FS3E: uma Ferramenta para Execução e Avaliação de Métodos de Seleção de Características para Detecção de Malwares Android

Estevão Costa[1], **Diego Kreutz**[2], **Vanderson Rocha**[1], **Luiza Leão**[1]
Sávio Sabóia[1], **Nicolas Neves**[1], **Eduardo Feitosa**[1]

[1]Universidade Federal do Amazonas (UFAM)
{ecc,lpml,savio.saboia,nicolas.neves,efeitosa}@icomp.ufam.edu.br
vanderson@ufam.edu.br

[2]Universidade Federal do Pampa (UNIPAMPA)
diegokreutz@unipampa.edu.br

Resumo. *Atualmente, há dezenas de métodos sofisticados para a seleção de características de um dataset. Entretanto, é difícil encontrarmos a implementação para avaliar a qualidade dos métodos em conjuntos diversos de datasets. Como forma de apresentar uma primeira solução para esse problema, implementamos a FS3E, uma ferramenta para catalogar implementações e automatizar a avaliação de métodos sofisticados de seleção de características. Para demostrar a funcionalidade e utilidade da FS3E, implementamos e disponibilizamos também cinco métodos sofisticados de seleção de características para o domínio de detecção de malwares Android.*

1. Introdução

Os modelos utilizados no contexto de detecção de *malwares* Android são treinados e validados com *datasets* que contém amostras de aplicativos rotuladas em benigna ou maligna. Essas amostras costumam vir acompanhadas de diferentes tipos de características, como permissões, intenções, chamadas de API e componentes do aplicativo [Damodaran et al., 2017, Qiu et al., 2020].

O principal desafio dos modelos de detecção de *malwares* Android é trabalhar com a quantidade de dados contida nos *datasets*. Por exemplo, um *dataset* pode conter mais de 1 milhão de amostras e mais de 1 milhão de características [Roy et al., 2015], o que representa um desafio muito grande para o treinamento, validação e aplicação prática de modelos de aprendizado de máquina. Por exemplo, um *dataset* pequeno (e.g., apenas 6 mil amostras e 643 características) pode consumir praticamente 6 horas de processamento de uma CPU moderna [Cai et al., 2021].

No contexto de detecção de *malwares* Android, existem várias dezenas de métodos sofisticados de seleção de características [Mahindru and Sangal, 2019, Xiao et al., 2019, Lee et al., 2021, Mahindru and Sangal, 2021, Wu et al., 2022, Smmarwar et al., 2022], cujo objetivo é reduzir a dimensionalidade do *dataset* através da identificação assertiva das características mais significativas para a detecção de aplicativos maliciosos. Por exemplo, métodos como o SigPID [Sun et al., 2016], *Robust Feature Generation* (RFG) [Moutaz, 2020] e JOWNDroid [Cai et al., 2021] desempenham um papel importante no desenvolvimento de modelos de classificação otimizados,

pois evitam ambiguidades (e.g., características como a permissão `INTERNET`, utilizada tanto por aplicativos maliciosos quanto benignos) e levam a uma seleção criteriosa das características mais relevantes para a detecção de *malwares*. Essa redução de características, que pode chegar a mais de 90% em diversos casos [Soares et al., 2022], reduz substancialmente a dimensionalidade dos *datasets*, tornando a quantidade de dados viável para o treinamento, validação e aplicação prática dos modelos. Menos dados, mas qualitativamente mais expressivos e significativos, reduz o tempo de computação, melhora a escalabilidade dos modelos e aumenta a capacidade de generalização da classificação [Feizollah et al., 2015, Remeseiro and Bolon-Canedo, 2019, Venkatesh and Anuradha, 2019, Golrang et al., 2021].

Trabalhos existentes na literatura (e.g., [Nivaashini et al., 2018, Xiao et al., 2019, Anggraeni et al., 2021, Mahindru and Sangal, 2021, Wu et al., 2022]) propõe recorrentemente novos métodos de seleção de características e os comparam com um conjunto limitado de soluções existentes. Em alguns casos, os novos métodos são comparados apenas com relação a técnicas clássicas e generalistas de seleção de características, como chi-quadrado, ganho de informação, regressão linear e ANOVA. Trabalhos recentes, como o [Soares et al., 2022], avaliam e discutem, comparativamente, apenas quatro métodos sofisticados de seleção de características para o contexto de detecção de *malwares* Android. Porém, a reprodutibilidade e evolução desses trabalhos é comprometida devido a falta de um arcabouço de software, suficientemente desacoplado e modular, que seja capaz de continuamente incorporar e avaliar sistematicamente novos métodos sofisticados de seleção de características.

Neste trabalho, apresentamos a FS3E, de *Feature Selection Execution and Evaluation Engine*, uma ferramenta que disponibiliza um arcabouço de software para a incorporação simplificada e avaliação sistemática e automatizada dos mais diversos métodos sofisticados de seleção de características para o domínio de detecção de *malwares* Android. Na versão atual, a ferramenta incorpora cinco métodos avançados de seleção de características (SigPID [Sun et al., 2016], RFG [Moutaz, 2020], SigAPI [Galib and Hossain, 2020], ALR [Şahin et al., 2021] e JOWNDroid [Cai et al., 2021]). Adicionalmente, a FS3E incorpora também um módulo de modelos de aprendizado de máquina para avaliação e comparação dos resultados dos métodos de seleção de características, que disponibiliza implementações utilizando os algoritmos SVM e RandomForest (RF), considerados os mais amplamente utilizados e, frequentemente, com os melhores desempenhos de classificação no domínio de detecção de *malwares* Android [Sharma and Rattan, 2021]. É importante destacarmos que a incorporação de novos métodos de seleção de características é simples e bem estruturada na FS3E, como detalharemos nas próximas seções. Com esse arcabouço de software, acreditamos que tornaremos mais factível a reprodutibilidade e a consolidação de trabalhos que propõe novos métodos sofisticados de seleção de características, simplificando e acelerando a análise comparativa dos métodos, por exemplo.

O restante do trabalho está organizado como segue. Nas Seções 2 e 3 apresentamos as etapas de processamento, o fluxo de dados e a implementação da FS3E. Por fim, nas Seções 4 e 5 apresentamos resultados de utilização prática da ferramenta, detalhes da demonstração e considerações finais.

2. FS3E: Etapas e Fluxo de Dados

Na Figura 1 apresentamos as etapas e o fluxo de dados da ferramenta FS3E. Como pode ser observado, ela é composta por três etapas distintas: (a) **métodos** de seleção de características, (b) **avaliação** com modelos de aprendizado de máquina, e (c) **visualização** dos resultados.

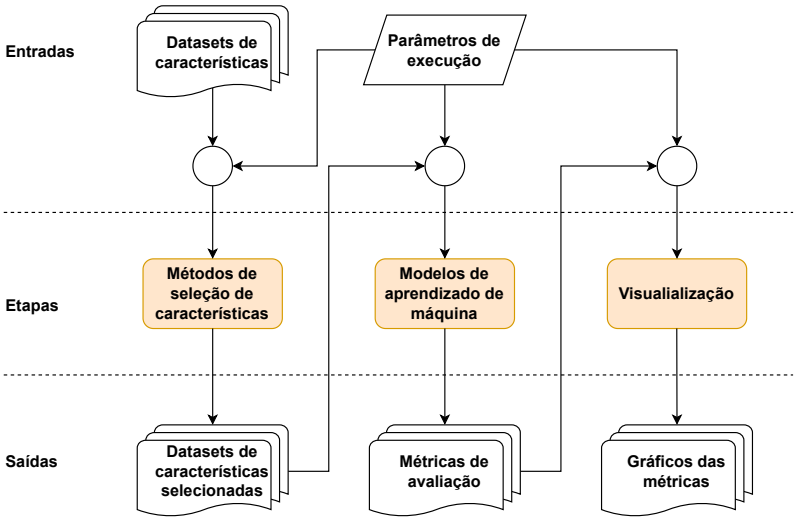


Figura 1. Etapas e fluxo de dados da FS3E

A primeira etapa são aplicados, sobre cada *dataset* de **entrada**, os **modelos** de seleção de características escolhidos pelo usuário. Cada método gera um novo conjunto de dados contendo as características identificadas como as mais relevantes, isto é, reduzindo a dimensionalidade do *dataset* original. As **saídas** desta etapa (i.e., novos *datasets*) servem de **entrada** para a etapa seguinte, a **avaliação** utilizando modelos de aprendizado de máquina. A **saída** desta segunda etapa são as métricas de avaliação (e.g., acurácia, revocação e precisão), que representam os dados de **entrada** da etapa de **visualização**. Nesta etapa final, são gerados gráficos das métricas para facilitar a interpretação e análise comparativas dos resultados gerados pela combinação de métodos de seleção de características e modelos de aprendizado de máquina.

3. Implementação

Na Figura 2 apresentamos uma visão geral da implementação da FS3E, desenvolvida e testada com Python versão 3.10.4, `pip` versão 22.0.2 e os módulos-versões `pandas-1.4.3`, `seaborn-0.11.2`, `scikit-learn-1.1.2`, `matplotlib-3.5.2`, `mlxtend-0.20.0`, `numpy-1.23.1` e `tqdm-4.64.0`. Os métodos de seleção de características estão organizados em um repositório específico do sistema de arquivos. Cada método precisa incorporar apenas um "invocador padrão", representado por um *shell script* denominado `run.sh`, contido no diretório da implementação de cada método. O "invocador padrão" recebe os parâmetros de execução da ferramenta (e.g., nomes dos arquivos de entrada e de saída) e executa o respectivo método. Essa organização torna possível um acoplamento fraco, isto é, não é necessário configurar ou mexer no código da FS3E, simplificando e agilizando o processo de inclusão de novos métodos de seleção de características. Resumidamente, para adicionar

um novo método, basta criar uma nova pasta para o método e disponibilizar o "invocador padrão" `run.sh`. A partir disso, o método passa a estar automaticamente disponível na lista de parâmetros de entrada da FS3E. Na versão atual, a ferramenta disponibiliza os métodos SigPID, SigAPI, ALR, RFG e JOWMDroid.

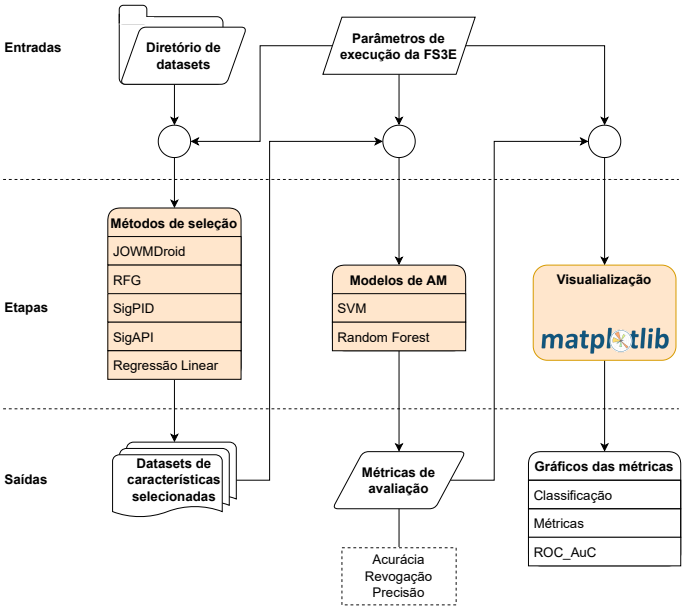


Figura 2. Implementação da ferramenta FS3E

Na segunda etapa, os modelos de aprendizado de máquina, utilizados na avaliação, recebem a saída dos métodos de seleção, isto é, um *dataset* contendo apenas as características selecionadas. Na versão atual da ferramenta, estão disponíveis dois modelos de aprendizado de máquina, o SVM e o RF.

Finalmente, na etapa de visualização, a ferramenta recebe as métricas de saída dos modelos de aprendizado de máquina e apresenta os resultados, agrupados, na forma de três tipos de gráficos, classificação, métricas e ROC_AuC.

4. Resultados

Esta seção apresenta os *datasets* e o ambiente de execução (Seção 4.1), os resultados das reduções de dimensionalidade dos cinco métodos de seleção de características (Seção 4.2) e os gráficos das métricas geradas pelos modelos de aprendizado de máquina (Seção 4.3).

4.1. *Datasets*

A relação dos *datasets* utilizados nos testes é sumarizada na Tabela 1. Ao todo, são nove *datasets* com diferentes tipos e quantidades (**# Cars**) de características e também variados números de amostras (**# Ams.**) e proporções entre benignos e malignos (**B:M**). Os *datasets* também estão disponíveis no repositório da FS3E[1].

Os testes foram realizado em um computador com processador Intel(R) Core(TM) i7-1185G7 3.00GHz da geração 11, memória RAM de 32GB, executando um Linux Ubuntu 20.04.3 LTS 64 bit.

[1] `https://github.com/Malware-Hunter/sf22-fs3e`

Tabela 1. Detalhamento dos *datasets*

Dataset	Tipos de características	# Cars.	# Ams.	B:M
motodroid_all	Permissões, Chamadas de API, Intenções, etc.	3048	46.670	15,4:1
motodroid_permissions	Permissões	1.316		
motodroid_api_calls	Chamadas de API	1.524		
androcrawl_all	Permissões, Chamadas de API, Intenções, etc.	81	96.732	8,5:1
androcrawl_permissions	Permissões	49		
androcrawl_api_calls	Chamadas de API	24		
drebin_215_all	Permissões, Chamadas de API, Intenções, etc.	215	15.031	1,7:1
drebin_215_permissions	Permissões	113		
drebin_215_api_calls	Chamadas de API	73		

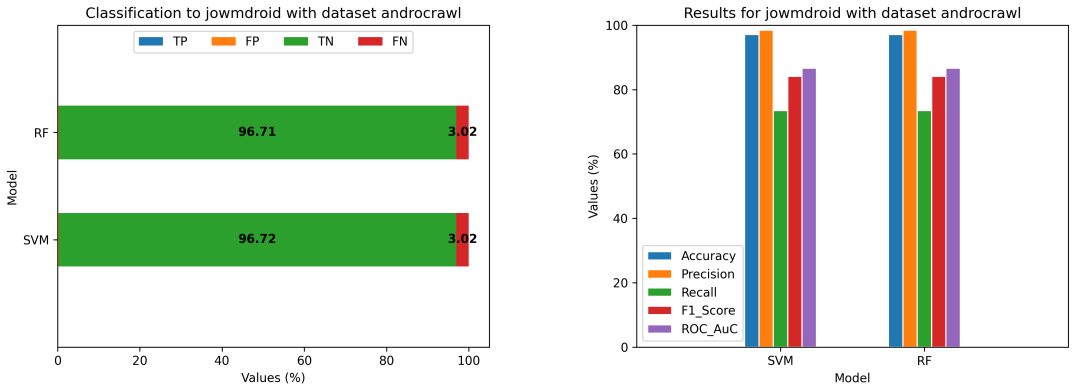
4.2. Métodos de Seleção

A Tabela 2 apresenta um resumo das reduções de dimensionalidade para os nove *datasets* apresentados na Tabela 1. É importante destacarmos que cada método é especializado para um (ou mais) tipo de características. Por exemplo, o método JOWMDroid foi projetado para trabalhar simultaneamente com diferentes tipos de características (e.g., permissões, chamadas de API, componentes do aplicativo). Este é o motivo pelo qual ele é o único método aplicado sobre os *datasets* finalizados por `_all`.

Tabela 2. Redução de dimensionalidade dos métodos de seleção de características

Dataset	Medida	JOWMDroid	SigPID	ALR	SigAPI	RFG
drebin_215_all	Redução (%)	82,30	–	–	–	–
	Tempo (s)	14,77	–	–	–	–
motodroid_all	Redução (%)	95,70	–	–	–	–
	Tempo (s)	788,89	–	–	–	–
androcrawl_all	Redução (%)	88,89	–	–	–	–
	Tempo (s)	49,76	–	–	–	–
drebin_215_permissions	Redução (%)	–	71,68	69,03	–	–
	Tempo (s)	–	47,96	2,06	–	–
motodroid_permissions	Redução (%)	–	99,77	46,05	–	–
	Tempo (s)	–	7106,16	745,83	–	–
androcrawl_permissions	Redução (%)	–	87,76	97,96	–	–
	Tempo (s)	–	176,43	3,02	–	–
drebin_215_api_calls	Redução (%)	–	–	–	79,45	71,23
	Tempo (s)	–	–	–	540,636	203,05
motodroid_api_calls	Redução (%)	–	–	–	–	8,07
	Tempo (s)	–	–	–	–	4322,94
androcrawl_api_calls	Redução (%)	–	–	–	95,83	12,50
	Tempo (s)	–	–	–	314,55	391,39

Outros métodos, como o SigPID e ALR, são especializados em reduzir a dimensionalidade de *datasets* contendo apenas permissões. Por fim, métodos como o SigAPI



(a) Gráfico de Classificação **(b) Gráfico de Métricas**

Figura 3. Gráficos gerados pela FS3E

e o RFG são especializados em reduzir a dimensionalidade de *datasets* contendo apenas chamadas de API.

Como podemos observar na Tabela 2, a redução de dimensionalidade e o tempo de computação necessário varia entre os *datasets* e métodos. Por exemplo, a redução é menor (e.g., 82,30%) em *datasets* como o drebin_215_all, pois ele possui um conjunto de características mais significativas para a detecção de *malwares* Android. Outros *datasets*, como o motodroid_all, possuem mais características, entretanto, a maioria delas mais esparsas e menos significativas. Consequentemente, a redução é mais significativa (e.g., 95,70%).

Nos *datasets* de tipos específicos de características ocorre algo similar. Por exemplo, o nível de redução cai ainda mais (71,68%) para o drebin_215_permissions, que contém 113 permissões. Isto ocorre por que o número de permissões relevantes, para detecção de *malwares*, é proporcionalmente maior quando comparada com todas as características contidas no *dataset* (i.e., drebin_215_all).

4.3. Gráficos das Métricas

A Figura 3 apresenta exemplos de saídas gráficas da FS3E para o *dataset* androcrawl_all e o método de seleção de características JOWMDroid. Como podemos observar na Figura 3b, as características selecionadas pelo JOWMDroid levam a modelos de aprendizado de máquina SVM e RF bons, ou seja, taxas de ROC_AuC acima de 80%. Apesar de a acurácia e precisão ficarem acima de 95%, o *recall* ficou abaixo de 80%. Para detecção de *malwares* Android, é importante que esta métrica seja elevada, pois ela identifica a porcentagem de amostras maliciosas corretamente classificadas. Os gráficos para os demais *datasets* e métodos podem ser encontrados no repositório da ferramenta.

Tanto o repositório de métodos de seleção de características, quanto a execução automatizada e os gráficos gerados, são recursos importantes e úteis para todos os desenvolvedores ou pesquisadores que desejam avaliar o estado da arte ou propor novos métodos de seleção de características. Acreditamos que um arcabouço de software deste tipo é um importante passo para termos pesquisas mais reprodutíveis e novos avanços na detecção de *malwares* Android sofisticados. Atualmente, a maioria dos métodos de

seleção de características, propostos na literatura, são frequentemente comparados apenas com técnicas tradicionais de uso geral (e.g., análise de componente principal, ganho de informação) e *datasets* limitados (e.g., construídos pelos próprios autores). Conforme identificamos em um levantamento bibliográfico, muitas vezes, os *datasets* dos trabalhos nem sequer estão disponíveis.

5. Considerações Finais

Demonstração. O código-fonte, documentação e instruções de instalação estão dispoıveis no repositório da FS3E[2]. A demonstração da ferramenta será realizada através de um ambiente em dispositivo próprio dos autores. O funcionamento da ferramenta será demonstrado através dos seguintes passos: (a) apresentação dos parâmetros de execução da ferramenta; (b) apresentação da organização da implementação (e.g., como incluir novo método de seleção de características); (c) apresentação da execução e resultados para um conjunto de *datasets*.

Conclusão. Neste trabalho apresentamos a FS3E, um arcabouço de software que permite a inclusão de novos métodos seleção de características e modelos de aprendizado de máquina, bem como sua execução e avaliação de forma completamente automatizada, potencializando a reprodutibilidade e a avaliação sistemática de novos métodos sofisticados de seleção de características. Como trabalhos futuros, podemos destacar: (a) inclusão de novos métodos de seleção de características; (b) implementação e disponibilização de outros modelos de aprendizado de máquina; e (c) inclusão e avaliação dos métodos de seleção de características em grupos maiores de *datasets*, buscando analisar e identificar o comportamento de cada método em diferentes conjuntos de dados.

Agradecimentos. Esta pesquisa foi parcialmente financiada, conforme previsto nos Arts. 21 e 22 do decreto no. 10.521/2020, nos termos da Lei Federal no. 8.387/1991, através do convênio no. 003/2021, firmado entre ICOMP/UFAM, Flextronics da Amazônia Ltda e Motorola Mobility Comércio de Produtos Eletrônicos Ltda. O presente trabalho foi realizado também com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

Referências

Anggraeni, A., Mustofa, K., and Priyanta, S. (2021). Comparison of filter and wrapper based feature selection methods on spam comment classification. *IJCCS*, 15(3):245–254.

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.

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.

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.

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 García-Castro, R., editor, *The 32nd SEKE*, pages 566–571. KSI Research Inc.

[2]`https://github.com/Malware-Hunter/sf22-fs3e`

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.

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).

Mahindru, A. and Sangal, A. L. (2019). DeepDroid: Feature selection approach to detect android malware using deep learning. In *ICSESS*, pages 16–19.

Mahindru, A. and Sangal, A. L. (2021). Semidroid: a behavioral malware detector based on unsupervised machine learning techniques using feature selection approaches. *International Journal of Machine Learning and Cybernetics*, 12:1411.

Moutaz, A. (2020). Automated malware detection in mobile app stores based on robust feature generation. *Electronics*, 9:435.

Nivaashini, M., Soundariya, R. S., Vidhya Shri, H., and Thangaraj, P. (2018). Comparative analysis of feature selection methods and machine learning algorithms in permission based android malware detection. In *I2C2SW*, pages 72–77.

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).

Remeseiro, B. and Bolon-Canedo, V. (2019). A review of feature selection methods in medical applications. *Computers in Biology and Medicine*, 112:103375.

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 *31st ACSAC*, page 81–90, New York, NY, USA. Association for Computing Machinery.

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

Smmarwar, S. K., Gupta, G. P., and Kumar, S. (2022). A hybrid feature selection approach-based android malware detection framework using machine learning techniques. In Agrawal, D. P., Nedjah, N., Gupta, B. B., and Martinez Perez, G., editors, *Cyber Security, Privacy and Networking*, pages 347–356, Singapore. Springer Nature Singapore.

Soares, T., Kreutz, D., Rocha, V., Costa, E., Leao, L., Pontes, J., Assolin, J., Rodrigues, G., and Feitosa, E. (2022). Uma análise de métodos de seleção de características aplicados à detecção de malwares android. In *SBSeg22*.

Sun, L., Li, Z., Yan, Q., Srisa-an, W., and Pan, Y. (2016). SigPID: significant permission identification for android malware detection. In *MALWARE*, pages 1–8.

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

Wu, Y., Li, M., Wang, J., Fang, Z., Zeng, Q., Yang, T., and Cheng, L. (2022). Droidrl: Reinforcement learning driven feature selection for android malware detection. `https://arxiv.org/abs/2203.02719`.

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. Association for Computing Machinery.

Şahin, D., Kural, O., Akleylek, S., and Kilic, E. (2021). A novel permission-based android malware detection system using feature selection based on linear regression. *Neural Computing and Applications*, pages 1–16.