Detecção e Classificação de Documentos contendo Macros Maliciosas com base em Processamento de Linguagem Natural

Anais do Computer on the Beach

Dario Simões Fernandes Filho
dariosfernandes@ufpr.br
Departamento de Informática
Universidade Federal do Paraná
Curitiba, Paraná, Brasil

João Pincovscy
pincovscy@cepesc.gov.br
Centro de Pesquisa e Desenv. para a Segurança das Comunicações
Brasília, DF, Brasil

Cláudio Torres Junior
claudio.torres@ufpr.br
Departamento de Informática
Universidade Federal do Paraná
Curitiba, Paraná, Brasil

Fabrício Ceschin
fceschin@gatech.edu
School of Electr. and Computer Eng.
Georgia Institute of Technology
Atlanta, Georgia, EUA

Paulo Lisboa de Almeida
paulorla@ufpr.br
Departamento de Informática
Universidade Federal do Paraná
Curitiba, Paraná, Brasil

André Grégio
gregio@ufpr.br
Departamento de Informática
Universidade Federal do Paraná
Curitiba, Paraná, Brasil

Abstract

Macros are functions written in Visual Basic for automation within MS Office documents. On the one hand, macros bring many facilities to home users and organizations. On the other hand, they have also stimulated the arise of macro viruses (malware that exploit macros to infect users loading compromised documents, spreadsheets, and presentations). Those viruses may delete data and steal information, and have been causing losses of billions of dollars in global attacks. Although more prevalent in the 1990s and 2000s, macro viruses resurged in the last decade and continue to threat current MS Windows/Office users. In this article, we present a natural language processing-based pipeline to detect macros in MS Office documents and classify them in malicious or benign. Using byte2vec as document representation, we outperform the state-of-the-art in Macro detection, reaching over 99% of Precision-Recall Area Under Curve (PRAUC) metric for four out of seven evaluated classifiers (and over 98% PRAUC in the remaining three classifiers).

Keywords

Detecção de Macros, Análise de Malware, Classificação Binária, Processamento de Linguagem Natural, Aprendizado de Máquina

1 Introdução

Os vírus de macro surgiram no início dos anos 90 como um novo tipo de programa malicioso (*malware*) cujo objetivo era o de explorar os recursos de programação de macro de aplicativos Microsoft Office, tais como arquivos para Word e Excel. Esses vírus eram escritos em uma linguagem de *script* chamada *Visual Basic for Applications* (VBA), a qual permitia automatizar tarefas e executar ações de dentro de aplicativos do Office [1]. Os vírus de macro foram responsáveis por prejuízos de bilhões de dólares, especialmente durante a década de 1990, impulsionados pela popularização do Microsoft Windows e da Suíte Microsoft Office [2, 3].

Em resposta à ameaça deste tipo de vírus, os desenvolvedores começaram a implementar recursos de segurança para proteger os usuários, tais como desabilitar macros por padrão, solicitar aos usuários que habilitassem macros apenas de fontes confiáveis e instalar programas antivírus para verificação de anexos de e-mail e arquivos em busca de vírus de macro conhecidos [4]. Com o tempo, essas medidas de segurança tornaram os vírus de macro menos eficazes, levando a um declínio em sua prevalência. O sistema operacional Windows também evoluiu seus mecanismos nativos de segurança, reforçando as configurações padrão com práticas menos permissivas a partir do Windows XP (lançado em 2001) [5], que introduziu a identificação de inserção de mídias removíveis e o serviço de monitoração contínua chamado de *Windows Security Center* [6]. Porém, os vírus de macro não desapareceram completamente e ressurgiram na última década, agora contando com funcionalidades evasivas como a detecção de execução dentro de máquinas virtuais.

Por ser um problema ainda presente atualmente, neste artigo é apresentado um sistema para detecção de macros em documentos do Office e sua classificação em maliciosos ou benignos, cujo fluxo de operação é baseado em técnicas de processamento de linguagem natural [7]. A arquitetura do sistema proposto é validada por conjuntos de dados públicos contendo macros em documentos diversos, totalizando 16.689 arquivos (15.571 amostras maliciosas e 1.118 amostras não-maliciosas) dos tipo MS Word, Excel e Power Point. Com o sistema proposto, foram obtidos resultados de Área sob a Curva *Precision-Recal* (AUC-PR) superiores a 0.9 (90%) e acurácia superiores a 93%, identificando assim mais documentos maliciosos do que os motores de antivírus disponíveis no VirusTotal (a média de acurácia da detecção dos arquivos maliciosos utilizados neste estudo por antivírus foi de 63%).

As principais contribuições deste artigo incluem: (i) definição e projeto de arquitetura para detecção de macros e classificação dos documentos que as contêm em maliciosos ou benignos; (ii) prototipação do sistema de detecção de macros em um fluxo modular, visando tratar documentos MS Office e classificá-los; (iii) proposta de abordagem para classificação de documentos baseada em diversos algoritmos de aprendizado de máquina aplicados de forma correta (do ponto de vista do treinamento, validação e teste, considerando-se boas práticas e métricas adequadas), em *dataset* mais amplo do que o do estado da arte, e cujos resultados superam a taxa acurácia na detecção de *malware* feita pelo VirusTotal [8], bem como do estado da arte em detecção de macros.

2 Conceitos e Trabalhos Relacionados

Embora os vírus de macro em documentos do Office tenham voltado recentemente a atacar sistemas modernos, não há um extenso corpo de trabalhos acadêmicos sobre o assunto. Um artigo diretamente relacionado a este é o SYMBEXCEL [9], que utiliza a execução simbólica para desobfuscar e analisar macros maliciosos do Excel 4.0 (XL4) de maneira automatizada. Para tanto, SYMBEXCEL opera em três etapas principais: *parsing* do documento malicioso, execução simbólica das fórmulas XL4 e concretização dos valores simbólicos encontrados durante a exploração simbólica. Essa abordagem supera significativamente as ferramentas de desobfuscação existentes, permitindo extrair de forma confiável Indicadores de Comprometimento (IoCs) e outras informações forenses críticas. Os resultados mostram que SYMBEXCEL é capaz de desobfuscar corretamente 23.931 de 24.537 amostras, comparado a 12.375 amostras desobfuscadas por ferramentas existentes. Porém, o escopo do SYMBEXCEL é delimitado em execução simbólica para desobfuscação de amostras identificadas como maliciosas. Por outro lado, o foco do presente artigo é o de detectar macros, ofuscadas ou não, em amostras quaisquer e classificá-las como maliciosas, de forma a alertar o usuário sobre se é possível executar seguramente o documento em questão.

Mais recentemente, Saha et al. conduziram uma análise sistemática de documentos maliciosos com *dataset* de 9.086 amostras [10]. Comparado a outros trabalhos, que se concentram em classificação binária (malicioso/benigno) ou em abordagens específicas para formatos (como OOXML ou RTF), o artigo destaca a diversidade de formatos e os desafios apresentados por técnicas de evasão e ofuscação. Foram implementados pipelines automatizados que combinaram ferramentas como *Yara*, *oletools* e *textract* para detectar indicadores de comprometimento (IoCs), atingindo uma taxa de detecção em 98,32% dos casos processados (para 60,55% dos arquivos RTF, a análise falhou devido a erros de parsing ou cabeçalhos malformados; 24,64% dos documentos com macros Excel 4.0 apresentaram desafios de extração devido a obfuscações complexas ou desvios no comportamento esperado). Os resultados obtidos revelam limitações nos métodos atuais de análise automatizada, especialmente para Excel macros XL4 e arquivos RTF malformados. Isto indica a necessidade de ferramentas mais robustas para a detecção e mitigação de ameaças documentais emergentes. Nesse sentido a proposta do presente artigo surge como uma abordagem que pode sanar algumas das limitações apresentadas, conforme apresentado nas Seções 3.1 e 4. No restante dessa seção, será apresentada a evolução da segurança sistemas operacionais Microsoft Windows, a estrutura dos arquivos do Microsoft Office e das Macros que permitem a proliferação de códigos maliciosos, um histórico sobre os vírus de macro mais impactantes das últimas décadas e, por fim, conceitos básicos sobre as técnicas de aprendizado de máquina usadas neste artigo.

2.1 Evolução do Microsoft Windows

Sistemas Operacionais MS *Windows* existem há aproximadamente quatro décadas e se tornaram adotados pela grande maioria dos usuários de computadores pessoais [11]. Começando a partir do *Windows 1* em 1985, cujo foco era prover uma interface gráfica para facilitar sua usabilidade, e evoluindo até o atual *Windows 11*, que introduz e reforça mecanismos de segurança e integração entre diferentes ambientes [12].

As soluções de segurança incorporadas ao longo do tempo têm contribuído para tornar o sistema *Windows* seguro e livre das ameaças mais comumente exploradas. A adoção de métodos como o *DEP (Data Execution Prevention)*—o qual previne a execução de código em áreas de memória que só contêm dados [13]—e *ASLR (Address Space Layout Randomization)*—que garante que bibliotecas do sistema sejam carregadas em regiões diferentes a cada execução, evitando assim que atacantes utilizem posições conhecidas de memória [14]—garantem um nível de proteção adicional contra ataques de corrupção de memória (e.g., *buffer overflow* [15]).

Além dos mecanismos de segurança já integrados ao sistema operacional, os sistemas *Windows* contam também com uma solução antivírus, o *Windows Defender*, cujo objetivo é identificar possíveis códigos maliciosos que tentam executar no ambiente [16]. O *Windows Defender*, que começou como uma solução *anti-spyware* adquirida pela *Microsoft* [17], atualmente é considerado como uma das melhores soluções antivírus disponíveis, provendo alta taxa de deteção de malware [18] e superando soluções comerciais tradicionais.

2.2 Arquivos do Microsoft Office

O Microsoft Office é um conjunto de aplicativos para criação, modificação e leitura de documentos de texto, apresentações e planilhas. Cada aplicativo responde por um tipo de documento de texto através de sua extensão e, embora provejam diferentes representações, todos compartilham uma base de formato comum. O *Office Open XML* (OOXML) é tanto um formato de arquivo aberto quanto um padrão internacional desenvolvido pela Microsoft para ser utilizado com o conjunto de aplicativos do Office. A ideia do OOXML, aprovado como um padrão internacional ISO/IEC em 2008, é ser uma alternativa ao formato de documento aberto *Open Document Format for Office Applications* (ODF), usado por ferramentas como o LibreOffice [19].

O padrão OOXML é projetado para permitir que os arquivos sejam lidos e escritos por vários programas e plataformas diferentes, provendo certo grau de interoperabilidade. Os documentos OOXML são, na verdade, uma coleção de vários arquivos compactados em um único arquivo .zip [20], cujos diferentes tipos existentes possuem sua própria hierarquia de diretórios. Na Figura 1, ilustra-se a estruturação de um DOCX [21], onde **[Content_Types].xml** é um arquivo de especificação que relaciona os arquivos presentes, permitindo assim o correto processamento destes pelos aplicativos. Há também três diretórios principais na raiz, listados a seguir:

- **docProps**: possui as propriedades do OOXML.
- **_rels**: contém o relacionamento dos arquivos dentro do OOXML.
- **word**: Além de possuir os arquivos relacionados ao conteúdo principal do OOXML, este diretório também tem um subdiretório chamado **_rels**, que contém os arquivos de relacionamento entre os arquivos de **word**, bem como o **theme**, que possui arquivos relacionados ao tema do documento.

Essa estrutura permite que os arquivos sejam facilmente manipulados, pois cada componente do documento (texto, imagens, gráficos, etc.) pode ser acessado e modificado individualmente. Os

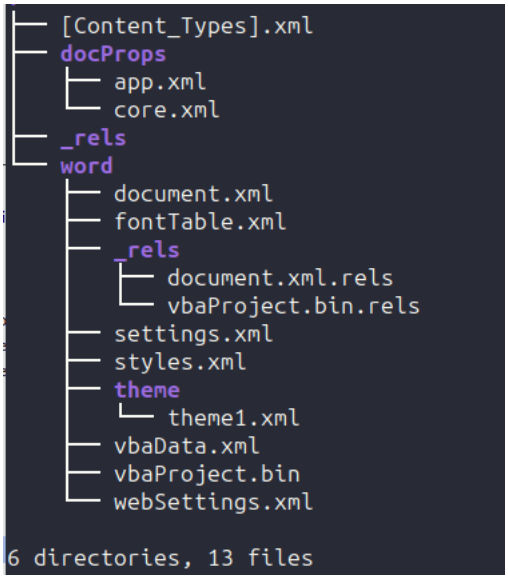

Figura 1: Anatomia de um documento no padrão OOXML

arquivos OOXML são usados principalmente por aplicativos como o Microsoft Word, Excel e PowerPoint. A introdução desses formatos de arquivo (DOCX, XLSX e PPTX) deu-se no Office 2007 [22] e sua estrutura visa ser mais segura do que os formatos binários mais antigos (DOC, XLS e PPT). Nos formatos baseados em XML [23], as macros são armazenadas separadamente do conteúdo do documento e os arquivos que contêm macros usam diferentes extensões de arquivo (DOC/XLS/PPT[M|X]). A extensão finalizada por "M" (por exemplo, DOCM) indica a presença de macros no documento.

2.2.1 Macros Excel 4.0: Estrutura e Funcionalidade. As macros do Excel 4.0, que usam uma linguagem de macro herdada anterior ao VBA, têm sido cada vez mais empregadas por agentes mal-intencionados nos últimos anos. O Excel 4.0, lançado em 1992, foi a primeira versão do Microsoft Excel a introduzir recursos de macro, permitindo aos usuários automatizar tarefas e realizar cálculos complexos. Essas macros foram criadas usando uma linguagem de macro específica, Excel 4.0 Macro Language (XLM) [24], que desde então foi substituída pela linguagem *Visual Basic for Applications* (VBA). Apesar da obsolescência do XLM, os últimos anos testemunharam um ressurgimento do uso de macros do Excel 4.0 para fins maliciosos.

O VBA é uma linguagem de programação incorporada aos aplicativos do Office e que permite aos usuários automatizar tarefas e personalizar funções. O VBA é executado em um ambiente de desenvolvimento integrado (IDE) fornecido pelos aplicativos do Office, como o Editor VBA no Excel. Os benefícios de flexibilidade e automação oferecidos pelo VBA apresentam a dualidade potencializar uma maior eficiência e produtividade por parte dos usuários, ao passo que também oferece caminhos que podem ser explorados por atacantes para desenvolver e disseminar *scripts* maliciosos. Esses *scripts* podem ser usados para infectar sistemas, roubar informações confidenciais e executar uma variedade de atividades mal-intencionadas. Na Tabela 1, descreve-se as utilidades das macros VBA nos diversos tipos de documentos do Office, as quais proveem uma variedade de vetores de comprometimento.

Tabela 1: Uso das macros por aplicativo do Office

Ferramenta	Descrição
Excel	criar e modificar documentos, formatar texto, gerar tabelas, gerenciar estilos e automatizar tarefas repetitivas.
Word	automatizar processos, analisar e manipular dados, criar gráficos, gerar relatórios e construir formulários e interfaces de usuário.
Power Point	criar apresentações dinâmicas, modificar slides e objetos, automatizar a inserção de elementos, como gráficos e tabelas, e personalizar a navegação e a interação com a apresentação.

2.3 Programas maliciosos

Programas maliciosos (*malwares*) podem ser definidos como um tipo de código que se instala em algum dispositivo computacional da vítima, seja sem consentimento ou por meio de enganação, resultando na violação da segurança desse dispositivo, sistema e/ou rede [25]. Existem vários tipos de *malware* categorizados de acordo com seu comportamento predominante, porém os programas maliciosos modernos, por sua modularidade e sofisticação, podem se enquadrar em mais de uma categoria [26]:

- **Vírus:** anexa-se a outros programas e necessitam de ativação do usuário.
- ***Worm:*** busca por vulnerabilidades em uma rede e as explora, disseminando-se autonomamente.
- ***Trojan:*** disfarça-se de uma aplicação ou objeto legítimo de forma que o usuário o instale.
- ***Bot:*** comunica-se com um servidor de comando e controle para receber ordens de ataque ou atualizações.
- **Outros:** podem ser transmitidos como carga útil de outros programas maliciosos e servem para abrir portas, esconder processos/arquivos/conexões, ou capturar teclas e *screenshots*, tais como *backdoor*, *rootkit*, *spyware*, etc.

O escopo deste trabalho é delimitado em programas maliciosos cuja carga útil explore as funcionalidades de Macro em documentos no formato Microsoft Office. Em geral, uma macro maliciosa pode ser um vírus que infectou um documento e, por enganar o usuário também ser um *Trojan*; caso a macro também tenha capacidade de propagar seu documento infectado pela rede, ela apresenta um comportamento de *worm* e assim por diante.

2.3.1 Evolução dos Vírus de Macro. O primeiro vírus de macro conhecido, *Concept*, foi descoberto em 1995 [27]. Esse vírus se espalhava por documentos do Word infectados e, quando um documento infectado era aberto, o vírus se copiava no modelo global do usuário, permitindo que infectasse outros documentos. À medida que os vírus de macro se tornaram mais sofisticados, eles começaram a se

espalhar por meio de anexos de e-mail, muitas vezes disfarçados de mensagens importantes para induzir os usuários a abrir os arquivos infectados.

Entre o final dos anos 1990 e início dos anos 2000, os vírus de macro continuaram a ser uma ameaça significativa, com alguns exemplos bem conhecidos como *Melissa* e *Love Letter* (descobertos em 1999 e 2000, respectivamente) [28]. O vírus Melissa tornou-se particularmente conhecido, pois se espalhou rapidamente por listas de contatos de e-mail, causando interrupções consideráveis nos sistemas de e-mail em todo o mundo. Já o *Love Letter*, também conhecido como"ILOVEYOU", foi um *worm* de macro que copiou o mecanismo de propagação do *Melissa*, causando a infecção de milhões de computadores e prejuízos de bilhões de dólares [29].

Dridex, Emotet, TrickBot e Ryuk são exemplares de *malware* que afetaram muitos usuários e organizações ao longo dos anos [30]. Dridex é um Trojan bancário projetado para roubar informações bancárias e realizar transações fraudulentas visto primeiramente em 2011. Para comprometer sua vítima, envia e-mail de *phishing* com anexos maliciosos do Word (DOCM) contendo *scripts* VBA que fazem o download e instalam a carga útil do Dridex no sistema da vítima. Emotet, identificado pela primeira vez em 2014, era inicialmente um trojan bancário que evoluiu para um vetor de infecção de malware mais versátil e perigoso. As técnicas de infecção utilizadas pelo Emotet são similares às do Dridex, podendo se propagar pela rede como um *worm* e baixar e instalar outras famílias de malwares, tais como TrickBot (Trojan bancário surgido em 2016) e Ryuk (ransomware descoberto em 2018).

Embora possam ter objetivos diferentes (Trojans bancários e ransomwares), todos os exemplares de malware supracitados compartilham a técnica de propagação via *phishing* contendo documentos do Office com macros maliciosas em VBA [31] e estão frequentemente relacionados atuando como vetores de infecção uns dos outros. Em 2020, os vírus de macro nos formatos DOCX, XLSX e PPTX eram relativamente incomuns em comparação com sua prevalência na década de 1990 e no início dos anos 2000, principalmente devido a medidas de segurança aprimoradas vigentes na versão 365 do Office. No entanto, eles não foram totalmente eliminados, uma vez que os atacantes encontraram maneiras inovadoras de explorar vulnerabilidades e distribuir macros maliciosas nesses formatos de arquivo. Um exemplo de vírus de macro direcionado aos usuários do Office 365 em 2020 foi o Trojan bancário Trickbot, que retornou evoluído em um *malware* modular e de múltiplos estágios [32], inaugurando um período de novos ataques usando técnicas de infecção clássicas.

2.4 Técnicas de Aprendizado de Máquina

A classificação é uma tarefa fundamental na aprendizagem de máquina, onde o objetivo é categorizar dados em classes predefinidas. Devido à natureza do trabalho proposto, são utilizados neste trabalho classificadores binários, que têm por objetivo classificar determinado objeto (documento com macro) entre uma de duas classes: malicioso ou benigno. Para tal, avaliamos o uso de diversos classificadores clássicos, entre eles:

(1) **Classificadores que assumem a independência entre as variáveis de entrada**:
- *Logistic Regression*, técnica amplamente utilizada para problemas de classificação binária. Ela modela a probabilidade de uma amostra pertencer a uma classe usando a função logística (sigmoide). A principal suposição é que as variáveis de entrada são linearmente relacionadas ao logit da variável resposta. Embora a Regressão Logística não assuma explicitamente a independência entre variáveis de entrada, ela se beneficia quando essas variáveis têm pouca correlação [33];
- *Gaussian Naive Bayes*, classificador probabilístico baseado no Teorema de Bayes, que assume a independência condicional entre as variáveis de entrada dado a classe. Além disso, assume que a distribuição das variáveis contínuas segue uma distribuição normal (Gaussiana). Devido à simplicidade dessa suposição, o Naive Bayes pode ser muito eficiente, especialmente em problemas de alta dimensionalidade [34].

(2) **Classificadores com fronteira linear**:
- *Support vector machine (SVM) com kernel Linear* é um classificador que busca encontrar um hiperplano de fronteira de decisão que maximiza a margem entre as classes. Com o uso de um *kernel* linear, o classificador assume que os dados são linearmente separáveis. Esse tipo de classificador é especialmente útil em problemas de altas dimensões com dados esparsos.

(3) **Classificadores com fronteira não-linear**:
- *SVM com kernel de Função de Base Radial (RBF)*, é um classificador similar ao SVM de fronteira linear. No entanto, devido ao kernel RBF, essa versão do SVM pode capturar relações não-lineares entre as variáveis de entrada ao transformar os dados para um espaço de maior dimensão onde uma fronteira linear pode ser encontrada [35];
- *Árvores de Decisão*, são classificadores que particionam iterativamente o espaço de entrada em subconjuntos homogêneos em termos da variável alvo. Cada nó da árvore representa uma condição sobre uma variável de entrada, e os ramos representam os resultados dessas condições. As Árvores de Decisão são capazes de capturar relações não-lineares, e possuem respostas interpretáveis, sendo essa uma de suas principais vantagens [36].

(4) **Métodos de classificação baseados em *ensemble***:
- *Random Forest*, método de *ensemble* que constrói múltiplas Árvores de Decisão em diferentes subconjuntos dos dados e combina suas previsões para melhorar a precisão e reduzir o *overfitting*. Cada árvore no *ensemble* é treinada em um subconjunto de amostras dos dados de treinamento, e a seleção das variáveis em cada nó é aleatória, o que contribui para a diversidade das árvores e a robustez do modelo [37];
- *Gradient Boosting*, técnica de *ensemble* que cria modelos sequenciais, onde cada modelo subsequente tenta corrigir os erros de seu predecessor. Em cada iteração, um novo modelo é ajustado aos resíduos dos modelos anteriores, e os modelos são combinados de forma

ponderada. Isso permite que o Gradient Boosting construa modelos muito precisos, embora também possa ser mais suscetível ao *overfitting* se não for adequadamente regulado [38]. Neste trabalho, usamos o Gradient Boosting com árvores de decisão de classificador base.

Dado que os classificadores usados dependem de técnicas de extração de características para geração de vetores de características de tamanho fixo, e os dados de macros podem ser considerados como texto livre em linguagem natural, os classificadores são alimentados com características geradas a partir de técnicas de Processamento de Linguagem Natural, como descrito na Seção 2.4.1.

2.4.1 Processamento de Linguagem Natural. Trabalhos recentes na literatura consideram utilizar apenas os *bytes* dos arquivos como características [39], o que torna possível a utilização de um único modelo, uma vez que a representação em *bytes* é comum a todos os formatos de arquivos utilizados. O estado da arte baseia-se em *word embeddings* (cujo precursor é o *Word2Vec*), uma forma de representar documentos e palavras definindo vetores numéricos que mantém palavras semelhantes próximas, baseando-se no contexto, em um espaço dimencional de tamanho N [40]. Estendendo-se para *bytes*, o *Byte2Vec* [41] é uma adaptação do *Word2Vec* que aprende a relação (contexto) entre *bytes* de arquivos afim de criar uma representação para os mesmos.

De forma similar ao *Word2Vec*, o *Byte2Vec* utiliza um *bag-of-bytes* (dicionário de bytes), que armazena os bytes conhecidos no conjunto de treinamento (*bag-of-words* no *Word2Vec*, um dicionário de palavras). Assim, a arquitetura da rede neural não supervisionada é baseada na *Continuous Bag of Words (CBOW)*, que tenta predizer a palavra em foco (ou alvo) baseando-se nas palavras de contexto, que são aquelas que ficam ao redor do alvo. A arquitetura *Continuous Bag-of-Bytes (CBB)* apresentada nesse trabalho pode ser vista na Figura 2, em que pode-se observar N arquivos que geram um dicionário de *bytes* (*bag-of-bytes*). Para cada arquivo, seus *bytes* presentes nesse dicionário são transformados em vetores *one-hot* e utilizados como entrada do algoritmo, uma rede neural é treinada para predizer o *byte* em foco. Ao final do processo, cada *byte* possui uma representação em um espaço dimensional de tamanho N, gerando assim o vetor de características que será interpretado pelos classificadores.

3 Fluxo do Sistema Proposto e Metodologia

O sistema proposto foi pensado para ser modular, de forma a facilitar sua atualização e manutenção. Tanto o fluxo das amostras quanto os módulos que compõem o sistema proposto estão descritos a seguir.

- **Identificador de Macro**: Antes da identificação da macro, ocorre a detecção do tipo de arquivo sendo analisado (outros formatos além do OOXML são detectados, como PDFs e vídeos). Isso é feito com base na extensão do arquivo ou no seu *MIME type*, que é um identificador que descreve o tipo de conteúdo do arquivo. Com base na definição do tipo de documento que estamos lidando, o sistema seleciona o módulo mais adequado para identificar macros. Para tanto, escolheu-se a ferramenta *oletools.olevba*, que é utilizada para análise de macros VBA em arquivos do tipo OOXML [42].
- **Identificador de Ofuscação**: Caso haja alguma macro ofuscada, esta é detectada utilizando a ferramenta *oletools.olevba* e após, realiza-se algumas tentativas de desofuscação. A ferramenta em questão consegue detectar e desofuscar técnicas de ofuscação amplamente utilizadas.
- **Extrator**: Após completar as etapas anteriores, detecta-se os possíveis scripts VBA. Para cada script, armazena-se o resultado final da extração de características através de Processamento de Linguagem Natural, como descrito na Seção 2.4.1 . Essa etapa é igual para todos os arquivos OOXML atualmente identificados pelo sistema (DOCX, XLSX e PPTX). As informações são armazenadas em um banco de características, juntamente com o documento original e seus respectivos *hashes* de identificação.
- **Classificador**: Para realizar a classificação dos documentos contendo macros como maliciosos ou não, foram postos à prova os classificadores descritos na Seção 2.4 (*Logistic Regression*, *Gaussian Naive Bayes*, SVM, Árvores de Decisão, *Random Forest* e *Gradient Boosting*) cujas técnicas e resultados obtidos estão discutidos na Seção 4.

3.1 Metodologia de Classificação

Para otimizar a forma como o *Byte2Vec* original [41] treina e não carregar todos os binários dos arquivos em memória, foi aplicada a estratégia de dividir o processamento em blocos de K *bytes*, como ilustrado na Figura 3. Desta forma, a rede é treinada com sequências de *bytes* do arquivo, traduzindo essas sequências para vetores na forma *one-hot encoding* para dar entrada no treinamento do modelo utilizando a arquitetura *CBAB* já apresentada.

Com o modelo treinado, divide-se cada arquivo da base de teste em blocos de K *bytes* e aplica-se o modelo *Byte2Vec* treinado. Como saída, tem-se os vetores de tamanho N de cada *byte*, que são somados e divididos por K, gerando um vetor médio por bloco. Ao final, todos os vetores de cada bloco são somados e divididos pelo número de blocos para gerar um único vetor de características que representa o arquivo de entrada, conforme apresentado na Figura 4. Finalmente, basta aplicar esse método de extração de características nos arquivos para se obter uma representação e utilizá-la em um modelo treinado.

4 Testes e Resultados

Como estudo de caso, foi usado neste trabalho o conjunto de dados *Malicious MS Office documents*, que contém inúmeros formatos de arquivos maliciosos e benignos disponíveis, tais como PDF, XLSX e DOCX [43]. No total, são 16.689 arquivos, nos quais 15.571 (93, 30%) são malignos e 1.118 (6, 7%) são benignos. A Figura 5 apresenta a distribuição de tipos de arquivos do *dataset* de acordo com seus *MIME types*, que em sua maioria é composto por documentos `application/msword` (cerca de 70%).

Como não há uma padronização entre diferentes formatos de arquivos, o método tradicional de se extrair características para a construção de modelos consiste em extrair seus metadados, tais como *strings* de um documento *word*, código *JavaScript* de um PDF, etc. Por conta disso, diferentes documentos não podem ser correlacionados em um único modelo, já que não compartilham os mesmos metadados, sendo necessário a criação de um modelo por

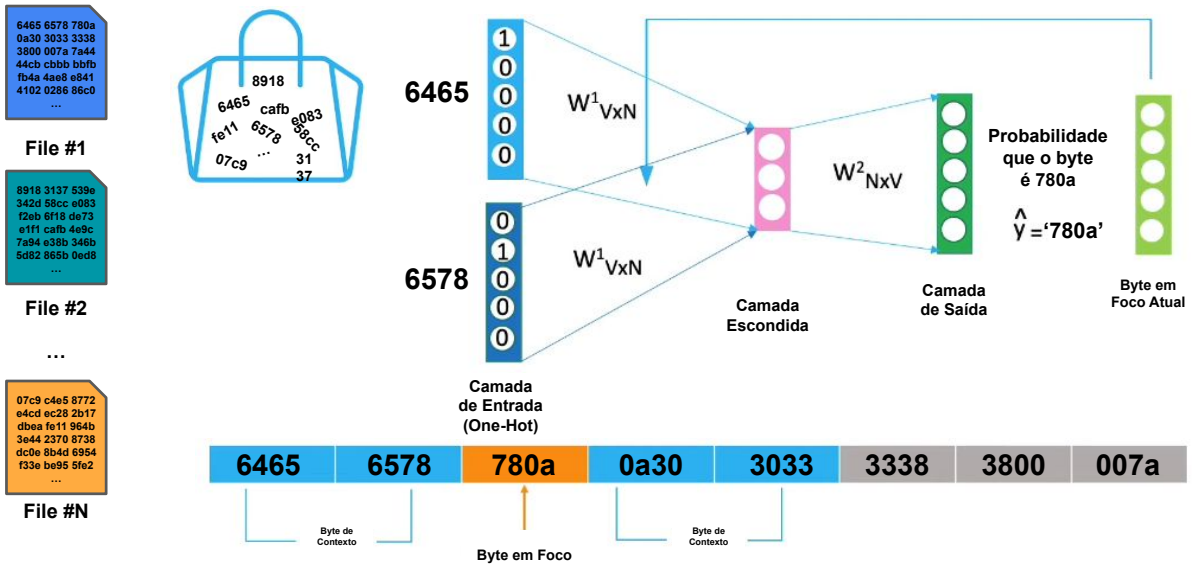

Figura 2: Arquitetura *Continuous Bag-of-Bytes* do *Byte2Vec*.

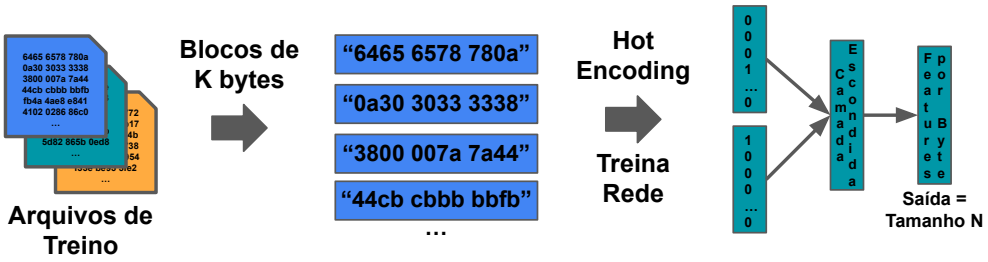

Figura 3: Treinamento do Byte2Vec utilizando blocos de *K bytes*.

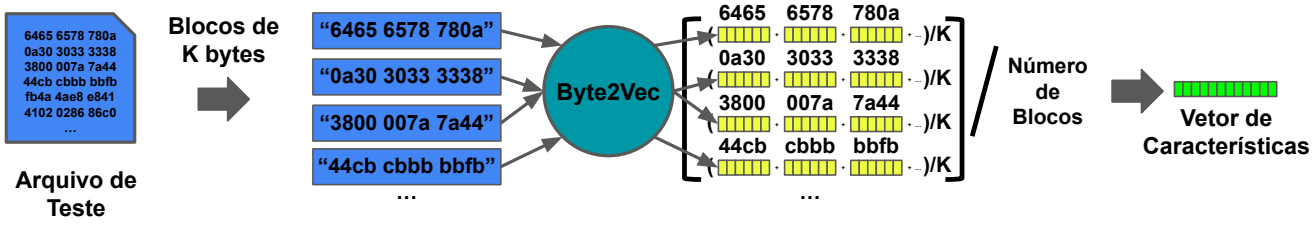

Figura 4: Extração de Características utilizando o *Byte2Vec*.

formato de arquivo (por exemplo, um detector de arquivo malicioso para PDF e outro para DOCX).

Neste trabalho, o conjunto de dados foi dividido ao meio por tipo de arquivo, com metade sendo utilizada para treinar e a outra metade para testar os modelos gerados. O processo de treinamento teste foi realizado 10 vezes, com seleção aleatória de exemplares, cálculo da média dos resultados e uso dos classificadores descritos na Seção 2.4. A Tabela 2 apresenta a *Area Under the Precision-Recall Curve* (*PRAUC*), métrica recomendada para problemas desbalanceados [44], para os classificadores mencionados e diferentes tamanhos do vetor de características N e números de bytes K por bloco. Como é possível observar, quanto maior o tamanho do vetor de características N e o números de bytes K por bloco, melhor o *PRAUC* no geral. O salto é ainda maior quando aumentamos o valor de ambas as variáveis, como apresentado a partir de $N = 64$ e $K = 64$, cujos valores se aproximam de 100%, resultado promissor considerando que a base é muito desbalanceada.

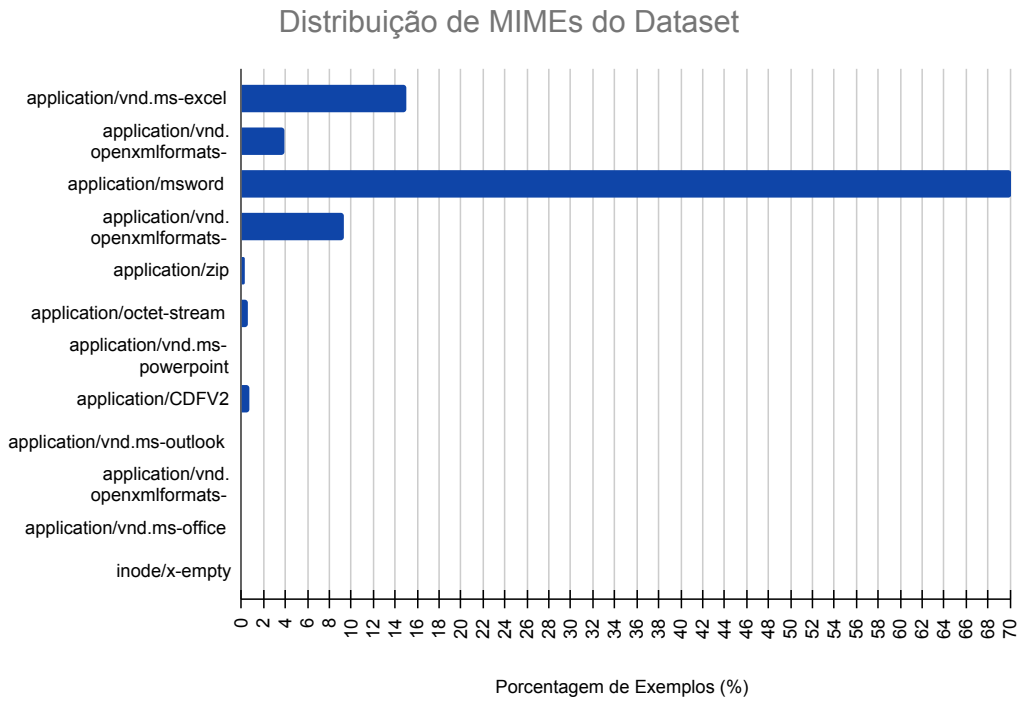

Figura 5: Distribuição de tipos de arquivos do *dataset* utilizado de acordo com seus MIME types.

Tabela 2: *Area Under the Precision-Recall Curve* (*PRAUC*) para diferentes classificadores, tamanhos do vetor de caraterísticas N e números de *bytes* K por bloco.

N	K	Random Forest	SVM RBF	Logistic Regression	Gaussian Naive Bayes	SVM Linear	Decision Tree	Gradient Boosting (DT)
2	**512**	98.17%	93.19%	95.53%	95.54%	96.92%	97.41%	97.74%
3	**512**	98.80%	96.92%	94.54%	95.73%	96.65%	97.87%	98.84%
3	**2048**	98.83%	96.67%	94.58%	95.39%	96.65%	97.86%	98.86%
2	**4096**	98.22%	93.61%	95.52%	95.56%	96.41%	97.28%	97.83%
64	**64**	99.79%	99.09%	99.33%	96.27%	97.01%	98.48%	99.53%
128	**128**	99.83%	99.65%	99.43%	97.68%	**98.52%**	98.55%	**99.75%**
512	**512**	**99.84%**	**99.67%**	**99.54%**	**98.27%**	97.38%	**98.61%**	99.74%

Vale ainda notar que o SVM com Kernel Linear foi um dos poucos classificadores incapazes de alcançar acurácias acima dos 99%, e apresentou uma queda de acurácia de mais de 1% quando o tamanho do vetor de características saltou de 128 para 512. Isso indica que, ao menos em altas dimensões, as fronteiras para o problema podem se tornar complexas e não lineares. Essa hipótese é corroborada pelos bons resultados gerados pelo SVM de kernel RBF considerando vetores de 512 dimensões, já que essa função de Kernel possibilita a aprendizagem de fronteiras não-lineares (Figura 6) e, consequentemente, a distinção entre as amostras.

5 Conclusão

Neste trabalho, apresentou-se um breve histórico da evolução dos sistemas operacionais MS Windows. Discutiu-se também uma linha do tempo dos vírus de macro desde seu surgimento até seu declínio, passando por sua volta nos tempos recentes. Com isso, motivou-se a introdução de um sistema para detecção de macros maliciosas em documentos do MS Office. Para a classificação dos documentos, usou-se sete algoritmos de aprendizado de máquina da literatura. O sistema foi prototipado e validado com *dataset* de dezenas de milhares de amostras com documentos maliciosos e benignos, e alcançou taxas de sucesso (medidas via AUC-PR) superiores à taxa de acurácia na detecção feita por motores antivírus disponíveis no VirusTotal, bem como da literatura no estado da arte, e utilizando mais amostras (>99% em 4 de 7 algorimos avaliados pela proposta deste artigo, comparado a 98,32% de [10], que falou em processar 60,55% de suas amostras RTF).

6 Agradecimentos

O presente trabalho foi parcialmente realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

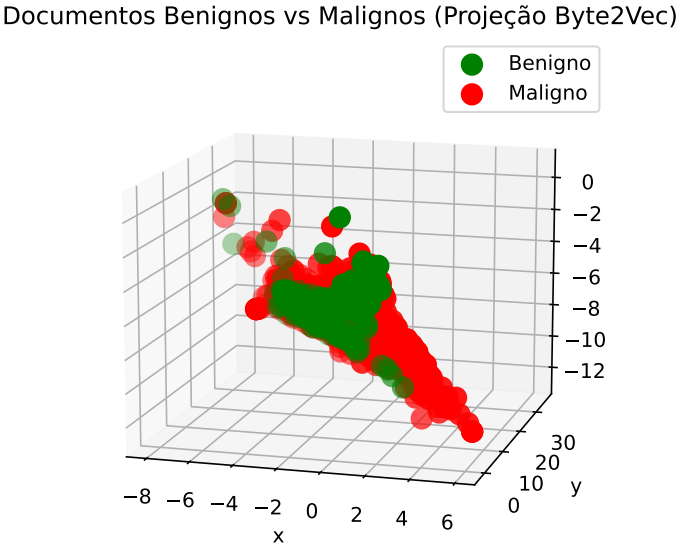

Figura 6: Projeção do *Byte2Vec* para $K = 3$ e $N = 512$.

Referências

[1] John McCumber. *Assessing and managing security risk in IT systems: A structured methodology*. CRC Press, 2004.
[2] F-Secure. Virus:w32/concept. https://www.f-secure.com/v-descs/concept.shtml, 1995.
[3] Lee Garber. Melissa virus creates a new type of threat. *Computer*, 32(06):16–19, 1999.
[4] James E Hunton. Facts and fables about computer viruses. *Journal of Accountancy*, 185(5):39, 1998.
[5] Tom Fout. Universal plug and play in windows xp. *Microsoft Corporation*, 2001.
[6] Haoyang Xie, Keyu Jiang, Xiaohong Yuan, and Hongbiao Zeng. Forensic analysis of windows registry against intrusion. *International Journal of Network Security & Its Applications*, 4(2):121, 2012.
[7] Filip Graliński, Krzysztof Jassem, and Marcin Junczys-Dowmunt. Psi-toolkit: A natural language processing pipeline. *Computational linguistics: Applications*, pages 27–39, 2013.
[8] Chronicle. Virustotal. https://www.virustotal.com, June 2023.
[9] Nicola Ruaro, Fabio Pagani, Stefano Ortolani, Christopher Kruegel, and Giovanni Vigna. Symbexcel: Automated analysis and understanding of malicious excel 4.0 macros. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 964–981. IEEE, 2022. doi: 10.1109/SP46214.2022.9833765.
[10] Aakanksha Saha, Jorge Blasco, and Martina Lindorfer. Exploring the Malicious Document Threat Landscape: Towards a Systematic Approach to Detection and Analysis. In *Proceedings of the 3rd Workshop on Rethinking Malware Analysis (WoRMA)*, 2024. doi: 10.1109/EuroSPW61312.2024.00065.
[11] Statista. Global market share held by operating systems for desktop pcs, from january 2013 to january 2023. https://www.statista.com/statistics/218089/global-market-share-of-windows-7/, Janeiro 2023.
[12] CNET. Microsoft windows through the years: Version 1.0 to windows 11. https://www.cnet.com/pictures/microsoft-windows-through-the-years-version-1-0-to-windows-11/null/, Setembro 2021.
[13] Microsoft. Prevenção de execução de dados. https://learn.microsoft.com/pt-br/windows/win32/memory/data-execution-prevention, março 2023.
[14] Microsoft. Address space layout randomization. https://learn.microsoft.com/en-us/windows/security/threat-protection/overview-of-threat-mitigations-in-windows-10#address-space-layout-randomization, março 2023.
[15] Aleph One. Smashing the stack for fun and profit. *Phrack*, 7(49), November 1996. URL http://www.phrack.com/issues.html?issue=49&id=14.
[16] Microsoft. Microsoft defender antivirus in windows. https://learn.microsoft.com/en-us/microsoft-365/security/defender-endpoint/microsoft-defender-antivirus-windows?view=o365-worldwide, março 2023.
[17] TechRepublic. Windows defender: Past, present, and future. https://www.techrepublic.com/article/windows-defender-past-present-and-future/, Novembro 2016.
[18] AV-TEST. Av-test product review and certification report – mar-apr/2023. https://www.av-test.org/en/antivirus/home-windows/windows-11/april-2023/microsoft-defender-antivirus-consumer-4.18-231214/, Abril 2023.
[19] ODF version 1.3. Open document format for office applications (opendocument) version 1.3. part 3: Opendocument schema. Standard, OASIS OPEN, abril 2021.
[20] Frederic P Miller, Agnes F Vandome, and John McBrewster. *Lossless Data Compression: Data Compression, Algorithm, Lossy Compression, Bit Rate, ZIP (File Format), Unix, Gzip, Portable Network Graphics, Graphics Interchange Format, Tagged Image File Format*. Alpha Press, 2009.
[21] Wikipedia. Office open xml file formats. https://en.wikipedia.org/wiki/Office_Open_XML_file_formats#:~:text=docProps%2Fcore.xml,word%2Fdocument.xml, Maio 2023.
[22] Bora Park, Jungheum Park, and Sangjin Lee. Data concealment and detection in microsoft office 2007 files. *Digital Investigation*, 5(3-4):104–114, 2009.
[23] Gerhard Post, Samad Ahmadi, Sophia Daskalaki, Jeffrey H Kingston, Jari Kyngas, Cimmo Nurmi, and David Ranson. An xml format for benchmarks in high school timetabling. *Annals of Operations Research*, 194:385–397, 2012.
[24] Martin Raubal, Bernhard Gaupmann, and Werner Kuhn. Demonstrating raster operations with spreadsheets.
[25] Ed Skoudis and Lenny Zeltser. *Malware: Fighting Malicious Code*. Prentice Hall PTR, USA, 2003. ISBN 0131014056.
[26] Eric Filiol. *Computer Viruses: from theory to applications*. Springer-Verlag, France, 2005. ISBN 2287239391.
[27] J. Hruska. Virus detection. In *European Conference on Security and Detection, 1997. ECOS 97.*, pages 128–130, 1997. doi: 10.1049/cp:19970437.
[28] James A. Whittaker and Andres De Vivanco. Neutralizing windows-based malicious mobile code. In *Proceedings of the 2002 ACM Symposium on Applied Computing*, SAC '02, page 242–246, New York, NY, USA, 2002. Association for Computing Machinery. ISBN 1581134452. doi: 10.1145/508791.508841. URL https://doi.org/10.1145/508791.508841.
[29] S.R. Subramanya and N. Lakshminarasimhan. Computer viruses. *IEEE Potentials*, 20(4):16–19, 2001. doi: 10.1109/45.969588.
[30] Femi Daramola. *The Threat and Economic Impact of Modular Malware on United States' Critical Infrastructures and Organizations*. PhD thesis, Utica College, 2020.
[31] Silviu Constantin Vitel, Gheorghe Balan, and Dumitru Bogdan Prelipcean. Improving detection of malicious office documents using one-side classifiers. In *2019 21st International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pages 243–247. IEEE, 2019.
[32] Joint Cybersecurity Advisory. Trickbot malware. Technical report, FBI/CISA, março 2021.
[33] David W Hosmer, Stanley Lemeshow, and Rodney X Sturdivant. *Applied Logistic Regression*. John Wiley & Sons, 2013.
[34] Kevin P Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.
[35] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.
[36] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.
[37] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
[38] Jerome H Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.
[39] Edward Raff, Jon Barker, Jared Sylvester, Robert Brandon, Bryan Catanzaro, and Charles Nicholas. Malware detection by eating a whole exe, 2017.
[40] Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013. URL http://arxiv.org/abs/1301.3781.
[41] Mahmood Yousefi-Azar, Len Hamey, Vijay Varadharajan, and Shiping Chen. Byte2vec: Malware representation and feature selection for android. *The Computer Journal*, 63(1):1125–1138, 2020. doi: 10.1093/comjnl/bxz121.
[42] Decalage. olevba - ole and openxml vba macro extraction. https://github.com/decalage2/oletools/wiki/olevba.
[43] Vasilios Koutsokostas, Nikolaos Lykousas, Gabriele Orazi, Theodoros Apostolopoulos, Amrita Ghosal, Fran Casino, Mauro Conti, and Constantinos Patsakis. Malicious ms office documents dataset. https://doi.org/10.5281/zenodo.4559436, February 2021. URL https://doi.org/10.5281/zenodo.4559436.
[44] Daniel Arp, Erwin Quiring, Feargus Pendlebury, Alexander Warnecke, Fabio Pierazzi, Christian Wressnegger, Lorenzo Cavallaro, and Konrad Rieck. Dos and don'ts of machine learning in computer security. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 3971–3988, Boston, MA, August 2022. USENIX Association. ISBN 978-1-939133-31-1. URL https://www.usenix.org/conference/usenixsecurity22/presentation/arp.