VOL.40 Nº4 2023
https://doi.org/10.22491/rmct.v40i4.12122.pt

Detecção de Novidades em Famílias de Malware

Ricardo Sant'Ana[1], Julio Cesar Cardoso Tesolin[1] e Julio Cesar Duarte[1]
[1]Instituto Militar de Engenharia (IME)
Praça General Tibúrcio, 80, 22290-270, Praia Vermelha, Rio de Janeiro, RJ, Brasil
***ricardo.santana@ime.eb.br**

RESUMO: Muitas pesquisas já apresentaram abordagens para a tarefa de detecção de malware. Classificá-los em famílias fornece uma melhor compreensão de seu comportamento, permitindo que empresas e pesquisadores otimizem seus esforços. No entanto, um problema ainda precisa ser tratado corretamente: como verificar se um artefato computacional detectado como malware pertence a uma família já conhecida? Este trabalho propõe o uso de dois classificadores amplamente conhecidos – GMM e SVM – para uma tarefa de detecção de novidade em análise de malware, em que o objetivo é direcionar esforços humanos e computacionais adequados para fornecer uma rápida contramedida. A principal contribuição deste trabalho está no uso de características diretamente extraídas do arquivo binário do malware detectado, como entropia e textura de imagem para a tarefa de detecção de novidade.

PALAVRAS-CHAVE: Modelo de Misturas Gaussianas. Detecção de Família de Malware. Detecção de Novidade. Máquina de Vetores de Suporte. Malware como imagem. Entropia.

ABSTRACT: Many researches have already presented approaches to the malware detection task. Classifying them into families provides a better understanding of their behavior, allowing companies and researchers to optimize their efforts. Nevertheless, an issue still needs to be proper addressed: how to verify if an artifact detected as a malware belongs to a known family? This work proposes the use of two widely known classifiers - GMM and SVM - for a novelty detection task in malware analysis to redirect proper human and computational efforts for a quick counter measure. The main contribution of this work is the use of features directly extracted from the detected malwares's binary file such as entropy and image's texture for novelty detection.

KEYWORDS: Gaussian Mixture Model. Malware Family Detection. Novelty Detection. Support Vector Machine. Malware as an image. Entropy.

1. Introdução

A detecção e a proteção contra artefatos computacionais maliciosos é um dos tópicos de grande visibilidade na área de segurança cibernética. A literatura apresenta várias definições sobre o que são artefatos maliciosos (ou *malware*) [12], de forma geral, são programas que podem causar danos a usuários, sistemas e redes, corrompendo seus códigos, prejudicando suas operações e/ou furtando suas informações.

Uma pesquisa realizada em 2017 por uma grande empresa produtora de programas de combate a artefatos maliciosos [1], envolvendo 1.300 profissionais de pequenas e grandes empresas de Tecnologia da Informação, revelou que 91% delas foram atacadas por algum tipo de *malware*, 45% estavam mal preparadas para ataques cibernéticos dedicados e 17% perderam dados financeiros como resultado desses ataques.

Normalmente, novos artefatos maliciosos são versões daqueles já existentes. Esse fato pode ser verificado em [21], em que menos de 10% dos *malware* relacionados em 2019 são novos. Assim, caracterizá-los em determinadas famílias de *malware* acelera a identificação de seu comportamento, informação de vital importância para a alocação de recursos computacionais e humanos que promoverão uma contramedida tão logo seja possível. Porém, nos casos em que o artefato malicioso não é uma variação dos já existentes, ou seja, uma novidade dentre os conjuntos de *malware* existentes, essa caracterização não se aplica. Assim, faz-se necessário iniciativas de pesquisa que busquem identificar automaticamente essas novidades de forma a acelerar o combate a essas ameaças.

Assim, o objetivo desta pesquisa é avaliar a capacidade dos modelos de classificação de artefatos nunca vistos nas famílias de *malware* conhecidas, utilizando dois modelos de aprendizado de máquina comuns para a detecção de novidade: Modelos de Misturas Gaussianas (GMM – *Gaussian Mixture Models*) e Máquinas de Vetores de Suporte (SVM – *Support Vector*

Machines). Tal tarefa impôs: (a) definir um conjunto mínimo de características dos *malware* de forma a permitir uma rápida detecção; (b) propor um método para a seleção do conjunto de dados; e (c) critérios de escolha dos hiperparâmetros do modelo, sempre considerando a tarefa de detecção de novidades. Estas são as principais contribuições deste artigo.

Este artigo foi estruturado da seguinte forma: a Seção 2 descreve os fundamentos teóricos de detecção de novidades, os modelos de aprendizado de máquina utilizados, além da base de dados de *malware* publicamente disponíveis. A Seção 3 relata os principais trabalhos relacionados com a utilização de aprendizado de máquina em análise de *malware*. A Seção 4 apresenta o método utilizado para definir os conjuntos de dados e selecionar os hiperparâmetros mais adequados para ajustar os modelos de detecção de novidade. Já a Seção 5 expõe os experimentos e os resultados obtidos. Finalmente, na Seção 6, são apresentadas as conclusões e as sugestões para trabalhos futuros.

2. Referencial Teórico

Nesta seção, são abordados, de forma sucinta, os fundamentos teóricos de análise de *malware*, de detecção de anomalias e novidades, dos métodos de aprendizados de máquina utilizados e do conjunto de dados usado para o experimento.

2.1 Análise estática e dinâmica

Existem duas abordagens fundamentais para a análise de *malware*: a abordagem estática e a abordagem dinâmica. A análise estática envolve examinar o *malware* sem executá-lo, enquanto a análise dinâmica envolve a execução do *malware*[22]. Ambas as técnicas têm vantagens e desvantagens e podem ser utilizadas de forma a complementar o processo de análise.

2.2 Detecção de novidade

De acordo com [13], anomalias são padrões nos dados que não se adaptam a uma noção bem definida do que é um comportamento normal. Trazendo essa definição para a detecção de *malware*, um artefato malicioso que não se enquadra em qualquer família conhecida é uma anomalia. No entanto, artefatos maliciosos (anomalias) nunca vistos são bastante comuns, e, portanto, de acordo com [13], esse tipo de anomalia é definido como novidade.

O uso de um termo ou outro não é uma mera questão semântica: os métodos de identificação de anomalias partem do princípio de que existem amostras anômalas – ainda que em quantidade bem reduzida – como parte do conjunto de treinamento. Já nos métodos de detecção de novidade, essas amostras anômalas não existem para parametrizar o modelo, tornando a detecção de um *malware* nunca antes visto ainda mais desafiadora. Obviamente, o problema de detecção de anomalias pode utilizar métodos de aprendizado de máquinas supervisionados, enquanto o problema de detecção de novidade utiliza comumente métodos de aprendizado de máquina não supervisionados.

2.3 Modelos de misturas gaussianas

O modelo de misturas gaussianas (*GMM – Gaussian Mixture Models*) é um modelo probabilístico que assume que todos os pontos de dados são gerados a partir de uma mistura de um número finito de distribuições gaussianas com parâmetros desconhecidos. Assim, um modelo GMM-1 utiliza apenas uma distribuição gaussiana para modelar a distribuição de dados, enquanto um modelo GMM-32 utiliza 32 distribuições gaussianas ponderadas. Nas técnicas estatísticas tradicionais, que utilizam modelos gaussianos para detecção de novidade, o limiar das regiões de probabilidade pode ser definido de duas maneiras: (i) como $\mu \pm 3\,\sigma$, em que μ é a média e σ é o desvio-padrão (ii), como prescreve o teste de Grubbs [13]. Contudo, em aprendizado de máquina não se estabelece, *a priori*, o limiar para as novidades.

A detecção de novidade utilizando o aprendizado não supervisionado envolve, principalmente, a tarefa de se estimar densidades. Claramente, o conhecimento prévio da densidade dos pontos de dados nos permitiria resolver qualquer problema com base nos dados [14].

2.4 Máquinas de suporte a vetor

As máquinas de vetores de suporte (SVM – *Support Vector Machines*) são um conjunto de métodos de aprendizado supervisionados usados para classificação, regressão e detecção de anomalias. Especificamente para a aplicação de deteção de novidades, a utilização de SVM pode ser encontrada em [14], em que é proposto um método para resolver o seguinte problema: estabelecido um conjunto de dados, extraído de uma distribuição de probabilidade subjacente P, deseja-se estimar um subconjunto S de forma que a probabilidade de uma amostra de teste retirada de P ficar fora de S seja igual a V. A utilização do SVM neste tipo de tarefa é conhecida como *OneClassSVM* (OC-SVM) e tem a vantagem de não fazer suposições sobre a forma da distribuição dos dados conhecidos [15], ou seja, ser aplicável em qualquer distribuição. De acordo com [23], a ideia é encontrar uma função que seja positiva para regiões com alta densidade de pontos e negativa para pequenas densidades.

Uma desvantagem do *OneClassSVM* é o fato de o método predizer, de forma discreta, se a amostra pertence ou não ao grupo modelado. É possível expandir essa limitação realizando modificações no *OneClassSVM*, ao permitir que a saída do SVM seja na forma de probabilidades de classes condicionais, conforme proposto em [16]. Uma abordagem probabilística oferece muitas vantagens em relação ao método convencional, incluindo a facilidade de selecionar automaticamente um limite de novidade probabilístico.

2.5 Base de dados de artefatos maliciosos para detecção de novidade

As técnicas de detecção de anomalias e novidades baseadas em aprendizado de máquina são claramente dependentes de bases de dados com volume representativo de amostras. Neste quesito, são poucas as bases de dados de *malware* disponíveis para o público.

Ainda, de acordo com [20], os artefatos maliciosos para a plataforma Windows, em agosto de 2019, representavam 74,49% do universo de *malware* existentes, o que motivou nossa escolha por essa plataforma.

3. Trabalhos relacionados

Nesta seção, são apresentados, de forma resumida, os principais trabalhos relacionados ao emprego de aprendizado de máquina à análise de *malware*.

A utilização de aprendizado de máquina em análise de *malware* é extensa e abrange diversos objetivos, como a detecção de artefatos maliciosos [2, 3, 4, 5, 6], a detecção de variantes de *malware* [7, 8], a detecção de categorias de *malware* [9,10,12], entre outros.

Em [2], um novo algoritmo baseado em grafos, construídos a partir de traços de instruções coletadas de forma dinâmica, é proposto para a tarefa de classificação de artefatos em maliciosos ou benignos. O resultado mostra que a utilização da análise dinâmica combinada com n-gramas é superior ao obtido por ferramentas de antivírus da época (2011). Já em [3], os autores propõem a utilização de análise estática e dinâmica de artefatos, assim como métricas de similaridade, com diversos *kernels* para a detecção de *malware*. Um peso é dado para cada *kernel*, de forma a encontrar a combinação pesos/*kernels* que obtenha a melhor acurácia. O resultado utilizando tal abordagem foi de 99.78% de acurácia. Em [4], o autor faz uso de máquinas de vetores de suporte para a tarefa de detecção de *malware*. O escopo do trabalho foi limitado a 398 amostras, obtendo uma acurácia entre 94% e 95%. Os autores em [5] utilizam técnicas estáticas e dinâmicas de forma conjunta para a detecção de artefatos maliciosos. Neste trabalho, foram utilizados os algoritmos C5.0 e *Random Forest*, sendo implementados no *framework* FAMA. A acurácia obtida foi de 95,75% para a classificação binária e 93,02% para a categorização múltipla. Uma arquitetura para análise automática de artefatos maliciosos (análise dinâmica) é proposta em [6]. Os resultados obtidos com *Random Forest* (baseados no ID3) apresentaram uma acurácia acima de 90%.

Em [7], um novo método para utilização de modelos ocultos de Markov é proposto: inicialmente, são selecionados conjuntos de *opcodes* relevantes para, em seguida, submeter esta sequência a um modelo oculto de Markov. O trabalho apresenta melhorias entre 8% e 42% nos resultados, em comparação aos trabalhos que usam Modelos Ocultos de Markov sem a seleção de *opcodes*. Já em [8],

é apresentada uma estrutura genérica que extrai informações estruturais de *malware* como gráficos de chamadas de função sendo codificadas como atributos em nível de função. Essa abordagem é avaliada em 11 famílias de *malware*, obtendo uma acurácia de 86,67% na detecção de amostras de variantes previamente escolhidas.

Em [9], são utilizados algoritmos de árvore de decisão e máquina de vetores de suporte para categorizar artefatos maliciosos nas classes *Trojan*, *Infector*, *Backdoor* e *Worm*. O resultado obteve uma acurácia acima de 98% para o algoritmo de árvores de decisão e de 97% para o algoritmo de máquina de vetores de suporte. Já em [10], os autores propõem a utilização de duas abordagens de aprendizado de máquina para fornecer uma correta classificação dos artefatos maliciosos apresentados no *Microsoft Malware Classification Challenge* [11]. Nesse trabalho, são extraídas várias características dos artefatos: referentes ao conteúdo binário do artefato malicioso e referentes ao arquivo desconstruído (*disassembled*). Com a utilização de um algoritmo para a seleção dos melhores atributos, o sistema obteve uma acurácia de 99,76%. Finalmente, em [12], os autores utilizam a abordagem de aprendizado de máquina não supervisionado para a classificação e a detecção de novas famílias de *malware*, denominadas *Shared Nearest Neighbor* – SNN. Utilizando uma base de dados com 20 mil amostras, os resultados mostram uma melhoria de 3% utilizando o algoritmo de *Random Forest* e de 18% utilizando o algoritmo de *Naive Bayes*, considerando uma combinação de três tipos de atributos em todos os casos (atributos de imagem em escala de cinza e de n-grama). Em síntese, esses últimos artigos mostram duas estratégias diferentes: [10] limita-se a classificar os *malware* nas famílias existentes e [12] propõe a classificação e a detecção utilizando uma abordagem não supervisionada.

Assim, apesar da extensa pesquisa na área de análise de *malware* utilizando aprendizado de máquina, nenhum dos artigos analisados aborda a tarefa de detecção de novidade, tema trabalhado no presente artigo.

4. Desenvolvimento da pesquisa

Nesta seção, são apresentados os processos e os métodos utilizados para ajustar os hiperparâmetros dos modelos de aprendizagem de máquina para detecção de novidades no conjunto de dados de *malware* selecionado.

4.1 Análise do conjunto de dados de malware

Neste trabalho, utilizou-se o conjunto de dados fornecido pela Microsoft para seu desafio de classificação de famílias *malware*, sendo este hospedado na plataforma *Kaggle* [11]. Originalmente, são fornecidos dois conjuntos de dados: um de treinamento, com 10.868 amostras de famílias conhecidas, e um de testes, com 10.873 amostras de famílias desconhecidas. Apenas o subconjunto de treinamento foi utilizado para esta pesquisa, uma vez que era o único a apresentar os rótulos de família para cada amostra. A distribuição por família deste subconjunto de treinamento é apresentada na Tabela 1.

Tabela 1 - Distribuição das famílias da malware na base de dados da microsoft [11]

Ordem	Nome da família	Quantidade de amostras
1	Ramnit	1.541
2	Lollipop	2.478
3	Kelihos_ver3	2.942
4	Vundo	475
5	Simda	42
6	Tracur	751
7	Kelihos_ver1	398
8	Obfuscator.ACY	1.228
9	Gatak	1.013

As amostras de *malware* desta base de dados estão disponibilizadas em dois formatos: binário bruto, que contém a representação hexadecimal do conteúdo binário do arquivo; e o código desconstruído, resultado do processo de *disassembler* do código original (com a ferramenta IDA [17, 10]). Em ambos os casos, o cabeçalho do arquivo (*PE header*) foi removido, inviabilizando qualquer tentativa de análise dinâmica do código.

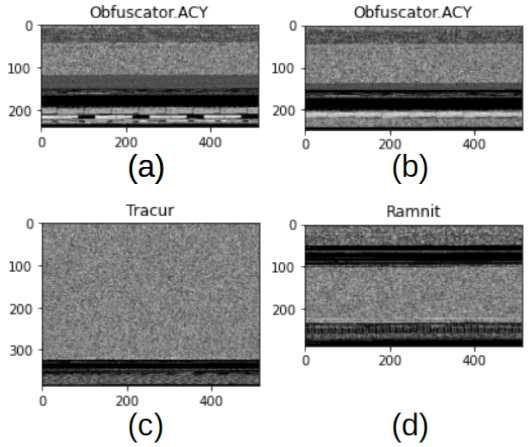
Como o objetivo da pesquisa é definir um classificador capaz de identificar novidades de forma rápida, considerou-se apenas um conjunto mínimo de atributos extraídos dos dados binários brutos a partir de duas premissas: (i) distribuição próxima à distribuição normal; e (ii) sucesso na utilização em outros trabalhos. Dessa forma, quatro classes de atributos foram escolhidas de [10]: as de entropia (ENT), as de metadados (MD1), as de Haralick (IMG1) e as de padrões binários locais (IMG2).

Os atributos de entropia (ENT) são utilizados para medir a quantidade de desordem do arquivo binário bruto. A entropia é calculada utilizando N janelas deslizantes de forma a representar o *malware* como uma medida de entropia $E = E_i$ com $i = \{1, 2, 3 \ldots N\}$, em que E_i é a entropia medida na janela i e N é o número de janelas. Em seguida, considerou-se a estatística do vetor de entropia obtido usando o método de janela deslizante, ou seja, foi calculada a entropia para cada janela de 10.000 *bytes* e então considerou-se as medidas estatísticas quantis, percentis, média e variância da distribuição obtida. Foram computados também a entropia de todos os *bytes* do *malware,* totalizando 202 características. Já os atributos de metadados extraídos (MD1) são o tamanho do arquivo e o endereço da primeira sequência de *bytes*, totalizando duas características.

Uma maneira original de representar uma amostra de *malware* é visualizar os *bytes* como escala de cinza de um pixel em uma imagem 2D [10]. A Figura 1 mostra um recorte da imagem em escala de cinza de (a) um *malware* da família Obfuscator.ACY, (b) outro *malware* da família Obfuscator.ACY, (c) um *malware* da família Tracur e (d) um *malware* da família Ramnit. Nota-se por (a) e (b) que dois *malware* da mesma família têm representação visual similar.

Da imagem de escala de cinza de cada amostra, foram extraídos dois conjuntos de atributos que descrevem as texturas: os atributos *Haralick* (IMG1), com 53 características extraídas; e os atributos padrões binários locais (IMG2), com 108 características extraídas, todas de fácil obtenção do ponto de vista de processamento [10]. Para essa extração, utilizou-se a biblioteca de processamento de imagens e visão computacional *Mahotas* [18].

Fig. 1 - Representação de *malware* em suas respectivas famílias por meio de imagens em escala de cinza.



Assim, as classes de atributos de entropia (ENT), metadados (MD1), *Haralick* (IMG1) e padrões binários locais (IMG2) totalizam 365 atributos individuais e o código fonte dos programas de extração desses atributos pode ser encontrado em [10]. Não foram realizadas iniciativas para reduzir a dimensionalidade, tendo em vista que estes 365 atributos foram selecionados dos 1.805 atributos utilizados em [10].

4.2 Definição do conjunto de treinamento e teste

A definição dos conjuntos de treinamento e teste foi feita em três passos, de forma a obter subconjuntos de dados adequados para a tarefa de detecção de novidade.

Inicialmente, a atividade de selecionar a família de *malware* para fazer o papel de novidade mostrou-se desafiadora, uma vez que, salvo melhor entendimento, não há uma definição ou método formal para execução dessa tarefa. Seria possível,, por exemplo, escolher tanto a família com o menor número de amostras quanto a que tivesse a maior distância em relação aos centroides das outras famílias (considerando que um conjunto de atributos foi selecionado). Para esta pesquisa, resolveu-se adotar como novidade a família com o menor número de amostras.

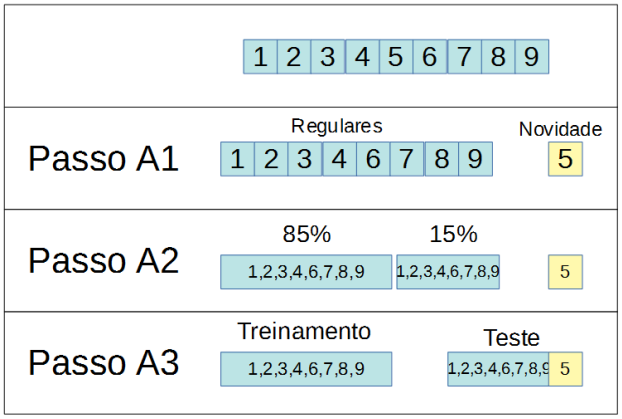
VOL.40 Nº4 2023
https://doi.org/10.22491/rmct.v40i4.12122.pt

Assim, no Passo A1, foram selecionadas as amostras pertencentes à família 5 (Simda), apresentada na Tabela 1. A partir deste ponto, foi convencionado indicar esta família como classe **novidade**, enquanto as demais foram reunidas em uma única classe denominada **regular**.

Em seguida, no Passo A2, a classe regular foi dividida em dois grupos, compostos respectivamente por 85% e 15% do total de amostras, por meio de um particionamento extratificado aleatório. Finalmente, no Passo A3, o primeiro grupo foi definido como o conjunto de treinamento (85% das amostras da classe regular), enquanto o conjunto de testes foi definido como a união do segundo grupo (15% das amostras da classe regular) e a classe novidade (Simda).

Como resultado deste particionamento (resultado do Passo 3), nenhuma amostra da classe novidade pertence ao conjunto de treinamento. A relação entre a quantidade de amostras do grupo de treinamento e teste ficou em torno de 6:1. A Figura 2 apresenta os passos descritos, sendo que os números de 1 a 9 representam as 9 famílias de *malware* da Tabela 1.

Fig. 2 - Representação do método de partição dos conjuntos de treinamento e teste.



4.3 Definição do conjunto de treinamento e validação

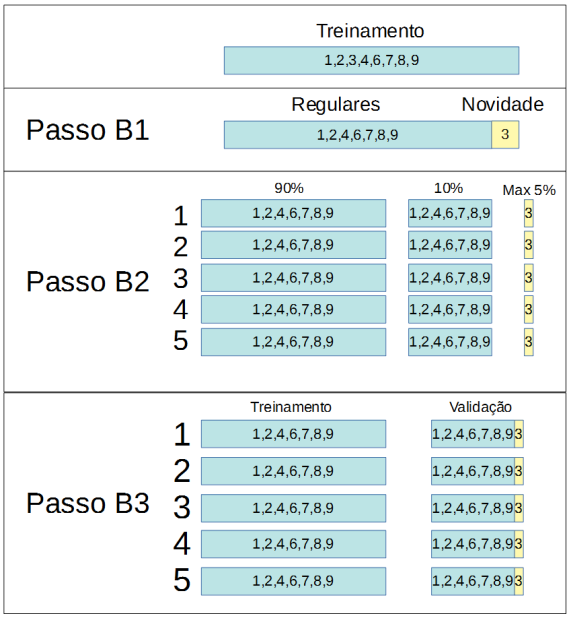
Como exposto na Seção 2.2, a detecção de uma novidade não é o mesmo que a detecção de uma anomalia. Enquanto nesta existem amostras para calibrar o processo de aprendizagem de máquina, naquela não há essa possibilidade. Logo, deve-se adaptar o processo de seleção de conjunto de treinamento e validação à tarefa de detecção de novidade, fazendo com que diversas famílias do conjunto de treinamento definido na Seção 4.2 desempenhem o papel de classe novidade no conjunto de validação. Haverá então uma iteração, sendo que cada família do conjunto de treinamento fará o papel de classe novidade. Da mesma forma que na seção anterior, dividiu-se este processo em três passos, como mostra a Figura 3.

No passo B1, foi escolhida uma família do conjunto de treinamento para representar a classe novidade, enquanto o restante das amostras foi utilizado para representar a classe regular. Na Figura 3, a título de exemplo, utilizou-se a família 3 como a classe escolhida para fazer o papel de família novidade. Já no Passo B2, dividiu-se a classe regular em cinco subconjuntos de igual tamanho, utilizando particionamento extratificado aleatório. A ideia de se utilizar cinco subconjuntos é que, durante a fase de treinamento e validação do modelo, o desempenho será o valor da média dos desempenhos obtidos em cada um dos cinco subconjuntos. Em seguida, cada subconjunto foi dividido em dois grupos: o primeiro com 90% das amostras e o segundo com 10% das amostras.

Finalmente, no Passo B3, designou-se o primeiro grupo (90% das amostras) como o conjunto de treinamento. Já o conjunto de validação é formado pela união do segundo grupo criado (10% das amostras) com um conjunto de amostras da família novidade definida no Passo B1 (exemplificado na Figura 3 como família 3), limitando seu volume em 5% da quantidade total do segundo grupo de forma aleatória. Portanto, o conjunto de **validação** tem amostras regulares e amostras novas. Novamente, buscou-se uma relação entre amostras de treinamento e validação próximo a 6:1.

Como cada família fará o papel de novidade, um total de oito experimentos serão realizados, dos quais serão definidos os conjuntos de treinamento/validação, conforme descrito anteriormente.

Fig. 3 - Método para selecionar o conjunto de treinamento e validação para a tarefa de detecção de novidade. A família 3 foi utilizada como classe novidade.



4.4 Seleção dos parâmetros do classificador

Os hiperparâmetros do modelo foram ajustados considerando, de forma simultânea, o resultado nos oito experimentos de treinamento/validação definidos na Seção 4.3. Assim, criou-se um critério para o aceite de determinado conjunto de hiperparâmetros de um modelo.

Inicialmente, define-se P_i como um conjunto de hiperparâmetros, em que i = {1, 2, 3 ... n} e n é o número de tentativas. Para avaliar o processo de aprendizado a partir desses hiperparâmetros, foram estabelecidas três métricas:

Porcentagem do Erro de Treinamento **(PETr)**: erro no conjunto de treinamento, calculado a partir da divisão do número de predições de que não pertence à classe regular e o total de elementos da classe de treinamento. No conjunto de dados de treinamento há somente elementos da classe regular.

Porcentagem do Erro de Validação **(PEV)**: erro no conjunto de validação, calculado a partir da divisão número de equívocos cometidos na predição das amostras do conjunto de validação dividido pelo número total de amostras do conjunto de validação. O conjunto de validação possui amostras regulares e amostras novidade.

Porcentagem do Erro de Novidade **(PEV_{Nov})**: o erro de novidade é definido dividindo o número de equívocos cometidos na predição apenas das amostras da classe novidade do conjunto de validação pelo número total de amostras da classe novidade do conjunto de validação.

Para aceitar um conjunto de parâmetros P_i, definiu-se o seguinte critério:

$$(PETr < 30\%) \wedge (PEV < 30\%) \wedge (PEV_{Nov} < 40\%)$$

Esse critério deve ser validado considerando a média das métricas PETr, PEV e PEV_{Nov} para cada um dos oito experimentos definidos na Seção 4.2. Cabe ressaltar que os limiares percentuais apresentados anteriormente foram estabelecidos após seguidas tentativas.

Caso P_i atenda ao critério estabelecido, ele é selecionado. Senão, um novo conjunto de parâmetros P_{i+1} é definido e o processo de seleção de parâmetros é reiniciado. Não houve a pretensão de escolher o melhor modelo possível. O objetivo foi escolher aquele com hiperparâmetros definidos como P_i, capaz de ser utilizado em todos os oito experimentos. A premissa adotada é que o critério de escolha de hiperparâmetros P_i é suficiente para generalizar o modelo final e obter um bom resultado para amostras nunca vistas anteriormente.

Finalmente, com o conjunto de hiperparâmetros P_i foi selecionado, um modelo com todo o conjunto de treinamento original (famílias 1, 2, 3, 4, 6, 7, 8 e 9) é treinado. Em seguida, o conjunto de teste (família 5) é apresentado para este modelo. Para estimar o desempenho do modelo no conjunto de teste, as seguintes métricas são definidas:

Porcentagem do Erro de Treinamento (**PETr**): erro no conjunto de treinamento, calculado a partir da divisão do número de predições de que não pertence à classe regular e ao total de elementos da classe de treinamento, conforme apresentado na Figura 2.

Porcentagem do Erro de Teste **(PET)**: erro no conjunto de teste, calculado a partir da divisão número de equívocos cometidos na predição das amostras do conjunto de teste dividido pelo número total de amostras do conjunto de validação. O conjunto de teste possui amostras regulares e amostras novidade da família 5.

Porcentagem do Erro de Novidade (**PET_{Nov}**): o erro de novidade, ou seja, família 5, que é definido dividindo o número de equívocos cometidos na predição apenas das amostras da classe novidade do conjunto de teste pelo número total de amostras da classe novidade do conjunto de teste.

5. Resultados

Esta seção apresenta os resultados a partir dos hiperparâmetros escolhidos na seção anterior, utilizando os modelos de máquina de vetores de suporte e de misturas gaussianas. Conforme apresentado nas seções 2.3 e 2.4, os modelos de máquina de vetores de suporte e o modelo de misturas gaussianas são utilizados em trabalhos de detecção de anomalia e novidades, apesar da sua abordagem bastante diferente. Assim, a escolha desses modelos se justifica para verificar o comportamento deles em um caso específico na área de *malware*. Cabe ressaltar que o desempenho dos experimentos será avaliado tanto em amostras de famílias previamente identificadas como em amostras nunca antes vistas pelo modelo.

5.1 Modelo de máquina de vetores de suporte

Para o modelo SVM, decidiu-se utilizar o *kernel* RBF conforme orientação apresentada em [19]. O restante dos hiperparâmetros possíveis de serem modificados são (i) *outliers*, que definem a quantidade de anomalias do conjunto de treinamento e (ii) *gamma* (γ), que define o *kernel* escolhido. Como não foi encontrada uma implementação probabilística do *OneClassSVM*, conforme apresentado em [16], optou-se por utilizar a versão convencional do algoritmo.

Depois, os melhores resultados foram obtidos utilizando *outliers* iguais a 13% e γ igual a 0,01, seguindo o método apresentado na Seção 4.4. Os resultados das métricas para os diversos conjuntos de validação é apresentado na Tabela 2. Em cada linha tem-se a média da métrica para 5 subconjuntos, em que uma das oito famílias faz o papel de novidade (ver Seção 4.3).

Durante a fase de seleção de parâmetros, identifica-se que o pior resultado ocorre quando a família 6 (família *Tracur*, com porcentagem de erro de novidade de 74,18%) é novidade. O possível motivo para isso é que os *malware* da família *Tracur* se diferenciam dos demais por, prioritariamente, serem capazes de redirecionar as pesquisas na *web* do computador infectado, além de serem capazes de baixar e executar arquivos, incluindo outros *malware*, e permitir ações de comando e controle no computador infectado. Assim, os artefatos da família *Tracur* se diferenciam dos demais por serem mais diversificados com relação à capacidade de atividades maliciosas. Com isso, o classificador pode, de forma equivocada, acabar classificando-os em outras famílias.

Observando que o resultado médio das métricas atende aos critérios definidos na Seção 4.4, selecionou-se este conjunto de hiperparâmetros para criar o modelo capaz de considerar todo o conjunto de treinamento descrito na Seção 4.2.

Tabela 2 - Erro treinamento e validação para SVM

Nome da família	PETr	PEV	PEV_{Nov}
Ramnit	13,14%	12,48%	6,08%
Lollipop	13,10%	13,97%	7,08%
Kelihos_ver3	13,21%	12,46%	0,00%
Vundo	13,17%	11,63%	1,41%
Tracur	13,18%	33,99%	74,18%
Kelihos_ver1	13,14%	12,48%	6,08%
Obfuscator.ACY	13,27%	12,61%	3,92%
Gatak	13,19%	16,44%	15,73%
Média:	**13,18%**	**15,86%**	**13,43%**

Em seguida, realizou-se a predição no conjunto de testes e o resultado das métricas neste conjunto é apresentado na Tabela 3. Pode-se observar que o modelo SVM se manteve sólido para detecção de amostras do conjunto regular, mas nota-se uma queda de desempenho na detecção de novidade.

Tabela 3 - Erro de teste para SVM

Nome da família	PETr	PET	PET_{Nov}
Simda	13,17%	17,29%	26,19%

5.2 Modelo de misturas gaussianas

Inicialmente, foi utilizado um modelo de misturas gaussianas com apenas uma gaussiana, o que é equivalente a utilizar um modelo de gaussiana simples para detecção de novidade. Diversas tentativas foram realizadas para encontrar o valor de limite de erro (ε), utilizando a abordagem descrita na Seção 4.4, na qual valores abaixo desse limiar indicam que a amostra não pertence à classe regular.

Após diversos experimentos, chegou-se a um limiar de erro $\varepsilon = \mu - 0{,}98\sigma$, em que μ é a média dos logaritmos ponderados das probabilidades para cada amostra de treinamento e σ é o desvio-padrão dos logaritmos ponderados das probabilidades para cada amostra de treinamento. Os resultados das métricas para os diversos conjuntos de validação estão presentes na Tabela 4. Em cada linha, a média da métrica para os cinco subconjuntos é apresentada, em que cada uma das oito famílias fez o papel de novidade.

Tabela 4 - Erro treinamento e validação para GMM-1

Nome da família	PETr	PEV	PEV_{Nov}
Ramnit	31,66%	30,91%	40,50%
Lollipop	21,64%	21,66%	10,97%
Kelihos_ver3	31,52%	32,51%	0,01%
Vundo	28,70%	29,04%	98,12%
Tracur	33,76%	35,16%	38,54%
Kelihos_ver1	28,30%	28,72%	13,31%
Obfuscator.ACY	29,40%	28,79%	88,71%
Gatak	26,25%	27,37%	24,37%
Média:	**28,95%**	**18,97%**	**39,34%**

Observando o resultado médio das métricas, os critérios definidos na Seção 4.4 foram atendidos. As famílias que obtiveram pior resultado foram Vundo e Obfuscator.ACY. O possível motivo para esse erro é que a distribuição dos atributos dessas famílias não pode ser aproximada de uma distribuição gaussiana. Assim, este conjunto de hiperparâmetros foi selecionado, criando-se um modelo que considera todo o conjunto de treinamento e, em seguida, realizou-se a predição no conjunto de testes. Os resultados para o conjunto de testes são apresentados na Tabela 5.

Tabela 5 - Erro de teste para GMM-1

Nome da família	PETr	PEV	PEV_{Nov}
Simda	27,79%	27,91%	19,05%

Pode-se concluir preliminarmente que o modelo com uma gaussiana atendeu à tarefa de detecção de novidade, considerando os critérios utilizados para a seleção de parâmetros utilizada.

De qualquer forma, realizaram-se diversos experimentos variando o número de gaussianas do GMM. Modelos com 2, 4, 8, 16, 32 e 64 gaussianas foram experimentados. A partir de 32 gaussianas, não foi observada nenhuma melhora no resultado do treinamento/validação. Dessa forma, os resultados das métricas para os diversos conjuntos de validação para 32 gaussianas é apresentado na Tabela 6.

Tabela 6 - Erro treinamento para GMM-32

Nome da família	PETr	PEV	PEV_{Nov}
Ramnit	16,00%	30,25%	0,08%
Lollipop	8,57%	20,25%	14,98%
Kelihos_ver3	8,14%	21,69%	0,19%
Vundo	8,54%	18,86%	97,92%
Tracur	11,27%	21,09%	29,97%
Kelihos_ver1	18,04%	27,26%	2,84%
Obfuscator.ACY	9,18%	18,84%	88,77%
Gatak	7,87%	17,74%	26,40%
Média:	**10,95%**	**22,00%**	**39,33%**

Novamente, pode-se identificar uma grande variação na porcentagem de erro de novidade, também observado no caso do GMM-1. Nota-se ainda que a média da porcentagem de erro do modelo com apenas uma gaussiana é próxima à do modelo com 32 gaussianas.

Realizou-se o processo de escolha e seleção de hiperparâmetros para os modelos com 2, 4, 8, 16 e 32 gaussianas e aplicou-se o conjunto de testes em cada um desses modelos. O resultado é apresentado na Tabela 7.

Tabela 7 - Erro de teste para GMM 2,4,8,16 e 32

GMM	Nome da família	PETr	PET	PET_{Nov}
2	Simda	17,09%	18,13%	30,95%
4	Simda	27,22%	30,79%	28,57%
8	Simda	26,58%	28,09%	28,57%
16	Simda	21,00%	27,01%	28,57%
32	Simda	15,52%	25,15%	26,19%

Se considerarmos apenas a quantidade de erros na detecção de novidades, tem-se 13 erros para GMM-2, 12 erros (28,57%) para GMM-4, GMM-8 e GMM-16 e 11 erros (26,19%) para GMM-32. Assim, os resultados no treinamento e teste de modelo mostraram o modelo GMM-32 um pouco melhor. Isso era esperado, se considerarmos que, quanto maior o número de gaussianas, melhor o modelo se adequa à distribuição das características.

É interessante observar que o modelo com uma única gaussiana se saiu melhor na detecção de novidade. Ou seja, um modelo que melhor se adapta à distribuição de atributos do conjunto regular não necessariamente é o melhor modelo para detecção de novidades. Ainda, em ambos os modelos (GMM-1 e GMM-32) e durante a fase de seleção de parâmetros, nota-se que os piores resultados são obtidos nas classes 4 (Vundo) e 8 (Gatak) no papel de novidade.

A partir dos resultados apresentados para os modelos SVM, GMM-1 e GMM-32, pode-se concluir que todos os modelos foram adequados para a tarefa de detecção de novidade, considerando os critérios de seleção de parâmetros escolhidos e considerando a dificuldade na tarefa de detectar amostras nunca vistas pelo modelo (novidade).

Em outras palavras, dizer que o SVM tenha obtido PET_{Nov} de 26,19% e o GMM-1, um PET_{Nov} 19,05%, é o mesmo que dizer que o SVM e o GMM-1 detectaram 73,81% e 80,95% das novidades, respectivamente. Muito melhor do que não detectar nada.

O modelo utilizando SVM parece ter um potencial maior para obter melhor resultado de detecção de novidade quando comparado ao modelo GMM, mas os modelos mostraram resultados compatíveis neste trabalho.

6. Conclusão

A detecção de novidade é uma tarefa complexa – não há uma relação óbvia entre os resultados na fase de seleção de parâmetros (usando conjunto de treinamento e validação) e os resultados na fase de teste (usando conjunto de teste). Assim, neste artigo, utilizou-se dois algoritmos de classificadores (GMM e SVM), que obtiveram um resultado satisfatório para a detecção de novidades. A definição da seleção do conjunto de treinamento, validação e teste, além de critérios de aceite de hiperparâmetros, foi essencial para a obtenção do resultado. O método utilizado para a seleção de parâmetros de cada um dos modelos foi adequado, ainda que seja necessário um estudo mais aprofundado, uma vez que alguns experimentos tiveram resultado ruins durante a fase de seleção de parâmetros (classe novidade Vundo e Gatak para GMM, classe novidade Tracur para SVM).

Como sugestão para trabalhos futuros, propomos a utilização de outros atributos, um aprofundamento sobre os resultados ruins em alguns experimentos na fase de seleção de parâmetros, a utilização de *datasets* com maior quantidade de famílias de *malware*, além do emprego de um classificador SVM de novidades probabilístico, conforme sugerido em [16].

Referências

[1] – KASPERKY LABS. Ready… or Not? Balancing future opportunities with future risks. Moscou: Kaspersky Labs, 2017. Disponível em: https://media.kaspersky.com/documents/business/brfwn/en/The-Kaspersky-Lab-Global-IT-Risk-Report_Kaspersky-Endpoint-Security-report.pdf. Acesso em: dd mmm aaaa.

[2] - ANDERSON, B.; QUIST, D.; NEIL, J.; STORLIE, C.; LANE, T. Graph-based malware detection using dynamic analysis. Journal in Computer Virology, v. 7, n. 4, p. 247-258, 2011. DOI: 10.1007/s11416-011-0152-x
[3] – ANDERSON, B.; STORLIE, C.; LANE, T. Improving malware classification: Bridging the static/dynamic gap. In: Proceedings of the workshop on security and artificial intelligence, 5., [S.l.: s.n.], 2012. p. 3-14.
[4] - KRUCZKOWSKI, M.; SZYNKIEWICZ, E. N. Support vector machine for malware analysis and classification. In: Proceedings Of The 2014 Ieee/Wic/Acm International joint conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT). Washington, DC: IEEE Computer Society, 2014. p. 415-420. DOI: 10.1109/WI-IAT.2014.127
[5] - MANGIALARDO, R. J.; DUARTE, J. C. Integrating static and dynamic malware analysis using machine learning. IEEE Latin America Transactions, v. 13, n. 9, p. 3080–3087, 2015.
[6] - ANDRADE, C. A. B.; MELLO, C. G.; DUARTE, J. C. Malware automatic analysis. In: 2013 BRICS Congress On Computational Intelligence And 11th Brazilian Congress on Computational Intelligence. [S.l.: s.n.], 2013. p. 681-686.
[7] - GHARACHEH, M.; DERHAMI, V.; HASHEMI, S.; HAZRATI FARD, S. M. Detection of metamorphic malware based on hmm: A hierarchical approach. International Journal of Intelligent Systems and Applications, v. 8, n. 4, p. 18-25, 2016.
[8] - KONG, D.; YAN, G. Discriminant malware distance learning on structural information for automated malware classification. In: Proceedings of the 19th acm sigkdd international conference on knowledge discovery and data mining, 19., New York: ACM, 2013. p. 1357-1365. DOI: 10.1145/2487575.2488219
[9] - CHEN, Z.; ROUSSOPOULOS, M.; LIANG, Z.; ZHANG, Y.; CHEN, Z.; DELIS, A. Malware characteristics and threats on the internet ecosystem. Journal of Systems and Software, v. 85, n. 7, p. 1650-1672, 2012. DOI: 10.1016/j.jss.2012.02.015
[10] - AHMADI, M.; ULYANOV, D.; SEMENOV, S.; TROFIMOV, M.; GIACINTO, G. Novel feature extraction, selection and fusion for effective malware family classification. In: ACM Conference on Data and Application Security and Privacy (CODASPY), 16., 2016. p. 183-194
[11] – PANCONESI, A.; MARIAN; CUKIERSKI, W. Microsoft Malware Classification Challenge (BIG 2015). [S. l.]: Kaggle, 2015. Disponível em: https://www.kaggle.com/c/malware-classification. Acesso em: 17 set. 2018.
[12] - LIU, L.; WANG, B.-S.; YU, B.; ZHONG, Q.-X. Automatic malware classification and new malware detection using machine learning. Frontiers of Information Technology & Electronic Engineering, v. 18, p. 1336-1347.
[13] CHANDOLA, V.; ARINDAM, B.; VIPIN, K. Anomaly Detection: A Survey. ACM Computing Surveys, v. 41, n. 3, p. 1-58.
[14] – SOLLA, S. A.; LEEN, T. K.; MÜLLER, K.-R. Support vector method for novelty detection. In: Advances in Neural Information Processing Systems, 12., Cambridge: MIT Press, 2000. p. 582-588.
[15] - SHAWE-TAYLOR J.; ŽLIČAR, B. Novelty Detection with One-Class Support Vector Machines. In: MORLINI, I.; MINERVA, T.; VICHI, M. (Ed.). Advances in Statistical Models for Data Analysis. Berlin: Springer, 2015. p. 231-257.
[16] – CLIFTON, L.; CLIFTON, D. A.; ZHANG, Y.; WATKINSON, P.; TARASSENKO, L.; YIN, H. Probabilistic novelty detection with support vector machines. IEEE Transactions on Reliability, v. 63, n. 2, p. 455-467, 2014.
[17] - RONEN, R.; RADU, M.; FEUERSTEIN, C.; YOM-TOV, E.; AH-MADI, M. Microsoft malware classification challenge. [S. l.: s. n.], 2018.
[18] - COELHO, L. P. 2013. Mahotas: Open source software for scriptable computer vision. Journal of Open Research Software, v. 1, n. 1, 2013.
[19] - CHANG, C. C. A practical guide to support vector classification, Technical report, n. 5. Taipei: National Taiwan University, 2005.
[20] – Heightened threat scenario: all the facts in the AV-TEST Security Report 2018/2019. Magdeburg. AVTEST – The Independent IT-Security Institute, 2019. Disponível em: https://www.av-test.org/en/news/heightened-threat-scenario-all-the-facts-in-the-av-test-security-report-2018-2019/. Acesso em: 23 ago. 2019.
[21] – Malware. AVTEST – The Independent IT-Security Institute, 2019. Disponível em: https://www.av-test.org/en/statistics/malware//. Acesso em: 10 set. 2019.
[22] SIKORSKI, M.; HONIG, A. Practical malware analysis: the hands-on guide to dissecting malicious software. San Francisco: No Starch Press, 2012.
[23] SCHÖLKOPF, B.; WILLIAMSON, R.; SMOLA, A.; SHAWE-TAYLOR, J.; PLATT, J. Support vector method for novelty detection. NIPS, v. 12. 1999.