Classificação de Malwares Android com Análise Dinâmica Segmentada e Estratégia Multivisualização

Ivson Ferreira[1], João Victor O. de Oliveira[1], Fernando Purkott[1], Jhonatan Geremias[1] e Eduardo K. Viegas[1]

[1]Programa de Pós-Graduação em Informática (PPGIa)
Pontifícia Universidade Católica do Paraná (PUCPR)
80.215-901 – Curitiba – PR

{ivson.sjunior, joao.oliveira, fernando.purkott,
jhonatan.geremias, eduardo.viegas}@ppgia.pucpr.br

Resumo. *Neste artigo, propomos um modelo dinâmico de detecção de malwares Android em duas fases: extração de características com base em janelas temporais e classificação multivisualização. A primeira fase segmenta a execução do app em curtos intervalos, permitindo extração rápida e de baixa latência. A segunda aplica classificadores em diferentes visualizações, cujos resultados são combinados por votação majoritária, aumentando a precisão sem sobrecarregar o sistema. Avaliamos a abordagem com um novo conjunto de dados contendo 4.128 APKs, abrangendo amostras benignas e maliciosas de nove famílias. Os resultados mostram ganho de até 0,06 na AUC em relação a métodos de visualização única.*

1. Introdução

Diversos estudos nos últimos anos propuseram novas estratégias de detecção de *malware* Android baseadas em abordagem dinâmica, com técnicas de Machine Learning (ML) apresentando os resultados mais promissores [Bashir et al. 2024]. Para isso, um modelo de ML comportamental é construído analisando um conjunto de dados de treinamento, normalmente composto por um grande número de variantes de comportamento de *malware* e *goodware*. Como a detecção baseada em ML depende da análise do comportamento do Android Application Package (APK) associado, uma tarefa crucial gira em torno da definição do conjunto de características (ou visualizações) que servirão como entrada para o modelo [Abreu et al. 2017]. Nesse caso, os pesquisadores frequentemente se baseiam em uma estratégia de visualização única, normalmente focando na análise do comportamento do APK com base em recursos como chamadas de sistema executadas, negligenciando os arquivos acessados, por exemplo [Xiao et al. 2017]. Essa limitação geralmente se deve aos desafios significativos envolvidos na extração de características comportamentais em tempo de execução e na combinação eficaz de múltiplas visualizações durante a fase de classificação.

A extração dinâmica de características requer a execução adequada do APK analisado durante um período definido, normalmente variando de alguns minutos a horas [Cui et al. 2023]. Embora o monitoramento do APK durante um intervalo prolongado em um ambiente *sandbox* seja viável, a execução no dispositivo apresenta desafios maiores. Dado o poder de processamento limitado dos *smartphones* e as restrições de bateria, a detecção de *malware* deve ser otimizada para minimizar os requisitos de tempo [Shrestha

et al. 2023]. Por outro lado, a literatura frequentemente assume que a tarefa de detecção pode ser realizada sem restrições de tempo, ignorando a necessidade de esquemas que permitam a execução no dispositivo [Simioni et al. 2025]. Pesquisadores normalmente assumem que o APK analisado é monitorado dentro de um ambiente *sandbox*, enquanto a implementação de seus esquemas no próprio *smartphone* é frequentemente negligenciada.

Para uma detecção confiável de *malware* Android baseado em dinâmica, as abordagens propostas devem avaliar o comportamento do APK incorporando visualizações múltiplas e complementares. Por exemplo, a identificação de um *malware* de exfiltração de dados geralmente requer a correlação do grande número de arquivos acessados com anomalias em recursos relacionados à rede para confirmar a intenção maliciosa. Isso só pode ser realizado avaliando conjuntamente os recursos relacionados aos arquivos e suas contrapartes relacionadas à rede [Sabir et al. 2021]. Embora a extração com múltiplas visualizações no dispositivo apresente desafios significativos, a combinação eficaz desses recursos durante tarefas de classificação é igualmente complexa. Isso ocorre porque os comportamentos de múltiplas visualizações devem ser analisados e integrados, minimizando as demandas de processamento [Horchulhack et al. 2024].

Contribuição. Diante disso, este artigo propõe uma nova ferramenta para detecção de *malware* em dispositivos Android por meio de análise dinâmica multivisualização, implementada de duas maneiras. Primeiramente, propomos um novo mecanismo de extração de características baseado em janelas temporais para minimizar a sobrecarga de processamento do monitoramento de APKs em dispositivos. O modelo proposto captura o comportamento do APK em cada janela de tempo avaliada, visando reduzir o custo computacional da extração. Cada janela de tempo encapsula o comportamento do aplicativo Android analisado, representado pelos recursos extraídos dentro do intervalo definido. Nossa principal ideia é reduzir os custos de processamento discretizando o comportamento do APK em intervalos de tempo distintos, permitindo que a extração de características seja realizada de forma mais eficiente em um período menor. Em segundo lugar, abordamos a tarefa de classificação usando uma estratégia de análise dinâmica multivisualização para Android.

2. Trabalhos Relacionados

Em geral, os esquemas propostos de detecção dinâmica de malware para Android focam em aumentar a precisão da detecção em um determinado conjunto de dados [Espindola et al. 2021, Filho et al. 2025]. Como exemplo, R. Yumlembam *et al.* [Yumlembam et al. 2023] avaliam o recurso de importância da chamada de API para melhorar a precisão de algoritmos de ML amplamente utilizados. Seu esquema pode reduzir o número de recursos utilizados sem comprometer a precisão, mas ignora os custos computacionais associados à sua extração. J. Li *et al.* [Li et al. 2024] pré-processa o conjunto de dados do Android para lidar com o desequilíbrio de classes. Seu esquema melhora a precisão com um classificador Random Forest (RF), mas ignora os custos de processamento associados à fase de extração de características. Os custos de processamento associados à detecção dinâmica de malware para Android raramente são considerados na literatura. Em geral, quando um novo conjunto de recursos é proposto, apenas os benefícios de precisão resultantes são avaliados, negligenciando os custos de processamento associados à sua extração. H. Liu *et al.* [Liu et al. 2024] concentra-se na extração de dependências de curto e longo prazo de sequências de Opcode. Seu esquema melhora a precisão por meio

de um modelo DNN, mas ignora os custos de processamento associados à extração de características. Da mesma forma, D. Zou *et al.* [Zou et al. 2021] baseia-se em uma abordagem baseada em grafos para calcular a intimidade entre chamadas de API sensíveis. As características resultantes permitem a detecção com alta precisão, mas negligenciam os custos de processamento associados à fase de construção do gráfico. Y. K. Sharma *et al.* [Sharma et al. 2025] extrai sequências de opcodes em nível de método como uma representação codificada do APKs. Sua abordagem de extração de características melhora a precisão, mas os custos de processamento associados à tarefa de codificação são ignorados.

3. Um modelo de detecção dinâmica de malware para Android em um intervalo de tempo limitado

Para abordar essas limitações, propomos um novo modelo dinâmico de detecção de malware para Android com múltiplas visualizações, baseado na extração de recursos por janela de tempo, conforme ilustrado na Figura 1, implementado de duas maneiras. O primeiro estágio, *Extração de características por Janela Temporal*, foi projetado para otimizar o processo de extração de recursos empregando uma abordagem de janela de tempo em cascata. Em vez de monitorar continuamente o comportamento do aplicativo por um longo período, o que é computacionalmente custoso, esse método discretiza o processo de extração de recursos em intervalos de tempo fixos. Em cada janela, recursos dinâmicos relevantes, como chamadas de API, atividades de rede e acessos a arquivos, são capturados e processados. Essa abordagem reduz significativamente o tempo necessário para a extração de recursos, pois o modelo pode coletar dados comportamentais suficientes para classificação sem a necessidade de observação prolongada.

O segundo estágio é a *Classificação por Multivisualização*, em que os recursos extraídos de cada janela de tempo são classificados usando uma abordagem de conjunto com múltiplas visualizações. Cada visualização representa uma perspectiva comportamental distinta da aplicação, como interações de rede, operações de arquivo ou chamadas de sistema, e está associada a um classificador leve dedicado. Esses classificadores operam de forma independente, analisando os recursos de suas respectivas visualizações e produzindo resultados de classificação. A decisão final é derivada da combinação das saídas desses classificadores por meio de um esquema de votação majoritária ou outra técnica de conjunto adequada. Este design permite a detecção leve de malware, mantendo alta precisão, já que cada classificador é especializado em analisar um aspecto específico do comportamento da aplicação.

3.1. Um modelo de extração de recursos de janela temporal

As abordagens tradicionais de detecção dinâmica de malware para Android normalmente monitoram o APK analisado por longos intervalos de tempo, capturando seu comportamento por um período prolongado para garantir que todas as ações maliciosas sejam observadas. No entanto, esse monitoramento contínuo demanda recursos computacionais substanciais, tornando-o inviável para smartphones e outros dispositivos com recursos limitados. Diante disso, nosso modelo proposto aborda esse desafio discretizando o processo de extração de recursos em janelas temporais fixas, onde cada janela representa um período de observação limitado. Os recursos comportamentais são extraídos do APK dentro de cada janela temporal, e esses recursos são então usados como entrada para o modelo

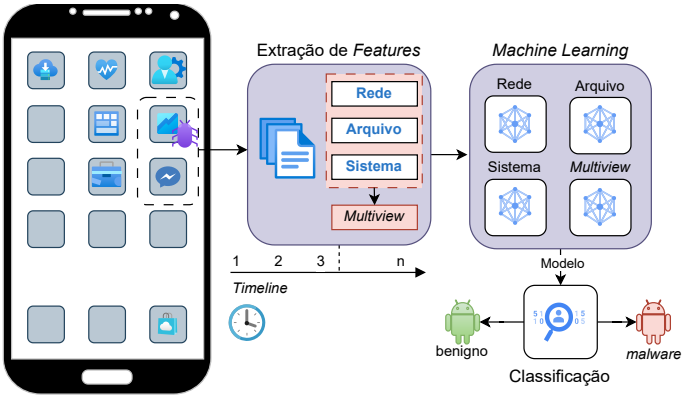

Figura 1. Arquitetura de alto nível do modelo proposto.

de classificação. Como resultado, essa abordagem reduz significativamente o tempo de processamento necessário para a extração de recursos, pois evita a necessidade de longos períodos de observação, ao mesmo tempo em que captura características comportamentais suficientes para uma detecção precisa de malware.

Seja x um vetor de recursos de tamanho N representando as características comportamentais de um aplicativo Android capturadas durante uma janela temporal específica. Nosso processo de extração de recursos de janela temporal é definido da seguinte forma: Dado um rastreamento de execução $T = \{e_1, e_2, \ldots, e_M\}$, onde cada e_i denota um evento registrado (como uma chamada de *API*, solicitação de rede ou acesso a um arquivo), o rastreamento T é dividido em uma sequência de janelas temporais não sobrepostas $W = \{w_1, w_2, \ldots, w_K\}$. Para cada janela w_i, um vetor de características de tamanho N é extraído. Por exemplo, o número de chamadas de API, o volume de dados de rede transmitidos ou a contagem de operações de leitura de arquivo. Esse processo produz uma sequência de vetores de características que são posteriormente usados como entradas para o modelo de classificação.

Por exemplo, um APK pode ser monitorado usando uma janela temporal inicial de 5 segundos, durante a qual seus recursos comportamentais — como o número de chamadas de API, o volume de dados de rede transmitidos e as frequências de acesso a arquivos — são capturados. Esses recursos são extraídos e imediatamente usados como entrada para o modelo de classificação. Se a classificação resultante for considerada não confiável — seja devido a uma pontuação de confiança baixa ou a resultados de classificação conflitantes de múltiplas visualizações — o modelo pode estender o período de monitoramento por mais 5 segundos. Essa extensão produz um novo conjunto de recursos extraídos da janela estendida de 10 segundos, que são então combinados com os recursos iniciais para reclassificação. Essa abordagem adaptativa permite que o modelo classifique rapidamente casos simples dentro da janela inicial de 5 segundos, mantendo a flexibilidade para realizar um monitoramento mais abrangente para casos desafiadores, garantindo maior confiabilidade da classificação sem impor custos de processamento desnecessários.

3.2. Classificação leve de múltiplas visualizações

A classificação multivisualização é essencial na detecção dinâmica de malware em Android para garantir uma classificação confiável, pois permite que o modelo de detecção avalie o comportamento do aplicativo sob múltiplas perspectivas, como atividade de rede, interações no sistema de arquivos e uso de API. No entanto, as abordagens existentes

na literatura frequentemente alcançam isso empregando modelos complexos, como Deep Neural Networks (DNNs), que aumentam significativamente os custos computacionais. Esses métodos que consomem muitos recursos são impraticáveis para implementação em dispositivos com recursos limitados, como smartphones, onde o poder de processamento, a memória e a duração da bateria são limitados. Para resolver esse problema, propomos uma abordagem de conjunto leve que mantém os benefícios da classificação multivisualização sem a sobrecarga computacional associada. Nossa abordagem aplica um classificador leve dedicado a cada visualização, permitindo que cada uma analise independentemente suas características comportamentais correspondentes. O resultado final da classificação é então determinado pela combinação dos resultados desses classificadores individuais, por exemplo, por votação majoritária. Esse design garante uma classificação multivisualização eficiente e escalável, adequada para implantação no dispositivo.

Seja $E = \{h_1, h_2, \ldots, h_m\}$ um conjunto de m classificadores, onde cada classificador h_i está associado a uma visão distinta do comportamento da aplicação, como atividade de rede, interações no sistema de arquivos ou uso da API. Dado um vetor de características de entrada x, cada classificador h_i produz um resultado de classificação independente $y_i \in \{0, 1\}$, onde $y_i = 1$ indica uma classificação de malware e $y_i = 0$ indica uma classificação benigna. A decisão final de classificação y é determinada usando um esquema de votação majoritária. Este procedimento de votação majoritária permite que o modelo aproveite os pontos fortes de múltiplas visualizações, mantendo a eficiência computacional, tornando-o adequado para implantação em dispositivos com recursos limitados.

4. Protótipo

Implementamos um protótipo proposto em um esquema dinâmico de detecção de malware para Android. Cada APK analisado foi executado em um ambiente sandbox usando o DroidBox $4.1.1$, o que nos permitiu capturar seu comportamento dinâmico sob condições controladas. Para aprimorar ainda mais o processo de extração de recursos, integramos o AndroPyTool [Martín et al. 2018], o que permitiu a coleta de dados comportamentais adicionais. Especificamente, utilizamos o AndroPyTool para extrair o rastro do sistema (strace) de cada execução do APK, fornecendo métricas associadas às chamadas do sistema. Além disso, essa ferramenta facilitou o monitoramento do uso da rede e dos padrões de acesso a arquivos. Cada APK foi executado por um período predefinido, durante o qual seu comportamento foi monitorado, e os dados resultantes foram processados para gerar os vetores de recursos para classificação. As características extraídas foram categorizados em três visualizações distintas, cada uma representando um aspecto específico do comportamento do aplicativo:

- ***Rede***. Essas características capturam as interações de rede do aplicativo, incluindo o número total de conexões de rede, o volume de dados transmitidos e recebidos, o número de endereços IP distintos acessados e os tipos de protocolos utilizados;
- ***Sistema***. Essa visão se concentra nos comportamentos em nível de sistema, incluindo a frequência e os tipos de chamadas de sistema realizadas, o número de criações de processos, alocações de memória e o uso de recursos específicos do sistema;

- ***Arquivo***. Características que descrevem as interações do aplicativo com o sistema de arquivos, como o número de arquivos criados, excluídos, lidos ou gravados, bem como o volume total de dados acessados no sistema de arquivos;

Cada visualização de recursos é extraída para cada janela temporal predefinida, permitindo que nosso protótipo capture o comportamento evolutivo do APK analisado ao longo do tempo. Especificamente, nossa implementação considera quatro janelas temporais, com cada janela representando um intervalo médio de 53 segundos da execução do APK. Consequentemente, o APK é monitorado por um período total de 212 segundos, durante os quais seu comportamento dinâmico é registrado. Após a fase de execução, os recursos extraídos de cada visualização — Rede, Sistema e Arquivo — são processados usando a API Pandas $v.2.2.3$, que facilita a manipulação de dados e a agregação de informação. Esses recursos processados são então usados como entrada para o módulo de classificação, que é implementado usando a API scikit-learn $v.1.6.1$.

5. Avaliação

Nossos experimentos conduzidos visam responder às seguintes perguntas: Research Questions (RQs):

- *RQ1 - Como nossa proposta de detecção de malware multivisualização para Android melhora a precisão da classificação?*
- *RQ2 - Qual é o impacto na precisão do nosso esquema proposto de extração de recursos de janela temporal?*

5.1. Construção de conjunto de dados

Os conjuntos de dados de detecção de malware existentes para Android concentram-se predominantemente em análises estáticas, nas quais os recursos são extraídos diretamente do arquivo APK sem executá-lo. Mesmo no caso menos comum de conjuntos de dados dinâmicos, a maioria não adota uma configuração de múltiplas visualizações, na qual os recursos comportamentais são categorizados com base em diferentes aspectos da atividade do aplicativo, como uso da rede, interações com o sistema e acesso a arquivos. Para superar essas limitações, construímos um novo conjunto de dados dinâmico e multivisualização usando o protótipo descrito anteriormente (consulte Seção 4). Nossa abordagem permite uma análise comportamental abrangente, capturando as atividades do aplicativo em múltiplas visualizações, fornecendo um conjunto diversificado de recursos para a detecção de malware.

O conjunto de dados construído consiste na análise de 4.128 amostras de APK, incluindo 2.145 amostras benignas e 1.983 amostras de malware. As amostras de malware são categorizadas em 9 famílias distintas, garantindo a representação de diversos comportamentos maliciosos. Amostras benignas foram selecionadas baixando os aplicativos mais populares da Google Play Store, garantindo que representassem padrões de uso legítimos. Em contraste, amostras de malware foram obtidas do VirusTotal, uma plataforma bem estabelecida que agrega malware de diversas fontes.

O conjunto de dados resultante foi dividido em três subconjuntos: *treinamento*, *teste* e *validação*, compreendendo 40%, 30%, e 30% do total de amostras, respectivamente. O conjunto *treinamento* é usado para ajustar o modelo de classificação, permitindo que ele aprenda os padrões e características que diferenciam aplicativos benignos

Tabela 1. Métricas de desempenho dos classificadores.

Feature View	Classifier	AUC	F1	TNR	FPR	FNR	TPR
Arquivo	DT	0.77	0.77	0.74	0.26	0.20	0.80
	RF	0.86	0.86	0.91	0.09	0.18	0.82
	XGB	0.83	0.83	0.84	0.16	0.18	0.82
Rede	DT	0.78	0.78	0.77	0.23	0.21	0.79
	RF	0.86	0.86	0.89	0.11	0.17	0.83
	XGB	0.82	0.82	0.84	0.16	0.20	0.80
Sistema	DT	0.76	0.76	0.73	0.27	0.21	0.79
	RF	0.85	0.85	0.91	0.09	0.20	0.80
	XGB	0.82	0.82	0.84	0.16	0.20	0.80
Completo	DT	0.79	0.79	0.78	0.22	0.20	0.80
	RF	0.88	0.88	0.94	0.06	0.18	0.82
	XGB	0.85	0.85	0.85	0.15	0.16	0.84
Nossa (Multi-view)	RF	0.88	0.88	0.93	0.07	0.18	0.82

de malware. O conjunto *teste* é usado para avaliar o desempenho do modelo, fornecendo uma avaliação imparcial de sua exatidão, precisão, recall e outras métricas em dados nunca vistos anteriormente. Finalmente, o conjunto *validação* é empregado para ajustar os hiperparâmetros do modelo, garantindo que ele generalize bem para novos dados sem sobreajuste.

5.2. Construção de modelos

Avaliamos o desempenho do nosso modelo proposto com 3 classificadores diferentes, a saber: Decision Tree (DT), RF e Gradient Boosted Decision Trees (XGB). O DT é configurado com o critério de impureza de Gini, uma profundidade máxima de 10 e um mínimo de 2 amostras necessárias para dividir um nó interno. O modelo RF é configurado com 100 árvores, utilizando o critério de Gini, uma profundidade máxima de 10 e bootstrapping habilitado. Por fim, o classificador XGB é configurado com 100 estimadores, uma taxa de aprendizado de $0,1$ e uma profundidade máxima de 6. Essas configurações foram selecionadas com base em parâmetros comumente utilizados na literatura e validadas por meio de testes empíricos no conjunto de validação.

5.3. Desempenho de extração de recursos por janela temporal

Nosso primeiro experimento visa responder à RQ1 e investiga o desempenho de precisão do nosso esquema de classificação multivisualização proposto. Para conduzir esta avaliação, aplicamos os classificadores selecionados usando uma abordagem de janela temporal integral, na qual cada classificador recebe como entrada os recursos concatenados extraídos de todas as janelas temporais. Nosso esquema proposto é implementado por meio de um procedimento de votação majoritária, onde cada visualização — Arquivo, Rede e Sistema — é associada a um classificador independente do mesmo tipo. Por exemplo, ao utilizar o classificador RF, instanciamos três modelos RF, cada um treinado em uma das visualizações, e agregamos suas previsões por votação majoritária para produzir a saída final da classificação. Este experimento nos permite analisar o impacto da combinação de múltiplas perspectivas comportamentais na precisão da detecção, mantendo as propriedades leves necessárias para a execução no dispositivo.

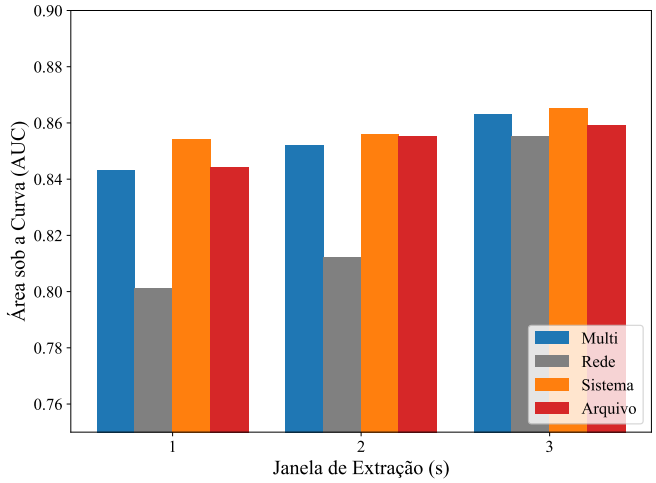

Figura 2. AUC do classificador Random Forest em diferentes visualizações.

Tabela 1 mostra a precisão da classificação para os classificadores selecionados em diferentes visualizações de recursos ao usar todas as janelas temporais. Os resultados indicam que há uma variação de desempenho entre as diferentes visualizações de recursos. Ao considerar o classificador RF como exemplo, observamos que a visualização de Rede produz uma AUC de entre $0,80$ e $0,86$, a visualização de Arquivo atinge uma AUC ligeiramente mais estável entre $0,84$ e $0,86$ e a visualização de Sistema fornece um resultado intermediário com uma AUC de entre $0,82$ e $0,85$. Essa variação destaca que cada perspectiva comportamental captura características distintas da execução do APKs. Por exemplo, a visualização de Rede pode oferecer padrões mais discriminativos para identificar comportamentos maliciosos que envolvem comunicação com servidores externos, enquanto a visualização de Arquivo pode ser mais eficaz na detecção de malware que manipula ou acessa o armazenamento local. As diferenças observadas enfatizam a importância da incorporação de múltiplas visualizações para uma compreensão mais abrangente do comportamento do malware.

Nossa abordagem de classificação *multi-view* proposta demonstra benefícios claros em termos de maior precisão de detecção. Ao aplicar a estratégia de votação majoritária nas visualizações de Arquivo, Rede e Sistema usando o classificador RF, o modelo atinge uma Area Under the Curve (AUC) de $0,88$. Este resultado reflete uma melhoria notável em comparação com o melhor resultado de visualização única, mostrando um ganho de até $0,02$ na AUC. Esta melhoria confirma que o aproveitamento de informações complementares de múltiplas visualizações comportamentais melhora o desempenho geral da classificação. É importante ressaltar que nossa abordagem de conjunto permanece leve e adequada para implantação em ambientes com recursos limitados, validando sua implementação em cenários reais de detecção de malware em dispositivos.

Nosso segundo experimento visa responder à RQ2 e investiga o desempenho da precisão de acordo com o tamanho da janela temporal utilizada para o módulo de extração de recursos. O objetivo é examinar se o aumento do intervalo de extração de recursos pode influenciar positivamente a precisão geral do sistema. Para tanto, treinamos os classificadores selecionados usando janelas temporais cumulativas que variam de 53 segundos a 212 segundos e avaliamos seus valores de AUC correspondentes. Quando uma janela

temporal maior é considerada, concatenamos os vetores de características extraídos de cada intervalo de tempo constituinte e usamos o vetor resultante como entrada para o modelo ML subjacente.

Figura 2 mostra a precisão da classificação de acordo com a janela temporal de extração de recursos utilizada e a visualização utilizada para o classificador RF. É possível observar que o aumento da janela temporal de extração de recursos tem um impacto benéfico na precisão resultante do modelo ML subjacente. Essa tendência é consistente em todas as visualizações consideradas. Por exemplo, quando a janela temporal aumenta de 53 segundos para 106 segundos, a AUC melhora em $0,06$ para a visualização de Rede, $0,03$ para a visualização de Sistema e $0,02$ para a visualização de Arquivo. Essas melhorias destacam a importância de observar o comportamento do aplicativo por períodos mais longos, pois permite que o modelo capture padrões de comportamento mais representativos. Consequentemente, isso contribui para um processo de classificação mais robusto, particularmente em cenários onde períodos curtos de observação podem não revelar sinais suficientes de comportamento malicioso.

Além das visualizações individuais, a solução de múltiplas visualizações também se beneficia dessa característica. Conforme demonstrado em nossos resultados, a AUC da classificação multivisualização aumenta de $0,85$ para $0,88$ quando a janela temporal de extração de recursos é estendida de 53 para 212 segundos — uma melhoria de $0,03$. Essa descoberta demonstra que nossa abordagem de janela temporal proposta não apenas melhora a precisão do modelo à medida que a janela de observação aumenta, mas também permite uma implementação eficaz em um ambiente *multi-view*.

6. Conclusão

Neste trabalho, propusemos um modelo dinâmico de detecção de malware para Android com múltiplas visualizações que integra uma estratégia leve de extração de recursos baseada em janela temporal. O modelo extrai dados comportamentais das visualizações de Rede, Sistema e Arquivo em curtos intervalos de tempo e os classifica usando um conjunto de classificadores independentes com votação majoritária. Este projeto permite uma estrutura flexível adequada para implantação em ambientes com recursos limitados. Nossa avaliação demonstrou que o modelo proposto melhora a AUC em comparação com as abordagens de visualização única, enquanto o aumento da janela de extração de recursos aumenta ainda mais a precisão em todas as visualizações.

Agradecimentos

Este trabalho foi parcialmente financiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), número do processo 302937/2023-4, 407879/2023-4, e 442262/2024-8.

Referências

Abreu, V., Santin, A. O., Viegas, E. K., and Stihler, M. (2017). A multi-domain role activation model. In *2017 IEEE International Conference on Communications (ICC)*, page 1–6. IEEE.

Bashir, S., Maqbool, F., Khan, F. H., and Abid, A. S. (2024). Hybrid machine learning model for malware analysis in android apps. *Pervasive and Mobile Computing*, 97:101859.

Cui, Y., Sun, Y., and Lin, Z. (2023). Droidhook: a novel api-hook based android malware dynamic analysis sandbox. *Automated Software Engineering*, 30(1).

Espindola, A., Viegas, E. K., Traleski, A., Pellenz, M. E., and Santin, A. O. (2021). A deep autoencoder and rnn model for indoor localization with variable propagation loss. In *2021 17th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE.

Filho, A. G., Viegas, E. K., Santin, A. O., and Geremias, J. (2025). A dynamic network intrusion detection model for infrastructure as code deployed environments. *Journal of Network and Systems Management*, 33(4).

Horchulhack, P., Viegas, E. K., Santin, A. O., and Simioni, J. A. (2024). Network-based intrusion detection through image-based cnn and transfer learning. In *2024 International Wireless Communications and Mobile Computing (IWCMC)*, page 386–391. IEEE.

Li, J., He, J., Li, W., Fang, W., Yang, G., and Li, T. (2024). Syndroid: An adaptive enhanced android malware classification method based on ctgan-svm. *Computers amp; Security*, 137:103604.

Liu, H., Gong, L., Mo, X., Dong, G., and Yu, J. (2024). Ltachecker: Lightweight android malware detection based on dalvik opcode sequences using attention temporal networks. *IEEE Internet of Things Journal*, 11(14):25371–25381.

Martín, A., Lara-Cabrera, R., and Camacho, D. (2018). A new tool for static and dynamic android malware analysis. In *Data Science and Knowledge Engineering for Sensing Decision Support*, page 509–516. WORLD SCIENTIFIC.

Sabir, B., Ullah, F., Babar, M. A., and Gaire, R. (2021). Machine learning for detecting data exfiltration: A review. *ACM Computing Surveys*, 54(3):1–47.

Sharma, Y. K., Tomar, D. S., Pateriya, R., and Bhandari, S. (2025). Mosdroid: Obfuscation-resilient android malware detection using multisets of encoded opcode sequences. *Computers amp; Security*, 152:104379.

Shrestha, S., Pathak, S., and Viegas, E. K. (2023). Towards a robust adversarial patch attack against unmanned aerial vehicles object detection. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page 3256–3263. IEEE.

Simioni, J. A., Viegas, E. K., Santin, A. O., and de Matos, E. (2025). An energy-efficient intrusion detection offloading based on dnn for edge computing. *IEEE Internet of Things Journal*, 12(12):20326–20342.

Xiao, X., Zhang, S., Mercaldo, F., Hu, G., and Sangaiah, A. K. (2017). Android malware detection based on system call sequences and lstm. *Multimedia Tools and Applications*, 78(4):3979–3999.

Yumlembam, R., Issac, B., Yang, L., and Jacob, S. M. (2023). Android malware classification and optimisation based on bm25 score of android api. In *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, page 1–6. IEEE.

Zou, D., Wu, Y., Yang, S., Chauhan, A., Yang, W., Zhong, J., Dou, S., and Jin, H. (2021). Intdroid: Android malware detection based on api intimacy analysis. *ACM Transactions on Software Engineering and Methodology*, 30(3):1–32.