Aprendizado de máquina adversário contra Detectores de Malware

Gabriel H. N. E. Silva[1], **Bruno Bogaz Zarpelão**[1]

[1]Departamento de Computação – Universidade Estadual de Londrina (UEL)
Caixa Postal 10.011 – CEP 86057-970 – Londrina – PR – Brasil

gabriel.henrique1@uel.br, brunozarpelao@uel.br

Abstract. *Malware detectors are the main line of defense against the growing threat of malicious software. Malicious software, known as malware, aim to steal confidential information, exploit security vulnerabilities and bring inconvenience to users. To help detect these programs, the use of machine learning (ML) is applied. ML with the help of data analysis and data extraction techniques can detect patterns attached to malware behavior. However, these algorithms are vulnerable through attacks of adversarial machine learning. These attacks have the ability to evade or manipulate the classifications of ML models, allowing malicious programs to be classified as benign. This work has the purpose to study the main attacks against malware detectors and the defense techniques used to combat them. For this, we will create an attack scenario and a defense scenario that will help us understand the impact of attacks on the ML model. With the results, we hope to contribute to the construction of models that are more robust to adversarial machine learning.*

Resumo. *Os detectores de malware são a principal linha de defesa contra a crescente ameaça dos programas maliciosos. Os programas maliciosos, conhecidos como malwares (Malicious Software), possuem o objetivo de roubar informações confidenciais, explorar brechas de segurança e trazer transtornos a usuários. Para auxiliar na detecção desses programas, é utilizado algoritmos de aprendizado de máquina (Machine Learning - ML). O ML, com a combinação de técnicas de análise e extração de dados, possui a capacidade de detectar padrões atribuídos ao comportamento dos malwares. Entretanto, esses algoritmos são suscetíveis a ataques por meio do aprendizado de máquina adversário. Esses ataques possuem a capacidade de evadir ou manipular as classificações dos modelos de ML, possibilitando que programas maliciosos sejam classificados como benignos. Esse trabalho tem como objetivo estudar os principais ataques contra detectores de malware e as técnicas de defesas empregadas para combatê-los. Para isso, criaremos um cenário de ataque e outro de defesa que irão nos ajudar a entender o impacto dos ataques aos modelo de ML. Com os resultados obtidos esperamos contribuir com a construção de modelos mais robustos ao aprendizado de máquina adversário.*

1. Introdução

Os computadores como smartphones, computadores pessoais, dispositivos inteligentes, servidores e muitos outros são importantes ferramentas de comunicação e de operação de serviços, e possuem um papel fundamental em diversas áreas da sociedade. Diante

dessa importante característica, esses tipos de dispositivos se tornam alvos de diversos ataques cibernéticos que buscam tirar vantagens desses sistemas, aproveitando de falhas de seguranças e da utilização de programas maliciosos, chamados de malwares (*malicious software*), que podem atuar como atores primários nesses ataques. São programas que possuem como objetivo trazer transtornos a usuários, explorar falhas e promover ataques massivos, como no caso das botnets. Esses programas escondem sua verdadeira intenção através de diversas técnicas, enganando usuários e até sistemas de segurança [3] [5] [18].

Para combater a propagação desses programas existem os detectores de malware. Os detectores utilizam algumas técnicas que auxiliam na tentativa de determinar se um programa é malicioso ou não. Uma vez que os malwares conseguem se passar por programas benignos, essa tarefa se torna muito difícil, provocando, muitas das vezes, falsos positivos [4] [3] [5]. Como um reforço aos detectores, no decorrer dos anos passaram a ser utilizadas técnicas de aprendizado de máquina (conhecidos como *Machine Learning-based*) que permitem a análise de padrões encontrados nos mais diversos malwares. Com as análises estática e dinâmica dos programas, e a combinação de técnicas de extração de dados, é possível construir diferentes bancos de dados que podem fornecer informações sobre os programas e suas características. A partir disso, os modelos de aprendizado se tornam capazes de auxiliar na classificação de determinado programa baseado nos padrões que ele apresenta.

Entretanto, ao mesmo tempo em que o ML é um reforço aos detectores, ele também é suscetível a ataques por meio de amostras adversárias. As amostras adversárias são bastante conhecidas no universo do ML, permitindo que criadores de malware explorem esta brecha a fim de invalidar sistemas que utilizam ML. A função dos ataques é subverter e enganar os modelos de ML realizando alterações mínimas em atributos dos programas de forma que um malware seja classificado como benigno [13], mantendo suas propriedades intactas e permitindo que o programa ainda seja nocivo ao sistema alvo [10] [8].

A detecção de malware é um desafio visto que novos malwares ainda podem subverter modelo robustos como os baseados em aprendizado de máquina. Diante disso, esse trabalho possui como foco explorar e analisar os detectores de malware baseados em ML elaborando um estudo que cria cenários de ataques e combinam métodos de defesas propostos na literatura. Ao final, a ideia é sumarizar os principais ataques e mecanismos de defesas apresentando métricas quantitativas para avaliar o impacto destes ataques e a eficácia da defesa.

Ao decorrer do documento teremos as seguintes seções que irão descrever o estudo proposto: A seção 2 apresenta alguns conceitos necessários para o desenvolvimento do trabalho; a seção 3 traz de forma direta os objetivos esperados ao final do estudo; a seção 4 traz os métodos que serão utilizados para o desenvolvimento; a seção 5 lista as atividades planejadas juntamente com os cronogramas; a seção 6 descreve os resultados esperados ao finalizar o trabalho.

2. Fundamentação Teórico-Metodológica e Estado da Arte

2.1. Detectores de Malware

Mediante a ameaça dos malwares, os detectores possuem um papel fundamental na linha de defesa contra a propagação e na identificação desse tipo de software. Entretanto, os

detectores não são capazes de detectar todos os tipos de malwares. Malwares são alterados facilmente e apresentam características muito semelhantes a programas benignos. Esse problema torna o processo de decidir se determinado programa é maligno ou não mais complexo [3]. Em suma, os detectores de malware podem ser definidos como sistemas que tentam identificar se determinado programa é um malware ou não [5].

Dadas essas circunstâncias, os detectores de malware se dividem em diversas técnicas que ajudam a analisar os programas. Para entender o funcionamento desses detectores, podemos generalizar o processo de detecção em 3 (três) etapas:

1. *Análise de arquivos*: a análise do malware pode ser feita de maneira estática, dinâmica ou uma combinação de ambas. A análise estática analisa informações dos programas sem a necessidade de executá-los, coletando informações como sequência de bytes, chamadas a bibliotecas e códigos de operação (*opcode*). A análise dinâmica é feita a partir da execução do código em um ambiente seguro. Dessa forma, é possível coletar informações sobre o comportamento do programa enquanto ele está em execução [3] [6].

2. *Extração de atributos*: o processo de extração de atributos utiliza técnicas de mineração de dados para obter informações atreladas a diversos atributos capturados durante as análises [3] [18]. É nessa etapa em que é realizada a seleção de atributos. A seleção tem por objetivo remover informações redundantes e irrelevantes, reduzindo a quantidade de atributos. No final desse processo é esperado um conjunto de dados mais significativo que represente informações que possam ser melhor aproveitadas na classificação [18].

3. *Detecção*: após a coleta e filtragem dos dados, é realizada a detecção do programa analisado. Nessa etapa várias técnicas são utilizadas. A escolha da técnica tem influência direta nos tipos e famílias de malwares que serão detectados [18] [3] [5].

Os malwares também podem ser classificados em vários tipos e famílias, considerando as diferentes estratégias e técnicas. Algumas dessas famílias são os vírus, *worms*, *ramsonwares*, *botnets*, *spywares* e muitos outros. Cada família é conhecida por ações específicas e podem, inclusive, atuar em conjunto. Para dificultar a sua detecção, os malwares utilizam de diversas técnicas de ofuscação. Essas técnicas podem tornar os malwares ainda mais complexos, aumentando a probabilidade de atuarem livremente nos sistemas.

Alguma das técnicas utilizadas pelos criadores de malware podem ser observadas a seguir:

- *Criptografia*: uso da criptografia para esconder partes importantes do código;
- *Polimórfica*: o método polimórfico permite que o malware possa assumir múltiplas formas. Cópias do mesmo malware nunca são iguais. Uma forma de obter esse comportamento é através de uma criptografia variável. A cada nova cópia, um novo algoritmo de criptografia é gerado, fazendo com que o malware sempre assuma formas distintas.
- *Código morto*: adiciona códigos que não possuem utilidade para a execução do malware, como, por exemplo, funções que nunca são executadas;

- *Furtivo*: recorre a diferentes técnicas para se proteger contra as análises. De forma simples, faz alterações no sistema para fugir da detecção.

Os detectores também possuem diferentes abordagens para combater as técnicas de ofuscação dos malwares. Os dados e atributos minerados durante as fases de análise e extração diferem baseado no tipo de técnica utilizada. Entre as técnicas encontradas na literatura, temos:

- *Baseadas em Assinatura*: observa a sequência de bytes nos arquivos, instruções, *strings* e listas de bibliotecas. Através disso consegue produzir assinaturas únicas para programas. As assinaturas são posteriormente utilizados para detectar malwares já conhecidos. É uma técnica bastante utilizada por ser rápida, porém falha ao tentar detectar malwares ainda desconhecidos pelo sistema [3] [18].
- *Baseadas em Comportamento*: observa o comportamento dos malwares através das chamadas de API, chamadas de sistemas, comportamento da rede, etc.. Resolve parte do problema do método de assinatura, pois, ao analisar o comportamento do programa, pode identificar variações da mesma família de malwares que adicionam, por exemplo, código morto e alteram sua assinatura. As desvantagens desse método estão ligadas ao tempo de processamento dos programas e à quantidade de programas benignos que são classificados como malwares [4] [3].
- *Baseadas em Heurística*: utiliza uma combinação de técnicas, entre elas o uso de algoritmos de ML, além da aplicação de mineração de dados na extração de dados. Analisa chamadas de API e chamadas de sistemas que não são comumente utilizadas. A partir disso, se baseia em uma série de regras que definem limites ao comportamento dos programas. Quando um limite é extrapolado, aquele programa pode apresentar características de um malware. É um modelo mais robusto e proativo na detecção, sendo bastante aplicado no mercado devido à sua capacidade de lidar com mutações de malwares da mesma família e detectar novos malwares que seguem comportamentos semelhantes. Entretanto, apresenta alta taxa de falsos positivos e falha ao detectar malwares que são bastante complexos, além da criação das regras consumirem bastante tempo [3] [18].

Além dessas técnicas existentes, temos outras como *mobile-based* e *cloud-based*, que cresceram recentemente devido à utilização de aparelhos mobile e a fácil escalabilidade dos ambientes em nuvem. A aplicação do ML para a classificação também se tornou uma ferramenta bastante efetiva atualmente auxiliando no processo de detecção. O sucesso dos detectores baseados em ML dependem fortemente das fases de análise e extração de atributos já que os algoritmos de ML são orientados a dados [12]. A escolha do algoritmo utilizado pelo modelo de ML também influencia no desempenho dos detectores, visto que o desempenho de diferentes algoritmos muda conforme a distribuição dos dados e a quantidade de atributos nos conjuntos de dados analisados [18]. A Figura 1 mostram um esquema simples de um detector baseado em ML. As fases de análise e extração de dados irão fornecer os dados necessários, já rotulados, para o treinamento e teste do modelo. Em seguida, é esperado que o modelo seja capaz de generalizar o que foi aprendido para dados desconhecidos.

2.2. Aprendizado de Máquina (Machine Learning - ML)

O aprendizado de máquina estuda algoritmos que melhoram com experiência [12]. São excelentes ferramentas de regressão e classificação de dados, visto que se ajustam ao

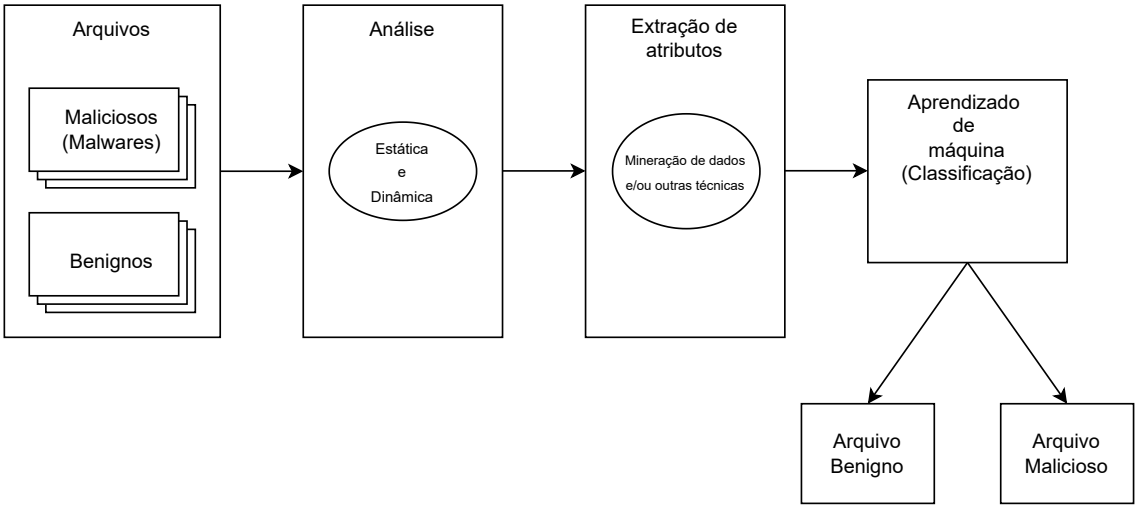

Figura 1. Exemplo de um detector baseado em Machine Learning.

aprendizado obtido sobre o conjunto de dados e não necessitam serem explicitamente programados para isso [16] [19]. Como ferramenta de regressão, os algoritmos tentam prever um valor numérico, como, por exemplo, salário de um funcionário ou a idade de uma pessoa. Como de classificação, os valores são categóricos, por exemplo, dizer se o comportamento de um programa é malicioso ou não. Os algoritmos de ML podem ser divididos em duas categorias: aprendizado supervisionado e aprendizado não supervisionado.

No aprendizado supervisionado, podemos dividir o processo em duas etapas: aprendizado e testes. No aprendizado, o algoritmo de ML recebe um conjunto de dados rotulado como entrada e realiza operações estatísticas sobre eles. Ao final desse processo, é criado um modelo que tenta predizer o rótulo pertencente a determinada entrada que compartilha similaridade com os dados utilizados nas operações. O objetivo é que os rótulos previsto pelo modelo se aproximem dos rótulos reais [19]. Após o aprendizado, é possível realizar testes sobre dados que não foram utilizados na fase de aprendizado. Nessa fase são coletadas métricas sobre o desempenho do modelo durante a classificação.

O aprendizado não supervisionado é aplicado em situações onde é custoso rotular os dados. Por exemplo, em um fluxo de rede onde existe um grande tráfego de dados. Nesse caso, os dados que apresentam características subjacentes semelhantes são colocados em grupos. Esses grupos são chamados de *clusters*. Esse processo conhecido como clusterização fornece informações sobre a organização e comportamento dos dados [19].

2.2.1. Redes Neurais

As redes neurais são um subconjunto do aprendizado de máquina. São utilizadas para o processo de aprendizagem. Possuem esse nome por conta da sua estrutura ser baseada no funcionamento do cérebro humano, com neurônios que se conectam e conversam entre si por meio das sinapses [2]. Uma rede neural pode ser organizada da seguinte maneira: uma camada de entrada, camadas ocultas e uma camada de saída. A Figura 2 exemplifica a organização da rede.

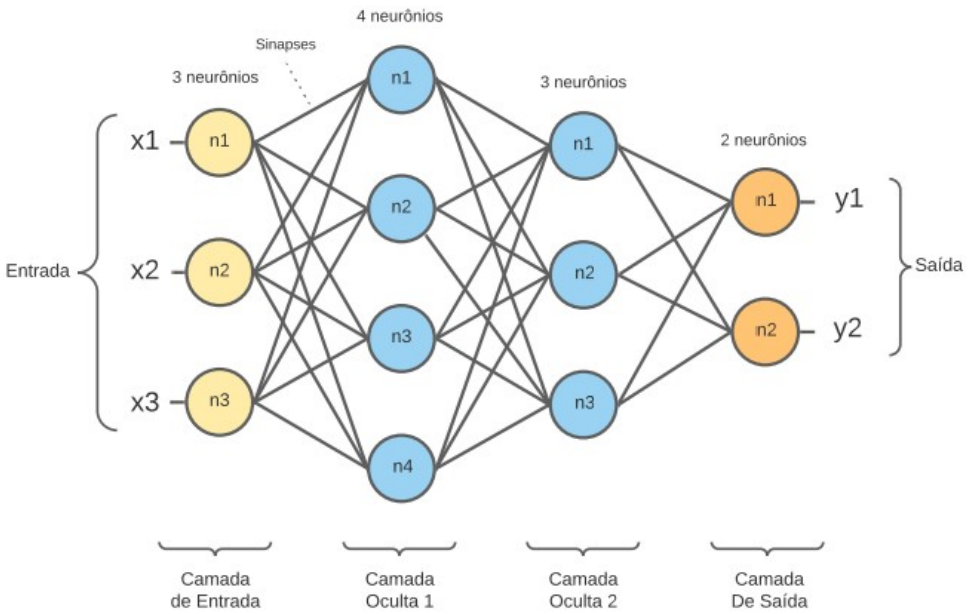

Figura 2. Rede neural

A camada de entrada é um conjunto de neurônios onde cada neurônio recebe um atributo do conjunto de dados. As saídas desses neurônios se conectam as estradas dos neurônios da primeira camada oculta. Na camada oculta é aplicada a função de ativação. A função de ativação esta presente dentro de cada neurônio e decidem quando ele deve ser ativado ou não [1]. Cada neurônio possui um peso e um viés (ou limite) atrelado a ele. O peso é multiplicado pelo valor de entrada. A função de ativação verifica se o valor obtido ultrapassa o limite estipulado e, caso ultrapasse, o neurônio é ativado. Quando ativado, o neurônio transmite informações para o próximo neurônio com quem ele se conecta. A camada oculta pode ter quantas camadas, e neurônios por camadas, forem desejados. Normalmente, quando existem mais de 3 camadas ocultas, as redes neurais são chamadas de *Deep Neural Network* (Rede Neural Profunda) [2]. Os neurônios da última camada oculta se conectam com os neurônios da camada de saída, que produz o resultado das operações. A quantidade de neurônios na saída é dada pela quantidade de classes no problema analisado. O processo de transmissão de informações de um neurônio para o outro na próxima camada é conhecido como *feedfoward*.

As redes neurais aprendem realizando ajustes nos pesos e vieses dos neurônios. Para isso, ao final da computação é realizado uma comparação entre os valores estipulados com o valores reais das saídas. A comparação é feita por uma função de custo que irá dizer o quão errado o modelo cria relações entre os valores de entrada e os valores de saída. O resultado da função de custo é repassado para os neurônios reajustarem seus pesos e vieses em um processo chamado de backpropagation. O objetivo é que o erro da relação entre a entrada e a saída seja otimizado [19].

2.3. Aprendizado de máquina adversário

O Aprendizado de máquina adversário (Adversarial machine learning - AML) é um conjunto de técnicas que possuem como objetivo enganar modelos de ML adicionando pe-

quenas manipulações em dados. Os dados adversários são construídos sobre modelos de ML e estimam uma pertubação mínima, mas efetiva, a ser adicionada aos atributos [7] [15]. A Figura 3 nos mostra um exemplo de pertubações sendo injetadas em uma imagem, as diferenças são imperceptíveis a um observador humano.

2.3.1. Tipos de ataques

Os ataques baseados em aprendizado de máquina adversário possuem diferentes abordagens e podem atuar com o conhecimento de informações do modelo-alvo ou não. Quando o atacante possui conhecimento dos parâmetros do modelo e do processo de treinamento, esse ataque é chamado *white box*. Caso contrário, é chamado de *black box*. Os ataques *black box* são os mais comuns em cenários reais, uma vez que os atacantes não possuem acesso às informações do modelo. Entretanto, segundo o conceito de *transferability*, atacantes podem aprender a manipular de maneira eficiente as entradas utilizando um algoritmo de ML diferente daquele utilizado pela vítima. Em outras palavras, o conhecimento obtido pelo atacante com um algoritmo pode ser transferido para o ataque a outros algoritmos. Dessa maneira, apesar da falta de informação, ataques *black box* também podem ter um grande impacto ao modelo-alvo [7] [17].

s ataques possuem finalidades diferentes. Eles podem buscar evadir classificações ou influenciar no desempenho dos modelos. Os ataques de evasão possuem o objetivo de desviar a classificação de determinada entrada. Esse desvio pode ser aleatório, qualquer classe menos a verdadeira, ou até mesmo tentar classificar por outras classes escolhidas previamente pelo atacante. Já os ataques que influenciam no desempenho, os dados alterados estão presentes durante a fase de aprendizado dos modelos. Esse tipo de ataque é conhecido como *poisoning* (envenenamento), onde o objetivo é corromper o modelo-alvo. Dessa forma, quando os testes são realizados, o modelo apresenta um desempenho ruim aos dados de teste. Os ataques de envenenamento podem ser indiscriminados se o objetivo é afetar qualquer classe. Ou de integridade, se o objetivo for afetar determinadas classes. Quando o ataque é de integridade, existe a possibilidade da criação de *backdoors* nos modelos-alvo. Um *backdoor* é basicamente uma falha de segurança deixada pelo atacante no sistema alvo o possibilitando de realizar ataques posteriormente. Na detecção de malware, foi demonstrado um ataque que leva a um *backdoor* [17]. Esse ataque permitiu que dados maliciosos pudessem ativar uma brecha no sistema, fazendo com que eles fossem classificados como benignos.

Quando focamos os ataques a detectores de malware, encontramos várias referências frisando a importância dos ataques não alterarem o funcionamento dos malwares. Com a inserção de pertubações nos atributos, os arquivos manipulados podem perder sua funcionalidade maligna, tornando o ataque ineficiente [9] [15] [11]. Existe outro problema também relacionado a área de aplicação. Esses ataques são comumente demonstrados na área de visão computacional [7]. Ao atacar modelos de classificação malwares, a organização dos dados difere de uma imagem. Os dados analisados são arquivos binários e possuem uma organização específica que difere entre sistemas. Dessa maneira, para que os ataques possam ser construídos, é necessário que seja feita uma análise sobre a organização dos dados [15] [10].

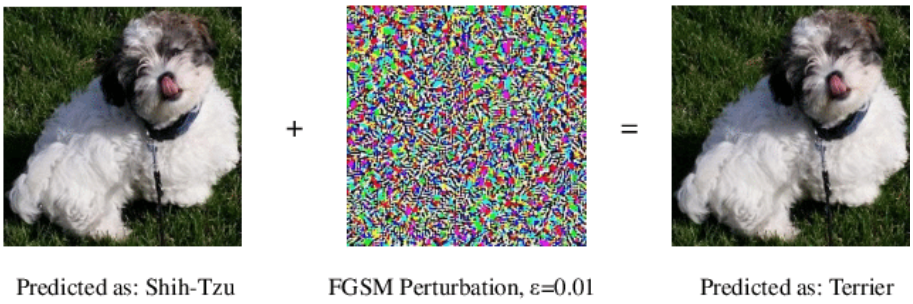

Figura 3. Ataque FGSM (baseado em gradiente) aplicado na visão computacional

2.3.2. Técnicas de defesa

As técnicas de defesa têm por objetivo aumentar a robustez dos classificadores, assim tornando-os mais resistentes aos ataques. As defesas encontradas na literatura são normalmente aplicadas na área de visão computacional, mas também podem ser transpostas aos detectores de malware como apresentado por [8]. As técnicas de defesa atuam principalmente na fase de treinamento dos classificadores como, por exemplo, o treinamento adversário, sugerido por [7] e o *defensive distillation* [14]. Estas técnicas de defesa tiveram impacto na eficiência dos ataques como apresentado por [11] [9].

O treinamento adversário determina que amostras adversárias sejam inseridas durante a fase de treinamento. Assim, os classificadores aprendem a classificar possíveis padrões relacionados a certos tipos de ataques. Entretanto, como apontado por [8] e [11], as amostras precisam ser cuidadosamente selecionadas, pois os algoritmos podem aprender a classificar determinado padrão de ataque e passar uma falsa sensação de proteção.

O *defensive distillation* utiliza um modelo a mais durante a fase de treinamento. Essa técnica, de forma simples, normaliza as saídas obtidas pelo primeiro modelo e utiliza esses valores como rótulos para os dados de entrada. Em seguida, esse conjunto é utilizado para treinar o próximo modelo que recebe o nome de modelo destilado [9]. Segundo [14] a experiência obtida não é apenas concentrada nos parâmetros de pesos da rede neural, mas também pelo vetor de probabilidade[1] produzido pelo modelo.

3. Objetivos

O objetivo desse trabalho é estudar os principais ataques contra os detectores de malware e as defesas que podem ser utilizadas para combatê-los. Sistemas baseados em aprendizado de máquina são vulneráveis a entradas adversárias, entretanto, existem técnicas de defesas que visam combater esse tipo de ataque diminuindo sua eficácia.

Ao final, a ideia é responder questões como:

- Quais mecanismos de defesas são mais eficazes?
- Quais tipos de ataques são mais severos?
- O volume de dados utilizado pelo atacante, interfere na eficácia do ataque?
- Existe algum tipo de malware que consegue melhor evadir a classificação?

[1] O vetor de probabilidade é criado por uma função de ativação na última camada de saída da rede neural. Essa função, conhecida como softmax, estipula a probabilidade da entrada pertencer as k classes do problema.

4. Procedimentos Metodológicos/Métodos e técnicas

Em um primeiro momento, será realizado um processo de análise bibliográfica levantando os principais ataques e defesas quando pensamos em aprendizado de máquina aplicado a detectores de malware. Durante a pesquisa, identificaremos quais os tipos de conjuntos de dados utilizados, os tipos de malwares que são encontrados nesses conjuntos, os algoritmos de ML utilizados pelo detectores, os tipos de ataques e técnicas de defesas. Após as pesquisas, será realizado a implementação dos algoritmos de ML escolhidos, das defesas e dos ataques.

Com a realização das implementações, iniciaremos o estudo criando dois cenários: um deles será o cenário de ataque, onde o atacante tentará através de ataques *black box* e *white box* evadir classificações. O outro cenário será o de defesa, onde, com os mesmos ataques do primeiro cenário, avaliaremos o quão eficiente foram os ataques quando elementos de defesa estão presentes. Com isso, teremos informações suficientes que contribuirão para responder às questões listadas na Seção 3.

É importante ressaltar que o processo de estudo e análise dos algoritmos será feito na fase de classificação dos detectores. Dessa forma, poderemos isolar os ambientes de ataque e defesa observando exclusivamente o efeito durante a classificação.

5. Cronograma de Execução

Atividades:

1. Levantamento bibliográfico;
2. Implementação dos Algoritmos de ML selecionados;
3. Implementação das defesas;
4. Implementação dos ataques;
5. Elaboração dos cenários de Ataque e defesa;
6. Organização e análise dos resultados obtidos;
7. Redação do TCC;
8. Revisão ortográfica;

Tabela 1. Cronograma de Execução

	set	out	nov	dez	jan	fev	mar	abr	mai
Atividade 1	x	x	x						
Atividade 2		x	x						
Atividade 3			x	x	x				
Atividade 4				x	x				
Atividade 5						x	x		
Atividade 6							x		
Atividade 7					x	x	x	x	
Atividade 8								x	x

6. Contribuições e/ou Resultados esperados

Com os resultados obtidos nesse trabalho, esperamos contribuir na segurança de futuras implementações de detectores de malware baseados em aprendizado de máquina. Com a sumarização dos principais ataques e defesas, e respondendo as questões abordadas na Seção 3, poderemos contribuir para ambientes mais robustos a ataques adversários.

7. Espaço para assinaturas

Londrina, *12 de setembro de 2022*.

Aluno

Orientador

Referências

[1] Deep learning book. https://www.deeplearningbook.com.br/funcao-de-ativacao/#:~:text=As%20fun%C3%A7%C3%B5es%20de%20ativa%C3%A7%C3%A3o%20s%C3%A3o,fornecida%20ou%20deve%20ser%20ignorada. Acessado: 2022-09-07.

[2] Redes neurais. https://www.ibm.com/br-pt/cloud/learn/neural-networks. Acessado: 2022-08-15.

[3] Ömer Aslan Aslan and Refik Samet. A comprehensive review on malware detection approaches. *IEEE Access*, 8:6249–6271, 2020.

[4] Zahra Bazrafshan, Hashem Hashemi, Seyed Mehdi Hazrati Fard, and Ali Hamzeh. A survey on heuristic malware detection techniques. In *The 5th Conference on Information and Knowledge Technology*, pages 113–120. IEEE, 2013.

[5] Mihai Christodorescu, Somesh Jha, Sanjit A Seshia, Dawn Song, and Randal E Bryant. Semantics-aware malware detection. In *2005 IEEE symposium on security and privacy (S&P'05)*, pages 32–46. IEEE, 2005.

[6] Ekta Gandotra, Divya Bansal, and Sanjeev Sofat. Malware analysis and classification: A survey. *Journal of Information Security*, 2014, 2014.

[7] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[8] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. Adversarial perturbations against deep neural networks for malware classification. *arXiv preprint arXiv:1606.04435*, 2016.

[9] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. Adversarial examples for malware detection. In *European symposium on research in computer security*, pages 62–79. Springer, 2017.

[10] Bojan Kolosnjaji, Ambra Demontis, Battista Biggio, Davide Maiorca, Giorgio Giacinto, Claudia Eckert, and Fabio Roli. Adversarial malware binaries: Evading deep learning for malware detection in executables. In *2018 26th European signal processing conference (EUSIPCO)*, pages 533–537. IEEE, 2018.

[11] Raphael Labaca-Castro, Luis Muñoz-González, Feargus Pendlebury, Gabi Dreo Rodosek, Fabio Pierazzi, and Lorenzo Cavallaro. Realizable universal adversarial perturbations for malware. 2021.

[12] Tom M Mitchell and Tom M Mitchell. *Machine learning*, volume 1. McGraw-hill New York, 1997.

[13] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

[14] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE, 2016.

[15] Fabio Pierazzi, Feargus Pendlebury, Jacopo Cortellazzi, and Lorenzo Cavallaro. Intriguing properties of adversarial ml attacks in the problem space. In *2020 IEEE symposium on security and privacy (SP)*, pages 1332–1349. IEEE, 2020.

[16] Arthur L Samuel. Some studies in machine learning using the game of checkers. ii—recent progress. *IBM Journal of research and development*, 11(6):601–617, 1967.

[17] Limin Yang, Zhi Chen, Jacopo Cortellazzi, Feargus Pendlebury, Kevin Tu, Fabio Pierazzi, Lorenzo Cavallaro, and Gang Wang. Jigsaw puzzle: Selective backdoor attack to subvert malware classifiers. *arXiv preprint arXiv:2202.05470*, 2022.

[18] Yanfang Ye, Tao Li, Donald Adjeroh, and S Sitharama Iyengar. A survey on malware detection using data mining techniques. *ACM Computing Surveys (CSUR)*, 50(3):1–40, 2017.

[19] Zhi-Hua Zhou. *Machine learning*. Springer Nature, 2021.