1% | 95.85 | 98.55 | 93 | 92.87 | 95.28 |

Table 3-8 Random forest model result of dynamic co-existence features

##### Logistic regression

In the Table 3-9 the model we used the Logistic regression model to detect malware using dynamic co-existence features. The accuracy, precision, recall, and F1 score of the model increase as the level increases as per thresholds. this model result demonstrates the analysis reveals a promising trend with most metrics improving as the model's complexity increases, suggesting enhanced capability to distinguish malware from benign samples. The accuracy of the model is highest at level3, with a score of 86.22%. The precision of the model is highest at level 3, with a score of 82.70%. The recall of the model is highest at level4, with a score of 87.45%. The F1 score of the model is highest at level4, with a score of 85.17%. The cross-validation of the model is highest at level3, with a score of 85.60%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Precision | Recall | Cross-validation | F1-score |
| Level 1 | 50% | 83.66 | 82.12 | 85.63 | 81.77 | 80.94 |
| Level 2 | 25% | 85.32 | 81.26 | 87.45 | 85.57 | 85.01 |
| Level 3 | 10% | 86.22 | 82.7 | 87.27 | 85.60 | 85.08 |
| Level 4 | 5% | 85.86 | 82.05 | 87.45 | 85.46 | 85.17 |
| Level 5 | 1% | 86.18 | 82.63 | 87.27 | 85.52 | 84.65 |

Table 3-9 Logistic regression model result of dynamic co-existence features

##### CNN

Table3-10 shows the performance of a CNN model on dynamic features of co-existence. The accuracy of the model increases from 88.77% at level 1 to 93.61% at level 2. The level 2 is the highest accuracy in this model. The precision also increases from 93.88% to 97.12%, and the F1 score increases from 88.7% to 94.5%. However, the Hamming loss decreases from 1.12 to 0.063. This model shows the ability to detect malware and learn the most effective features. However, it depends on the task that we have performed on it. these APIs about this result.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Average Precision | F1-score | Hamming Loss |
| Level 1 | 50% | 88.77 | 93.88 | 88.7 | 1.122 |
| Level 2 | 25% | 93.61 | 97.1 | 93.57 | 0.063 |
| Level 3 | 10% | 93.5 | 97.12 | 93.47 | 0.064 |
| Level 4 | 5% | 93.42 | 97.07 | 93.33 | 0.065 |
| Level 5 | 1% | 93.53 | 97 | 94.5 | 0.064 |

Table 3-10 CNN model result of dynamic co-existence features

##### RNN

In Table 3–11, the results of the RNN model show the accuracy, average precision, F1 weight, and Hamming loss of the RNN model at five different levels of dynamic co-existence features. As we observe, the RNN model performs well on API permissions, with an accuracy of over 89% at all levels. This means that the model is able to correctly classify over 92.7% of the malware and benign samples in the dataset. The average precision is also high, varing from 93% to 96.49%, indicating that the model is good at identifying true-positive cases of malware. The F1 weight, which takes into account both precision and recall, is also above 89% for all levels, suggesting the model has a good balance between these two metrics. Finally, the Hamming loss is relatively low, ranging from 1.061 to 0.072, which means that the model is making few mistakes in its predictions

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Average Precision | F1-score | Hamming Loss |
| Level 1 | 50% | 89.3 | 93.92 | 89 | 1.06 |
| Level 2 | 25% | 92.44 | 96.49 | 92.43 | 0.75 |
| Level 3 | 10% | 92.7 | 96.47 | 92.63 | 0.72 |
| Level 4 | 5% | 92.65 | 96.39 | 92.62 | 0.73 |
| Level 5 | 1% | 91.24 | 95.82 | 91.18 | 0.87 |

Table 3-11 RNN model result of dynamic co-existence features

##### ANN

In the result of the ANN model using dynamic features of co-existence, Table3-12 displayed the accuracy, average precision, F1 weight, and Hamming loss of the ANN model at different levels with thresholds. These metrics are all used to measure the performance of a deep learning model. The accuracy of the model ranges from 89% to 93.42%, the average precision is 89% to 93.5%, and the F1 weight is 89% to 93.29%. the highest accuracy, precision, and F1 score in level 5. F1 weight is a harmonic mean of precision and recall. Hamming loss is a measure of the difference between the predicted labels and the true labels.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Average Precision | F1-score | Hamming Loss |
| Level 1 | 50% | 89.43 | 94.17 | 89.3 | 0.15 |
| Level 2 | 25% | 93.05 | 96.64 | 93 | 0.69 |
| Level 3 | 10% | 93.1 | 96.8 | 93.5 | 0.67 |
| Level 4 | 5% | 93.32 | 96.93 | 93.29 | 0.66 |
| Level 5 | 1% | 93.42 | 97.13 | 93.4 | 0.65 |

Table 3-12 ANN model result of dynamic co-existence features

#### Drebin, Only Permissions Dynamic Co-existence Features Result

##### Decision tree

In Table3-13 we used the decision classifier model to detect malware using dynamic co-existence features. The accuracy, precision, recall, and F1 score of the model increase as the level increases as per thresholds. This model result shows the ability of malware detection increase as per level, and also shows it can be solved complex patterns of malware, overfitting training data is also improved. The accuracy of the model is highest at level 3, with a score of 75.65%. The precision of the model is highest at level 4, with a score of 81.19%. The recall of the model is highest at level 2, with a score of 73.21%. The F1 score of the model is highest at level 3, with a score of 77.06%. The cross-validation of the model is highest at level 3, with a score of 76.14%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Precision | Recall | Cross-validation | F1-score |
| Level 1 | 50% | 71.96 | 52.22 | 87.07 | 71.74 | 65.29 |
| Level 2 | 25% | 75.15 | 80 | 73.21 | 76.2 | 76.49 |
| Level 3 | 10% | 75.65 | 81.5 | 73.4 | 75.7 | 77.06 |
| Level 4 | 5% | 74.8 | 81.19 | 72.34 | 76.1 | 76.55 |
| Level 5 | 1% | 74 | 78.43 | 72.52 | 76.14 | 73.37 |

Table 3-13 Decision tree model result of dynamic co-existence features

##### Random Forest

In Table3-14 we used the random forest model to detect malware using dynamic co-existence features. The accuracy, precision, recall, and F1 score of the model increase as the level increases as per thresholds. This model result shows the ability of malware detection increase as per level, and also shows it can be solved complex patterns of malware, overfitting training data is also improved. The accuracy of the model is highest at level 2, with a score of 75.92%. The precision of the model is highest at level 4, with a score of 82.79%. The recall of the model is highest at level 1, with a score of 86.75%. The F1 score of the model is highest at level 4, with a score of 77.571%. The cross-validation of the model is highest at level 4, with a score of 72.33%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Precision | Recall | Cross-validation | F1-score |
| Level 1 | 50% | 71.24 | 50.8 | 86.75 | 63.08 | 64.08 |
| Level 2 | 25% | 75.92 | 81.9 | 73.46 | 64.32 | 77.45 |
| Level 3 | 10% | 75.6 | 81.37 | 73.27 | 72.08 | 77.11 |
| Level 4 | 5% | 75.83 | 82.79 | 72.97 | 72.33 | 77.57 |
| Level 5 | 1% | 75.02 | 80.21 | 72.99 | 72.18 | 76.43 |

Table 3-14 Random Forest model result of dynamic co-existence features

##### Logistic regression

In Table3-15 we used the logistic regression model to detect malware using dynamic co-existence features. The accuracy, precision, recall, and F1 score of the model increase as the level increases as per thresholds. This model result shows the ability of malware detection increase as per level, and also shows it can be solved complex patterns of malware, overfitting training data is also improved. The accuracy of the model is highest at level 2, with a score of 74.84%. The precision of the model is highest at level 4, with a score of 81.19%. The recall of the model is highest at level 2, with a score of 73.21%. The F1 score of the model is highest at level 3, with a score of 77.06%. The cross-validation of the model is highest at level 2, with a score of 76.2%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Precision | Recall | Cross-validation | F1-score |
| Level 1 | 50% | 71.29 | 50.22 | 87.07 | 71.74 | 63.95 |
| Level 2 | 25% | 74.84 | 80 | 73.21 | 76.2 | 76.49 |
| Level 3 | 10% | 74.57 | 81.5 | 73.4 | 75.7 | 77.06 |
| Level 4 | 5% | 74.8 | 81.19 | 72.34 | 76.1 | 76.55 |
| Level 5 | 1% | 74 | 78.43 | 72.52 | 76.14 | 73.37 |

Table 3-15 Logistic regression model result of dynamic co-existence features

##### CNN

Table3-16 shows the performance of a CNN model on dynamic features of co-existence. The accuracy of the model increases from 71.29% at level 1 to 77.64% at level 3. The precision also increases from 80.15% to 85.13%, and the F1 score increases from 70.72% to 77.41%. However, the Hamming loss decreases from 0.29 to 0.22. This model shows the ability to detect malware and learn the most effective features. However, it depends on the task that we have performed on it. these APIs and permissions about this result.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Average Precision | F1-score | Hamming Loss |
| Level 1 | 50% | 71.29 | 80.15 | 70.72 | 0.29 |
| Level 2 | 25% | 74.84 | 84.15 | 74.25 | 0.26 |
| Level 3 | 10% | 76.54 | 84.19 | 76.70 | 0.24 |
| Level 4 | 5% | 77.64 | 85.13 | 77.41 | 0.22 |
| Level 5 | 1% | 77.55 | 84.82 | 77.39 | 0.22 |

Table 3-16 CNN model result of dynamic co-existence features

##### RNN

In Table3–17, the results of the RNN model show the accuracy, average precision, F1 weight, and Hamming loss of the RNN model at five different levels of dynamic co-existence features. As we observe, the RNN model performs well on API permissions, with an accuracy of over 77.78% at all levels. This means that the model is able to correctly classify over 77.78% of the malware and benign samples in the dataset. The average precision is also high, ranging from 80% to 83.40%, indicating that the model is good at identifying true-positive cases of malware. The F1 weight, which takes into account both precision and recall, is also above 77.59% for all levels, suggesting the model has a good balance between these two metrics. Finally, the Hamming loss is relatively low, ranging from 0.22 to 0.29, which means that the model is making few mistakes in its predictions.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Average Precision | F1-score | Hamming Loss |
| Level 1 | 50% | 70.21 | 80.5 | 71.23 | 0.29 |
| Level 2 | 25% | 77.64 | 83.40 | 74.25 | 0.23 |
| Level 3 | 10% | 77.76 | 81.19 | 75.78 | 0.24 |
| Level 4 | 5% | 77.51 | 81.43 | 76.71 | 0.22 |
| Level 5 | 1% | 77.78 | 81.2 | 77.59 | 0.22 |

Table 3-17 RNN model result of dynamic co-existence features

##### ANN

In the result of the ANN model using dynamic features of co-existence, Table3-18 displayed the accuracy, average precision, F1 weight, and Hamming loss of the ANN model at different levels with thresholds. These metrics are all used to measure the performance of a deep learning model. The accuracy of the model ranges from 70.88% to 77.78%, the average precision is 79.15% to 82.42%, and the F1 weight is 71.31% to 77.80%. the highest accuracy, precision, and F1 score in level 2, 4 and 5 respectively. F1 weight is a harmonic means of precision and recall. Hamming loss is a measure of the difference between the predicted labels and the true labels

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Average Precision | F1-score | Hamming Loss |
| Level 1 | 50% | 70.88 | 79.15 | 71.31 | 0.30 |
| Level 2 | 25% | 77.78 | 82.40 | 75.45 | 0.24 |
| Level 3 | 10% | 77.46 | 82.13 | 75.88 | 0.24 |
| Level 4 | 5% | 77.77 | 83.43 | 76.88 | 0.22 |
| Level 5 | 1% | 77.74 | 83.21 | 77.80 | 0.22 |

Table 3-18 ANN model result of dynamic co-existence features

### Malgenome Dynamic Co-existence Features Results

In this section we are collecting the result of Malgenome Dataset. These datasets most famous in the malware detection world. Are basically try each model and get the result and analysis about the result.

#### Malgenome, API-Permission Combination Dynamic Co-existence Features Result

In this section, we took a dataset file API and Permission together interact about the malware. we need to check this dataset. We are applying our six model with each datasets. The threshold of the dynamic features.

##### Decision tree

In Table3-19 we used the decision classifier model to detect malware using dynamic co-existence features. The accuracy, precision, recall, and F1 score of the model increase as the level increases as per thresholds. This model result shows the ability of malware detection increase as per level and shows it can be solved complex patterns of malware, overfitting training data is also improved. The accuracy of the model is highest at level 4, with a score of 98.05%. The precision of the model is highest at level 4, with a score of 99.77%. The recall of the model is highest at level 4, with a score of 99.79%. The F1 score of the model is highest at level 2, with a score of 93.63%. The cross-validation of the model is highest at level 4, with a score of 93.83%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Precision | Recall | Cross-validation | F1-score |
| Level 1 | 50% | 94.64 | 97.46 | 97.85 | 91.71 | 92.12 |
| Level 2 | 25% | 96.81 | 99.15 | 99.35 | 93.54 | 93.63 |
| Level 3 | 10% | 97.8 | 99.55 | 99.45 | 93.72 | 92.62 |
| Level 4 | 5% | 98.05 | 99.77 | 99.79 | 93.83 | 93.59 |
| Level 5 | 1% | 97.59 | 99.43 | 99.63 | 93.81 | 93.53 |

Table 3-19 Decision tree model result of dynamic co-existence features

##### Random Forest

In Table3-20 we used the random forest classifier model to detect malware using dynamic co-existence features. The accuracy, precision, recall, and F1 score of the model increase as the level increases as per thresholds. This model result shows the ability of malware detection increase as per level and shows it can be solved complex patterns of malware, overfitting training data is also improved. The accuracy of the model is highest at level 2, with a score of 99.80%. The precision of the model is highest at level 2, 3 and 5, with a score of 100%. The recall of the model is highest at level 2, 3 and 5, with a score of 100%. The F1 score of the model is highest at level 2, with a score of 93.631%. The cross-validation of the model is highest at level 4, with a score of 93.83%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Precision | Recall | Cross-validation | F1-score |
| Level 1 | 50% | 98.01 | 98.46 | 98.85 | 91.71 | 92.12 |
| Level 2 | 25% | 99.80 | 100 | 100 | 93.54 | 93.63 |
| Level 3 | 10% | 99.71 | 100 | 100 | 93.72 | 92.62 |
| Level 4 | 5% | 99.65 | 99.87 | 99.89 | 93.83 | 93.59 |
| Level 5 | 1% | 99.65 | 100 | 100 | 93.81 | 93.53 |

Table 3-20 Random forest model result of dynamic co-existence features

##### Logistic regression

In Table3-21 we used the logistic regression model to detect malware using dynamic co-existence features. The accuracy, precision, recall, and F1 score of the model increase as the level increases as per thresholds. This model result shows the ability of malware detection increase as per level and shows it can be solved complex patterns of malware, overfitting training data is also improved. The accuracy of the model is highest at level 5, with a score of 95.85%. The precision of the model is highest at level 3, with a score of 98.44%. The recall of the model is highest at level 3, with a score of 98.74%. The F1 score of the model is highest at level 4, with a score of 96.83%. The cross-validation of the model is highest at level 4, with a score of 96.83%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Precision | Recall | Cross-validation | F1-score |
| Level 1 | 50% | 92.85 | 90.31 | 94.71 | 91.31 | 91.30 |
| Level 2 | 25% | 95.29 | 97.7 | 94.09 | 92.97 | 94.19 |
| Level 3 | 10% | 95.41 | 98.74 | 95.62 | 93.03 | 96.18 |
| Level 4 | 5% | 94.67 | 98 | 96.68 | 93.17 | 96.83 |
| Level 5 | 1% | 95.85 | 98.55 | 96.5 | 92.87 | 96.68 |

Table 3-21 Logistic regression model result of dynamic co-existence features

##### CNN

Table3-22 shows the performance of a CNN model on dynamic features of co-existence. The accuracy of the model increases from 92.88% at level 1 to 94.44% at level 5. The precision also increases from 9415% to 98.43%, and the F1 score increases from 92.31% to 97.80%. However, the Hamming loss decreases from 0.071 to 0.045. This model shows the ability to detect malware and learn the most effective features. However, it depends on the task that we have performed on it. these APIs and permissions about this result.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Average Precision | F1-score | Hamming Loss |
| Level 1 | 50% | 92.88 | 94.15 | 92.31 | 0.071 |
| Level 2 | 25% | 94.68 | 97.40 | 95.45 | 0.046 |
| Level 3 | 10% | 95.16 | 97.13 | 94.88 | 0.056 |
| Level 4 | 5% | 94.77 | 98.43 | 96.88 | 0.046 |
| Level 5 | 1% | 95.44 | 98.21 | 97.80 | 0.045 |

Table 3-22 CNN model result of dynamic co-existence features

##### RNN

In Table3–23, the results of the RNN model show the accuracy, average precision, F1 weight, and Hamming loss of the RNN model at five different levels of dynamic co-existence features. As we observe, the RNN model performs well on API permissions, with an accuracy of over 92.64% at all levels. This means that the model is able to correctly classify over 92.64% of the malware and benign samples in the dataset. The average precision is also high, ranging from 91% to 97.53%, indicating that the model is good at identifying true-positive cases of malware. The F1 weight, which takes into account both precision and recall, is also above 96.88% for all levels, suggesting the model has a good balance between these two metrics. Finally, the Hamming loss is relatively low, ranging from 0.075 to 0.091, which means that the model is making few mistakes in its predictions.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Average Precision | F1-score | Hamming Loss |
| Level 1 | 50% | 91.38 | 91.15 | 92.21 | 0.091 |
| Level 2 | 25% | 92.48 | 96.30 | 96.45 | 0.077 |
| Level 3 | 10% | 92.56 | 96.33 | 96.18 | 0.076 |
| Level 4 | 5% | 92.57 | 97.53 | 96.88 | 0.075 |
| Level 5 | 1% | 92.64 | 97.21 | 96.50 | 0.075 |

Table 3-23 RNN model result of dynamic co-existence features

##### ANN

In the result of the ANN model using dynamic features of co-existence, Table3-24 displayed the accuracy, average precision, F1 weight, and Hamming loss of the ANN model at different levels with thresholds. These metrics are all used to measure the performance of a deep learning model. The accuracy of the model ranges from 93% to 95.68%, the average precision is 96% to 98.51%, and the F1 weight is 93% to 95.26%. the highest accuracy, precision, and F1 score in level 2. F1 weight is a harmonic means of precision and recall. Hamming loss is a measure of the difference between the predicted labels and the true labels.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Average Precision | F1-score | Hamming Loss |
| Level 1 | 50% | 94.38 | 97.15 | 94.21 | 0.056 |
| Level 2 | 25% | 95.28 | 98.31 | 95.26 | 0.047 |
| Level 3 | 10% | 95.68 | 98.51 | 95.26 | 0.047 |
| Level 4 | 5% | 93.32 | 96.53 | 93.98 | 0.065 |
| Level 5 | 1% | 93.44 | 97.13 | 93.40 | 0.065 |

Table 3-24 ANN model result of dynamic co-existence features

#### Malgenome, API Calls Dynamic Co-existence Features Result

##### Decision tree

In Table3-25 we used the decision classifier model to detect malware using dynamic co-existence features. The accuracy, precision, recall, and F1 score of the model increase as the level increases as per thresholds. This model result shows the ability of malware detection increase as per level and shows it can be solved complex patterns of malware, overfitting training data is also improved. The accuracy of the model is highest at level 4, with a score of 99.2%. The precision of the model is highest at level 4, with a score of 99.29%. The recall of the model is highest at level 3, with a score of 98.95%. The F1 score of the model is highest at level 4, with a score of 99.391%. The cross-validation of the model is highest at level 4, with a score of 93.83%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Precision | Recall | Cross-validation | F1-score |
| Level 1 | 50% | 97.02 | 97.16 | 96.45 | 92.37 | 97.22 |
| Level 2 | 25% | 98.61 | 98.27 | 97.27 | 93.54 | 98.27 |
| Level 3 | 10% | 98.41 | 98.29 | 98.95 | 93.72 | 98.39 |
| Level 4 | 5% | 99.2 | 99.29 | 97.94 | 93.83 | 99.39 |
| Level 5 | 1% | 99 | 97.29 | 96.45 | 92.81 | 97.59 |

Table 3-25 Decision tree model result of dynamic co-existence features

##### Random Forest

In Table3-26 we used the random forest classifier model to detect malware using dynamic co-existence features. The accuracy, precision, recall, and F1 score of the model increase as the level increases as per thresholds. This model result shows the ability of malware detection increase as per level and shows it can be solved complex patterns of malware, overfitting training data is also improved. The accuracy of the model is highest at level 3 and level 5, with a score of 99.4%. The precision of the model is highest at level 2, with a score of 100%. The recall of the model is highest at level 4, with a score of 96.09%. The F1 score of the model is highest at level 2, with a score of 96.63%. The cross-validation of the model is highest at level 4, with a score of 93.83%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Precision | Recall | Cross-validation | F1-score |
| Level 1 | 50% | 96.82 | 97.46 | 91.45 | 91.37 | 93.72 |
| Level 2 | 25% | 99.2 | 100 | 95.27 | 93.54 | 96.63 |
| Level 3 | 10% | 99.4 | 99.79 | 94.95 | 93.72 | 96.08 |
| Level 4 | 5% | 99.2 | 99.59 | 96.09 | 93.83 | 96.59 |
| Level 5 | 1% | 99.4 | 99.40 | 95.45 | 93.81 | 96.53 |

Table 3-26 Random Forest model result of dynamic co-existence features

##### Logistic regression

In Table3-27 we used the Logistic regression model to detect malware using dynamic co-existence features. The accuracy, precision, recall, and F1 score of the model increase as the level increases as per thresholds. This model result shows the ability of malware detection increase as per level and shows it can be solved complex patterns of malware, overfitting training data is also improved. The accuracy of the model is highest at level 5, with a score of 94.64%. The precision of the model is highest at level 4 and level 5, with a score of 96.59%. The recall of the model is highest at level 4, with a score of 96.49%. The F1 score of the model is highest at level 2, with a score of 96.63%. The cross-validation of the model is highest at level 4, with a score of 93.83%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Precision | Recall | Cross-validation | F1-score |
| Level 1 | 50% | 88.88 | 90.24 | 87.45 | 91.37 | 93.72 |
| Level 2 | 25% | 94.04 | 96.57 | 96.27 | 93.54 | 96.63 |
| Level 3 | 10% | 93.84 | 96.29 | 96.35 | 93.72 | 96.08 |
| Level 4 | 5% | 93.65 | 96.59 | 96.49 | 93.83 | 96.59 |
| Level 5 | 1% | 94.64 | 96.59 | 96.45 | 93.81 | 96.53 |

Table 3-27 Logistic regression model result of dynamic co-existence features

##### CNN

Table3-28 shows the performance of a CNN model on dynamic features of co-existence. The accuracy of the model increases from 91.47% at level 1 to 95.36% at level 2 and 4. The precision also increases from 93.47% to 98.52%, and the F1 score increases from 94.47% to 98.19%. However, the Hamming loss decreases from 0.06 to 0.04. This model shows the ability to detect malware and learn the most effective features. However, it depends on the task that we have performed on it. these APIs and permissions about this result.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Average Precision | F1-score | Hamming Loss |
| Level 1 | 50% | 91.47 | 93.47 | 94.47 | 0.06 |
| Level 2 | 25% | 95.36 | 98.36 | 95.96 | 0.05 |
| Level 3 | 10% | 94.52 | 98.52 | 97.25 | 0.04 |
| Level 4 | 5% | 95.36 | 98.36 | 98.19 | 0.04 |
| Level 5 | 1% | 95.15 | 98.15 | 98.15 | 0.04 |

Table 3-28 CNN model result of dynamic co-existence features

##### RNN

In Table3–29, the results of the RNN model show the accuracy, average precision, F1 weight, and Hamming loss of the RNN model at five different levels of dynamic co-existence features. As we observe, the RNN model performs well on API permissions, with an accuracy of over 94.1% at all levels. This means that the model is able to correctly classify over 94.1% of the malware and benign samples in the dataset. The average precision is also high, ranging from 95% to 98.66%, indicating that the model is good at identifying true-positive cases of malware. The F1 weight, which takes into account both precision and recall, is also above 98.19% for all levels, suggesting the model has a good balance between these two metrics. Finally, the Hamming loss is relatively low, ranging from 0.04 to 0.06, which means that the model is making few mistakes in its predictions.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Average Precision | F1-score | Hamming Loss |
| Level 1 | 50% | 92.74 | 95.47 | 94.47 | 0.06 |
| Level 2 | 25% | 93.68 | 98.56 | 95.96 | 0.05 |
| Level 3 | 10% | 93.68 | 98.62 | 97.25 | 0.04 |
| Level 4 | 5% | 94.1 | 98.66 | 98.19 | 0.04 |
| Level 5 | 1% | 94 | 98.65 | 98.15 | 0.04 |

Table 3-29 RNN model result of dynamic co-existence features

##### ANN

In the result of the ANN model using dynamic features of co-existence, Table3-30 displayed the accuracy, average precision, F1 weight, and Hamming loss of the ANN model at different levels with thresholds. These metrics are all used to measure the performance of a deep learning model. The accuracy of the model ranges from 93% to 95.16%, the average precision is 95% to 98.56%, and the F1 weight is 93% to 95.66%. the highest accuracy, precision, and F1 score in level 5. F1 weight is a harmonic means of precision and recall. Hamming loss is a measure of the difference between the predicted labels and the true labels.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Average Precision | F1-score | Hamming Loss |
| Level 1 | 50% | 93.68 | 95.68 | 93.28 | 0.08 |
| Level 2 | 25% | 94.94 | 98.14 | 94.34 | 0.07 |
| Level 3 | 10% | 94.84 | 98.44 | 94.44 | 0.05 |
| Level 4 | 5% | 95.15 | 98.45 | 95.55 | 0.05 |
| Level 5 | 1% | 95.16 | 98.56 | 95.66 | 0.05 |

Table 3-30 ANN model result of dynamic co-existence features

#### Malgenome, Only Permissions Dynamic Co-existence Features Result

##### Decision tree

In Table3-31 we used the decision classifier model to detect malware using dynamic co-existence features. The accuracy, precision, recall, and F1 score of the model increase as the level increases as per thresholds. This model result shows the ability of malware detection increase as per level and shows it can be solved complex patterns of malware, overfitting training data is also improved. The accuracy of the model is highest at level 3, with a score of 81.74%. The precision of the model is highest at level 4, with a score of 84.56%. The recall of the model is highest at level 4, with a score of 82.39%. The F1 score of the model is highest at level 4, with a score of 81.76%. The cross-validation of the model is highest at level 4, with a score of 75.38%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Precision | Recall | Cross-validation | F1-score |
| Level 1 | 50% | 74.6 | 73.3 | 73.13 | 69.4 | 74.4 |
| Level 2 | 25% | 80.55 | 82.55 | 79.53 | 75.15 | 79.55 |
| Level 3 | 10% | 81.74 | 83.74 | 80.74 | 75.24 | 80.74 |
| Level 4 | 5% | 78.76 | 84.56 | 82.39 | 75.38 | 81.76 |
| Level 5 | 1% | 78.17 | 84.17 | 82.17 | 75.32 | 81.47 |

Table 3-31 Decision tree model result of dynamic co-existence features

##### Random Forest

In Table3-32 we used the random forest classifier model to detect malware using dynamic co-existence features. The accuracy, precision, recall, and F1 score of the model increase as the level increases as per thresholds. This model result shows the ability of malware detection increase as per level and shows it can be solved complex patterns of malware, overfitting training data is also improved. The accuracy of the model is highest at level 3, with a score of 99.6%. The precision of the model is highest at level 3, with a score of 98.2%. The recall of the model is highest at level 3, with a score of 97.9%. The F1 score of the model is highest at level 3, with a score of 98.41%. The cross-validation of the model is highest at level 3, with a score of 96.3%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Precision | Recall | Cross-validation | F1-score |
| Level 1 | 50% | 75 | 72 | 75 | 75 | 75 |
| Level 2 | 25% | 79.96 | 79.46 | 79.96 | 79.96 | 79.96 |
| Level 3 | 10% | 99.6 | 98.2 | 97.9 | 96.3 | 98.4 |
| Level 4 | 5% | 80.15 | 85.15 | 82.15 | 82.45 | 82.43 |
| Level 5 | 1% | 79.56 | 80.56 | 80.56 | 80.56 | 80.56 |

Table 3-32 Random Forest model result of dynamic co-existence features

##### Logistic regression

In Table3-33 we used the Logistic regression model to detect malware using dynamic co-existence features. The accuracy, precision, recall, and F1 score of the model increase as the level increases as per thresholds. This model result shows the ability of malware detection increase as per level and shows it can be solved complex patterns of malware, overfitting training data is also improved. The accuracy of the model is highest at level 4, with a score of 80.55%. The precision of the model is highest at level 4, with a score of 80.55%. The recall of the model is highest at level 4, with a score of 80.76%. The F1 score of the model is highest at level 4, with a score of 78.15%. The cross-validation of the model is highest at level 4, with a score of 76.15%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Precision | Recall | Cross-validation | F1-score |
| Level 1 | 50% | 74.8 | 72.8 | 73.52 | 69.98 | 72.8 |
| Level 2 | 25% | 79.16 | 80.16 | 79.55 | 73.16 | 76.16 |
| Level 3 | 10% | 78.76 | 80.26 | 80.55 | 74.76 | 77.76 |
| Level 4 | 5% | 80.55 | 80.55 | 80.76 | 76.55 | 78.15 |
| Level 5 | 1% | 78.17 | 80.37 | 80.47 | 76.17 | 76.27 |

Table 3-33 Logistic regression model result of dynamic co-existence features

##### CNN

Table3-34 shows the performance of a CNN model on dynamic features of co-existence. The accuracy of the model increases from 66% at level 1 to 80.78% at level 5. The precision also increases from 68% to 82.28%, and the F1 score increases from 62% to 80.78%. However, the Hamming loss decreases from 0.32 to 0.25. This model shows the ability to detect malware and learn the most effective features. However, it depends on the task that we have performed on it. these APIs and permissions about this result.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Average Precision | F1-score | Hamming Loss |
| Level 1 | 50% | 66.84 | 68.84 | 62.84 | 0.32 |
| Level 2 | 25% | 79.47 | 80.17 | 74.17 | 0.29 |
| Level 3 | 10% | 79.47 | 80.37 | 76.37 | 0.25 |
| Level 4 | 5% | 79.78 | 81.18 | 78.48 | 0.25 |
| Level 5 | 1% | 80.78 | 82.28 | 80.78 | 0.25 |

Table 3-34 CNN model result of dynamic co-existence features

##### RNN

In Table3–35, the results of the RNN model show the accuracy, average precision, F1 weight, and Hamming loss of the RNN model at five different levels of dynamic co-existence features. As we observe, the RNN model performs well on API permissions, with an accuracy of over 79.78% at all levels. This means that the model is able to correctly classify over 79.78% of the malware and benign samples in the dataset. The average precision is also high, ranging from 79% to 87.41%, indicating that the model is good at identifying true-positive cases of malware. The F1 weight, which takes into account both precision and recall, is also above 76.14% for all levels, suggesting the model has a good balance between these two metrics. Finally, the Hamming loss is relatively low, ranging from 0.25 to 0.33, which means that the model is making few mistakes in its predictions.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Average Precision | F1-score | Hamming Loss |
| Level 1 | 50% | 70.42 | 79.12 | 69.42 | 0.33 |
| Level 2 | 25% | 79.68 | 87.38 | 74.68 | 0.28 |
| Level 3 | 10% | 79.78 | 87.38 | 75.18 | 0.26 |
| Level 4 | 5% | 79.51 | 87.41 | 75.59 | 0.25 |
| Level 5 | 1% | 78.94 | 86.54 | 76.14 | 0.25 |

Table 3-35 RNN model result of dynamic co-existence features

##### ANN

In the result of the ANN model using dynamic features of co-existence, Table3-36 displayed the accuracy, average precision, F1 weight, and Hamming loss of the ANN model at different levels with thresholds. These metrics are all used to measure the performance of a deep learning model. The accuracy of the model ranges from 70% to 81.62%, the average precision is 77.42% to 88.92%, and the F1 weight is 69.22% to 72.67%. the highest accuracy, precision, and F1 score in level 5. F1 weight is a harmonic means of precision and recall. Hamming loss is a measure of the difference between the predicted labels and the true labels.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Average Precision | F1-score | Hamming Loss |
| Level 1 | 50% | 70.42 | 77.42 | 69.22 | 0.31 |
| Level 2 | 25% | 80.1 | 85.11 | 75.1 | 0.27 |
| Level 3 | 10% | 79.37 | 88.77 | 73.33 | 0.25 |
| Level 4 | 5% | 79.26 | 88.56 | 75.24 | 0.24 |
| Level 5 | 1% | 81.62 | 88.92 | 75.67 | 0.25 |

Table 3-36 ANN model result of dynamic co-existence features

## Brief Summary

We explored machine learning algorithms and deep learning algorithms, including Random Forest, Decision Tree, Logistic Regression, CNN, RNN, and ANN, to identify the best performers for malware detection using dynamic co-existence features. As we discussed our reach aim, malware hidden their pattern into another application. For these reasons, it can get our personal data, which is a very serious issue, even our bank information and countries defense information leakage. This is our proposed experiment, which has been successfully completed.

We are using tools in Python, and Python has some libraries, such as Pandas and Numpy, which we mentioned in our proposal model. Then we use the FPM-IN algorithm to find the hidden malware shape. We make a level 1 to level 5 detection of dynamic co-existence; each level contains a threshold level of the code for how features will be shared. These are works with permissions, API calls, and combinations of APIs and permissions. Which is smoothly divided to make dynamic co-existence.

Our experiments revealed that Random Forest yielded the highest-performance Malgenome dataset, demonstrating its effectiveness rather than static features. We are addressing the result of 99%, and the Drebin datasets have a remarkable performance of 98.8%, which is a better than previous researchers model. The API-permission combination to detect the malware, although 99% at the API calls, permission is at 98%, which can detect the malware in cases of co-existence. As we observe the threshold, the level of 5 should increase, but our experiment shows level 4 and level 3 are the average highest accuracy levels. The deep learning model has remarkable performance. RNN, CNN, and ANN in the Drebin level of co-existence ANN and CNN have the highest accuracy and precision (91% to 98%), while Malgenome datasets have more accuracy than Drebin datasets 90% to 98.5%. Logistic and RNN models are good at the dataset of permissions, with almost 80% to 88 accuracy and precision

# Dynamic Co-existence Features And FPMA Algorithm

## Introduction

In this chapter we are going to describe dynamic features of co-existence using Frequent Pattern Mining Approximate Algorithm (FPMAA) with Fp-Max. Before we start our another new approach, we can discuss about the pervious research. In previous research that belongs to static features and dynamic based approach to detect the malware but in the detection always a room for the research to make it good finding about the detection of malware. Previous method is not very large observations and experimental to make novel based approach however novel based detection still investigation in the process. Traditional frequent pattern mining (FP-Max) focuses on identifying maximal frequent itemsets that appear frequently within a dataset. However, leveraging approximation algorithms can indeed be a viable solution for handling large datasets when exact results are not crucial, and to accept a certain level of tolerance in terms of accuracy.

This chapter about the tool techniques, what we are using model, language, programming APIs and user interfaces we need to explore for dynamic co-existence features for malware detection. We are using dynamic analysis and implement into the real-world for malware detection.

The limitation of approximation algorithm and may not find all true frequent itemset with the same guarantee for all itemset. The choice of affects the trade-off between accuracy and efficiency. Our aim of use this algorithm sample algorithms require processing only a portion of data significantly reducing runtime compared to exact algorithms and handle the large efficiently for making suitable for data scenarios A higher α leads to faster execution but potentially less accurate results. we are going to explain in proposal statement.

## Problem Statement

The previous research depended on static features, dynamic analysis, and a dynamic approach the features not correctly defined even the static 150 features are less to investigate. The previous research does not mention about the deep learning if applied. No limitations describe by the author means it can be works on cloud, the answer is no. the other problems in this previous research we have identified multiple problems, we are going to mention every problem which authors has done below

1）Lacks in-depth analysis and validation of assumptions about the co-existence features

2）The author of malware does not define all frequent elements they were using

3）Limitation has not been described and computational cost of feature generation

4）Only machine learning analysis about the co-existence features not deep learning.

5）Not using hyperparameters.

7）They focus on static datasets instead of dynamic features that will enhance the accuracy

In the current Android malware detection approaches, only those employing the FP-Max algorithm using Approximation of algorithm, generally concentrate on identifying frequent itemsets. These represent combinations of features that frequent item within a dataset. However, leveraging approximation algorithms allows us to benefit from its core principles while making it feasible to handle large datasets at the cost of some tolerance for inexact results

## Dynamic Co-existence and FPMA Algorithm Proposed Model

The focus of generate dynamic co-existence. comparable data mining algorithms, FP-Max algorithm with Approximate algorithms. These methods examine how different applications interact with each other and detect malicious applications while changing their patterns of interaction when applying deep learning to co-existence features. Figure 4-1 is defined the step to solve the dynamic features of co-existence malware in Android application.

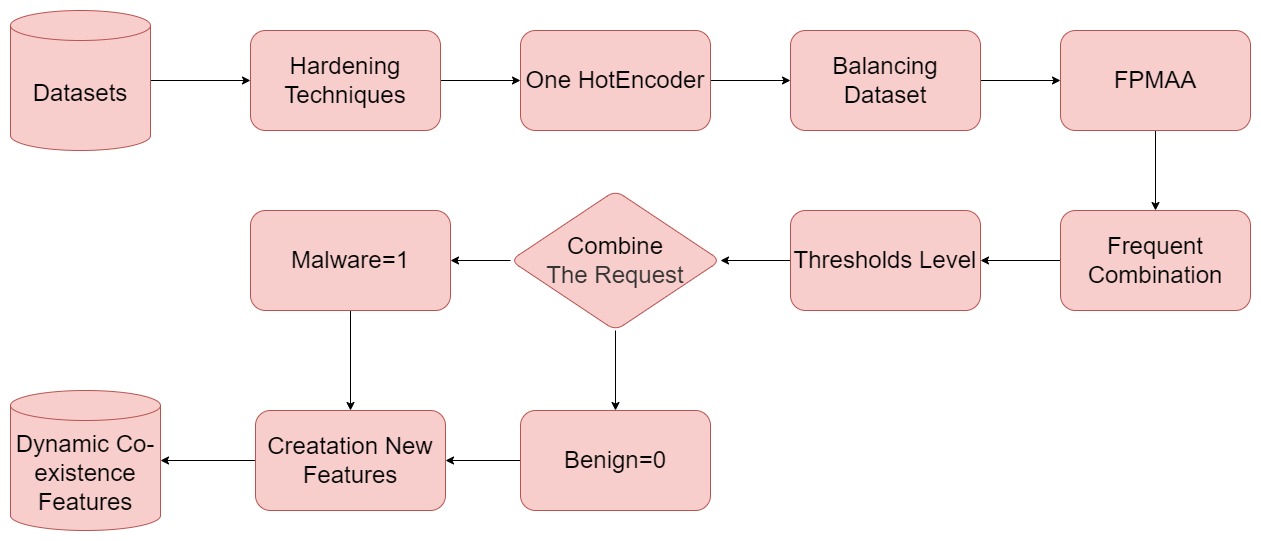


Figure 4-1 The research workflow of dynamic features of malware detection using dynamic co-existence

In this proposal method we are going describe each work, these are the introduction and information what techniques we will apply to solve our problem. This chapter has main research on about FP-max using approximation of algorithm to detect the malware using dynamic co-existence features. Steps are common but variation of the algorithm creation of new datasets and new finding.

### Frequent Pattern Mining Approximate Algorithm (FPMAA)

FP-Max, a powerful upgrade to the FP-growth algorithm, excels at pinpointing the most informative frequent patterns within datasets. It surpasses traditional methods by prioritizing the discovery of "maximal frequent itemsets," which are unique patterns that can't be expanded further without losing their frequency [2, 8, 11]. The enhancement of this algorithm using sampling-based approach. We don’t use directly this method because analyzing massive datasets due to its computational limitations. However, leveraging approximation algorithms can indeed be a viable solution for handling large datasets when exact results are not crucial, and we willing to accept a certain level of tolerance in terms of accuracy.

### Challenges with FPMA Algorithm

In this thesis we are going to explain our FPMA algorithm are several challenges. We are going to explain each expected challenge. The word alpha is a directly implies on balance between accuracy and efficiency. The higher alpha leads the execution fast but potentially misses true frequent item. Random sampling with replacement might not always create a representative sample of the entire data, potentially affecting the accuracy of the identified frequent itemsets This approach might not be ideal for datasets containing rare itemsets or exhibiting a highly skewed distribution. Rare itemsets might be missed in the sample, and skewed data might lead to inaccurate results for less frequent itemsets research While the code provides an approximation guarantee, it doesn't give information about the magnitude of potential errors. Understanding the potential deviation from exact results dealing with the large datasets is crucial for reliable interpretation

### Approaches for Addressing the Challenges while using FPAMAA

Exercise caution while solving the alpha. The balance between accuracy and efficiency is necessary for our special usage in the dynamic co-existence feature to identify malware. Experimenting with various values may influence the outcomes. We choose a random sample to investigate further in order to possibly enhance the accuracy of the approximation technique. We are mindful of the limitations while using these algorithms.

#### Sample-Based Approach

We are utilizing a sample-based technique with the FP-Max algorithm, which we can directly integrate since we are evaluating huge datasets owing to its processing restrictions. Our suggestion model relies on the equation and the workflow. Data, for which we are utilizing the Malgenome and Drebin datasets. Minimum thresholds, and we need an estimated ratio. The approximate ratio is termed α, and this alpha value between 0 and 1 specifies the intended trade-off between accuracy and efficiency. A higher α leads to quicker execution but perhaps less accurate outcomes. As we adjusted the parameters, we computed the sample and determined the number of transactions to sample based on the desired approximation ratio (α) and the total number of transactions (N).

........................................................(4-1)

After executing this sample size of data, we changed the minimum support of the threshold. While our approach about malware co-existence dynamic characteristics was using the level of thresholds, now we are using the adjusted minimum support. Here is the additional equation for the adjusted minimum**.**

**................**(4-2)

**After**After these equations we need to execute the code, run the algorithm and produce more frequent things using thresholds however this FP-Max only select those which often items with high accuracy. We analyze this approach for dynamic co-existence malware utilizing threshold level. In this chapter we are utilizing altered thresholds which is lowest when our threshold level exceeds 1% we have observed FP-Max method in detail below these parts

### Frequent pattern FPMAA

FPMAA Frequent Pattern Mining Approximate Algorithm, a frequent pattern mining algorithm is high efficient to detect malware. its the best when information streams about the sequence of feature such as system calls or network traffic. FP-max may detect patterns of harmful behavior of malware by finding often persistent combinations of these features [8]. In the frequent pattern when will use FPMAA algorithm. random sample of features among them using the sample size equation. Another hand we were using thresholds, while using thresholds. Apply the equation about the features. Than the settings of name and save into the new files as per level 1 to 5, it means if a file API calls, it will dissociate into 5 level. And each level has detection malware of dynamic co-existence into 6 detection algorithms.

### New Datasets Generated Dynamic Co-Existence Features

Our methodology is about dynamic co-existence of features that we need some libraries in Python and make an environment in Python where all the libraries are installed next. Download the main datasets, Drebin or Malgenome. Check if these files are suitable or not; if not, take the data into pieces and merge the data. After the data, use the pre-processing technique of one hotencode operation: if the value of the cell is greater than 1, use 1; if the data is less than 0, or in double format, consider 0, in the representation of malware 1, and 0 is a benign app. Next step. Balance the data and apply the FPMA Algorithm within a loop. A loop has certain values. [50, 25, 10, 5, 1]. These are the thresholds of our method. Then applied the equations and general analysis like as error and system handle the process file our datasets are very large. Than the request that each variable be checked and then dumped into a csv file. But make sure one variable is a dependent variable. We have malware classes of variables to classify malware and benign apps

## Experimental of Dynamic Co-existence Result and Analysis

We are going to discuss about experiment first, we took the datasets from Malgenome, Drebin and only two datasets about API and permissions of datasets these are very popular now a days in research of malware In this experiment about dynamic co-existence in a dataset of permissions and call APIs for Android, Our main objective is to identify frequent elements in these datasets. For the initial step, we install Python and its libraries, like Numpy, Pandas, and Scikit-Learn, and for the pattern analysis, we use Mxltend These are useful interfaces for our research work.

Next, we load the CSV file, which contains data about permissions or API calls. These files have a specific variable whose name is class. that label denotes an app category. Check the name to ensure the class variable exists or not. and also some files as permission contain class variables, so we name them by the programming and make them correct for our needs. The class variable employs one-hot encoding to transform the categorical into a numerical representation to make it easy for analysis by the machine and deep learning, then we fetch the data and remove the duplication for redundancy.

FPMAA emerges as a valuable tool to analyze the proposed co-existence attributes for call the class of sample random, after the cast into integer, when casting of format alpha multiply by data, alpha we supposed 0.8, now data length it will generate automatically after this step. We can put the result of random function it will generate the sample. After that we will use adjusted minimum support minimum support alpha after these data sample and alpha use in FP-Max algorithm to geneterate the features Android malware detection. Each app sample becomes a transaction in this framework, with co-existence attributes forming the individual items. By setting varying a minimum support threshold of 50% to 1%, FP-max efficiently identifies frequently occurring combinations of permissions, API calls, and permission-API pairs within these transactions. Establishing a baseline based on these frequent patterns in benign apps allows flagging new samples exhibiting significant deviations. Careful analysis of these outliers, leveraging domain expertise and threat intelligence, helps understand potential malicious behavior and potentially refine malware signatures for enhanced detection. However, mindful consideration of false positives and evolving malware tactics necessitates combining this approach with other security measures for robust protection.

Our exploration continues to generate co-existence features. It loops through the frequently created patterns, and for each permission or API call within a pattern, it adds a new column to a dedicated dataframe. This new column holds the related information of permission or API call value from the original data and generates a binary representation of whether an app contains a specific information permission combination.

To better understand insights, co-existence is categorized into five levels. It defines the five thresholds that determine varying levels of co-existence. For each level threshold, the code meticulously examines the average value for each co-existence feature. If the average value surpasses the threshold, it shows that the individual permission or API call combination is frequently associated, and the corresponding feature name is added to a list to show that level.

Finally, our experimental model, executed meticulously, saves the data for further analysis. loop iterates through each level, showing the list of co-existence features. For each level, it creates a new dataframe containing only those features deemed significant based on the chosen threshold, along with the original class label for context. These dataframes were saved as separate CSV files, each with a descriptive filename indicating the level and the FP-Growth algorithm used, providing a clear organization for further exploration and potential use in understanding app behavior or security implications.

### Drebin Dynamic Co-existence Features Results

In this section we are collecting the result of Drebin dataset. These datasets most famous in the malware detection world. Are basically try each model and get the result and analysis about the result.

#### Drebin, API-Permission Combination Dynamic Co-Existence Features Result

In this section, we took a dataset file API and Permission together interact about the malware. We need to check this dataset. We are applying our six models with each dataset. The threshold of the dynamic features.

##### Decision tree

w In Table3-1 the mode we used the decision classifier model to detect malware using dynamic co-existence features. The accuracy, precision, recall, and F1 score of the model increase as the level increases as per thresholds. This model result shows the ability of malware detection increase as per level and shows it can be solved complex patterns of malware, overfitting training data is also improved. The accuracy of the model is highest at level 4, with a score of 80.89%. The precision of the model is highest at level 1, with a score of 85.82%. The recall of the model is highest at level 4, with a score of 96.80%. The F1 score of the model is highest at level 4, with a score of 79.70%. The cross-validation of the model is highest at level 4, with a score of 79.83%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Precision | Recall | Cross-validation | F1-score |
| Level 1 | 50% | 66.78 | 85.82 | 62.45 | 66.91 | 72.29 |
| Level 2 | 25% | 79.38 | 75.23 | 91.43 | 79.44 | 78.73 |
| Level 3 | 10% | 79.61 | 75.85 | 93.83 | 79.72 | 78.63 |
| Level 4 | 5% | 80.89 | 77.84 | 96.80 | 79.83 | 79.70 |
| Level 5 | 1% | 79.83 | 76.81 | 95.83 | 79.81 | 79.53 |

Table4-1 Decision tree model result of dynamic co-existence features using API-permissions

##### Random Forest

In Table3-2 the mode we used the random forest model to detect malware using dynamic co-existence features. The accuracy, precision, recall, and F1 score of the model increase as the level increases as per thresholds. This model result shows the ability of malware detection increase as per level and shows it can be solved complex patterns of malware, overfitting training data is also improved. The accuracy of the model is highest at level 4, with a score of 80.89%. The precision of the model is highest at level 1, with a score of 85.82%. The recall of the model is highest at level 4, with a score of 96.80%. The F1 score of the model is highest at level 4, with a score of 79.70%. The cross-validation of the model is highest at level 4, with a score of 79.83%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Precision | Recall | Cross-validation | F1-score |
| Level 1 | 50% | 66.56 | 85.38 | 62.32 | 67.12 | 72.05 |
| Level 2 | 25% | 80.87 | 78.07 | 83.03 | 79.22 | 80.47 |
| Level 3 | 10% | 79.29 | 74.95 | 82.4 | 79.07 | 78.52 |
| Level 4 | 5% | 78.93 | 74.24 | 82.31 | 7968 | 78.06 |
| Level 5 | 1% | 79.65 | 75.66 | 82.58 | 79.93 | 78.97 |

Table 4-2 Random Forest model result of dynamic co-existence features using API-permissions

##### Logistic regression

This model we examine in Table4-3 the effectiveness of a logistic regression model utilizing dynamic co-existence features for malware detection. The model's performance across five levels of complexity is assessed using various metrics: threshold, accuracy, precision, recall, F1 score, and cross-validation. The analysis reveals a promising trend with most metrics improving as the model's complexity increases, suggesting enhanced capability to distinguish malware from benign samples. Notably, levels 1 and 2 achieve peaks in accuracy and precision (76.90% and 75.52%, respectively), while level 1 and 2 takes the lead in recall and F1 score (80.13% and 75.40%, respectively). However, the cross-validation F1 score paints a different picture. While level 5 remains strong, the highest score of 88% is achieved at level 2, indicating a potential trade-off between complexity and generalizability. Increased complexity leads to better performance on training data but might introduce overfitting, hindering real-world effectiveness.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Precision | Recall | Cross-validation | F1-score |
| Level 1 | 50% | 76.90 | 71.23 | 80.13 | 70.20 | 73.51 |
| Level 2 | 25% | 74.42 | 75.52 | 78.43 | 87.41 | 75.40 |
| Level 3 | 10% | 75.46 | 68.8 | 78.96 | 87.50 | 74.58 |
| Level 4 | 5% | 75.46 | 72.99 | 78.89 | 88.00 | 74.80 |
| Level 5 | 1% | 74.42 | 68.29 | 78.19 | 87.99 | 74.52 |

Table4-3 Logistic regression model result of dynamic co-existence features using API-permissions

##### CNN

Table 4-4 shows the performance of a CNN model on dynamic features of co-existence. The accuracy of the model increases from 75.90% at level 1 to 80.46% at level 3. The precision also increases from 79.23% to 92.99%, and the F1 score increases from 81.73% to 88.15%. However, the Hamming loss decreases from 0.19 to 0.14. This model shows the ability to detect malware and learn the most effective features. However, it depends on the task that we have performed on it. these APIs and permissions about this result.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Average Precision | F1-score | Hamming Loss |
| Level 1 | 50% | 75.90 | 79.23 | 81.73 | 0.19 |
| Level 2 | 25% | 78.42 | 90.52 | 84.80 | 0.15 |
| Level 3 | 10% | 80.46 | 90.8 | 87.59 | 0.14 |
| Level 4 | 5% | 79.46 | 92.99 | 88.15 | 0.14 |
| Level 5 | 1% | 79.42 | 92.29 | 87.05 | 0.14 |

Table4-4 CNN model result of dynamic co-existence features using API-permissions

##### RNN

In Table4–5, the results of the RNN model show the accuracy, average precision, F1 weight, and Hamming loss of the RNN model at five different levels of dynamic co-existence features. As we observe, the RNN model performs well on API permissions, with an accuracy of over 80.36% at all levels. This means that the model is able to correctly classify over 80.36% of the malware and benign samples in the dataset. The average precision is also high, ranging from 80% to 92.67%, indicating that the model is good at identifying true-positive cases of malware. The F1 weight, which considers both precision and recall, is also above 86.08% for all levels, suggesting the model has a good balance between these two metrics. Finally, the Hamming loss is relatively low, ranging from 0.43 to 0.51, which means that the model is making few mistakes in its predictions.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Average Precision | F1-score | Hamming Loss |
| Level 1 | 50% | 75.95 | 80.00 | 76.27 | 0.43 |
| Level 2 | 25% | 78.32 | 90.21 | 78.84 | 0.51 |
| Level 3 | 10% | 78.81 | 92.51 | 84.60 | 0.51 |
| Level 4 | 5% | 78.89 | 92.67 | 84.01 | 0.51 |
| Level 5 | 1% | 80.36 | 92.47 | 86.08 | 0.50 |

Table4-5 RNN model result of dynamic co-existence features using API-permissions

##### ANN

In the result of the ANN model using dynamic features of co-existence, Table4-6 displayed the accuracy, average precision, F1 weight, and Hamming loss of the ANN model at different levels with thresholds. These metrics are all used to measure the performance of a deep learning model. The accuracy of the model ranges from 94.68% to 96.54%, the average precision is 96.05% to 98.43%, and the F1 weight is 91% to 96.82%. the highest accuracy, precision, and F1 score in level 3. F1 weight is a harmonic means of precision and recall. Hamming loss is a measure of the difference between the predicted labels and the true labels.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Average Precision | F1-score | Hamming Loss |
| Level 1 | 50% | 94.68 | 96.05 | 91.79 | 0.061 |
| Level 2 | 25% | 95.18 | 98.21 | 95.27 | 0.047 |
| Level 3 | 10% | 96.54 | 98.43 | 96.82 | 0.041 |
| Level 4 | 5% | 95.58 | 97.87 | 95.54 | 0.044 |
| Level 5 | 1% | 95.38 | 98.03 | 94.82 | 0.048 |

Table4-6 ANN model result of dynamic co-existence features using API-permissions

#### Drebin, Only API Calls Dynamic Co-Existence Features Result

In this research we are going to explain the API Calls for malware detection. Hance we are giving the data analysis of API only

##### Decision Tree

In Table 4-7 the model we used the decision classifier model to detect malware using dynamic co-existence features. The accuracy, precision, recall, and F1 score of the model increase as the level increases as per thresholds. This model result shows the ability of malware detection increase as per level, and also shows it can be solved complex patterns of malware, overfitting training data is also improved. The accuracy of the model is highest at level 5, with a score of 84.88%. The precision of the model is highest at level 5, with a score of 89.42%. The recall of the model is highest at level 4, with a score of 86.62%. The F1 score of the model is highest at level 5, with a score of 85.41%. The cross-validation of the model is highest at level 5, with a score of 83.42%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Precision | Recall | Cross-validation | F1-score |
| Level 1 | 50% | 78.91 | 78.92 | 78.93 | 78.91 | 78.91 |
| Level 2 | 25% | 82.42 | 86.42 | 86.42 | 83.42 | 84.41 |
| Level 3 | 10% | 83.32 | 88.42 | 86.32 | 73.38 | 82.41 |
| Level 4 | 5% | 84.71 | 85.42 | 86.62 | 74.88 | 84.41 |
| Level 5 | 1% | 84.88 | 89.42 | 86.12 | 74.52 | 85.41 |

Table 4-7 Decision tree model result of dynamic co-existence features

##### Random Forest

In Table 4-8 the model we used the Random classifier model to detect malware using dynamic co-existence features. The accuracy, precision, recall, and F1 score of the model increase as the level increases as per thresholds. This model result demonstrates a smaller advantage, indicating a potential trade-off between complexity and generalizability. While higher complexity enhances performance on training data, it might lead to overfitting, hindering real-world effectiveness. The accuracy of the model is highest at level 4, with a score of 78.87%. The precision of the model is highest at level 4, with a score of 77.82%. The recall of the model is highest at level 4, with a score of 77.15%. The F1 score of the model is highest at level 5, with a score of 74.83%. The cross-validation of the model is highest at level 4, with a score of 74.88%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Precision | Recall | Cross-validation | F1-score |
| Level 1 | 50% | 60.93 | 59.92 | 59.93 | 60.91 | 64.91 |
| Level 2 | 25% | 72.43 | 71.42 | 71.42 | 73.42 | 71.41 |
| Level 3 | 10% | 73.79 | 72.32 | 72.33 | 73.38 | 73.33 |
| Level 4 | 5% | 78.87 | 77.82 | 77.15 | 74.88 | 73.83 |
| Level 5 | 1% | 75.83 | 75.83 | 75.83 | 74.52 | 74.83 |

Table 4-8 Random Forest model result of dynamic co-existence features

##### Logistic regression

In Table 4-9 the model we used the Logistic regression model to detect malware using dynamic co-existence features. The accuracy, precision, recall, and F1 score of the model increase as the level increases as per thresholds. This model result demonstrates the analysis reveals a promising trend with most metrics improving as the model's complexity increases, suggesting enhanced capability to distinguish malware from benign samples. The accuracy of the model is highest at level 3, with a score of 76.7%. The precision of the model is highest at level 5, with a score of 76.83%. The recall of the model is highest at level 4, with a score of 77.15%. The F1 score of the model is highest at level 5, with a score of 74.83%. The cross-validation of the model is highest at level 4, with a score of 74.88%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Precision | Recall | Cross-validation | F1-score |
| Level 1 | 50% | 61.3 | 60.92 | 59.93 | 60.91 | 64.91 |
| Level 2 | 25% | 73.4 | 71.42 | 71.42 | 73.42 | 71.41 |
| Level 3 | 10% | 76.7 | 72.32 | 72.33 | 73.38 | 73.33 |
| Level 4 | 5% | 75.8 | 74.82 | 77.15 | 74.88 | 73.83 |
| Level 5 | 1% | 77.8 | 76.83 | 75.83 | 74.52 | 74.83 |

Table 4-9 Logistic regression model result of dynamic co-existence features

##### CNN

Table4-10 shows the performance of a CNN model on dynamic features of co-existence. The accuracy of the model increases from 61.9% at level 1 to 76.4% at level 5. Level 5 is the highest accuracy in this model. The precision also increases from 70.92% to 83.42%, and the F1 score increases from 63.9% to 75.4%. However, the Hamming loss decreases from 1.181 to 0.062. This model shows the ability to detect malware and learn the most effective features. However, it depends on the task that we have performed on it. these APIs about this result.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Average Precision | F1-score | Hamming Loss |
| Level 1 | 50% | 61.9 | 70.92 | 63.9 | 1.181 |
| Level 2 | 25% | 74.4 | 80.53 | 73.4 | 0.083 |
| Level 3 | 10% | 72.4 | 80.32 | 72.4 | 0.074 |
| Level 4 | 5% | 74.4 | 83.42 | 74.4 | 0.075 |
| Level 5 | 1% | 76.4 | 86.4 | 75.4 | 0.062 |

Table 4-10 CNN model result of dynamic co-existence features

##### RNN

In Table 4–11, the results of the RNN model show the accuracy, average precision, F1 weight, and Hamming loss of the RNN model at five different levels of dynamic co-existence features. As we observe, the RNN model performs well on API permissions, with an accuracy of over 76.49% at all levels. This means that the model is able to correctly classify over 76.49% of the malware and benign samples in the dataset. The average precision is also high, varying from 80% to 85.93%, indicating that the model is good at identifying true-positive cases of malware. The F1 weight, which takes into account both precision and recall, is also above 75.63% for all levels, suggesting the model has a good balance between these two metrics. Finally, the Hamming loss is relatively low, ranging from 1.06 to 0.73, which means that the model is making few mistakes in its predictions.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Average Precision | F1-score | Hamming Loss |
| Level 1 | 50% | 70.19 | 80.92 | 70.1 | 1.06 |
| Level 2 | 25% | 73.48 | 81.42 | 71.42 | 0.75 |
| Level 3 | 10% | 72.45 | 82.43 | 72.43 | 0.72 |
| Level 4 | 5% | 76.49 | 85.93 | 75.63 | 0.73 |
| Level 5 | 1% | 75.44 | 85.82 | 75.52 | 0.87 |

Table 4-11 RNN model result of dynamic co-existence features

##### ANN

In the result of the ANN model using dynamic features of co-existence, Table4-12 displayed the accuracy, average precision, F1 weight, and Hamming loss of the ANN model at different levels with thresholds. These metrics are all used to measure the performance of a deep learning model. The accuracy of the model ranges from 70.1% to 77.4%, the average precision is 84.17% to 87.13%, and the F1 weight is 67.1% to 75.4%. the highest accuracy, precision, and F1 score in level 5, 2 and 3 respectively. F1 weight is a harmonic means of precision and recall. Hamming loss is a measure of the difference between the predicted labels and the true labels.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Average Precision | F1-score | Hamming Loss |
| Level 1 | 50% | 70.1 | 84.17 | 67.1 | 1.15 |
| Level 2 | 25% | 73.4 | 86.64 | 70.4 | 1.05 |
| Level 3 | 10% | 74.4 | 86.8 | 72.4 | 0.87 |
| Level 4 | 5% | 75.2 | 86.3 | 73.2 | 0.86 |
| Level 5 | 1% | 77.4 | 87.13 | 75.4 | 0.5 |

Table4-12 ANN model result of dynamic co-existence features

### Malgenome, Dynamic Co-existence Features Results

In this section we are collecting the result of Malgenome Dataset. These datasets most famous in the malware detection world. Are basically try each model and get the result and analysis about the result

#### Malgenome, API-Permission Combination Dynamic Co-Existence Features Result

In this section, we took a dataset file API and Permission together interact about the malware. We need to check this dataset. We are applying our six models with each dataset. The threshold of the dynamic features.

##### Decision tree

In Table4-13 we used the decision classifier model to detect malware using dynamic co-existence features. The accuracy, precision, recall, and F1 score of the model increase as the level increases as per thresholds. This model result shows the ability of malware detection increase as per level and shows it can be solved complex patterns of malware, overfitting training data is also improved. The accuracy of the model is highest at level 4, with a score of 97.8%. The precision of the model is highest at level 1, with a score of 50%. The recall of the model is highest at level 4, with a score of 97.89%. The F1 score of the model is highest at level 4, with a score of 96.80%. The cross-validation of the model is highest at level 4, with a score of 96.84%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Precision | Recall | Cross-validation | F1-score |
| Level 1 | 50% | 88.9 | 50% | 88.95 | 88.72 | 88.63 |
| Level 2 | 25% | 92.4 | 25% | 92.47 | 91.23 | 91.43 |
| Level 3 | 10% | 93.8 | 10% | 93.87 | 92.85 | 93.83 |
| Level 4 | 5% | 97.8 | 5% | 97.89 | 96.84 | 96.80 |
| Level 5 | 1% | 96.8 | 1% | 96.86 | 95.81 | 95.83 |

Table 4-13 Decision tree model result of dynamic co-existence features

##### Random Forest

In Table4-14 we used the random forest classifier model to detect malware using dynamic co-existence features. The accuracy, precision, recall, and F1 score of the model increase as the level increases as per thresholds. This model result shows the ability of malware detection increase as per level and shows it can be solved complex patterns of malware, overfitting training data is also improved. The accuracy of the model is highest at level 4, with a score of 99%. The precision of the model is highest at level 4, with a score of 98.93%. The recall of the model is highest at level 4, with a score of 98.88%. The F1 score of the model is highest at level 4, with a score of 98.85%. The cross-validation of the model is highest at level 4, with a score of 95.96%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Precision | Recall | Cross-validation | F1-score |
| Level 1 | 50% | 91.55 | 90.42 | 90.31 | 89.21 | 90.52 |
| Level 2 | 25% | 96.83 | 95.82 | 95.53 | 94.68 | 95.67 |
| Level 3 | 10% | 98.45 | 98.42 | 98.32 | 95.43 | 98.33 |
| Level 4 | 5% | 99.00 | 98.93 | 98.88 | 95.96 | 98.85 |
| Level 5 | 1% | 98.53 | 98.44 | 98.34 | 95.66 | 98.55 |

Table 4-14 Random Forest model result of dynamic co-existence features

##### Logistic regression

In Table4-15 we used the logistic regression model to detect malware using dynamic co-existence features. The accuracy, precision, recall, and F1 score of the model increase as the level increases as per thresholds. This model result shows the ability of malware detection increase as per level and shows it can be solved complex patterns of malware, overfitting training data is also improved. The accuracy of the model is highest at level 4, with a score of 90.83%. The precision of the model is highest at level 4, with a score of 89.99%. The recall of the model is highest at level 4, with a score of 89.89%. The F1 score of the model is highest at level 4, with a score of 89.80%. The cross-validation of the model is highest at level 4, with a score of 88%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Precision | Recall | Cross-validation | F1-score |
| Level 1 | 50% | 78.90 | 76.23 | 76.13 | 70.20 | 76.51 |
| Level 2 | 25% | 88.42 | 87.52 | 87.43 | 87.41 | 87.40 |
| Level 3 | 10% | 88.58 | 87.8 | 87.96 | 87.50 | 87.58 |
| Level 4 | 5% | 90.83 | 89.99 | 89.89 | 88.00 | 89.80 |
| Level 5 | 1% | 89.59 | 89.29 | 89.19 | 87.99 | 89.52 |

Table 4-15 Logistic regression model result of dynamic co-existence features

##### CNN

Table4-16 shows the performance of a CNN model on dynamic features of co-existence. The accuracy of the model increases from 91.95% at level 1 to 95.82% at level 4. The precision also increases from 96.12% to 98.49%, and the F1 score increases from 92.73% to 97.59%. However, the Hamming loss decreases from 0.042 to 0.081. This model shows the ability to detect malware and learn the most effective features. However, it depends on the task that we have performed on it. these APIs and permissions about this result.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Average Precision | F1-score | Hamming Loss |
| Level 1 | 50% | 91.95 | 96.12 | 91.73 | 0.081 |
| Level 2 | 25% | 94.42 | 98.21 | 96.84 | 0.046 |
| Level 3 | 10% | 93.84 | 98.43 | 97.59 | 0.045 |
| Level 4 | 5% | 95.82 | 98.49 | 98.15 | 0.042 |
| Level 5 | 1% | 95.32 | 98.13 | 97.05 | 0.049 |

Table 4-16 CNN model result of dynamic co-existence features

##### RNN

In Table4–17, the results of the RNN model show the accuracy, average precision, F1 weight, and Hamming loss of the RNN model at five different levels of dynamic co-existence features. As we observe, the RNN model performs well on API permissions, with an accuracy of over 93.95% at all levels. This means that the model is able to correctly classify over 93.95% of the malware and benign samples in the dataset. The average precision is also high, ranging from 97.21% to 98%, indicating that the model is good at identifying true-positive cases of malware. The F1 weight, which considers both precision and recall, is also above 96.84% for all levels, suggesting the model has a good balance between these two metrics. Finally, the Hamming loss is relatively low, ranging from 0.043 to 0.051, which means that the model is making few mistakes in its predictions.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Average Precision | F1-score | Hamming Loss |
| Level 1 | 50% | 93.95 | 98.00 | 96.27 | 0.043 |
| Level 2 | 25% | 92.32 | 97.21 | 96.84 | 0.051 |
| Level 3 | 10% | 90.81 | 97.51 | 96.60 | 0.051 |
| Level 4 | 5% | 92.89 | 97.67 | 96.01 | 0.051 |
| Level 5 | 1% | 92.36 | 97.47 | 96.08 | 0.050 |

Table 4-17 RNN model result of dynamic co-existence features

##### ANN

In the result of the ANN model using dynamic features of co-existence, Table4-18 displayed the accuracy, average precision, F1 weight, and Hamming loss of the ANN model at different levels with thresholds. These metrics are all used to measure the performance of a deep learning model. The accuracy of the model ranges from 94.68% to 96.54%, the average precision is 96% to 98.43%, and the F1 weight is 91% to 96.82%. the highest accuracy, precision, and F1 score in level 3. F1 weight is a harmonic means of precision and recall. Hamming loss is a measure of the difference between the predicted labels and the true labels.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Average Precision | F1-score | Hamming Loss |
| Level 1 | 50% | 94.68 | 96.05 | 91.79 | 0.061 |
| Level 2 | 25% | 95.18 | 98.21 | 95.27 | 0.047 |
| Level 3 | 10% | 96.54 | 98.43 | 96.82 | 0.041 |
| Level 4 | 5% | 95.58 | 97.87 | 95.54 | 0.044 |
| Level 5 | 1% | 95.38 | 98.03 | 94.82 | 0.048 |

Table 4-18 ANN model result of dynamic co-existence features

#### Malgenome, Only API Calls Dynamic Co-existence Features Result

In this research we are going to explain the API Calls for malware detection. Hance we are giving the data analysis of API only

##### Decision tree

In Table4-19 we used the decision classifier model to detect malware using dynamic co-existence features. The accuracy, precision, recall, and F1 score of the model increase as the level increases as per thresholds. This model result shows the ability of malware detection increase as per level and shows it can be solved complex patterns of malware, overfitting training data is also improved. The accuracy of the model is highest at level 5, with a score of 77.88%. The precision of the model is highest at level 4, with a score of 76.82%. The recall of the model is highest at level 4, with a score of 76.85%. The F1 score of the model is highest at level 5, with a score of74.83%. The cross-validation of the model is highest at level 4, with a score of 74.88%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Precision | Recall | Cross-validation | F1-score |
| Level 1 | 50% | 60.91 | 59.92 | 59.93 | 60.91 | 64.91 |
| Level 2 | 25% | 72.42 | 71.42 | 71.42 | 73.42 | 71.41 |
| Level 3 | 10% | 73.32 | 72.32 | 72.33 | 73.38 | 73.33 |
| Level 4 | 5% | 77.88 | 76.82 | 76.85 | 74.88 | 73.83 |
| Level 5 | 1% | 75.81 | 75.83 | 75.83 | 74.52 | 74.83 |

Table 4-19 Decision tree model result of dynamic co-existence features

##### Random Forest

In Table4-20 we used the random forest classifier model to detect malware using dynamic co-existence features. The accuracy, precision, recall, and F1 score of the model increase as the level increases as per thresholds. This model result shows the ability of malware detection increase as per level and shows it can be solved complex patterns of malware, overfitting training data is also improved. The accuracy of the model is highest at level 4, with a score of 78.87%. The precision of the model is highest at level 4, with a score of 77.82%. The recall of the model is highest at level 4, with a score of 77.15%. The F1 score of the model is highest at level 5, with a score of 74.83%. The cross-validation of the model is highest at level 4, with a score of 74.88%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Precision | Recall | Cross-validation | F1-score |
| Level 1 | 50% | 60.93 | 59.92 | 59.93 | 60.91 | 64.91 |
| Level 2 | 25% | 72.43 | 71.42 | 71.42 | 73.42 | 71.41 |
| Level 3 | 10% | 73.79 | 72.32 | 72.33 | 73.38 | 73.33 |
| Level 4 | 5% | 78.87 | 77.82 | 77.15 | 74.88 | 73.83 |
| Level 5 | 1% | 75.83 | 75.83 | 75.83 | 74.52 | 74.83 |

Table 4-20 Random Forest model result of dynamic co-existence features

##### Logistic regression

In Table4-21 we used the Logistic regression model to detect malware using dynamic co-existence features. The accuracy, precision, recall, and F1 score of the model increase as the level increases as per thresholds. This model result shows the ability of malware detection increase as per level and shows it can be solved complex patterns of malware, overfitting training data is also improved. The accuracy of the model is highest at level 5, with a score of 77.8%. The precision of the model is highest at level 5, with a score of 76.83%. The recall of the model is highest at level 4, with a score of 77.15%. The F1 score of the model is highest at level 5, with a score of 74.83%. The cross-validation of the model is highest at level 4, with a score of 74.88%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Precision | Recall | Cross-validation | F1-score |
| Level 1 | 50% | 61.3 | 60.92 | 59.93 | 60.91 | 64.91 |
| Level 2 | 25% | 73.4 | 71.42 | 71.42 | 73.42 | 71.41 |
| Level 3 | 10% | 76.7 | 72.32 | 72.33 | 73.38 | 73.33 |
| Level 4 | 5% | 75.8 | 74.82 | 77.15 | 74.88 | 73.83 |
| Level 5 | 1% | 77.8 | 76.83 | 75.83 | 74.52 | 74.83 |

Table 4-21 Logistic regression model result of dynamic co-existence features

##### CNN

Table4-22 shows the performance of a CNN model on dynamic features of co-existence. The accuracy of the model increases from 70.1% at level 1 to 77.4% at level 5. The precision also increases from 84.17% to 87.13%, and the F1 score increases from 67.1% to 75.4%. However, the Hamming loss decreases from 1.15 to 0.5. This model shows the ability to detect malware and learn the most effective features. However, it depends on the task that we have performed on it. these APIs and permissions about this result.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Average Precision | F1-score | Hamming Loss |
| Level 1 | 50% | 70.1 | 84.17 | 67.1 | 1.15 |
| Level 2 | 25% | 73.4 | 86.64 | 70.4 | 1.05 |
| Level 3 | 10% | 74.4 | 86.8 | 72.4 | 0.87 |
| Level 4 | 5% | 75.2 | 86.3 | 73.2 | 0.86 |
| Level 5 | 1% | 77.4 | 87.13 | 75.4 | 0.5 |

Table 4-22 CNN model result of dynamic co-existence features

##### RNN

In Table4–23, the results of the RNN model show the accuracy, average precision, F1 weight, and Hamming loss of the RNN model at five different levels of dynamic co-existence features. As we observe, the RNN model performs well on API permissions, with an accuracy of over 76.49% at all levels. This means that the model can correctly classify over 76.49% of the malware and benign samples in the dataset. The average precision is also high, ranging from 80% to 85.93%, indicating that the model is good at identifying true-positive cases of malware. The F1 weight, which considers both precision and recall, is also above 75.63% for all levels, suggesting the model has a good balance between these two metrics. Finally, the Hamming loss is relatively low, ranging from 0.72 to 1.06, which means that the model is making few mistakes in its predictions.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Average Precision | F1-score | Hamming Loss |
| Level 1 | 50% | 70.19 | 80.92 | 70.1 | 1.06 |
| Level 2 | 25% | 73.48 | 81.42 | 71.42 | 0.75 |
| Level 3 | 10% | 72.45 | 82.43 | 72.43 | 0.72 |
| Level 4 | 5% | 76.49 | 85.93 | 75.63 | 0.73 |
| Level 5 | 1% | 75.44 | 85.82 | 75.52 | 0.87 |

Table 4-23 RNN model result of dynamic co-existence features

##### ANN

In the result of the ANN model using dynamic features of co-existence, Table4-24 displayed the accuracy, average precision, F1 weight, and Hamming loss of the ANN model at different levels with thresholds. These metrics are all used to measure the performance of a deep learning model. The accuracy of the model ranges from 93% to 95.15%, the average precision is 95% to 98.56%, and the F1 weight is 93% to 95.66%. the highest accuracy, precision, and F1 score in level 5. F1 weight is a harmonic means of precision and recall. Hamming loss is a measure of the difference between the predicted labels and the true labels.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combination | Threshold | Accuracy | Average Precision | F1-score | Hamming Loss |
| Level 1 | 50% | 93.68 | 95.68 | 93.28 | 0.08 |
| Level 2 | 25% | 94.94 | 98.14 | 94.34 | 0.07 |
| Level 3 | 10% | 94.84 | 98.44 | 94.44 | 0.05 |
| Level 4 | 5% | 95.15 | 98.45 | 95.55 | 0.05 |
| Level 5 | 1% | 95.16 | 98.56 | 95.66 | 0.05 |

Table 4-24 ANN model result of dynamic co-existence features

## Brief Summary

We explored machine learning algorithms and deep learning algorithms, including Random Forest, Decision Tree, Logistic Regression, CNN, RNN, and ANN, to identify the best performers for malware detection using dynamic co-existence features. As we discussed our reach aim, malware hidden their pattern into another application. For these reasons, it can get our personal data, which is a very serious issue, even our bank information and countries defense information leakage. This is our proposed experiment, which has been successfully completed.

We are using tools in Python, and Python has some libraries, such as Pandas and Numpy, which we mentioned in our proposal model. Then we use the FP-Max algorithm with Approximation algorithm. (FPMA) algorithm contain the sample-based technique using the equation and to find the approximate accuracy and efficiency of malware patterns. We construct from level 1 to level 5 based on thresholds malware detection of dynamic co-existence This works with permissions, API calls, and combinations of APIs and permissions. Which is smoothly divided to make dynamic co-existence

Our experiments revealed that Random Forest yielded the highest-performance Drebin dataset. We are addressing the result of 99 %, and the Malgenome datasets have a remarkable performance of 98.5% in ANN model, which is a better model for the API-permission combination to detect the malware. although 99% at the API calls, permission is at 98.5%, which can detect the malware in cases of co-existence. As we observe the threshold, the level of 5 should increase, but our experiment shows level 4 and level 3 are the average highest accuracy levels. The deep learning model has remarkable performance. RNN, CNN, and ANN in the Drebin level of co-existence ANN and CNN have the highest accuracy and precision (91% to 98%), while Malgenome datasets have more accuracy than Drebin datasets. Logistic and RNN models are good at the dataset of permissions, with almost 80% to 88 accuracy and precision.

Our finding from the whole chapter we knows FP-Max will not get much more features of datasets but our finding FP-Max with approximate algorithm work remarkable with 99% accuracy in ANN model. Another the API datasets permission API is not capable to use with this algorithm FP-Max algorithm do not find the permissions datasets frequent items to detect malware. because In the introduction passage its limitations and some datasets FP-Max has no guarantee to find the itemsets. so here the set of permissions not found. These are our contribution in this chapter with experiment to detect the malware.

For instance [15], demonstrate their effectiveness in analyzing network traffic data for anomaly detection, highlighting their ability to identify subtle deviations from normal patterns indicative of potential threats. In another study [15] showcase their success in classifying Android apps as malicious or benign, emphasizing their interpretability as a valuable asset for understanding the factors contributing to their classifications.

.

.

# Conclusions and Suggestions

## Conclusion

Our research employed a combination of machine learning and deep learning techniques to address the dynamic co-existence of features. We utilized six different AI models, comprising three machine learning algorithms and three deep learning algorithms. These algorithms were further enhanced by incorporating GRU layers and optimized machine learning hyperparameters. Additionally, we explored two advanced data mining algorithms, FPM-IN and FPMAA, which offer user or researcher-controlled configuration options. The specific choice of algorithm depended on the research question being addressed. Our primary focus was to conduct an in-depth investigation into the dynamic co-existence of features and detection methods, an area ripe for further exploration. Beyond the core algorithms, we also employed a sample-based approach algorithm and native association rules. Finally, was implemented within the programming stage to identify dynamic co-existence. This comprehensive approach yielded new datasets, with each dataset file containing five new sub-datasets. In total, six dataset files were utilized, resulting in the generation of 30 new datasets. These 30 datasets were then evaluated using the six machine and deep learning models mentioned earlier

Our research delves into the power of co-existence features a novel approach to analyzing dynamic patterns within malware datasets. Drebin, and Malgenome datasets yielded intriguing findings. While Malgenome, combining API and permissions, exhibited the highest accuracy 99.2% in revealing hidden patterns, even Drebin API calls 99% at FPMA algorithm. combinations surpassed single features. By dissecting these co-existence features, we can expose the unreveal pattern of malware, leading to more effective identification and mitigation. Deep learning models employing co-existence features further boost accuracy in pinpointing hidden malware behavior.

Our research impact on malware detection using dynamic co-existence features the best results than pervious and better innovation despite it like old traditional methods but the equation and the idea of concept is unique these dynamic co-existence with level approaches and real time environment.

## Suggestions

In the future change the method use the heuristic approach in data minging. In this research read carefully it has multiple ideas for the future work. As data mining has various algorithm to change the idea make the equation use the laitance of the features it will gives the another research. Even modify this research another approach for data-mining dynamic co-existence with image process patterns these are the new research areas. We will find in future works co-existence of fault detection on space exploration software.

# Acknowledgements

I am deeply grateful to my supervisor, Dr. Xiaozhi Du, for their invaluable guidance and support throughout my research journey. Their critical feedback and encouragement were instrumental in shaping this thesis.

I would also like to express my sincere thanks to my committee members, Dr. [Name] and Dr. [Name], for their insightful suggestions and constructive criticism.

Additionally, I am grateful to [Name of organization] for providing me with a research grant, which enabled me to [mention how the grant helped your research].

Finally, I would like to thank my family and friends for their unwavering love and support during this challenging but rewarding experience

# References

1. Synk. [Online]. Available: https://snyk.io/blog/python-security-best-practices-cheat-sheet/.
2. E. O. A. Q. M. YASEEN, "A Novel Machine Learning Approach for Android Malware Detection Based on the Co-Existence of Features," *IEEE,* 2023.
3. L. T. R. P. R. Paula Branco, "A Survey of Predictive Modeling on Imbalanced Domains," *ACM,* 2016.
4. "Python Library," no. http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html.
5. Microsoft, "alerts," Microsoft, 2022. [Online]. Available: https://learn.microsoft.com/en-us/mem/configmgr/protect/deploy-use/endpoint-configure-alerts.
6. T. Doc, "paloaltonetworks," 2023. [Online]. Available: https://docs.paloaltonetworks.com/pan-os/10-2/pan-os-admin/zone-protection-and-dos-protection/zone-defense/take-baseline-cps-measurements-for-setting-flood-thresholds.
7. Cynet, "Malware Protection: 6 Technologies to Protect Your Organization," [Online]. Available: https://www.cynet.com/malware/malware-protection-6-technologies-to-protect-your-organization/.
8. "mlxtend," mlxtend python, [Online]. Available: https://rasbt.github.io/mlxtend/user\_guide/frequent\_patterns/fpgrowth/.
9. S. Documentation, "Mining Frequent Maximal Itemsets Using The FPMax Algorithm," [Online]. Available: https://www.philippe-fournier-viger.com/spmf/FPMax.php.
10. UNB, "Canada Institute for Cybersecurity," [Online]. Available: https://www.unb.ca/cic/datasets/maldroid-2020.html.
11. s. t. help, "Frequent Pattern (FP) Growth Algorithm In Data Mining," 2023. [Online]. Available: https://www.softwaretestinghelp.com/fp-growth-algorithm-data-mining/.
12. X. Y. J. H. C.-W. H. Hong Cheng, "Discriminative Frequent Pattern Analysis for Effective Classification," *IEEE,* pp. 4- 7, 2007.
13. 2. J. W. 1. XUEQIN ZHANG 1, "Detection of Android Malware Based on Deep Forest and Feature Enhancement," *IEEE,* 20 Feb 2023.
14. \*. A. 2. K. Y. 1. a. S. 3. Sohrab Mokhtari 1, "A Machine Learning Approach for Anomaly Detection in Industrial Control Systems Based on Measurement Data," *MDPI,* 2021.
15. R. R. R. R. Amuthan Prabakar Muniyandia, "Network Anomaly Detection by Cascading K-Means Clustering and C4.5 Decision Tree algorithm," *science direct,* 2021.
16. M. O. D. S. S. J. B. A. &. D. P. T. Roseline Oluwaseun Ogundokun, "A Novel PCA-Logistic Regression for Intrusion Detection System," *Springer Link,* 2023.
17. L. i. S. a. H. Xiaojun, "Android Malware Detection Based on Logistic Regression and XGBoost," *IEEE,* 2019.
18. C. C. F. T. L. W. Binghui Zou, "IMCLNet: A lightweight deep neural network for Image-based Malware Classification," *Journal of Information Security and Applications,* 2022.
19. C.-M. H. ·. M. Z. A. ·. H.-Y. H. ·. S. W. P. ·. J.-S. Leu1, "Robust Network Intrusion Detection Scheme Using Long-Short Term Memory Based Convolutional Neural Networks," in *Mobile Networks and Applications*, 2020.
20. C. R. H. S. S. Z. ,. Weiping Wang, "FGL\_Droid: An Efficient Android Malware Detection Method Based on Hybrid Analysis," *Research Article,* 2022.
21. M. A. c. S. W. d. H. N. e. B. S. f. Q. Z. Danish Vasan a b, "IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture," *Science Direct,* 2020.
22. A. F. A. ,. G. Asmaa Ahmed Awad, "An improved long short term memory network for intrusion detection," *Plos one,* 2023.
23. I. I. s. institute, "avtest," 2023. [Online]. Available: https://www.av-test.org/en/statistics/malware.
24. s. NEWS, "state of raansomeware 2023," 2023. [Online]. Available: https://news.sophos.com/en-us/2023/05/10/the-state-of-ransomware-2023/.
25. B. Y. &. S. M. S. M. A. A. S. E. Tokekar, "Deep Learning in Malware Identification and Classification," in *Malware analysis using artificial intellegence and deep learning*, Springer, p. 123 to 139.
26. Y. P. Leonid Popryho, "Behaviour-based detection of ransomware attacks in the Cloud using machine learning," *Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden,* 2023.
27. E. T. M. T. E. M. A. M. R. S. N. C. C. –. E. U. A. f. C. Ifigeneia Lella, ENISA THREAT LANDSCAPE 2023, EUROPEAN UNION AGENCY, 2023.
28. J. W. J. X. C. G. XUEQIN ZHANG, "Detection of Android Malware Based on Deep Forest and Feature Enhancement," *IEEE,* 2023.
29. S.-j. C. H. H. W. C. K. S. Hojun Lee, "Enhancing Sustainability in Machine Learning-based Android Malware Detection using API calls," *IEEE, publically availible,* 2022.
30. W. L. L. W. A. K. Fred Guyton, "Android Feature Selection based on Permissions, Intents, and API Calls," *IEEE,* 2022.
31. L. Huang, J. Xue, Y. Wang, D. Qu, J. Chen, N. Zhang and L. Zhang, "EAODroid: Android Malware Detection Based on Enhanced API Order," *IEEE,* 2023.
32. V. P. V. G. M. A. K. E. R. A. S. A. V. A. S. Anandhi V, "Malware Detection using Dynamic Analysis," *IEEE,* 2023.
33. J. P. S. Priya Raghuvanshi, "Android Malware Detection Using Machine Learning Techniques," *IEEE,* 2022.

# Appendix

附录编号依次编为附录A，附录B。附录标题各占一行，按一级标题编排。每一个附录一般应另起一页编排，如果有多个较短的附录，也可接排。附录中的图表公式另行编排序号，与正文分开，编号前加“附录A-”字样。

本部分内容非强制性要求，如果论文中没有附录，可以省略《附录》。

# Decision of Defense Committee

# General Reviewers List

本学位论文共接受X位专家评阅，其中常规评阅人X名，名单如下：

|  |  |  |
| --- | --- | --- |
| 王XX | 教授 | 西安交通大学 |
| 李XX | 教授 | XXXX大学 |
| 田XX | 教授 | XXXX大学 |

学位论文独创性声明（1）

本人声明：所呈交的学位论文系在导师指导下本人独立完成的研究成果。文中依法引用他人的成果，均已做出明确标注或得到许可。论文内容未包含法律意义上已属于他人的任何形式的研究成果，也不包含本人已用于其他学位申请的论文或成果。

本人如违反上述声明，愿意承担以下责任和后果：

1．交回学校授予的学位证书；

2．学校可在相关媒体上对作者本人的行为进行通报；

3．本人按照学校规定的方式，对因不当取得学位给学校造成的名誉损害，进行公开道歉。

4．本人负责因论文成果不实产生的法律纠纷。

论文作者（签名）： 日期： 年 月 日

学位论文独创性声明（2）

本人声明：研究生 所提交的本篇学位论文已经本人审阅，确系在本人指导下由该生独立完成的研究成果。

本人如违反上述声明，愿意承担以下责任和后果：

1．学校可在相关媒体上对本人的失察行为进行通报；

2．本人按照学校规定的方式，对因失察给学校造成的名誉损害，进行公开道歉。

3．本人接受学校按照有关规定做出的任何处理。

指导教师（签名）：日期： 年 月 日

学位论文知识产权权属声明

我们声明，我们提交的学位论文及相关的职务作品，知识产权归属学校。学校享有以任何方式发表、复制、公开阅览、借阅以及申请专利等权利。学位论文作者离校后，或学位论文导师因故离校后，发表或使用学位论文或与该论文直接相关的学术论文或成果时，署名单位仍然为西安交通大学。

论文作者（签名）： 日期： 年 月 日

指导教师（签名）： 日期： 年 月 日

(本声明的版权归西安交通大学所有，未经许可，任何单位及任何个人不得擅自使用)