

DETECÇÃO DE MALWARES EM DISPOSITIVOS MÓVEIS UTILIZANDO MACHINE LEARNING

Marcello Cainelli Filho

Faculdade de Ciências (FC) - Campus Bauru
Universidade Estadual Paulista “Júlio de Mesquita Filho”

Orientador: Prof. Dr. Kelton Augusto Pontara da Costa

Apresentação Banca
13 de novembro de 2018

Sumário

- 1 Introdução
- 2 Fundamentação Teórica
 - Conceitos-chave
 - Segurança
 - Support Vector Machines - SVM
- 3 Desenvolvimento
 - Base de Dados
 - Modelo de classificação
 - Utilização do modelo no Android
 - Aplicação - Implementação
 - Aplicação - Interface
 - Experimentos e Resultados
- 4 Conclusão
- 5 Referências

Introdução

A evolução e popularização dos dispositivos móveis facilitaram a execução de diversas necessidades que a humanidade possui nos dias atuais, sendo poucas as pessoas que não possuem consigo um dispositivo deste tipo durante a maior parte do tempo.

A segurança na área de dispositivos móveis não consegue acompanhar a produção e comercialização acelerada de aparelhos como os *smartphones*, atraindo indivíduos com intenções maliciosas que infectam os dispositivos com *malwares* por meio da internet.

O Globo 2017

Uma análise feita pela empresa Kaspersky durante o primeiro trimestre de 2017 identificou 1,333 milhão de ataques de vírus em *smartphones* no mundo.

A quantidade de ataques por vírus em *smartphones* exige a busca de soluções mais efetivas no combate aos *malwares* em dispositivos móveis, como a integração de softwares antivírus com técnicas de IA.

Portanto, este trabalho propôs a aplicação *Mal-Intentfier* para *smartphones Android* que utiliza um classificador de padrões à fim de identificar a existência de algum *malware* instalado no dispositivo.

Conceitos-chave

Classificador de padrões

Procedimento que separa amostras em diferentes categorias de acordo com suas características.

Manifesto Android

O Manifesto Android é um arquivo texto responsável por apresentar todo tipo de informação importante sobre o aplicativo ao sistema Android para que o este consiga executar a aplicação com suas devidas permissões e configurações (CORDEIRO, 2016).

Intent-filter

Os intent-filters são expressões no Manifesto que especificam o tipo de *intents* que o componente de uma aplicação deseja receber (DEVELOPERS, 2018).

```
01. <intent-filter>
02.     <action android:name="android.intent.action.EDIT" />
03.     <action android:name="android.intent.action.INSERT" />
04.     <action android:name="android.intent.action.DELETE" />
05. </intent-filter>
```

Figura 1: Exemplo de Intent-Filter

Matriz de Confusão

A matriz de confusão tem como objetivo mostrar o número de classificações corretas e predições para cada classe (FAWCETT, 2005 apud NAKATANI, 2017, p. 20).

Curva ROC

A curva ROC tem como base as taxas de verdadeiros positivos e falsos positivos (PRATI; BATISTA; MONARD, 2008).

Validação

A validação verifica a estabilidade do modelo de aprendizado, garantindo que este não super-ajuste ou sub-ajuste os dados (GUPTA, 2017).

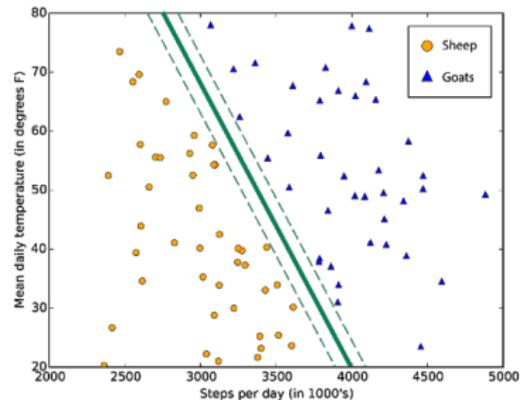
Segurança

Uma aplicação mal-intencionada pode usar os *intents* e *intent-filters* para ações maliciosas como roubar dados, e em conjunto com outra aplicação utilizar do envio de *intents* para prejudicar o dispositivo (KHAN et al., 2012).

Support Vector Machines - SVM

SVM é uma técnica de classificação de padrões. Seu funcionamento consiste em fazer uma separação dos dados por meio de uma lacuna em um espaço. Ao colocar novas amostras neste espaço, elas são categorizadas de acordo com o lado que foram distribuídas (LORENA; CARVALHO, 2007).

Figura 2: Exemplo de Support Vector Machines



Base de Dados Malgenome

A base de dados **Malgenome** possui aproximadamente 3700 aplicações catalogadas e foi essencial para este projeto por possuir como uma de suas características os *intent-filters* (ZHOU; JIANG, 2012).

A base **Malgenome** possui 26 tipos de *intent-filters*.

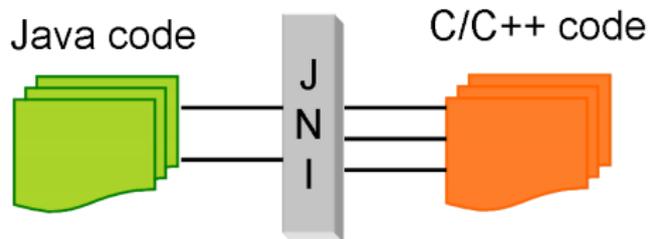
Modelo de classificação

- O modelo foi gerado em linguagem C utilizando o Matlab e suas *toolboxes*;
- Técnica de classificação utilizada: SVM;
- Método de validação: *K-fold Cross Validation*.

Utilização do modelo no Android

- Componente JNI (*Java Native Interface*);
- Compilação do modelo em uma biblioteca.

Figura 5: JNI



Aplicação - Implementação

A aplicação *Mal-Intentfier* foi desenvolvida em linguagem *JAVA* na *IDE Android Studio*.

Etapas da implementação:

- Lista de aplicações;
- Extrair o Arquivo Manifesto;
- Coletar informações;
- Classificador;
- Classificação.

Aplicação - Interface

Figura 6: Tela Inicial

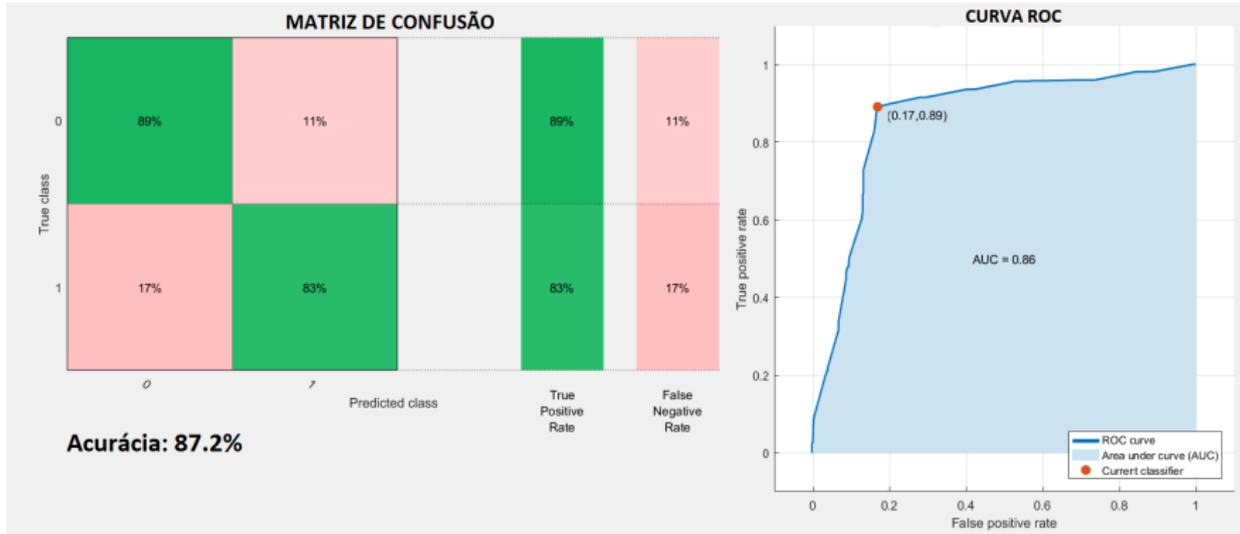


Experimentos e Resultados

Como dito anteriormente, foi utilizado o método *K-fold Cross Validation* com $K = 5$ para validação durante o experimento.

Utilizando o Matlab, foram geradas análises estatísticas do modelo obtido após a fase de treinamento e testes, localizadas na Figura 7.

Figura 7: Resultados do experimento - SVM



Testes em dispositivos

Foram criados dois cenários para testes, o primeiro envolveu analisar 60 amostras de *malwares* da própria base *Malgenome* e o segundo foi a execução do aplicativo *Mal-Intentfier* em 4 dispositivos diferentes.

Resultados

No primeiro cenário de teste a aplicação detectou 52 amostras corretamente, apresentando uma taxa de acerto de 86.66%.

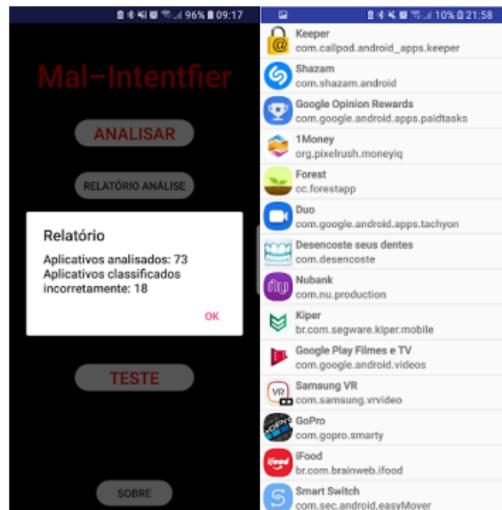
Os resultados do segundo teste se encontram na tabela abaixo.

Dispositivos	Nº de aplicações	Malwares	Tempo de execução (s)
Samsung S8	73	18	70
Samsung S6	15	4	10
Samsung A5	22	2	20
Motorola G6	11	2	8

Tabela 1: Resultados da execução da análise nos dispositivos

Exemplo de análise

Figura 8: Exemplo Galaxy S8



Conclusão

- Altas taxas positivas nas métricas de avaliação como acurácia, curva ROC e matriz de confusão;
- Resultados satisfatórios em dispositivos reais;
- Baixo tempo de execução;
- Os resultados fornecem oportunidade para trabalhos futuros aprimorarem os resultados deste projeto utilizando diferentes bases de dados ou adicionando a verificação dos *intent-filters* para potencializar outros tipos de classificadores.

Referências I

 CORDEIRO, F. *Os Segredos do AndroidManifest*. 2016. Disponível em: [〈https://www.androidpro.com.br/blog/desenvolvimento-android/androidmanifest〉](https://www.androidpro.com.br/blog/desenvolvimento-android/androidmanifest).

 DEVELOPERS, A. *Intents e filtro de intents*. 2018. Disponível em: [〈https://developer.android.com/guide/components/intents-filters?hl=pt-br〉](https://developer.android.com/guide/components/intents-filters?hl=pt-br).

 FAWCETT, T. An introduction to roc analysis. In: *Pattern Recognition Letters*. Palo Alto: Institute for the Study of Learning and Expertise, 2005.

 GUPTA, P. *Cross-Validation in Machine Learning*. Towards Data Science, 2017. Disponível em: [〈https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f〉](https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f).

Referências II

 KHAN, S.; NAUMAN, M.; OTHMAN, A. T.; MUSA, S. How secure is your smartphone: An analysis of smartphone security mechanisms. In: *2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec)*. [S.l.]: IEEE, 2012. p. 76–81.

 LORENA, A. C.; CARVALHO, A. C. P. L. F. Uma introdução às support vector machines. *RITA*, XIV, n. 2, p. 44–67, 2007. Acesso em: 23 out. 2018.

 NAKATANI, M. H. *Avaliação de Classificadores para Detecção de Phising em E-mails*. 2017.

 PRATI, R. C.; BATISTA, G. E. A. P. A.; MONARD, M. C. Curvas roc para avaliação de classificadores. *Revista IEEE America Latina*, IEEE, v. 6, n. 2, 2008. Acesso em: 23 out. 2018.

Referências III

 ZHOU, Y.; JIANG, X. Dissecting android malware: Characterization and evolution. In: *Proceedings of the 33rd IEEE Symposium on Security and Privacy*. [S.l.: s.n.], 2012.