
UFAM - Engenharia da Computação

# AVALIANDO MÉTODOS DE SELEÇÃO DE CARACTERÍSTICAS ESPECÍFICAS APLICADOS À DETECÇÃO DE MALWARES ANDROID

Luiza Paula Moreira Leão

Monografia de Graduação apresentada à Coordenação de Engenharia da Computação, UFAM, da Universidade Federal do Amazonas, como parte dos requisitos necessários à obtenção do título de Engenheiro da Computação.

Orientadores:
Eduardo Luzeiro Feitosa

Manaus

Junho de 2023

AVALIANDO MÉTODOS DE SELEÇÃO DE CARACTERÍSTICAS ESPECÍFICAS APLICADOS À DETECÇÃO DE MALWARES ANDROID

Luiza Paula Moreira Leão

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO CURSO DE ENGENHARIA DA COMPUTAÇÃO DA UNIVERSIDADE FEDERAL DO AMAZONAS COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO.

Aprovada por:

Prof. Eduardo Luzeiro Feitosa , Ph.D.

Prof. Eduardo James Pereira Souto, Ph.D.

Prof. César Augusto Viana Melo, Ph. D.

Manaus

Junho de 2023

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Leão, Luiza Paula Moreira
L437a Avaliando Métodos de Seleção de Características Específicas aplicados à Detecção de Malwares Android / Luiza Paula Moreira Leão . 2023
40 f.: 31 cm.

Orientador: Eduardo Luzeiro Feitosa
TCC de Graduação (Engenharia da Computação) - Universidade Federal do Amazonas.

1. Métodos de seleção. 2. Chamadas de api. 3. Permissões. 4. Modelos de Aprendizado. 5. Datasets. I. Feitosa, Eduardo Luzeiro. II. Universidade Federal do Amazonas III. Título

# Agradecimentos

Primeiramente à minha família, meus professores e amigos que estiveram ao meu lado durante esta caminhada.

Resumo da Monografia apresentada à UFAM como parte dos requisitos necessários para a obtenção do grau de Engenheiro

# AVALIANDO MÉTODOS DE SELEÇÃO DE CARACTERÍSTICAS ESPECÍFICAS APLICADOS À DETECÇÃO DE MALWARES ANDROID

Luiza Paula Moreira Leão

Junho/2023

Orientadores: Eduardo Luzeiro Feitosa

Curso: Engenharia da Computação

As soluções mais encontradas na literatura acadêmica atualmente para detecção de *malwares* em dispositivos com sistema Android envolve o uso de modelos de classificação baseados em aprendizado de máquina. Tipicamente, este tipo de solução demanda o treinamento de modelos através de conjuntos de dados (comumente chamados de datasets) que contém números significativos de amostras (e.g., 100k, 1M) e características (e.g., 3k, 500k). Contudo, avaliar esses *datasets* demandam muito tempo de computação e podem impactar a qualidade dos modelos. Para mitigar esses problemas, pesquisadores vêm propondo diferentes métodos para selecionar características específicas (como permissões e chamadas de API), reduzindo o tamanho do dataset sem perder a capacidade de identificação. Neste trabalho, propomos avaliar diferentes métodos recentes de seleção focados em características específicas, como, por exemplo, o SigPID (Sun *et al*., 2016), que reduz a dimensionalidade de permissões; e o SigAPI (Galib e Hossain, 2020), que reduz a dimensionalidade de chamadas de API.

Abstract of Monograph presented to UFAM as a partial fulfillment of the requirements for the degree of Engineer

# EVALUATING METHODS OF SPECIFIC FEATURE SELECTION APPLIED TO ANDROID MALWARE DETECTION

Luiza Paula Moreira Leão

June/2023

Advisors: Eduardo Luzeiro Feitosa

Course: Computer Engineering

The most common solutions found in current academic literature for detecting malware on Android devices involve the use of machine learning-based classification models. Typically, such solutions require training models on datasets containing a significant number of samples (e.g., 100k, 1M) and features (e.g., 3k, 500k). However, evaluating these datasets can be computationally time-consuming and can impact model quality. To mitigate these issues, researchers have proposed various methods for selecting specific features (such as permissions and API calls), reducing the dataset size without sacrificing identification capability. In this work, we propose to evaluate different recent methods for feature selection focused on specific characteristics, such as SigPID (Sun et al., 2016), which reduces the dimensionality of permissions, and SigAPI (Galib and Hossain, 2020), which reduces the dimensionality of API calls.

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Capítulo 1

# Introdução

Atualmente, o principal meio de detecção de *malwares* para sistemas operacionais Android é o emprego de modelos de aprendizado de máquina, que objetivam classificar aplicativos (APKs) em malignos e benignos (kakavand et al.,2018; Agrawal e Trivedi, 2021; Rana et al., 2018). Tais modelos são treinados e validados com conjuntos de dados, mais conhecidos como *datasets*, que contém amostras rotuladas (i.e., benigno ou maligno) de aplicativos Android. As amostras vêm acompanhadas de conjuntos de características, como permissões, intenções, chamadas de API, chamadas de sistema, tráfego de rede, componentes de hardware, componentes do aplicativo, comandos de sistema, bytecodes e strings semânticas (Damodaran et al., 2017; Shijo e Salim, 2015; Qiu et al., 2020), por exemplo.

Um dos principais desafios enfrentados na construção de modelos de detecção de *malwares* Android é a quantidade de dados que um *dataset* pode conter. Por exemplo, um dos maiores *datasets* já construídos para o treinamento de modelos de detecção de *malwares* Android contém mais de 1 milhão de amostras e mais de 1 milhão de características (Roy et al., 2015). Em termos computacionais, é demasiadamente custoso treinar e validar modelos com *datasets* dessa magnitude (e.g., matriz de 1 milhão por 1 milhão). Na prática, um *dataset* contendo apenas 6 mil amostras e 643 características pode consumir até 5,8 horas de computação em um computador moderno (Cai et al., 2021). Em outras palavras, trabalhar com *datasets* grandes torna-se inviável em termos de custo financeiro e tempo computacional. Sem falar que um treinamento contínuo de modelos precisa ser eficiente para tornar viável a detecção de *malwares* em tempo real (ou próximo disso).

Reduzir o tamanho de um *dataset* (e.g., reduzir a dimensão através da seleção das características mais significativas) faz parte de um conjunto de otimizações possíveis e necessárias para viabilizar e dar qualidade ao treinamento e a validação dos modelos.

Enquanto que, na literatura de aprendizado de máquina, há um conhecimento vasto sobre preparação dos dados (e.g., remoção de amostras duplicadas, adaptação de tipos de dados, remoção de outros ruídos) e otimização dos parâmetros dos algoritmos classificadores (e.g*., grid search* e *random search*), há uma quantidade menor de estudos que avaliam e comparam o impacto e os desafios dos diferentes métodos de seleção das características, como SigPID (Sun et al., 2016), SigAPI (Galib e Hossain, 2020) e *Robust Feature Generation* (RFG) (Moutaz, 2020), atividade importante para o desenvolvimento de modelos otimizados e viáveis na prática.

A seleção de características é tipicamente utilizada com o objetivo de reduzir o tamanho do *dataset*. Essa redução melhora o desempenho dos modelos, pois diminui o tempo de aprendizado (aumenta a velocidade) e aumenta a capacidade de generalização da classificação. Também os tornam melhores em termos de escalabilidade, já que a representação e compreensão dos dados é superior (Remeseiro e Bolon-Canedo,2019; Venkatesh e Anuradha, 2019; Golrang et al., 2021; Feizollah et al., 2015).

Técnicas de seleção de características têm sido utilizadas para auxiliar os modelos de detecção de *malwares* Android (Mahindru e Sangal, 2019; Xiao t al., 2019; Lee et al., 2021). Essas técnicas são implementadas através de métodos otimizados para encontrar o melhor e menor subconjunto de características no *dataset* capaz de reduzir a complexidade do processamento e melhorar o desempenho do modelo. Cada um desses métodos pode incorporar diversas técnicas de seleção (e.g., regressão linear, ganho de informação, chi-quadrado, análise de variância) para trabalhar com diferentes tipos de características. Por exemplo, o método SigAPI utiliza diferentes técnicas de seleção para reduzir entre 75% e 80% o número de características de *datasets* (Galib Hossain, 2020).

Um ponto importante é que os métodos de seleção de características existentes implementam e avaliam diferentes técnicas. Por exemplo, tanto o SigAPI quanto o RFG são utilizados para selecionar as características de **chamada de API** mais relevantes. Contudo, eles são comparados e avaliados somente com técnicas clássicas de seleção de características, como ganho de informação e qui-quadrado. Resumidamente, esses trabalhos demonstram o resultado positivo da seleção de características para um conjunto limitado de *datasets* (e.g., um único conjunto de dados) e consideram apenas

um sub-conjunto de técnicas clássicas de características na comparação, mas nunca foram comparados com outros métodos que selecionam características similares.

## 1.1 Objetivo Geral

Este Trabalho de Conclusão de Curso (TCC) visa avaliar, empírica e exploratoriamente, diferentes grupos de métodos de seleção de características (por exemplo, voltados para características como permissões, chamadas API, intenções, entre outros) recentes e significantes para detecção de *malwares* Android.

## 1.2 Objetivos Específicos

Especificamente, pretende-se:

- Identificar e utilizar os *datasets* originais de cada estudo para avaliações dos conjuntos de métodos;
- Definir métricas para avaliação dos métodos.

## 1.3 Organização do Trabalho

Esta pesquisa está organizada da seguinte forma: No capítulo 2 é apresentada a metodologia que foi usada para atingir os objetivos deste trabalho. Ainda no capítulo 2 é explicado o desenvolvimento do projeto, onde constam, de forma resumida e simplificada, os métodos a serem utilizados para a seleção das características e avaliação dos mesmos. No capítulo 3 são apresentados os resultados obtidos por cada um dos métodos de seleção e há uma explicação sobre os *datasets* utilizados. No capítulo 4 são feitas as considerações finais e as conclusões obtidas a partir dos experimentos realizados.

# Capítulo 2

# Conceitos e Metodologia

Este capítulo apresenta os principais conceitos e técnicas utilizadas para a elaboração da pesquisa, bem como a metodologia utilizada para alcançar os objetivos propostos.

## 2.1. Fundamentação

*Malwares* Android são programas maliciosos desenvolvidos para dispositivos móveis que executam o sistema operacional Android. Eles podem assumir várias formas e representam uma ameaça significativa, pois podem comprometer a privacidade do usuário, roubar informações pessoais e financeiras, além de causar danos ao dispositivo e ao sistema operacional.

A detecção de *malwares* Android é um desafio complexo devido à natureza dinâmica e evasiva dos *malwares*. Diversas abordagens têm sido propostas para enfrentar esse desafio, incluindo técnicas baseadas em assinaturas, comportamentais, heurísticas e de aprendizado de máquina. Via de regra, essas técnicas exploram características específicas dos aplicativos para identificar comportamentos maliciosos ou padrões de código suspeitos.

No contexto da detecção de *malwares* Android, a seleção de características é um processo de identificação das características mais relevantes e discriminativas para uma tarefa específica, visando escolher as que melhor diferenciam aplicativos maliciosos dos benignos. Essa etapa é crucial para reduzir a dimensionalidade dos dados e melhorar a eficiência e eficácia dos algoritmos de detecção.

Existem diversas formas de selecionar características para a detecção de *malwares* Android. Uma delas é analisando a importância de características, atribuindo uma pontuação ou ranking à elas com base na sua importância para a detecção de *malwares*. Algoritmos como ganho de informação (*Information Gain*), qui-quadrado ou coeficiente

de correlação podem ser utilizados para realizar essa análise. Outra forma é aplicando filtros, onde se avaliam as características independentemente do classificador utilizado. Métricas estatísticas, como a correlação de Pearson ou a análise de componentes principais *(PCA)*, podem ser aplicadas para selecionar as características mais relevantes. Por fim, empregar aprendizado de máquina, utilizando algoritmos como árvores de decisão ou redes neurais para selecionar e classificar as características mais relevantes, é outra forma encontrada para selecionar características para a detecção de *malwares* Android.

É evidente a importância da avaliação de métodos de seleção de características específicas aplicados à detecção de *malwares* Android. A escolha adequada dos métodos de seleção de características pode contribuir para a identificação precisa de aplicativos maliciosos, fortalecendo a proteção dos dispositivos móveis e a segurança dos usuários.

## 2.2. Metodologia da Pesquisa

Para atingir os objetivos propostos, este projeto segue uma metodologia de pesquisa baseada em quatro etapas distintas, conforme a Figura 1.

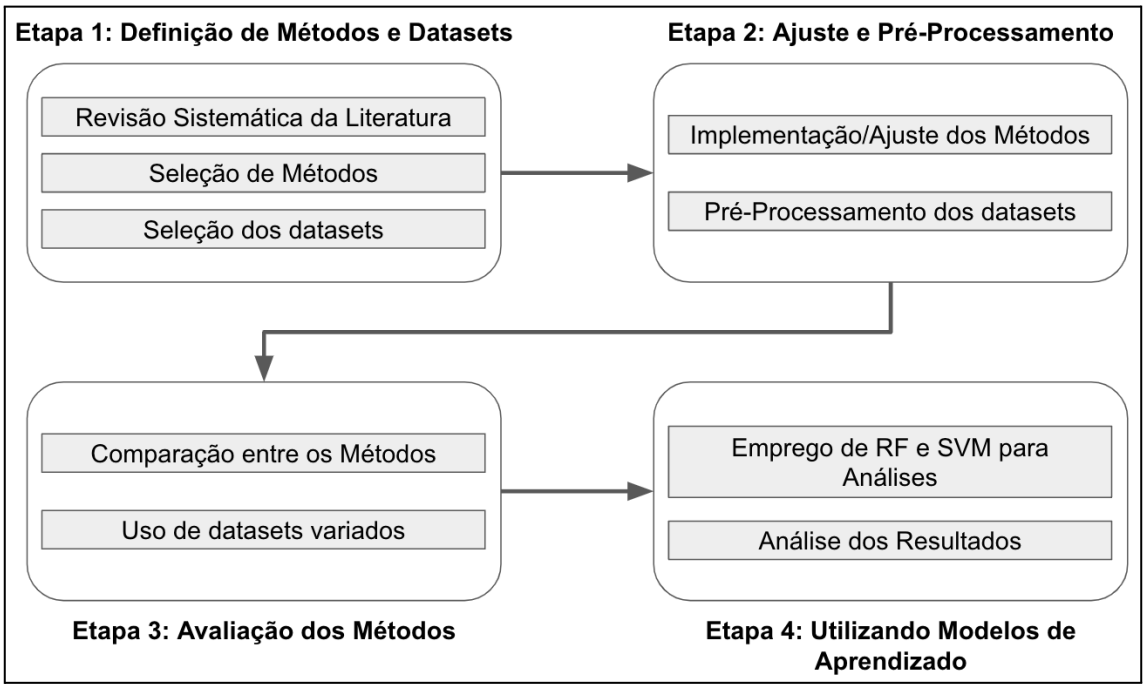


**Figura 1. Etapas da metodologia**

A **primeira etapa** consistiu de uma revisão sistemática da literatura (RSL), que desempenhou um papel crucial na definição dos métodos de seleção a serem avaliados e dos *datasets* a serem utilizados nos experimentos. Essa revisão envolveu a busca por estudos relevantes, como artigos científicos, conferências, teses e dissertações, que apresentem métodos de seleção e descrevam a implementação e avaliação desses métodos. Durante a revisão sistemática, foi considerada a disponibilidade do código fonte ou a descrição da implementação dos métodos. Isso permitiu que fosse possível replicar os experimentos e compreender detalhadamente como cada método funciona. A disponibilidade do código fonte também facilita a comparação e a reprodutibilidade dos resultados obtidos em diferentes estudos. Além disso, foi necessário levar em conta a disponibilidade dos *datasets* utilizados na avaliação dos métodos.

A **segunda etapa** do processo envolveu a realização de ajustes, configurações ou até mesmo a implementação dos métodos de seleção de características escolhidos, caso não existisse o código fonte disponível, mas apenas a descrição do algoritmo. Essa etapa foi essencial para garantir que os métodos pudessem ser executados corretamente nos experimentos. Se o código fonte dos métodos estivesse disponível, foi preciso analisar e compreender o funcionamento do algoritmo. Nesse caso, foram realizados ajustes ou configurações nos parâmetros do método, com base na RSL e no conhecimento de outros pesquisadores. Esses ajustes adaptaram o método às características específicas dos *datasets* selecionados e otimizaram os desempenhos. No entanto, se o código fonte dos métodos não estivesse disponível, mas apenas a descrição do algoritmo, foi necessário implementá-lo.

Além dos ajustes e implementações dos métodos de seleção de características, também foi fundamental realizar o pré-processamento dos *datasets* antes de aplicá-los aos métodos. O pré-processamento visou evitar resultados enviesados e garantir que os dados estivessem em formato adequado para a análise, o que envolveu a limpeza dos dados, remoção de amostras que possuíam pelo menos um valor faltante. Isso foi importante para garantir a integridade e a qualidade dos dados, evitando que valores ausentes influenciassem indevidamente nos resultados dos métodos de seleção.

A **terceira etapa** consistiu na avaliação dos métodos de seleção selecionados em comparação com diversas técnicas de seleção. Essa etapa envolveu a utilização dos *datasets* empregados na comprovação dos métodos, bem como a incorporação de *datasets* famosos e disponíveis na literatura, como o Androcrawl e o Drebin 215. Para realizar a avaliação, foram utilizadas diferentes técnicas de seleção, incluindo a *Recursive Feature Elimination* (RFE), *Mutual Information Gain*, Análise de Variância (ANOVA), qui-quadrado e SelectKBest com qui-quadrado. Essas técnicas são comumente empregadas na seleção de características e foram escolhidas para permitir uma comparação abrangente dos métodos selecionados.

Na **etapa final** da metodologia, os *datasets* gerados com as características selecionadas pelos métodos de seleção foram utilizados para treinar modelos de aprendizado de máquina. Especificamente, os algoritmos *Random Forest* (RF) e *Support Vector Machine* (SVM) foram escolhidos como classificadores de referência para avaliar a qualidade dos métodos de seleção de características em conjuntos de dados distintos. A avaliação dos modelos de RF e SVM nos *datasets* gerados com as características selecionadas permitiu analisar a qualidade dos métodos de seleção. Comparando os resultados obtidos com diferentes conjuntos de dados de entrada, foi possível avaliar a capacidade dos métodos em selecionar características relevantes que contribuem para o desempenho dos modelos de aprendizado de máquina.

Essa etapa final do processo fornece uma visão abrangente sobre a eficácia dos métodos de seleção de características, considerando diferentes conjuntos de dados e aplicando modelos de referência populares como RF e SVM. Os resultados obtidos permitiram identificar os métodos mais eficazes e apropriados para a tarefa de seleção de características, além de fornecer informações valiosas para a escolha de características relevantes em futuras aplicações de aprendizado de máquina.

## 2.3. Métodos de Seleção

Com base na RSL realizada nesta pesquisa, foram selecionados quatro (4) métodos de seleção de características. Como será observado nesta seção, cada método possui suas particularidades e foi projetado e avaliado sobre um grupo específico de características e um conjunto limitado de *datasets*.

### 2.3.1 SigAPI

O SigAPI (Galib and Hossain, 2020) utiliza uma combinação de técnicas para seletivamente identificar as chamadas de API mais significativas para a detecção de *malwares* Android. O método funciona em duas etapas. Na primeira etapa, o SigAPI aplica e avalia diversas técnicas básicas de seleção, como *Mutual Information Gain*, *Based on Univariate ROC-AUC Score*, *Recursive Feature Elimination with Gradient Boosting Classifier*, *Recursive Feature Elimination with Random Forest Classifier*, *SelectKBest with x2* e *SelectFromModel*. O melhor resultado entre elas é selecionado para a segunda etapa do método.

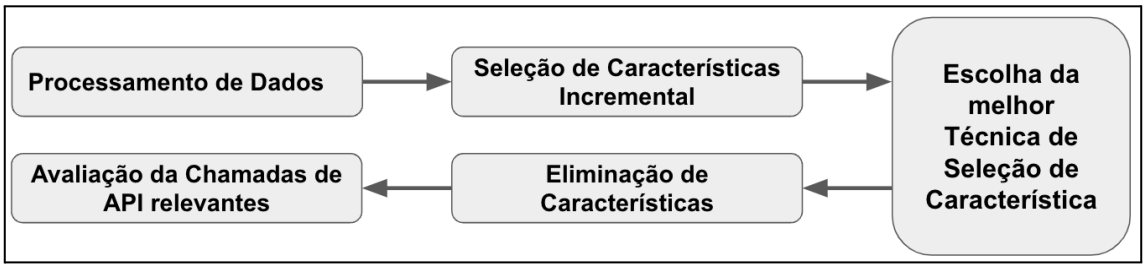


**Figura 2. Etapas do SigAPI.**

A segunda etapa do SigAPI é implementada utilizando o coeficiente de correlação de Pearson, que é aplicado para todas as combinações de pares de chamadas de API derivadas do melhor resultado da primeira etapa. Os pares que apresentam correlação maior que 0,85 são encaminhados para um sub-processo de eliminação de características, que é baseado em um estimador implementado através do algoritmo RF. O subprocesso procura determinar a importância relativa da característica para a classificação, eliminando aquela que for considerada a menos importante e reduzindo ainda mais a quantidade de chamadas de API do *dataset*.

Como as duas características em cada par são fortemente correlacionadas, o pressuposto do método é que a remoção daquela de menor importância não afeta o desempenho da classificação dos modelos. Resultados experimentais indicam que método é capaz de reduzir em mais de 75% o número de chamadas de API, para *datasets* como o *Android Malware Genome Project* (Mal Genome) (Zhou and Jiang, 2012), sem prejudicar o desempenho dos modelos de detecção de *malwares*.

### 2.3.2 SigPID

O SigPID (Sun *et al.*, 2016) utiliza três níveis de poda para reduzir o número de permissões e opera com duas matrizes, sendo uma de permissões utilizadas por amostras de *malwares* e outra de características utilizadas por aplicativos benignos.

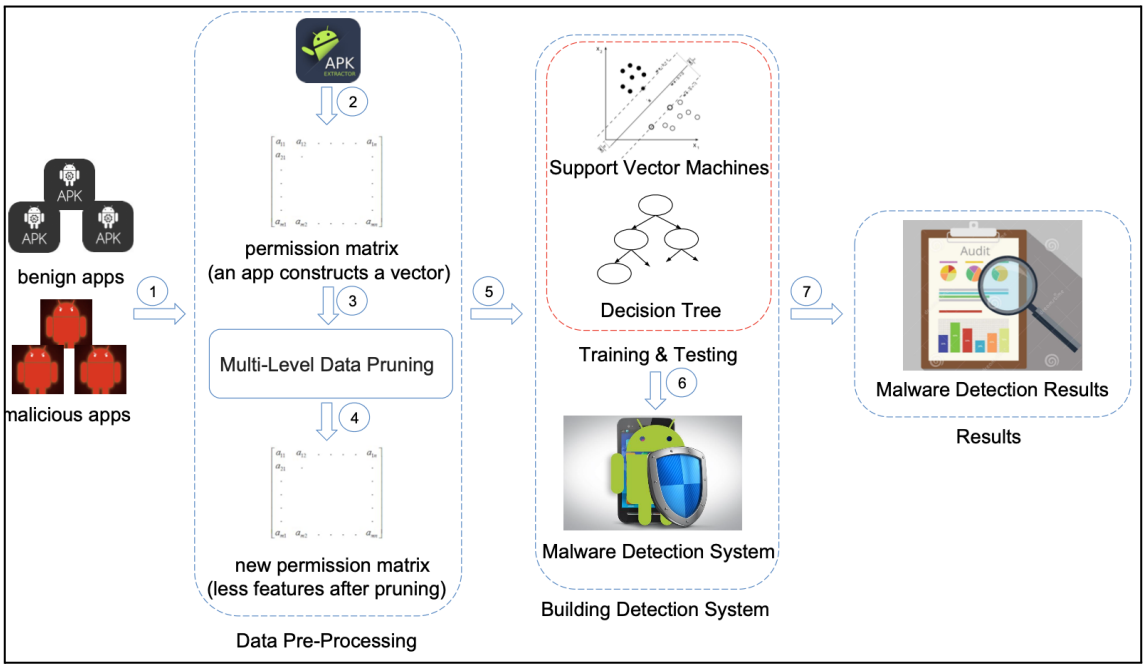


**Figura 3. Etapas do SigPID** (Retirada do artigo original).

No primeiro nível, o SigPID classifica permissões com taxa negativa através de um sistema incremental que utiliza o SVM. O objetivo nesta etapa é alcançar a melhor acurácia com um número reduzido de permissões, removendo as que geram ambiguidade ao modelo (e.g., INTERNET, CÂMERA). No artigo, os autores relatam que o primeiro nível de corte possibilitou reduzir o *dataset* de 135 para 95 permissões.

No segundo nível ocorre a classificação de permissões com base em suporte, que busca avaliar a recorrência de uma permissão no intervalo entre 0 e 1. Uma recorrência muito baixa (e.g., 0) indica que seu impacto será baixo no desempenho do modelo, isto é, essas permissões podem ser excluídas do *dataset*. No artigo foram eliminadas 70 permissões, restando apenas 25 para o terceiro nível de poda.

Finalmente, no terceiro nível ocorre a mineração de permissões com regras de associação, através do algoritmo apriori, com os parâmetros de 96,5% de confiança mínima e 10% de suporte mínimo. O objetivo é identificar as permissões que possuem

maior probabilidade de estarem associadas (por exemplo, **WRITE SMS** e **READ SMS**). A permissão com o menor suporte é eliminada. Como exemplo, a permissão **WRITE SMS** é considerada menos relevante que **READ SMS**. Ao final do terceiro nível, os autores chegaram a apenas 22 permissões, isto é, uma redução de 83,70% quando consideramos o *dataset* original, que contém 135 permissões.

### 2.3.3 RFG

O método *Robust Feature Generation* (RFG) (Moutaz, 2020) é composto de duas etapas. Na primeira etapa, o RFG utiliza o qui-quadrado e o ANOVA para selecionar as *n* melhores características, em que *n* pertence a um conjunto definido pelo usuário, que serão avaliadas na etapa seguinte. O qui-quadrado e o ANOVA são funções que associam uma pontuação para cada característica. Como resultado, características altamente dependentes na classe da amostra (*malware* ou benigno) recebem uma pontuação alta, criando um *ranking* de importância das características.

A segunda etapa do RFG consiste em uma avaliação das *n* melhores características com os algoritmos *Naive Bayes*, *k-Nearest Neighbors* (KNN), *Random Forest*, J48, *Sequential Minimal Optimization* (SMO), *Logistic Regression*, AdaBoost *decision-stump*, *Random Committee*, JRip e *Simple Logistics*. Cada um dos modelos é testado para cada valor de *n* em validação cruzada. O objetivo desta etapa é encontrar o menor valor de *n* cujo desempenho dos modelos ainda seja considerado aceitável, isto é, fique acima de 90% de acurácia.

Em uma avaliação empírica utilizando um *dataset* com 36.915 amostras (19000 benignas e 17915 malignas) e 9000 características e o conjunto {10, 25, 50, 100, 200, 300, 500, 1000, 3000, 5000, 7000, 9000} de valores para *n* (Moutaz, 2020), os autores demonstraram que o modelo foi capaz de manter uma acurácia 98,1% com apenas 500 características, o que representa uma redução de 94,44% no tamanho original do *dataset*.

### 2.3.4 SemiDroid

No método SemiDroid são consideradas permissões como características que ajudam no desenvolvimento de um modelo de detecção de *malware*. Para selecionar as características ou conjuntos de características apropriados são implementadas seis abordagens distintas de seleção de características, chamadas de técnicas de *ranking* e outros quatro métodos chamados de técnicas de subconjuntos ou subseleção.

As técnicas de *ranking* são:

- *Gain-Ratio Feature Selection*, uma medida que combina os conceitos de ganho de informação e taxa de divisão, levando em consideração a relevância e a independência das características do conjunto de dados;
- *Chi-Squared Test*, teste estatístico utilizado para determinar se existe uma associação significativa entre duas variáveis categóricas. É aplicado para verificar se há uma diferença significativa entre as frequências observadas e as frequências esperadas de cada categoria das variáveis;
- *Information-gain Feature Selection*, baseado no conceito de entropia, mede a incerteza ou a impureza de um conjunto de dados, ajudando a reduzir a dimensionalidade dos dados, melhorando a eficiência computacional e evitando a inclusão de características irrelevantes;
- *OneR Feature Selection*, método de seleção de características que busca identificar a característica mais relevante para uma tarefa de classificação. Tem uma abordagem simples e de fácil interpretação, que se baseia em criar um modelo de regra única para cada característica e selecionar aquela que resulta no melhor desempenho de classificação;
- *Principal Component Analysis*, técnica estatística utilizada para transformar um conjunto de variáveis correlacionadas em um novo conjunto de variáveis não correlacionadas chamadas de componentes principais. Os componentes principais são calculados de forma que o primeiro componente principal explique a maior parte da variabilidade dos dados; o segundo componente principal explique a segunda maior parte; e assim por diante;
- *Logistic Regression Analysis*, que aplica a função logística à combinação linear das variáveis independentes, com o objetivo de estimar a probabilidade de um evento ou resultado ocorrer com base nos valores das variáveis independentes. É

um método estatístico usado para modelar a relação entre uma variável dependente categórica e uma ou mais variáveis independentes.

As técnicas de subconjunto são:

- *Correlation Based Feature Selection*, que busca encontrar as características que possuem uma forte relação com a variável alvo, enquanto minimiza a presença de características redundantes. É um método utilizado para identificar as características mais relevantes ou importantes em um conjunto de dados, com base na medida de correlação entre as características e a variável alvo;
- *Rough Set Analysis*, se baseia na ideia de que a informação de um conjunto de dados pode ser dividida em informações precisas e informações aproximadas. Lida com a imprecisão dos dados, permitindo a identificação de padrões, regularidades e dependências entre as características;
- *Consistency Subset Evaluation Approach*, onde a ideia é que um subconjunto de características é considerado consistente se todas as instâncias com os mesmos valores para essas características pertencerem à mesma classe. Visa encontrar um equilíbrio entre a relevância das características e a consistência com a classe de destino. É útil para reduzir a dimensionalidade dos dados, remover características redundantes e melhorar a eficiência dos modelos de classificação;
- *Filtered Subset Evaluation*, uma abordagem utilizada para selecionar características relevantes em conjuntos de dados. As características são avaliadas e classificadas independentemente de um modelo de aprendizado, utilizando critérios estatísticos ou heurísticos, permite a identificação rápida de características relevantes, reduzindo a dimensionalidade dos dados e melhorando a eficiência de algoritmos de aprendizado subsequentes.

No entanto, para nossos experimentos foram utilizadas apenas as técnicas de *ranking,* já que os modelos de subconjuntos são aplicáveis apenas aos conjuntos de dados utilizados no experimento original, aos quais o acesso não é disponibilizado.

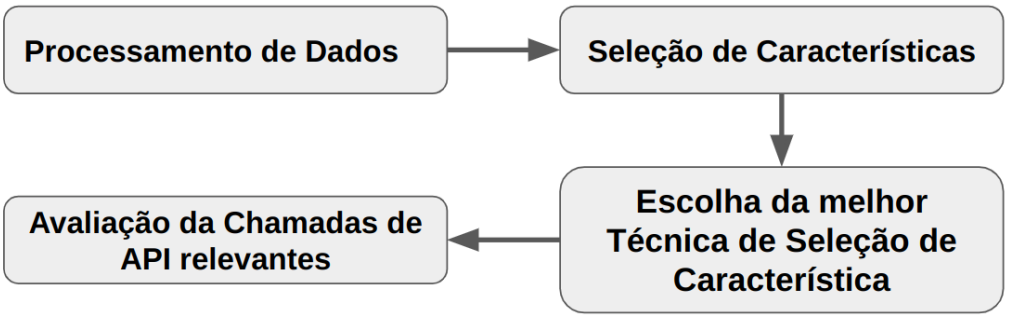


**Figura 4. Etapas do SemiDroid**

Ainda no experimento original, são implementados cinco modelos de aprendizado de máquina para a classificação das técnicas: *Self-organizing maps (SOM), K-mean, Farthest first, Filtered cluster* e *Density-based cluster.*

Os resultados empíricos dos autores revelam que o modelo construído considerando a análise de conjuntos aproximados como uma abordagem de seleção de características *Rough set analysis* e o algoritmo de aprendizado *Farthest first clustering* alcançaram a taxa de detecção mais alta de 98,8% para detectar *malware* em aplicativos do mundo real.

### 2.3.5 Breve Comparação

A Tabela 1 apresenta um resumo sobre as técnicas de seleção que foram implementadas. Vale destacar que os quatro trabalhos utilizam *datasets* próprios, geralmente criado a partir de outros conjuntos dados, como o Drebin e o Androcrawl, e outras fontes de aplicativos, como Google Play e APKPure.

**Tabela 1. Informações sobre os modelos de seleção**

| Método | Características | Técnicas de Seleção | Redução (%) |
|---|---|---|---|
| SigAPI | Chamadas de API | *Mutual InformationGain, Univariate ROC-AUC Score, Recursive Feature Elimination with Gradient Boosting Classifier, Recursive Feature Elimination with Random Forest Classifier, Select KBest with Chi2, SelectFrom-Model (Tree-based)*, Correlação | 65,8 - 79,46<br>63,77 - 78,28 |
| SigPID | Permissões | *Permission Ranking with Negative Rate, Support Based Permissions Ranking, Permission Mining* | 83,7 |

| | | | |
|---|---|---|---|
| | | *with Association Rules* | |
| RFG | Chamadas de API | qui-quadrado, ANOVA e algoritmos de aprendizado de máquina | 94.44 |
| SemiDroid | Permissões | *Gain-ratio*, qui-quadrado, I*nformation Gain, OneR*, PCA, *Logistic regression, Correlation based, Rough set analysis, Consistency subset e Filtered subset* | - |

Observa-se, na Tabela 1, que o método SemiDroid não apresenta dados sobre a sua capacidade de redução, conforme o artigo original. No entanto, segundo os autores, a capacidade de detecção de *malwares* da técnica chega a 98,8%. Outro ponto a ser observado na Tabela é a variedade de técnicas que podem ser utilizadas para a redução e seleção de características. A divergência entre as técnicas de seleção e os resultados de redução destacam a variedade de abordagens utilizadas na literatura. Além disso, o uso de conjuntos de dados diferentes, em cada estudo, dificulta ainda mais a comparação dos resultados. Portanto, a escolha de utilizar os mesmos três conjuntos de dados para todos os quatro métodos neste estudo tem como objetivo estabelecer uma base de comparação mais consistente e justa. Isso nos permitirá avaliar de forma mais precisa e confiável a eficácia e as diferenças entre esses métodos de seleção.

# Capítulo 3

# Protocolo e Resultados

Neste capítulo são apresentados os protocolos experimentais e os resultados obtidos a partir das técnicas e métodos implementados. Serão discutidas a capacidade de redução de cada técnica de seleção de características e também os resultados de suas métricas, a partir do algoritmo de aprendizado de máquina aplicado.

## 3.1 Protocolo Experimental

Como protocolo neste estudo, avaliamos quatro métodos de seleção de características em três conjuntos de dados distintos para a detecção de *malwares* Android. Realizamos uma limpeza nos *datasets*, removendo amostras com valores faltantes, duplicatas e características redundantes. Utilizamos os modelos RF e SVM para avaliar a qualidade das reduções de características. Métricas como acurácia, precisão, *recall*, F1-score e curva ROC-AUC foram utilizadas para analisar o desempenho dos modelos. Esses processos experimentais forneceram revelações sobre a capacidade dos métodos em lidar com diferentes conjuntos de dados e identificar padrões relevantes. Os resultados contribuirão para avanços na pesquisa de detecção de malwares e desenvolvimento de abordagens mais eficientes para seleção de características.

### 3.1.1 Datasets

Neste trabalho utilizamos três *datasets* publicamente disponíveis (Drebin 215[1], AndroCrawl[2] e Adroit[3]), o que é essencial para a reprodutibilidade do trabalho. Dos dois primeiros, selecionamos apenas as categorias de características consideradas nos trabalhos originais dos quatro métodos de seleção de características avaliados, ou seja, permissões e chamadas de API. Já no Adroit, as características utilizadas foram apenas as permissões, que são as únicas características presentes no *dataset*. Nosso objetivo foi

[1] https://figshare.com/articles/dataset/Android_malware_dataset_for_machine_learning_2/5854653
[2] https://github.com/phretor/ransom.mobi/blob/gh-pages/f/filter.7z
[3] https://www.kaggle.com/datasets/saurabhshahane/android-malware-dataset

eliminar características contidas nesses *datasets* que originalmente não foram consideradas pelos métodos de seleção, como, por exemplo, intenções e *opcodes*. É importante destacar que as permissões e chamadas de API estão entre as características consideradas mais relevantes para detecção de *malwares* Android (Wang *et al.*, 2019).

A Tabela 2 apresenta os detalhes dos *datasets* derivados e utilizados nos experimentos. Como pode ser observado, é apresentado o tipo e o número de características, bem como o número de amostras de cada *dataset*. Adicionalmente, a Tabela apresenta também a proporção entre amostras benignas e malignas (**B:M**).

**Tabela 2. Informações sobre os datasets**

| | Características | | | |
|---|---|---|---|---|
| **Dataset** | **Tipo** | **Número** | **Amostras** | **B:M** |
| adroit | Permissões | 166 | 11476 | 2,4:1 |
| androcrawl_permissions | Permissões | 49 | 96744 | 8,5:1 |
| androcrawl_api_calls | Api Calls | 24 | | |
| drebin_215_permissions | Permissões | 113 | 15031 | 1,7:1 |
| drebin_21_api_calls | Api Calls | 73 | | |

Além do agrupamento das características utilizadas nos experimentos, os *datasets* passaram também por uma limpeza, ou pré-processamento, uma etapa importante para a utilização do dataset em modelos de aprendizado de máquina. O processo de limpeza dos *datasets* incluiu (Han and Kamber, 2006):

- Remoção de amostras com um ou mais valores faltantes;
- Remoção de amostras duplicadas, isto é, amostras contidas repetidas vezes no *dataset*;
- Remoção de características que possuem o mesmo valor em todas as amostras (e.g., apenas 0s ou 1s);
- Conversão dos tipos de dados para o tipo numérico inteiro.

### 3.1.2 Algoritmos de aprendizado de máquina

Utilizamos os modelos classificadores Random Forest *(RF)* e Support Vector Machine *(SVM)* para verificar a qualidade dos resultados das reduções de características realizadas pelos métodos. A escolha do RF e SVM é pelo fato de eles serem

considerados os modelos mais frequentemente utilizados no domínio de detecção de malwares Android (Sharma and Rattan, 2021).

O RF é um algoritmo de aprendizado de máquina que combina múltiplas árvores de decisão para realizar classificações. Ele é capaz de lidar com grandes conjuntos de dados e pode fornecer uma boa precisão na classificação. Ao aplicar a redução de características, podemos remover características menos importantes e melhorar o desempenho do modelo. O SVM é um algoritmo de aprendizado supervisionado que mapeia os dados em um espaço dimensional superior e encontra um hiperplano que separa as classes. O SVM é conhecido por ser eficaz em conjuntos de dados de alta dimensionalidade. Ao realizar redução de características, podemos reduzir a dimensionalidade do problema e melhorar a eficiência computacional do algoritmo.

Ao utilizar esses dois modelos classificadores, podemos comparar a qualidade dos resultados obtidos antes e depois da redução de características.

### 3.1.3 Métricas utilizadas

Neste trabalho, iremos utilizar métricas de avaliação, como a acurácia, precisão, *recall*, F1-score e ROC-AUC, para determinar se a redução de características teve um impacto positivo nos resultados dos modelos.

A acurácia mede a proporção de exemplos classificados corretamente em relação ao total de exemplos (Equação 1). É uma métrica bastante utilizada, principalmente em problemas de classificação binária ou multiclasse, e fornece uma visão geral do desempenho geral do modelo.

$$Acurácia = \frac{VP + VN}{VP + FN + VN + FP} \qquad \text{Equação (1)}$$

A precisão é a proporção de exemplos positivos classificados corretamente em relação ao total de exemplos classificados como positivos (Equação 2). Essa métrica é útil quando o foco é minimizar os falsos positivos, ou seja, evitar classificar incorretamente exemplos negativos como positivos.

$$Precisão = \frac{VP}{VP + FP} \qquad \text{Equação (2)}$$

O *recall*, também conhecido como sensibilidade, é a proporção de exemplos positivos classificados corretamente em relação ao total de exemplos que realmente são positivos (Equação 3). O *recall* é relevante quando o objetivo é minimizar os falsos negativos, ou seja, evitar classificar incorretamente exemplos positivos como negativos.

$$Recall = \frac{TP}{TP + FN} \quad \text{Equação (3)}$$

O F1-score é a média harmônica da precisão e do *recall (Equação 4)*. Ele combina essas duas métricas para fornecer uma medida balanceada entre a precisão e o recall. É especialmente útil quando o conjunto de dados está desequilibrado em relação às classes.

$$F1\ Score = \frac{2\ x\ (Precisão\ x\ Recall)}{Precisão\ x\ Recall} \quad \text{Equação (4)}$$

A curva ROC (*Receiver Operating Characteristic*) e a métrica AUC (*Area Under the Curve*) são utilizadas principalmente para avaliar o desempenho de modelos de classificação binária. A curva ROC mostra a taxa de verdadeiros positivos em relação à taxa de falsos positivos em diferentes limiares de classificação, enquanto a métrica AUC quantifica a área sob a curva ROC (Equação 5). Quanto maior o valor de AUC, melhor a performance do modelo em distinguir entre as classes.

$$TPR = \frac{TP)}{TP + FN} \quad \text{Equação (5)}$$

Ao medir essas métricas antes e depois da redução de características, é possível determinar se essa redução teve um impacto positivo nos resultados dos modelos. Por exemplo, se a acurácia, precisão, *recall*, F1-score e/ou ROC-AUC melhorarem após a redução de características, isso indica que a redução foi eficaz e contribuiu para a melhoria dos modelos.

### 3.1.4 Experimentação

A ilustração apresentada na Figura 5 descreve toda a experimentação.

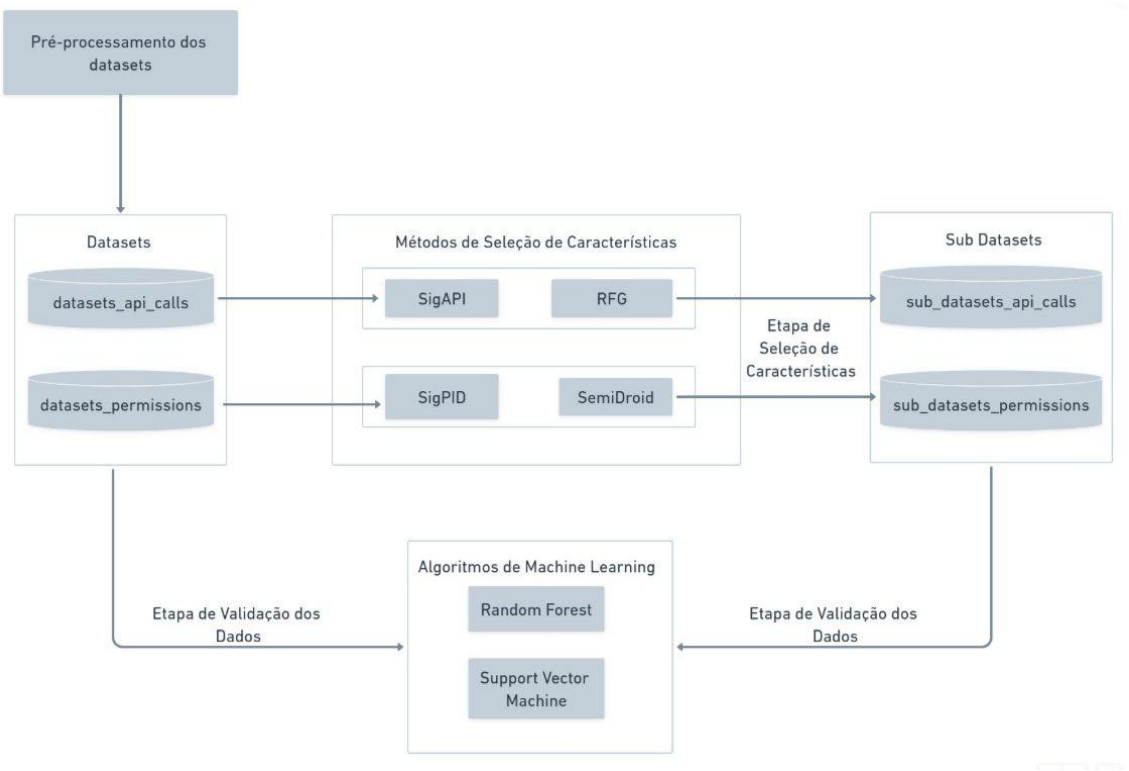


**Figura 5. Etapas do experimento**

Os *datasets* mostrados na Tabela 2 são utilizados como entrada para os quatro métodos escolhidos. Cada método, para cada conjunto de dados de entrada, produz um novo conjunto de dados contendo apenas as características selecionadas. Esses novos subconjuntos são então empregados em algoritmos de aprendizado de máquina para avaliar as métricas resultantes.

Na etapa final, todos os conjuntos de dados derivados são utilizados em modelos de aprendizado de máquina que se baseiam nos algoritmos RF e SVM. Essa escolha é feita devido à ampla utilização desses modelos no campo da detecção de *malwares* em dispositivos Android.

A partir dos resultados das métricas obtidas pelos dois modelos, analisamos a qualidade dos métodos de seleção de características em diferentes conjuntos de dados de entrada. Além disso, é possível comparar as diferenças entre as métricas resultantes e as métricas dos conjuntos de dados originais, ou seja, antes da redução.

Essa comparação permite verificar se a redução de dimensionalidade afetou negativamente ou positivamente o desempenho dos modelos de aprendizado de máquina. Se as métricas dos conjuntos de dados derivados forem consistentemente próximas ou melhores do que as métricas dos conjuntos de dados originais, isso indicaria que os métodos de seleção de características foram eficazes na identificação e preservação das informações relevantes para a detecção de *malwares* Android.

No entanto, se as métricas dos conjuntos de dados derivados apresentarem uma diminuição significativa em relação às métricas dos conjuntos de dados originais, seria necessário investigar a causa desse impacto negativo. Pode ser que os métodos de seleção de características tenham removido características importantes ou que a redução de dimensionalidade tenha causado perda de informação relevante para a detecção de *malwares*. Nesse caso, seria necessário reavaliar os métodos utilizados ou explorar outras técnicas de seleção de características que possam mitigar esses efeitos indesejados.

A análise comparativa das métricas também pode revelar conclusões sobre as diferenças de desempenho entre os modelos RF e SVM no contexto da detecção de *malwares* Android. A comparação permite avaliar qual modelo se destaca em termos de precisão, eficiência e capacidade de lidar com conjuntos de dados reduzidos.

# 3.2 Resultados

Esta seção apresenta os resultados obtidos ao aplicarmos os métodos de seleção nos conjuntos de dados de chamadas de API e permissões. Discutimos os efeitos desses métodos na seleção das características mais relevantes nos *datasets* de chamadas de API e permissões. Vamos examinar como esses métodos impactaram os resultados e avaliar se houve melhorias na qualidade das características selecionadas.

## 3.2.1 Validação dos Métodos de Seleção de Chamadas de API

A Tabela 3 apresenta os resultados obtidos pelos algoritmos de aprendizado de máquina quando aplicados no conjunto de dados original.

**Tabela 3. Resultado dos Classificadores nos *Datasets* originais de Chamada de API.**

| RF | Acurácia | Precisão | Recall | F1 Score | Roc auc |
|---|---|---|---|---|---|
| Androcrawl_api_calls | 97,61 | 93,57 | 83,05 | 87,99 | 91,19 |
| Drebin_215_api_calls | 97,98 | 98,57 | 95,93 | 97,24 | 97,56 |
| **SVM** | **Acurácia** | **Precisão** | **Recall** | **F1 Score** | **Roc auc** |
| Androcrawl_api_calls | 97,58 | 94,99 | 81,29 | 87,61 | 90,40 |
| Drebin_215_api_calls | 97,14 | 97,76 | 94,44 | 96,07 | 96,58 |

Os algoritmos RF e SVM foram aplicados nos conjuntos de dados *androcrawl_api_calls* e *drebin_215_api_calls*. No *androcrawl_api_calls*, ambos obtiveram valores elevados e parecidos para acurácia, precisão, *recall*, F1-Score e ROC AUC. O mesmo comportamento é observado em relação ao *drebin_215_api_calls*, onde também foram percebidos resultados similares entre as métricas para os dois classificadores.

Na Tabela 4 temos os dados obtidos, após aplicarmos os métodos de seleção nos conjuntos de dados originais.

**Tabela 4. Resultados do SigAPI e RFG.**

| RF | Acurácia | Precisão | Recall | F1 Score | Roc auc | Features selecionadas | Redução(%) |
|---|---|---|---|---|---|---|---|
| **Androcrawl_api_calls** | | | | | | | |
| SigAPI | 95,87 | 95,88 | 63,45 | 76,37 | 81,56 | 1 | 95,84 |
| RFG | 57,88 | 15,52 | 67,66 | 25,24 | 62,19 | 21 | 12,5 |
| **Drebin_215_api_calls** | | | | | | | |
| SigAPI | 94,95 | 93,92 | 92,31 | 93,11 | 94,40 | 15 | 79,46 |
| RFG | 97,26 | 97,63 | 94,89 | 96,24 | 96,77 | 41 | 43,84 |
| **SVM** | **Acurácia** | **Precisão** | **Recall** | **F1 Score** | **Roc auc** | **Features selecionadas** | **Redução(%)** |
| **Androcrawl_api_calls** | | | | | | | |
| SigAPI | 95,87 | 95,88 | 63,45 | 76,37 | 81,56 | 1 | 95,84 |
| RFG | 79,07 | 14,38 | 20,00 | 16,73 | 53,00 | 21 | 12,5 |
| **Drebin_api_calls** | | | | | | | |
| SigAPI | 94,41 | 93,21 | 91,55 | 92,38 | 93,84 | 15 | 79,46 |
| RFG | 96,06 | 96,81 | 92,39 | 94,55 | 95,30 | 41 | 43,84 |

Analisando o *androcrawl_api_calls*, após aplicada a redução do SigAPI (apenas uma característica de 24 originais) e do RFG (21 características de 24 originais), é possível observar a alta diferença nos resultados das métricas de avaliação para os dois classificadores.

Para o classificador RF, quando aplicado no *dataset* reduzido proveniente do SigAPI, às métricas tiveram valores próximos aos valores na execução do dataset original. Das 5

métricas, apenas a precisão teve valor superior, de 93,57% para 95,88%. As outras obtiveram porcentagens um pouco menores. A acurácia caiu de 97,61% para 95,87%; o *recall* caiu de 83,05% para 63,45%; F1-Score de 87,99% para 76,37% ; e ROC AUC de 91,19% para 81,56%. O mesmo comportamento é visto para o classificador SVM.

Para o RFG, houve uma grande discrepância entre os valores das métricas. Para o RF, a acurácia caiu de 97,61% para 67,66%; a precisão caiu de 93,57% para 15,52%; o *recall* caiu de 83,05% para 63,45%; F1-Score de 87,99% para 25,24%; e ROC AUC de 91,19% para 62,19%. Já no SVM, os resultados foram um pouco melhores. A acurácia caiu de 97,61% para 79,07%; a precisão caiu de 93,57% para 14,38%; o *recall* caiu de 83,05% para 20%; F1-Score de 87,99% para 16,73% ; e ROC AUC de 91,19% para 53%.

Já analisando o *drebin_15_api_calls*, o SigAPI teve uma redução de 79,46% (15 características de 73) e o RGF uma redução de 43,84% (41 características de 73). No âmbito dos classificadores, ambos apresentaram resultados similares, todas com percentuais maiores que 90%. Em relação ao SigAPI o RF, a acurácia caiu de 97,98% para 94,95%; a precisão caiu de 98,57% para 93,92%; o *recall* caiu de 95,93% para 92,31%; F1-Score de 97,24% para 93,11% ; e ROC AUC de 97,56% para 94,40%. O mesmo comportamento é visto para o classificador SVM, e apresentam valores muito próximos.

Já em relação ao RFG, os resultados das métricas também foram muito próximos dos obtidos pelos *datasets* originais. O RF, a acurácia caiu de 97,98% para 97,26%; a precisão caiu de 98,57% para 97,63%; o *recall* caiu de 95,93% para 94,89%; F1-Score de 97,24% para 96,24%; e ROC AUC de 97,56% para 96,77%. O mesmo comportamento é visto para o classificador SVM

Analisando os resultados obtidos com *datasets* de chamadas de API, podemos concluir que o conjunto de dados selecionado é um fator decisivo no comportamento do método. O *drebin_215_api_calls* apresenta resultados melhores em relação ao *androcrawl_api_calls*, o que se pode ser explicado pelo fato de que o *drebin_215_api_calls* contém um conjunto reduzido de apenas 73 chamadas de API sensíveis e de maior relevância. Por isso, é possível obter uma representação mais compacta dos dados, reduzindo a dimensionalidade do problema e possivelmente melhorando a eficiência dos algoritmos de aprendizado de máquina aplicados. Além

disso, a seleção de chamadas de API mais relevantes pode ajudar a focar a análise em aspectos-chave, tornando os resultados mais interpretáveis e compreensíveis.

Outro fator para os melhores resultados apresentados pelo *drebin_215_api_calls* é o fato de que o *androcrawl_api_calls* ser mais desbalanceado, em relação ao número de amostras benignas e malignas. O desbalanceamento pode levar a um viés nos resultados, onde o modelo pode ter uma tendência a favorecer a classe majoritária, no caso, a classe benigna, em detrimento da classe minoritária, a classe maligna. Isso ocorre porque os algoritmos de aprendizado de máquina são projetados para otimizar medidas como a acurácia geral, e em conjuntos de dados desbalanceados, uma alta taxa de acurácia pode ser alcançada simplesmente classificando a maioria das amostras como pertencentes à classe majoritária.

Por fim, analisando as duas tabelas, concluímos que os conjuntos de dados originais têm melhores resultados que os *datasets* reduzidos, o que é esperado já que o principal objetivo dos métodos de seleção de características é reduzir a dimensionalidade dos conjuntos de dados, mantendo apenas as características mais relevantes e informativas para o problema em questão, porém essa redução da dimensionalidade pode levar à perda de informações em relação ao conjunto de dados original. No entanto, vale ressaltar que o SigAPI possui resultados muito próximos dos que foram obtidos pelos conjuntos originais.

### 3.2.2 Validação da Seleção de Permissões

A Tabela 5 apresenta os resultados obtidos pelos algoritmos de aprendizado de máquina quando aplicados no conjunto de dados original.

**Tabela 5. Resultado dos Classificadores nos *Datasets* originais de Permissões.**

| **RF** | **Acurácia** | **Precisão** | **Recall** | **F1 Score** | **Roc auc** |
|---|---|---|---|---|---|
| Adroit | 81,90 | 66,32 | 79,73 | 72,41 | 81,27 |
| Androcrawl_permissions | 89,11 | 3,31 | 0,12 | 0,24 | 49,84 |
| Drebin_215_pemissions | 96,52 | 97,12 | 93,36 | 92,20 | 95,87 |
| **SVM** | **Acurácia** | **Precisão** | **Recall** | **F1 Score** | **Roc auc** |
| Adroit | 89,07 | 87,16 | 74,25 | 80,19 | 84,81 |
| Androcrawl_permissions | 89,34 | 0,69 | 0,00 | 0,00 | 49,92 |
| Drebin_215_pemissions | 95,15 | 97,41 | 89,26 | 93,16 | 93,93 |

Os algoritmos RF e SVM foram aplicados nos conjuntos de dados *adroit, androcrawl_permissions* e *drebin_215_permissions*. No *adroit* e no *drebin_215_permissions,* ambos obtiveram valores elevados para acurácia, precisão, *recall*, F1-Score e ROC AUC. Já para o *androcrawl_permissions,* a única métrica com um valor significativo foi a acurácia, com 89,11%. Para a precisão, o valor foi de 3,31%; para o *recall* foi de 0,12%; para o F1-Score foi 0,24% e o ROC-AUC foi 49,84%. E no classificador SVM, os resultados das métricas foram inferiores.

Na Tabela 6 temos os dados obtidos após aplicarmos os métodos de seleção nos conjuntos de dados originais.

**Tabela 6. Resultados SigPID e Semidroid.**

| RF | Acurácia | Precisão | Recall | F1 Score | Roc auc | Features selecionadas | Redução(%) |
|---|---|---|---|---|---|---|---|
| **Adroit** | | | | | | | |
| SigPID | 89,18 | 94,44 | 67,64 | 78,83 | 82,98 | 4 | 97,6 |
| Semidroid | 88,67 | 90,41 | 69,30 | 78,47 | 83,09 | 17 | 89,76 |
| **Androcrawl_permissions** | | | | | | | |
| SigPID | 89,49 | 50,00 | 0,00 | 0,00 | 50,00 | 6 | 87,77 |
| Semidroid | 89,48 | 0,00 | 0,00 | 0,00 | 50,00 | 5 | 89,8 |
| **Drebin_215_permissions** | | | | | | | |
| SigPID | 94,76 | 95,08 | 90,52 | 92,75 | 93,89 | 27 | 76,11 |
| Semidroid | 88,57 | 95,86 | 72,21 | 82,37 | 85,19 | 12 | 89,39 |
| **SVM** | **Acurácia** | **Precisão** | **Recall** | **F1 Score** | **Roc auc** | **Features selecionadas** | **Redução(%)** |
| **Adroit** | | | | | | | |
| SigPID | 89,18 | 94,48 | 67,64 | 78,84 | 82,98 | 4 | 97,6 |
| Semidroid | 88,67 | 90,45 | 69,28 | 78,41 | 83,08 | 17 | 89,76 |
| **Androcrawl_permissions** | | | | | | | |
| SigPID | 89,48 | 66,66 | 0,00 | 0,00 | 50,00 | 6 | 87,77 |
| Semidroid | 89,48 | 66,66 | 0,00 | 0,00 | 50,00 | 5 | 89,8 |
| **Drebin_permissions** | | | | | | | |
| SigPID | 93,82 | 95,66 | 87,25 | 91,26 | 92,46 | 27 | 76,11 |
| Semidroid | 88,44 | 95,69 | 71,97 | 82,16 | 85,04 | 12 | 89,39 |

Analisando o *adroit*, após aplicada a redução do SigPID (4 características de 166 originais) e do SemiDroid (17 características de 166 originais), é possível observar a semelhança nos resultados das métricas de avaliação para os dois classificadores.

Para o classificador RF, quando aplicado no *dataset* reduzido proveniente do SigPID, as métricas tiveram valores próximos aos valores na execução do *dataset* original. Das 5 métricas, apenas o *recall* caiu, de 79,73% para 67,64%. As outras porcentagens apresentaram valores superiores. A acurácia subiu de 81,90% para 89,18%; a *precisão*

subiu de 66,32% para 94,44%; F1-Score de 72,41% para 78,83%; e ROC AUC de 81,27% para 82,98%. O classificador SVM teve resultados superiores apenas para a acurácia que foi de 89,07% para 89,18% e para a precisão que teve um aumento de 87,16% para 94,48%. Todas as outras métricas tiveram resultados inferiores se comparadas com o conjunto de dados original. O *adroit* para o método Semidroid, apresentou o mesmo comportamento que para o SigPID, e também apresentou métricas com valores muito parecidos.

Analisando o *drebin_215_permissions*, após aplicada a redução do SigPID (27 características de 113 originais) e do SemiDroid (12 características de 113 originais). Para o classificador RF, quando aplicado no *dataset* reduzido proveniente do SigPID, as métricas tiveram valores próximos aos valores na execução do *dataset* original. Das 5 métricas houve aumento no F1-Score. As outras porcentagens apresentaram valores um pouco inferiores. A acurácia caiu de 96,52% para 94,76%; a *precisão* caiu de 97,12% para 95,08%;o *recall* caiu de 93,36% para 90,52%; F1-Score subiu 92,20% para 92,75%; e ROC AUC caiu de 95,87% para 85,19%. O classificador SVM teve o mesmo comportamento. Os resultados do Semidroid, para o classificador RF e SVM, foram bastante semelhantes, e todas as métricas do conjunto original apresentaram valores superiores aos valores obtidos pelo método.

Analisando o *androcrawl_permissions*, após aplicada a redução do SigPID (6 características de 49 originais) e do SemiDroid (5 características de 49 originais). Nos dois métodos e para os dois classificadores, tanto o RF e o SVM, o *dataset* apresenta valores significantes apenas na acurácia, que foi 89,49%; a precisão foi de 50% e o ROC AUC foi de 50%. Nas outras métricas os valores foram próximos de 0%, seguindo o mesmo comportamento que foi observado nos dados obtidos pelo conjunto original.

Observa-se, que o *androcrawl_permissions* sofre do mesmo efeito já observado no *androcrawl_api_calls,* por ser um *dataset* com amostras bastante desbalanceadas, isso acaba atrapalhando o desempenho obtido nos classificadores.

# Capítulo 4

# Conclusões

Neste trabalho, realizamos uma avaliação de quatro (04) métodos de seleção de características: SigPID, Semidroid, SigAPI e RFG. Esses métodos foram testados em características do tipo permissões e chamadas de API, utilizando três conjuntos de dados distintos. No entanto, os resultados obtidos indicaram que, embora esses métodos tenham sido projetados e especializados para um único tipo de característica (permissões e chamadas de API), eles não foram suficientemente abrangentes para identificar padrões em diferentes conjuntos de dados.

Isso sugere que, apesar da eficácia desses métodos em determinados contextos específicos, eles podem apresentar limitações ao serem aplicados em conjuntos de dados com características diferentes. Essas limitações podem estar relacionadas à capacidade dos métodos de capturar e representar os padrões presentes nos dados de entrada.

Uma forma de explicar os resultados é olhando para os conjuntos de dados utilizados. Eles apresentam variações em termos de dimensões, densidade, balanceamento e diversidade de características. Essas variações podem afetar o desempenho dos métodos de seleção de características, impedindo que sejam generalistas o suficiente para lidar com diferentes conjuntos de dados.

Os resultados ao avaliarmos os quatro métodos destacam a importância de considerar as particularidades dos conjuntos de dados ao escolher e aplicar métodos de seleção de características. Um método que funciona bem em um determinado contexto pode não ter um desempenho satisfatório em outro, devido às diferentes características e padrões presentes nos dados. Portanto, é fundamental realizar avaliações cuidadosas e adaptar os métodos de seleção de características de acordo com as características específicas de cada conjunto de dados.

Para ilustrar essa instabilidade, podemos observar os resultados obtidos pelo método RFG. Ele apresenta uma grande discrepância na taxa de redução entre os dois conjuntos

de dados utilizados, assim como nas métricas aplicadas. No conjunto de dados drebin 215 api calls, o RFG alcança uma taxa de corte de 43,84%, mantendo todas as métricas com valores acima de 90%. Por outro lado, no conjunto de dados androcrawl api calls, a redução é de apenas 12,5%, resultando em um desempenho insatisfatório nas métricas de acurácia, precisão, *recall*, f1-score e ROC-AUC. Esses resultados destacam a falta de consistência e estabilidade do método RFG em relação à redução de dimensionalidade e ao desempenho das métricas.

Dentre os métodos avaliados, o Semidroid foi o único a apresentar resultados mais consistentes entre os diferentes conjuntos de dados. Esse método conseguiu generalizar a seleção de chamadas de API para conjuntos de dados diversos, utilizando seis técnicas de seleção diferentes. Cada uma dessas técnicas gera subconjuntos de características, e o método avalia qual deles apresenta melhor eficiência para selecionar as características a serem aplicadas nos modelos de aprendizado de máquina.

Essa abordagem abrangente e adaptável do Semidroid permitiu que ele se destacasse ao lidar com diferentes conjuntos de dados, garantindo resultados mais consistentes em comparação com os outros métodos avaliados. Isso ressalta a importância de utilizar métodos que sejam capazes de se ajustar e adaptar às características específicas dos conjuntos de dados para obter os melhores resultados. Portanto, este estudo enfatiza a importância de métodos de seleção de características que possam lidar com a diversidade de conjuntos de dados.

## 4.1 Dificuldades Encontradas

Durante a realização da metodologia descrita, foram encontradas algumas dificuldades que precisaram ser superadas. Uma das dificuldades comuns é a falta de disponibilidade do código fonte dos métodos de seleção de características selecionados. Isso pode dificultar a replicação dos experimentos e a compreensão detalhada da implementação dos métodos, exigindo que os fossem implementados os métodos a partir de descrições algorítmicas.

Além disso, a qualidade dos *datasets* apresentou desafios. *Datasets* podem conter valores faltantes, ruídos ou inconsistências nos dados. Isso requer um pré-processamento adicional para lidar com essas questões, como imputação de dados faltantes, tratamento de *outliers* ou limpeza de dados ruidosos.

Outra dificuldade foi a escassez de *datasets* relevantes para a avaliação dos métodos de seleção de características. Em alguns domínios específicos, como em *datasets* com apenas chamadas de api, houve uma falta de *datasets* disponíveis ou os *datasets* disponíveis não se adequaram perfeitamente ao problema de pesquisa em questão. Isso limitou a diversidade dos conjuntos de dados utilizados e a generalização dos resultados obtidos.

Por fim, limitações computacionais surgiram, especialmente em relação à seleção de características e à avaliação dos modelos de aprendizado de máquina. Dependendo da quantidade de características e do tamanho dos conjuntos de dados utilizados, a execução dos experimentos exigiram recursos computacionais significativos, como memória e poder de processamento.

## 4.2. Trabalhos Futuros

Existem várias direções para trabalhos futuros com base na metodologia realizada. Uma possível abordagem é explorar e avaliar métodos de seleção de características adicionais, além dos selecionados nesta pesquisa. Novos métodos podem ser propostos e comparados com os métodos existentes, buscando identificar abordagens mais eficientes e eficazes. Além disso, a investigação de técnicas de seleção de características específicas para problemas ou domínios particulares pode ser explorada, a fim de desenvolver métodos mais adaptados a contextos específicos.

Além disso, a aplicação da metodologia em conjuntos de dados ainda mais diversos e desafiadores pode fornecer uma visão mais abrangente sobre a eficácia dos métodos de seleção de características. A combinação de diferentes métodos de seleção também pode ser investigada para explorar estratégias de seleção de características mais robustas e de alto desempenho. Por fim, é importante considerar a interpretabilidade dos métodos de seleção de características e como eles podem ser utilizados para melhorar a

compreensão dos modelos de aprendizado de máquina. Em geral, há um vasto campo de pesquisa aberto para aprimorar e expandir o conhecimento sobre a seleção de características e sua aplicação em modelos de aprendizado de máquina.

# Referências Bibliográficas


Sun, L., Li, Z., Yan, Q., Srisa-an, W., and Pan, Y. (2016). Sigpid: significant permission identification for android malware detection. In 2016 11th *International Conference on Malicious and Unwanted Software (MALWARE)*, pages 1–8.

Galib, A. H. and Hossain, B. M. M. (2020). Significant API calls in android malware detection (using feature selection techniques and correlation based feature elimination). In Garcia-Castro, R., editor, The 32nd *International Conference on Software Engineering and Knowledge Engineering*, SEKE 2020, KSIR Virtual Conference Center, USA, July 9-19, 2020, pages 566–571.

Kakavand, M., Dabbagh, M., and Dehghantanha, A. (2018). Application of machine learning algorithms for android malware detection. In Proceedings of the 2018 *International Conference on Computational Intelligence and Intelligent Systems*, pages 32–36.

Agrawal, P. and Trivedi, B. (2021). Machine learning classifiers for android malware detection. In *Data Management, Analytics and Innovation*, pages 311–322. Springer.
Rana, M., Rahman, S. S. M. M., Sung, A. H., et al. (2018). Evaluation of tree based machine learning classifiers for android malware detection. In *International Conference on Computational Collective Intelligence*, pages 377–385.

Springer.Damodaran, A., Di Troia, F., Visaggio, C. A., Austin, T., and Stamp, M. (2017). A comparison of static, dynamic, and hybrid analysis for malware detection. *Journal of Computer Virology and Hacking Techniques*, 13.

Shijo, P. and Salim, A. (2015). Integrated static and dynamic analysis for malware detection. Procedia Computer Science, 46:804–811. Proceedings of the *International Conference on Information and Communication Technologies*, ICICT 2014, 3-5 December, Kochi, India.

Qiu, J., Zhang, J., Luo, W., Pan, L., Nepal, S., and Xiang, Y. (2020). A survey of android malware detection with deep neural models. *ACM Comput. Surv.,* 53(6).

Roy, S., DeLoach, J., Li, Y., Herndon, N., Caragea, D., Ou, X., Ranganath, V. P., Li, H., and Guevara, N. (2015). Experimental study with real-world data for android app security analysis using machine learning. In Proceedings of the 31st *Annual Computer Security Applications Conference*, ACSAC 2015, page 81–90, New York, NY, USA.

Cai, L., Li, Y., and Xiong, Z. (2021). Jowmdroid: Android malware detection based on feature weighting with joint optimization of weight-mapping and classifier parameters. *Computers & Security*, 100:102086.

Moutaz, A. (2020). Automated malware detection in mobile app stores based on robust feature generation. *Electronics*, 9:435.
Remeseiro, B. and Bolon-Canedo, V. (2019). A review of feature selection methods in medical applications. *Computers in Biology and Medicine*, 112:103375.

Venkatesh, B. and Anuradha, J. (2019). A review of feature selection and its methods. *Cybernetics and Information Technologies*, 19(1):3–26.

Golrang, A., Yayilgan, S. Y., and Elezaj, O. (2021). The multi-objective feature selection in android malware detection system. In Yildirim Yayilgan, S., Bajwa, I. S., and Sanfilippo, F., editors, Intelligent Technologies and Applications, pages 311–322, Cham. *Springer International Publishing.*

Feizollah, A., Anuar, N. B., Salleh, R., and Wahab, A. W. A. (2015). A review on feature selection in mobile malware detection. *Digital Investigation*, 13:22–37.

Mahindru, A. and Sangal, A. L. (2019). Deepdroid: Feature selection approach to detect android malware using deep learning. In 2019 IEEE 10th *International Conference on Software Engineering and Service Science (ICSESS)*, pages 16–19.
Montgomery, D. C., Peck, E. A., and Vining, G. G. (2021). Introduction to linear regression analysis. *John Wiley & Sons*.

Xiao, J., Xu, K., and Duan, J. (2019). Malicious android application detection based on composite features. In *Proceedings of the 3rd International Conference on Computer Science and Application Engineering*, CSAE 2019, New York, NY, USA.

Lee, J., Jang, H., Ha, S., and Yoon, Y. (2021). Android malware detection using machine learning with feature selection based on the genetic algorithm. *Mathematics*, 9(21).

Wang, W., Zhao, M., Gao, Z., Xu, G., Xian, H., Li, Y., and Zhang, X. (2019). Constructing Features for Detecting Android Malicious Applications: Issues, Taxonomy and Directions. IEEE Access, 7:67602–67631.

Sharma, T. and Rattan, D. (2021). Malicious application detection in android—a systematic literature review. *Computer Science Review*, 40:100373.

Zhou, Y. and Jiang, X. (2012). Dissecting Android malware: Characterization and evolution. In 2012 *IEEE Symposium on Security and Privacy*, pages 95–109.