APKAnalyzer: Ferramenta de Classificação de *Malwares Android* Baseada em *Multi-view* e Seleção de Características Multiobjetivo

Philipe Fransozi
Jhonatan Geremias
Eduardo K. Viegas
Altair Santin

Pontifícia Universidade Católica do Paraná

Motivação

- Considerando o modelo de classificação de *malware Android* apresentado no artigo *Seleção de Características Multiobjetivo para Detecção de Malwares Android* - (SBSeg'24 - WTICG), construir uma prova de conceito de uma ferramenta que implemente esse modelo e classifique arquivos de aplicativos *Android*.

Tópicos

- **Introdução**
- Trabalhos Relacionados
- Descrição da Ferramenta
- Demonstração da Ferramenta
- Disponibilização da Ferramenta
- Conclusão
- Perguntas

Introdução

- **Contexto Acadêmico**: pesquisa foi realizada durante o PIBIC 2023-24, como resultado três artigos foram produzidos:
  - *Seleção de Características Multiobjetivo para Detecção de Malwares Android,* aceito na SBSeg 2024, categoria WTICG;
  - *APKAnalyzer: Ferramenta de Classificação de Malwares Android Baseada em Multi-view e Seleção de Características Multiobjetivo*, aceito na SBSeg 2024, categoria Salão de Ferramentas;
  - *A Multi-view Android Malware Detection Model Through Multi-objective Optimization*, submetido a ICMLA.

Introdução

- **Contexto dos dispositivos móveis Android:**
  - Em 2024, domínio do mercado com 3/4 de marketshare, aproximadamente 3 bilhões de usuários;
  - Em 2023, houve um aumento na detecção de aplicativos maliciosos, aproximadamente 53% de aumento; grande parte dessas amostras oriundas da loja oficial de aplicativos *Google Play*.

Introdução

- **Contexto dos métodos de análise de malware Android:**
  - **Análise estática** obtém indícios de um comportamento malicioso a partir do conteúdo do aplicativo (por exemplo, permissões obtidas do arquivo manisfest.xml):
    - Trata-se de um método mais simples, rápida implementação e escalável.

Introdução

- **Contexto dos métodos de detecção de malware Android:**
  - Na literatura, encontramos inúmeros métodos para a tarefa de detecção, dentre os quais temos os métodos baseados em modelos de aprendizado de máquina;
  - Em geral: um vetor de características é utilizado como entrada de um modelo de classificação que o classifica como malicioso ou benigno;
  - Vetores de características, originados de diferentes conteúdos de um arquivo, são amplamente utilizados na classificação, e a abordagem *multi-view* tem mostrado melhorar a generalização e confiança do sistema.

Introdução

- **Objetivo Geral:**
  - Implementar ferramenta que utiliza modelos de aprendizado de máquina previamente treinados com a abordagem *multi-view* e seleção de características com otimização multiobjetivo a partir de um novo dataset;
- **Objetivos Específicos:**
  - Dataset com 40 mil amostras de arquivos, 20 mil maliciosas e 20 mil benignas;
  - Um novo modelo de detecção de *malware Android multi-view* implementado com uma estratégia de otimizacção multiobjetivo.

Tópicos

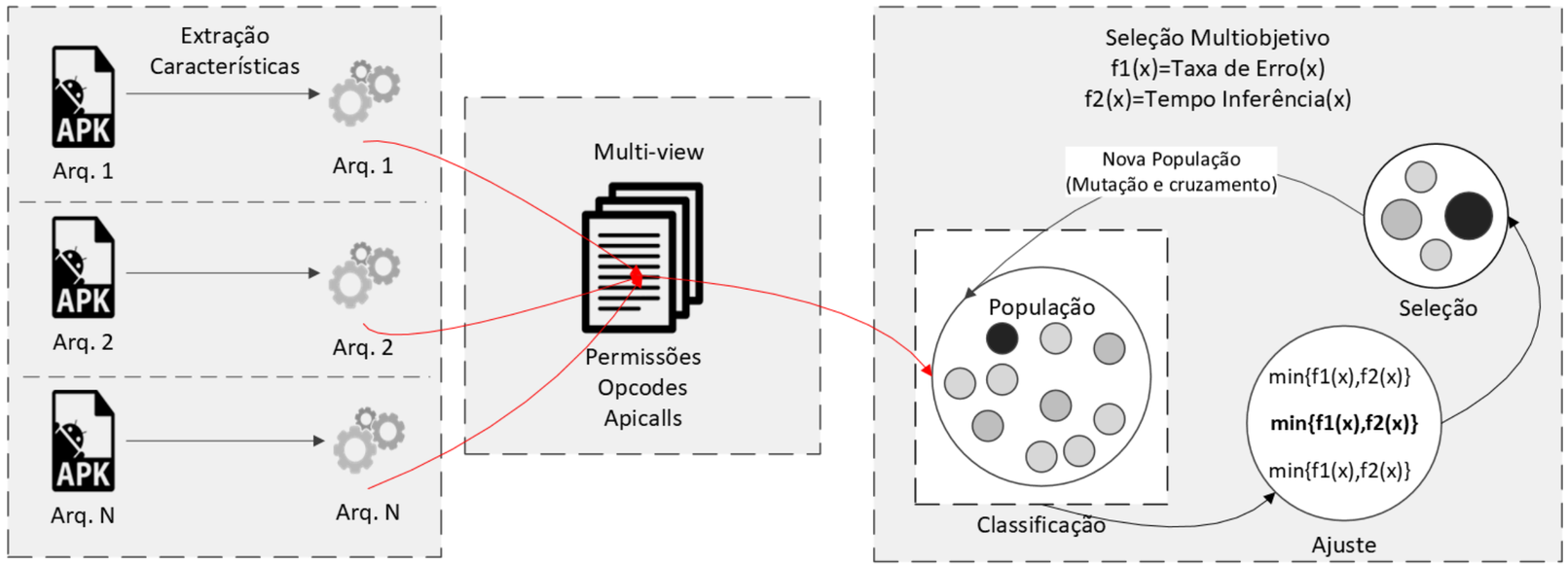
- Introdução
- **Trabalhos Relacionados**
- Descrição
- Demonstração
- Disponibilização
- Conclusão
- Perguntas

Trabalhos Relacionados

- **Artigo Base:**
  - *Seleção de Características Multiobjetivo para Detecção de Malwares Android*;
  - Contribuição na discussão do problema de deteccção de *malware* para *Android* em um cenário *multi-view* utilizando otimização multiobjetivo.

Trabalhos Relacionados

- **Artigo Base:**
  - Para superar o desafio da detecção de *malware Android* em um cenário *multi-view*, o esquema proposto é implementado através de uma abordagem de otimização multiobjetivo:



Trabalhos Relacionados

- **Artigo Base:**
  - Um módulo de extração de características processa cada arquivo de aplicativo *Android* através da ferramenta AndroPyTool;
  - As características extraídas são organizadas em três *datasets* de acordo com cada *view*: permissões, chamadas de API (API Calls) e código de operações (Opcodes);
  - As três *views* que compõem a técnica *multi-view*.

Trabalhos Relacionados

- **Artigo Base:**
  - O módulo de seleção multiobjetivo utiliza *framework* (pymoo) de código genético (NSGA-2) para encontrar um subconjunto de características que otimize dois objetivos, ou seja, reduza a taxa de erro e reduza o tempo de inferência dos modelos de classficação;
  - Nossa proposta melhorou a taxa verdadeiro positivo em uma média de 5,2, exigindo até 65% dos custos com processamento.

Tópicos

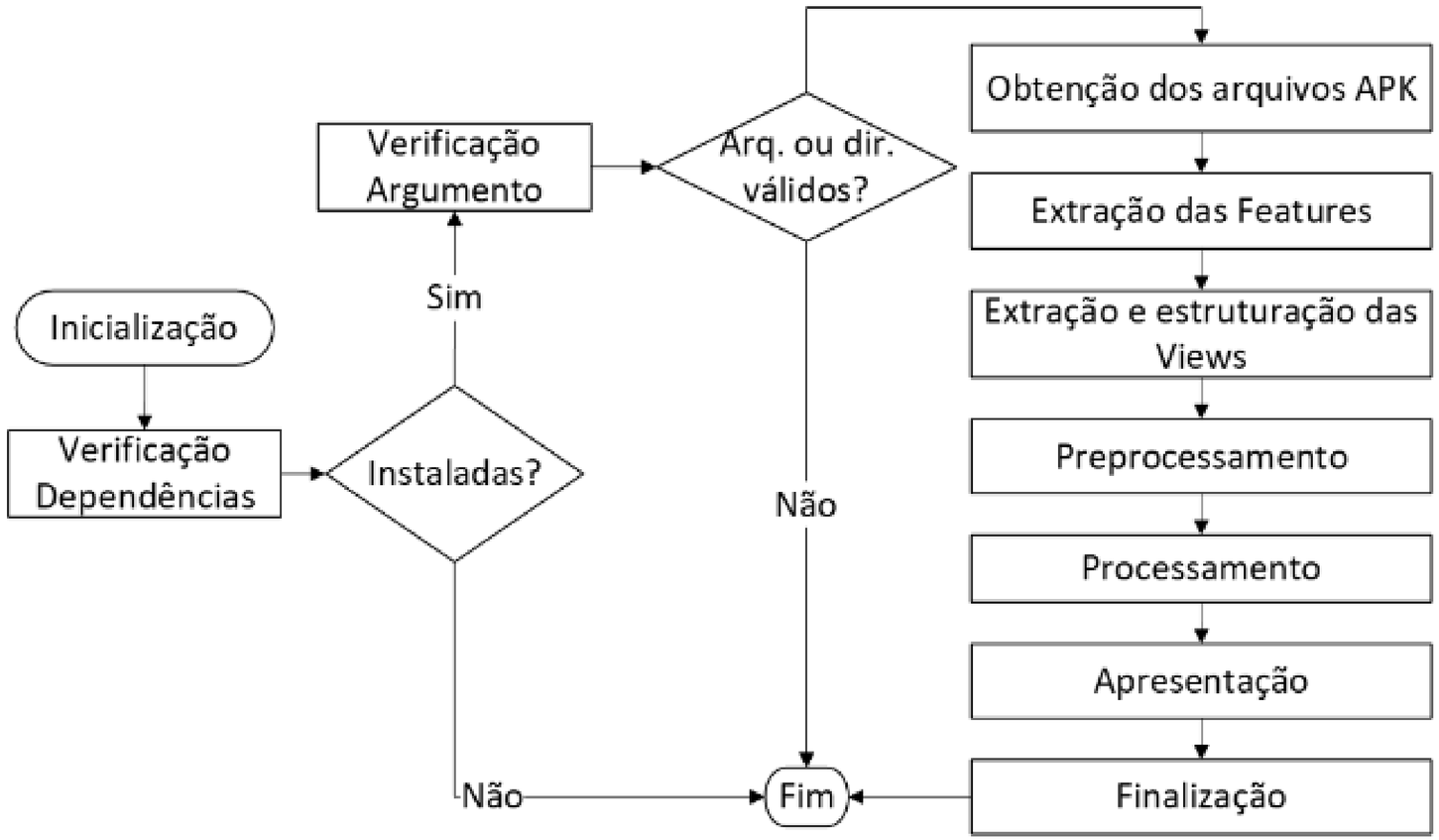
- Introdução
- Trabalhos Relacionados
- **Descrição**
- Demonstração
- Disponibilização
- Conclusão
- Perguntas

Descrição

- **APKAnalyzer:**
  - Desenvolvida na linguagem Python 3.10;
  - Principais Dependências:
    - Numpy v1.26;
    - Pandas v2.2;
    - Scikit-learn v1.4;
    - AndroPyTool: ferramenta de análise estática e dinâmica para aplicativos *Android* (configurado utilizando docker contêiner).

Descrição

- **APKAnalyzer:** fluxograma dos processos da ferramenta;



Descrição

- **APKAnalyzer:**
  - Módulo de Inicialização:
    - Verifica se todas as dependências estão adequadas e se o parâmetro recebido pela aplicação é válido. Caso alguma das dependências não seja encontrada ou o argumento é inválido, a execução da ferramenta é abortada;

Descrição

- Dentre as dependências:
  - Bibliotecas python;
  - Imagem docker alexmyg/andropytool:latest;
  - Diretórios e arquivos operacionais:
    - ./dumps: armazena os arquivos serializados;
    - ./schemas: armazena arquivos que representam a estrutura de todas as *features* de cada *view* antes da normalizacção e dimensionamento;
- O comando esperado é:
  - pyhton apkanalyzer.py arquivo_apk.apk;
  - pyhton apkanalyzer.py dir_apks.

Descrição

- **Módulo de Obtenção dos Arquivos APK:**
  - Diretório principal ./apks;
  - Faz cópia seguindo como referência o parâmentro passado por linha de comando;
  - Garantir as permissões necessárias para execução da ferramenta AndroPyTool;
  - Os arquivos permanecem no diretório ./apks até a conclusão da execução da aplicação.

Descrição

- **Módulo de Obtenção dos Arquivos APK:**
  - Diretório principal ./apks;
  - Faz cópia seguindo como referência o parâmentro passado por linha comando;
  - Garantir as permissões necessárias para execução da ferramenta AndroPyTool;
  - Os arquivos permanecem no diretório ./apks até a conclusão da execução da aplicação.

Descrição

- **Módulo de Extração das features:**
  - Inicializar a ferramenta AndroPyTool, utilizando o comando:

    docker run --volume ./apks:/apks alexmyg/andropytool -s /apks/ -fw;
  - Andropytool é inicializado no modo análise estática, consumindo os arquivos do diretório ./apks;
  - AndroPyTool produz como resultado alguns diretórios, dentro os quais ./apks/Features files/ conterá os arquivos com todas as *features* de cada um dos arquivos APK processados.

Descrição

- **Módulo de Extração e estruturação das views:**
  - A partir do resultado do processamento da ferramenta AndroPyTool, são extraídas e estruturadas as três *views* utilizadas no modelo: permissões, chamadas de API e códigos de operação;
  - A estrutura de cada *view* utiliza um gabarito disponível no diretório ./schemas. Assim mantém-se a integridade dos dados igual ao que foi utilizado na fase de treinamento do modelo.

Descrição

- **Módulo de Preprocessamento:**
  - Aplica-se a normalização utilizando o método *minmaxscaler;*
  - Aplica-se o dimensionamento utilizando o método PCA, reduzindo todos os *datasets* para 100 *features*;
  - Aplica-se a seleção de características em cada *view* com base em um subconjunto de características gerado previamente pelo algoritmo genético multiobjetivo NSGA-II:
    - Esse subconjunto é o mais eficiênte em termo de reduzir a taxa de erro e reduzir o tempo de inferência;
    - O garabarito foi serializado e está disponível no diretório ./dumps.

Descrição

- **Módulo de Processamento:**
  - Os modelos de classificação são executados utilizando como vetor de entrada os subconjuntos de cada *view*;
  - São utilizados três modelos:
    - Decision Tree;
    - Random Forest;
    - K-nearest Neighbors;
  - Para cada um dos modelos, é feita a predição utilizando cada uma das três *views*, totalizando 9 predições;

Descrição

- O voto majoritário é utilizado para obter resultados parciais e o resultado final. Considera-se *goodware* se a soma do resultado dos classificadores for menor que 2, caso contrário, *malware*.

	Permissões	*Opcodes*	Chamadas de API	Resultado Parcial
DT	0	0	1	*Goodware*
RF	1	1	0	*Malware*
kNN	1	1	1	*Malware*
Resultado Final				Malware

Descrição

- **Módulo de Apresentação:**
  - A etapa de apresentação consiste em exibir os resultados obtidos no terminal, apresentando:
    - Número sequencial para indexar o resultado;
    - Hash identificador do arquivo APK;
    - Resultado da classificação;
    - Link para o VirusTotal.

Descrição

- **Módulo de Apresentação:**
  - Exemplo de exibição dos resultados:

SEQ.	APK HASH	STATUS	VIRUSTOTAL LINK
1	1287BE51CB63E9F1CE4480227892965654 18DD97AEF934363086502 24A1C22A6	malware	VT
2	12B8377771B7FF0F30C13A66135FACA894 15649EBB1F0CCE5B2D84116FF47B06	goodware	VT
3	20E37718E9BA3DE0690A8989AF241540C 79BC42D06666B975925756 96ACD9D00	goodware	VT
4	84DF77A598833F46164198F65514B0597 28DBD2EFBEEF627C8B76A16519A338E	malware	VT

Tópicos

- Introdução
- Trabalhos Relacionados
- Descrição
- **Disponibilização**
- Conclusão
- Perguntas

Disponibilização

- **Github:**
  - A ferramenta está disponível no repositório:

    https://github.com/pFransozi/APKAnalyzer

Tópicos

- Introdução
- Trabalhos Relacionados
- Descrição
- Demonstração
- Disponibilização
- **Conclusão**
- Perguntas

Conclusão

- APKAnalyzer é uma ferramenta que tem o objetivo de implementar um modelo de detecção de *malware* para *Android* baseado em aprendizagem de máquina, incorporando duas técnicas: vetor de características da aplicação baseado em *multi-view* e seleção de características baseada em algoritmo genético com otimização multiobjetivo.

Tópicos

- Introdução
- Trabalhos Relacionados
- Descrição
- Demonstração
- Disponibilização
- Conclusão
- **Perguntas**

Obrigado!

Philipe Fransozi
Jhonatan Geremias
Eduardo K. Viegas
Altair Santin

{philipe.hfransozi, jgeremias, eduardo.viegas, santin}@ppgia.pucpr.br

Perguntas!

Patrocinadores do SBSeg 2024!

nic.br

cgi.br

Google

Tempest

CAPES

SiDi

FAPESP

zscaler™

BugHunt

CNPq

C.E.S.A.R

FACULDADE IBPTECH