

# AMGenerator e AMExplorer: Geração de Metadados e Construção de Datasets Android

Vanderson Rocha<sup>2(‡)</sup>, Joner Assolin<sup>2(‡)</sup>, Hendrio Bragança<sup>2(‡)</sup>  
Diego Kreutz<sup>1(‡)</sup>, Eduardo Feitosa<sup>2</sup>

<sup>1</sup>Universidade Federal do Pampa (UNIPAMPA)  
diegokreutz@unipampa.edu.br

<sup>2</sup>Universidade Federal do Amazonas (UFAM)  
{joner.assolin,hendrio.luis,efeitosa}@icomp.ufam.br, vanderson@ufam.edu.br

**Resumo.** *ADBUILDER é uma ferramenta recente que permite a construção de datasets atualizados para detecção de malwares Android. Entretanto, ela carece de um fluxo mais rigoroso de rotulação de amostras e recursos adicionais para explorar metadados e construir datasets mais ricos. Por esse motivo, apresentamos uma evolução da ADBUILDER, que resulta em duas novas ferramentas especializadas, a AMGenerator e AMExplorer. Enquanto a AMGenerator implementa as etapas de aquisição de dados, extração de metadados e características e rotulação robusta, a AMExplorer é especializada em explorar coleções de metadados e construir datasets mais completos e ricos para especialistas em detecção de malwares Android.*

## 1. Introdução

Existem diversos estudos recentes (e.g., [Wang et al., 2019, Soares et al., 2021a, Soares et al., 2021b]) que apontam problemas em *datasets* amplamente utilizados para o treinamento e a validação de modelos preditivos para a detecção de *malwares* Android. Como exemplo de problema podemos destacar a incorreta marcação (rotulação) das amostras em *datasets* como MalGenome, Drebin, Piggybacking e AMD [Wang et al., 2019]. Uma marcação incorreta das amostras compromete severamente todo o processo de treinamento e validação de modelos preditivos baseados em aprendizado de máquina supervisionado, o método mais frequentemente utilizado para classificação de aplicativos Android.

A rotulação correta de amostras malignas e benignas é uma etapa crucial para a implementação e avaliação de novos métodos de detecção de *malwares* Android [Wei et al., 2017, Salem et al., 2021, Vilanova et al., 2022]. Frequentemente, a plataforma online VirusTotal<sup>1</sup> é utilizada para categorizar aplicativos Android com base nos resultados de mais de 60 *scanners* antivírus. Entretanto, há pelo menos dois grandes desafios na rotulagem de aplicativos Android utilizando resultados de plataformas como a VirusTotal.

O primeiro desafio é obter dados atualizados. Infelizmente, a maioria dos trabalhos, incluindo ferramentas recentes como a ADBUILDER [Vilanova et al., 2022], simplesmente utilizam os dados retornados pelo VirusTotal. O problema é que o

---

(‡) Os quatro primeiros autores contribuíram igualmente para o desenvolvimento do trabalho.

<sup>1</sup><https://www.virustotal.com/>

VirusTotal armazena em *cache* os dados da última análise do aplicativo. Consequentemente, eles podem estar completamente defasados. Por exemplo, o aplicativo “Corrida de Fogo de Vento”, da loja chinesa Anzhi de aplicativos Android, possui dados de análise datados de 27/05/2014 na *cache* do VirusTotal. Nesta análise o aplicativo possui apenas 1 *scanner* positivo. Entretanto, ao enviar o aplicativo para re-análise no dia 09/06/2023, utilizando a AMGenerator, os resultados obtidos apontam que 11 *scanners* detectaram este aplicativo como positivo, o que passa a ser potencialmente determinante para classificá-lo como malicioso, por exemplo. Até o momento, utilizando a AMGenerator para coletar metadados do VirusTotal, já identificamos um problema similar em 540.470 aplicativos. Nossa meta é gerarmos metadados atualizados do VirusTotal para mais de 1M de aplicativos. Portanto, diferentemente do praticado pela maioria dos trabalhos, é necessário solicitarmos novas análises dos aplicativos para termos resultados atuais e mais confiáveis.

O segundo desafio da rotulagem é a interpretação dos resultados da verificação do VirusTotal para determinar se uma amostra é maligna ou benigna. A maioria dos trabalhos (e.g., [Wei et al., 2017, Zhu et al., 2020]) simplesmente define um *threshold* de *scanners* positivos (e.g., 2, 4, 8, 10) a partir de premissas e hipóteses informais [Choo et al., 2022]. Embora não haja consenso sobre o número ideal do *threshold*, pequenos limiares como 1 ou 2 têm sido utilizados com alguma frequência [Wang et al., 2018, Zhu et al., 2020].

O *framework* Maat [Salem et al., 2021] é uma das primeiras soluções a endereçar o problema da interpretação dos metadados do VirusTotal para rotular automática e sistematicamente amostras de aplicativos Android. A partir de dados atualizados do VirusTotal, Maat emprega técnicas de aprendizado de máquina para gerar rótulos para cada amostra de aplicativo. Entretanto, no próprio artigo do Maat, os autores simplesmente utilizam os dados retornados pelo VirusTotal, isto é, não há uma preocupação em garantir dados atualizados do VirusTotal.

Recentemente, apresentamos a ADBuilder [Vilanova et al., 2022], uma das ferramentas mais atuais e completas de geração de *datasets* de amostras de aplicativos Android. A ADBuilder é composta por quatro módulos principais: download de APKs, extração de características, rotulação das amostras e geração do *dataset*. Entretanto, existem pelo menos dois grandes desafios em dois módulos cruciais da ferramenta. Primeiro, assim como os demais trabalhos existentes na literatura, o módulo de rotulação utiliza apenas o resultados da análise em *cache* do VirusTotal, isto é, retorna dados potencialmente defasados de análise do APK. Segundo, o módulo de geração do *dataset* constrói somente um conjunto de dados binário, incluindo apenas características clássicas como permissões, chamadas de API, intenções e *opcodes*. No entanto, há uma infinidade de dados e metadados de repositórios, resultantes do download, extração de características do APK e rotulação que podem ser utilizados na construção de *datasets* mais ricos.

Como forma de garantir dados atualizados para a rotulação de aplicativos Android e a construção de *datasets* mais expressivos e personalizados a partir da exploração de metadados e dados gerados ou disponibilizados por diferentes sistemas,

propomos as ferramentas AMGenerator<sup>2</sup> e AMExplorer<sup>3</sup>. Enquanto a AMGenerator é uma refatoração e evolução da ADBuilder, que incorpora um novo módulo de coleta de dados atualizados de rotulagem, a AMExplorer é uma ferramenta nova especializada em explorar dados e metadados para construir *datasets* mais completos e ricos de informação para especialistas do domínio de detecção de *malwares* Android. Exemplos de três *datasets* completos (metadados, discreto e binário), contendo 8K amostras, estão disponíveis no repositório da AMExplorer. Enquanto que o *dataset* de metadados contém 25 características de dados ricos, diversos e complexos para ser utilizado por especialistas para uma ampla coleção de análises, os outros dois contêm 195.154 características clássicas e estão prontos para serem utilizados em métodos de aprendizado de máquina convencionais.

O restante deste trabalho está organizado como segue. Na Seção 2 apresentamos a arquitetura e implementação das ferramentas AMGenerator e AMExplorer. Na Seção 3 descrevemos a metodologia e analisamos os resultados de uma avaliação. Por fim, na Seção 4 apresentamos as informações básicas sobre a demonstração e as considerações.

## 2. Arquitetura e implementação

A arquitetura das ferramentas AMGenerator e AMExplorer é ilustrada na Figura 1. Como podemos observar, a AMGenerator é composta por três módulos principais: **Aquisição**, **Extração** e **Rotulação**. Já a ferramenta AMExplorer explora os dados e metadados gerados por cada módulo da AMGenerator para criar (inicialmente) três *datasets* distintos: **metadados**, **discreto** e **binário**.

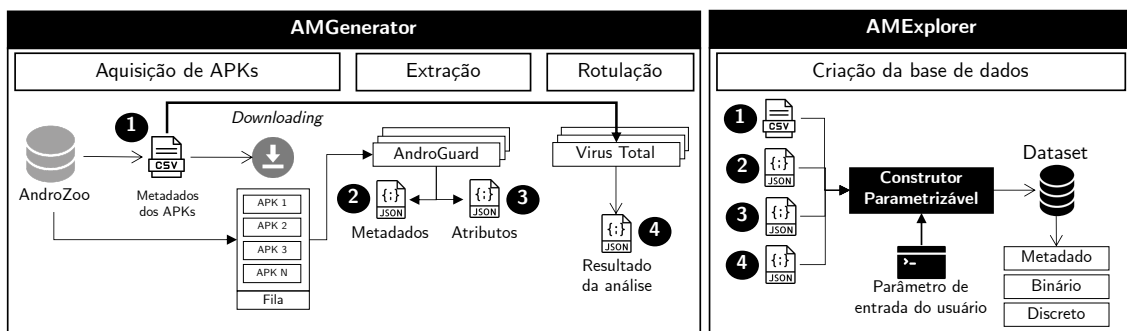


Figura 1. Visão geral das ferramentas AMGenerator e AMExplorer.

### 2.1. AMGenerator

Enquanto que as etapas de **Aquisição** e **Extração** foram incorporadas da ferramenta ADBuilder, a etapa de **Rotulação** foi re-projetada e implementada para contemplar novos requisitos, com particular atenção para a atualidade dos dados.

Para a geração de *datasets* atualizados e confiáveis, é necessário realizar uma rotulagem atualizada (data recente) dos APKs. O módulo de **rotulação** faz uso de serviços *online* (e.g., VirusTotal) para obter informações sobre o APK, de modo que

<sup>2</sup><https://github.com/Malware-Hunter/SF23-AMGenerator>

<sup>3</sup><https://github.com/Malware-Hunter/SF23-AMExplorer>

o usuário da ferramenta possa posteriormente determinar se um APK é benigno ou maligno.

Os serviços *online* podem conter dezenas de *scanners* e caberá ao usuário definir o limiar (e.g., 5 *scanners*) para classificar um APK como benigno ou malicioso. Por exemplo, o VirusTotal é um serviço *online* que disponibiliza, em julho de 2023, mais de 60 *scanners* para classificar um APK em benigno ou malicioso. Cada *scanner* do serviço retorna um resultado para o APK. Como os resultados dos *scanners* podem divergir entre si, é necessário também um método de decisão (e.g., maioria). O usuário pode atribuir rótulos às amostras a partir da contagem e limiar inferior ou utilizando soluções como a Maat, que exploram e analisam os metadados produzidos pelo VirusTotal para determinar automaticamente o rótulo das amostras.

O módulo de rotulação recebe dois parâmetros de entrada: a lista de SHA256 (identificadores dos APKs) e uma chave de API do VirusTotal. O resultado final da análise de cada APK é um arquivo JSON que contém metadados gerais do aplicativo (e.g., nome, características) e os resultados de análise de cada *scanner* disponível no VirusTotal, informando se o APK foi classificado como benigno ou maligno.

O processo da aquisição dos metadados de rotulação fornecidos pelo VirusTotal ocorre conforme o fluxo ilustrado na Figura 2. Para APKs analisados recentemente pelo VirusTotal (e.g., 01/01/2023 - data definida pelo usuário da ferramenta), é necessária apenas uma requisição, isto é, são retornados e utilizados os metadados da última análise (*cache*) realizada pelo VirusTotal. O resultado da requisição é armazenado no repositório de metadados de rotulação.

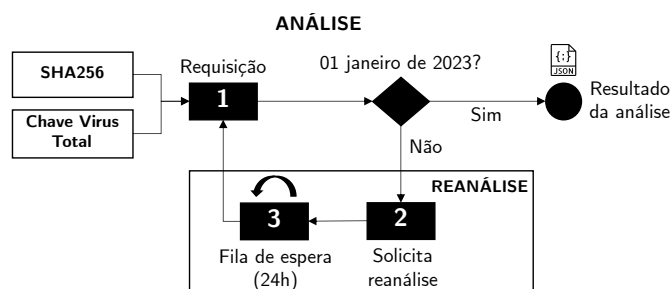


Figura 2. Fluxograma do módulo de rotulação.

Se o aplicativo nunca foi analisado ou foi analisado em um tempo passado não aceitável (e.g., 12/12/2021 - data inferior a definida pelo usuário) pelo VirusTotal, são necessárias três requisições: (1) requisição, (2) solicitação de re-análise do APK, (3) nova requisição após um tempo pré-definido de espera (e.g., 24 horas), conforme ilustrado no fluxo de análise e reanálise da Figura 2. Assumindo que praticamente todos os APKs precisam de reanálise, como tem sido o caso na nossa coleta de dados de rotulação para milhares de APKs, o número máximo de análises de APKs por dia por chave de API do VirusTotal cai para menos de 200. Em resumo, são necessárias mais chaves de API (ou contrato específico com o VirusTotal) para escalar o processo. É importante destacarmos que esse é um processo necessário para assegurar a atualidade dos metadados que serão utilizados para a rotulação dos aplicativos.

## 2.2. AMExplorer

A ferramenta AMExplorer (Figura 3) realiza a exploração e compilação dos dados de saída dos três módulos da AMGenerator: (a) **aquisição** disponibiliza os metadados fornecidos pelo repositório AndroZoo (1); (b) **extração** disponibiliza metadados (2) e atributos (3) do AndroGuard; e (c) **rotulação** disponibiliza os metadados resultantes do VirusTotal (4). Atualmente, a ferramenta oferece um conjunto de parâmetros de entrada que geram três saídas possíveis: (i) um *dataset* global contendo um **agrupamento dos metadados** para análises ou tabulações diversas (a critério do usuário ou especialistas), (ii) um *dataset* contendo valores discretos de características como chamadas de API e *opcodes*, e (iii) um *dataset* contendo apenas valores binários (0 ou 1) para os conjuntos de características mais frequentemente empregadas na classificação de *malwares* Android, como permissões, intenções e chamadas de API. Para a geração dos *datasets* (ii) e (iii) o usuário deve informar também um limiar de número de *scanners* (e.g., 10) para rotulagem das amostras a partir dos metadados do VirusTotal.

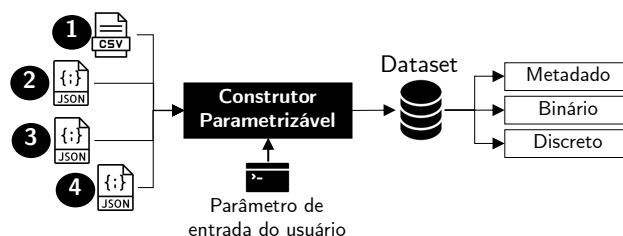


Figura 3. AMExplorer.

## 2.3. Tecnologias

Para a implementação, utilizamos as linguagens Python (3.8 e 3.10) e Bash *scripting* (5.0.17). Utilizamos as bibliotecas AndroGuard (3.3.5) para implementar o módulo de extração e a NetworkX (2.2) para criar o grafo das chamadas de API. Para manipular os arquivos JSON e CSV, utilizamos a biblioteca Pandas (1.3.5) do Python. Também utilizamos outros módulos Python, como o Lxml (4.5.0), Numpy (1.22.3) e Termcolor (2.3.0), para manipulação, tratamento e destaque de dados.

Testamos as ferramentas AMGenerator e AMExplorer em diferentes ambientes, como as distribuições Ubuntu 20.04 e 22.04 e Debian Buster. Os detalhes dos ambientes de teste e a relação completa das dependências podem ser encontrados nos respectivos repositórios GitHub das ferramentas.

## 3. Avaliação

Atualmente, estamos utilizando a ferramenta AMGenerator principalmente para obter metadados atualizados do VirusTotal para mais de 1M de aplicativos. Até o momento, estamos com dados atualizados de 577.050 aplicativos, sendo que 93,66% deles foram para re-análise. Isto demonstra a importância da nova versão do módulo de rotulação. Além dos metadados do VirusTotal, estamos realizando o download, a coleta e a extração de metadados e características desses aplicativos. Ao final, nossa meta é disponibilizar um enorme repositório de dados que poderão ser utilizados para os mais diversos fins pela comunidade de detecção de *malwares* Android.

No restante da avaliação, iremos focar na ferramenta AMExplorer. Na Tabela 1 apresentamos uma comparação entre três conjuntos de dados distintos, nomeadamente *metadata* e *binary* mais *discrete*, em relação ao uso de recursos ao processar diferentes quantidades de APKs. Cada linha fornece *insights* sobre a média do uso da CPU (*CPU\_AVG*, em porcentagem), a memória máxima utilizada (*MEM\_MAX*, em megabytes) e o tempo total de execução (*EXEC\_TIME*, em segundos) para um dado número de APKs. É importante ressaltarmos que os *datasets* binário e discreto foram agrupados devido à relação próxima em termos de geração. Na prática, os valores discretos podem ser facilmente transformados em binários.

**Tabela 1. Consumo de recursos e tempo para diferentes quantidades de APKs.**

N. APKs	Dataset	CPU_AVG (%)	MEM_MAX (MB)	EXEC.TIME (s)
256	metadata	103	19,73	2
	binary+discrete	115	72,36	17
512	metadata	124	39,47	4
	binary+discrete	106	151,30	29
1024	metadata	125	65,78	6
	binary+discrete	104	296,03	91
2048	metadata	120	131,56	12
	binary+discrete	100	677,58	345
4096	metadata	114	249,98	23
	binary+discrete	100	1605,14	1021
8192	metadata	102	486,80	46
	binary+discrete	100	5249,60	3433

A primeira observação dos resultados é a discrepância entre os dois conjuntos. Independentemente da quantidade de APKs, o conjunto *binary + discrete* consistentemente consome mais recursos se comparado ao conjunto de *metadata*. Esta diferença é mais evidente na coluna *MEM\_MAX*, onde o *binary + discrete* exhibe um uso significativamente maior de memória em comparação com o conjunto *metadata*. Por exemplo, para 8.192 APKs, o *binary + discrete* utiliza mais de 5 GB em memória, enquanto o *metadata* utiliza apenas pouco mais de 480 MB. Essa diferença está intimamente associada a necessidade de explosão no número de colunas entre os *datasets*. Enquanto o conjunto *metadata* agrupa características como permissões em uma única coluna (e.g., uma lista de 50 permissões), o conjunto *binary + discrete* explode as colunas de todas as características básicas de aplicativos Android, como permissões, chamadas de API, intenções e *opcodes*. Na prática, para o conjunto de 8.192 APKs, enquanto que o *dataset metadata* contém apenas 25 colunas, devido ao agrupamento por tipo de característica, os *datasets binary* e *discrete* contém 195.154 colunas.

Diferentemente da memória, o consumo médio de CPU tende a reduzir com o aumento no número de APKs. Por exemplo, enquanto a ferramenta chega a consumir 125% (i.e., 1,25 cores) para 1.024 APKs para o conjunto *metadata*, consome apenas na média 102% para 8.192 APKs. Isto ocorre devido ao fato da ferramenta possuir mais operações intensivas de memória do que de CPU. Quanto maior o número de APKs a ser processado, maior será o reflexo sobre a memória, isto é, a ferramenta gasta muito mais tempo em operações de memória do que ciclos de CPU.

Podemos observar também que o tempo de execução aumenta de forma linear para o conjunto *metadata* e exponencial para o conjunto *binary + discrete*. Esse

comportamento está diretamente relacionado à explosão no número de colunas dos *datasets* binário e discreto. Na prática, mais aplicativos podem levar a novas colunas de chamadas de API e *opcodes*, por exemplo.

Também realizamos uma avaliação de um *dataset* gerado pela ferramenta AMExplorer, denominado MH-8K e disponibilizado no GitHub, que contém 8.145 amostras, sendo 7.514 benignas e 631 malignas. A rotulagem levou em consideração o número mínimo de 4 *scanners* positivos do VirusTotal, conforme identificado em estudo recente [Bragança et al., 2023]. Vale ressaltar que este é um parâmetro configurável da ferramenta. O MH-8K possui 195.152 características das categorias permissões, atividades, serviços, receptores, provedores, intenções, chamadas de API e *opcodes*.

Como o número de características do MH-8K é muito elevado, realizamos uma seleção utilizando o Qui-quadrado (*chi-square*), que mede a dependência entre variáveis estocásticas (i.e., entre cada característica e a classe alvo). Para a nossa avaliação, selecionamos as 2.000 melhores características segundo o Qui-quadrado.

Na avaliação do MH-8K utilizamos o classificador *Random Forest* e particionamos os dados entre treino (80%) e teste (20%). O classificador apresentou uma precisão de 97,83%, *recall* de 98,61% e F1-score de 98,22% para a classe benigna de aplicativos, como pode ser visto na Tabela 2. Esses resultados sugerem que modelo consegue identificar e categorizar corretamente os aplicativos benignos. No entanto, o desempenho do modelo é inferior para aplicativos malignos, pois a precisão reduz para 80,56%, o *recall* para 72,50%, e o F1-score para 76,32%.

**Tabela 2. Resultados do classificador RandomForest.**

Classes	<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>	Suporte
Benigno	0.9783	0.9860	0.9821	1509
Malware	0.8055	0.7250	0.7631	120
Acurácia				0.9668

Uma explicação para essa diferença de desempenho pode estar relacionada à quantidade de características selecionadas com o Chi-quadrado ou ao desbalanceamento no conjunto de dados, que é significativamente menor para *malwares*. A utilização de técnicas de balanceamento de dados e a incorporação de características adicionais pode melhorar a capacidade de classificação de amostras malignas.

## 4. Considerações Finais

Neste trabalho apresentamos duas ferramentas: AMGenerator e AMExplorer. Enquanto a ferramenta AMGenerator automatiza e sistematiza a coleta de metadados de aplicativos Android, a AMExplorer permite a exploração dos metadados e a geração de diferentes conjuntos de dados. É importante destacarmos que a AMGenerator, uma evolução da ADBuilder, é a primeira ferramenta a sistematizar e garantir a automação e a correta coleta de dados de rotulação atualizados de serviços como o VirusTotal, um problema antigo e de longa data. Acreditamos que estas ferramentas são um importante passo na direção de modelos de aprendizado de máquina mais eficazes e realistas, isto é, treinados e otimizados com dados atualizados.

Como **trabalhos futuros** podemos destacar: (a) incorporar novos serviços de rotulação na AMGenerator; (b) otimizar o consumo de recursos da AMExplorer na geração de *datasets* com enormes quantidades de APKs; e (c) incorporar na AMExplorer técnicas como a Maat [Salem et al., 2021] para rotular as amostras.

Finalmente, a **demonstração** das ferramentas será realizada através de um ambiente hospedado em um dispositivo próprio dos autores. As funcionalidades das ferramentas serão apresentadas através dos seguintes passos: (i) apresentação dos parâmetros e opções de execução; (ii) demonstração da execução fim-a-fim de cada ferramenta; (iii) apresentação e discussão sobre os metadados coletados pela AMGenerator; e (iv) apresentação dos *datasets* gerados pela AMExplorer.

## 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 CAPES – Código de Financiamento 001.

## Referências

- Bragança, H., Rocha, V., Souto, E., Kreutz, D., and Feitosa, E. (2023). Capturing the behavior of android malware with mh-100k: A novel and multidimensional dataset. In *XXIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*.
- Choo, E., Nabeel, M., De Silva, R., Yu, T., and Khalil, I. (2022). A large scale study and classification of virustotal reports on phishing and malware urls. *arXiv preprint arXiv:2205.13155*.
- Salem, A., Banescu, S., and Pretschner, A. (2021). Maat: Automatically analyzing virus-total for accurate labeling and effective malware detection. *ACM TOPS*, 24(4).
- Soares, T., Mello, J., Barcellos, L., Sayyed, R., Siqueira, G., Casola, K., Costa, E., Gustavo, N., Feitosa, E., and Kreutz, D. (2021a). Detecção de Malwares Android: Levantamento empírico da disponibilidade e da atualização das fontes de dados. In *WRSeg21*.
- Soares, T., Siqueira, G., Barcellos, L., Sayyed, R., Vargas, L., Rodrigues, G., Assolin, J., Pontes, J., Feitosa, E., and Kreutz, D. (2021b). Detecção de Malwares Android: datasets e reprodutibilidade. In *WRSeg21*.
- Vilanova, L., Kreutz, D., Assolin, J., Quincozes, V., Miers, C., Mansilha, R., and Feitosa, E. (2022). Adbuilder: uma ferramenta de construção de datasets para detecção de malwares android. In *Anais Estendidos do XXII SBSeg*, pages 143–150. SBC.
- Wang, H., Liu, Z., Liang, J., Vallina-Rodriguez, N., Guo, Y., Li, L., Tapiador, J., Cao, J., and Xu, G. (2018). Beyond google play: A large-scale comparative study of chinese android app markets. In *IMC*, pages 293–307.
- Wang, H., Si, J., Li, H., and Guo, Y. (2019). Rmvdroid: towards a reliable android malware dataset with app metadata. In *IEEE/ACM 16th MSR*, pages 404–408. IEEE.
- Wei, F., Li, Y., Roy, S., Ou, X., and Zhou, W. (2017). Deep ground truth analysis of current android malware. In *Int. Conf. on DIMVA*, pages 252–276. Springer.
- Zhu, S., Shi, J., Yang, L., Qin, B., Zhang, Z., Song, L., and Wang, G. (2020). Measuring and modeling the label dynamics of online engines. In *USENIX Security*.