Evasão em Modelos de Detecção de Ameaças de Rede Usando Propriedades do Espaço de Decisão

Rafael Dias Campos [1], **Michele Nogueira** [1,2], **Marcio Costa Santos** [1]

[1]Departamento de Ciência da Computação - Universidade Federal de Minas Gerais

[2]Departamento de Informática - Universidade Federal do Paraná

{rafaelcampos,michele,marciocs}@dcc.ufmg.br

Abstract. *Machine learning techniques are frequently employed in detecting network threats, such as denial-of-service attacks, XSS, and Ransomware. However, the trained models can themselves become targets for attacks aimed at evading detection when sending malicious network traffic in the environment and preventing alerts or defensive actions from taking place. Current methods for conducting such attacks present limitations during their training phases and are susceptible to generating traffic samples that differ significantly from the original ones, limiting their utility. To address these issues, this work presents a new method for evading traffic detection by mapping the decision space of the network threat detection models as a set of convex polytopes. The developed method was able to generate samples closer to the original ones when compared with the other methods and presented more stable results among multiple runs.*

Resumo. *Técnicas de Aprendizado de Máquina são frequentemente empregadas para detectar ameaças à nível de rede, como ataques de negação de serviço, XSS e Ransomware. Entretanto, os modelos treinados podem se tornar alvos de ataques adversariais, visando a evasão da detecção de tráfego de rede malicioso e a execução de ações no ambiente sem gerar alertas ou bloqueios. Os métodos atuais para realizar esses ataques apresentam limitações durante sua etapa de treinamento e podem gerar amostras de tráfego malicioso que diferem muito das amostras originais, limitando sua utilidade. Para lidar com esses problemas, este trabalho apresenta um novo método para a evasão dessa detecção, a partir do mapeamento do espaço de decisão dos modelos de detecção de ameaças de rede como um conjunto de politopos convexos. O método desenvolvido foi capaz de gerar amostras de tráfego malicioso mais próximas do tráfego original em comparação com os demais métodos testados e apresenta uma maior estabilidade de resultados entre diversas execuções.*

1. Introdução

Soluções de segurança em nível de rede, como *Firewall* e Sistemas de Detecção / Prevenção de Invasão (IDS / IPS), são amplamente utilizadas na identificação e bloqueio de ataques cibernéticos, como escaneamento de portas, ataques de negação de serviço, ataques de XSS e *Ransomware*. Para detectar essas ameaças, algumas soluções empregam técnicas de aprendizado de máquina [Fortinet 2025, Progress Software Corporation 2025, Enea 2025], utilizando modelos treinados para classificar o tráfego de rede com base em

diversas características de amostras obtidas, como porta de origem e destino, tamanho do pacote e frequência de envio.

Entretanto, modelos de aprendizado de máquina podem se tornar alvos de ataques adversariais, que consistem na adição de perturbações em algumas características das amostras, com o objetivo de ocasionar erros de classificação. Essa classe de ataques pode ser aplicada no contexto de sistemas de detecção de ameaças de rede, com o objetivo de enviar tráfego malicioso ao ambiente de forma evasiva, evitando a ocorrência de alertas e bloqueios. Nesse caso, pacotes de rede relacionados aos ataques são modificados, produzindo amostras adversariais com diferentes valores de *features* que são classificadas incorretamente como tráfego normal.

O trabalho em [Ibitoye et al. 2019] aplica o método *Fast Gradient Sign Method* (FGSM) para realizar ataques adversariais nesse contexto; entretanto, essa estratégia requer acesso aos pesos utilizados pelo modelo de classificação de tráfego de rede, tornando o ataque inviável em diversos cenários reais. Por outro lado, os trabalhos em [Lin et al. 2022] e [Shu et al. 2020] apresentam, respectivamente, os métodos *IDSGAN* e *Gen-AAL*, que podem ser aplicados em ataques *black-box*, pois utilizam apenas os resultados de classificação do modelo alvo do ataque para treinar redes neurais do tipo GAN (*Generative adversarial network*). Entretanto, eles apresentam algumas limitações, como o envio de um grande número de amostras de rede durante o processo de treinamento no caso do *IDSGAN*, ou não permitem escolher quais características do tráfego de rede devem permanecer inalteradas no caso do *Gen-AAL*. Além disso, ambos os métodos de ataque *black-box* podem adicionar perturbações significativas nas amostras originais, limitando a funcionalidade das amostras adversariais geradas.

Neste trabalho, propõe-se um método denominado *PolyMap* para realizar ataques de evasão em modelos de detecção de ameaças de rede, com base em propriedades estruturais de diversas estratégias de aprendizado de máquina. Diversas amostras são geradas e enviadas para classificação durante o processo de treinamento, para mapear o espaço de decisão do modelo alvo como um conjunto de politopos. Após essa etapa, as amostras de tráfego malicioso são modificadas de forma a se encontrarem dentro de um dos politopos encontrados, evitando sua detecção.

Depois de implementado, o método proposto foi comparado com os demais nos *datasets* Bot-IoT [Koroniotis et al. 2020], TON_IoT [Moustafa 2021, Booij et al. 2022], CTU-13 [García et al. 2014] e NSL-KDD [Tavallaee et al. 2009]. Os resultados obtidos demonstraram que o *PolyMap* apresenta uma taxa de sucesso de ataque comparável à dos métodos *IDSGAN* e *Gen-AAL* e, simultaneamente, adiciona perturbações menores nas amostras originais. Ele também oferece uma redução em relação ao *IDSGAN* no número de requisições enviadas para classificação durante o processo de treinamento e, diferentemente do *Gen-AAL*, permite escolher *features* para permanecer inalteradas.

2. Trabalhos Relacionados

Diversas soluções comerciais de *Firewall* e Sistemas de Detecção / Prevenção de Intrusão (IPS/IDS) atualmente utilizam técnicas de aprendizado de máquina, como redes neurais, para auxiliar no processo de identificação de ameaças de rede [Fortinet 2025, Progress Software Corporation 2025, Enea 2025]. Para essa tarefa, são utilizados modelos treinados para classificar o tráfego de rede com base em diversas características das

amostras obtidas, como porta de origem e destino, tamanho do pacote e frequência de envio. Entretanto, esses modelos podem se tornar alvos de ataques adversariais, com o objetivo de evadir a detecção de tráfego de rede malicioso e permitir a execução de ataques no ambiente sem gerar alertas ou bloqueios.

Para avaliar a suscetibilidade a ataques adversariais de diferentes tipos de Redes Neurais utilizadas para classificar tráfego de rede, os autores de [Ibitoye et al. 2019] desenvolveram e treinaram redes dos tipos *Feed Forward Neural Network* (FNN) e *Self-Normalizing Neural Network* (SNN). Em seguida, ataques foram realizados em ambos os modelos, utilizando FGSM, observando-se uma maior resiliência das redes SNN em comparação com as redes FNN. Apesar de bem-sucedidos, esses ataques foram executados em uma metodologia *white-box* e requerem visibilidade dos gradientes de cada modelo durante o processo de classificação, o que pode não ser possível em diversos cenários reais nos quais se tem conhecimento apenas do resultado binário da rede (tráfego malicioso ou não malicioso).

De uma perspectiva diferente, o trabalho apresentado em [Lin et al. 2022] demonstra uma técnica *black-box* para realizar ataques contra modelos de análise de tráfego de rede. Os autores desenvolveram uma rede neural do tipo GAN que recebe como entrada uma amostra de tráfego de rede malicioso e modifica algumas de suas características (a escolha do atacante) para evitar sua detecção. Essa rede GAN foi treinada apenas com o resultado da classificação de amostras pelo modelo alvo do ataque e foi capaz de obter uma taxa de sucesso superior a 99% em alguns cenários. Entretanto, a etapa de treinamento do modelo adversarial requer o envio de um grande volume de amostras, incluindo amostras maliciosas, o que pode resultar na detecção desse ataque e no bloqueio da conexão do atacante. Além disso, esse método não busca minimizar os valores das perturbações adicionadas às amostras, o que pode resultar em amostras geradas muito distintas das originais e que não representam mais o tipo de ataque de rede desejado.

Com o objetivo de reduzir a probabilidade de detecção de ataques de rede durante o processo de treinamento do modelo adversarial, os autores de [Shu et al. 2020] desenvolveram o método *Gen-AAL*, que utiliza um processo de treinamento ativo para limitar a quantidade de amostras enviadas ao modelo de classificação. Esse método inicialmente treina um modelo de rede neural para reproduzir o comportamento do modelo de classificação a partir de amostras de rede categorizadas. Em seguida, o modelo adversarial baseado em GAN é treinado para atacar essa rede, enquanto apenas um pequeno número de amostras é enviado ao modelo de classificação real para atualizar o comportamento da rede reproduzida. Esse método foi capaz de atingir uma taxa de sucesso de 98.86%, enviando apenas 25 amostras para classificação pelo modelo real; entretanto, ele não permite a escolha de algumas características das amostras originais para permanecerem inalteradas nas amostras adversariais. Além disso, apesar do método buscar minimizar a distância L2 entre as amostras originais e adversariais, essa distância foi maior do que a obtida por outros métodos, como o *DeepFool Adversarial Learning* (DFAL) [Ducoffe and Precioso 2018], o que pode reduzir a efetividade de um ataque.

3. Metodologia do Mecanismo de Evasão

O problema tratado neste trabalho e a solução proposta são descritos nesta seção. Primeiramente, é formulada uma definição matemática dos objetivos do mecanismo de evasão

desenvolvido. Em seguida, são detalhados os processos de treinamento e evasão do mecanismo proposto.

3.1. Definição do problema e objetivos de otimização

Neste trabalho, foram considerados métodos de evasão em modelos de detecção de ameaças de rede que recebem como entrada uma amostra x de tráfego de rede malicioso, adicionam pequenas perturbações em suas características e retornam uma amostra adversarial $\tilde{x}$, que deve ser classificada incorretamente como tráfego normal. Essa amostra representa um pacote de tráfego de rede e possui características intrínsecas do pacote (como tamanho, portas de origem e destino e *flags* de conexão) e dados estatísticos (como a quantidade total de *bytes* enviados para o IP de destino e sua frequência). Para simular mais precisamente cenários reais, foi utilizada uma abordagem *black-box* em relação ao modelo de detecção de ameaças de rede, de forma que sua estrutura interna e seus pesos são desconhecidos, enquanto é possível deduzir apenas o resultado da classificação de uma amostra (normal / maliciosa), ao observar se ela foi bloqueada ou não. Com base nisso, foram definidos os principais objetivos de otimização dos métodos de ataques adversariais em redes, considerando as etapas de treinamento e evasão.

Durante a fase de treinamento do modelo adversarial, amostras maliciosas produzidas são enviadas para o modelo de detecção de ameaças para classificação. Esse comportamento pode ser detectado como uma ameaça, expondo o atacante e possivelmente resultando em alertas e no bloqueio de seu acesso antes de realizar o ataque desejado. Dessa forma, podem ser considerados como objetivos dessa classe de métodos de ataques adversariais a redução do número de amostras de tráfego enviadas para classificação (N_{req}) e do número de amostras classificadas como tráfego malicioso (N_{mal}).

Durante a realização de ataques na rede, o algoritmo de evasão é aplicado a cada amostra de tráfego relacionada ao ataque, e as amostras adversariais são enviadas na rede. Nesse ponto, seu objetivo principal é minimizar a probabilidade de que a amostra $\tilde{x}$ seja classificada como tráfego malicioso, possibilitando a execução furtiva do ataque. Entretanto, grandes perturbações na amostra original x podem resultar em amostras $\tilde{x}$ que não desempenham a função desejada; portanto, também é desejável minimizar as diferenças entre x e $\tilde{x}$, calculadas pela distância L2 entre os valores de suas características. A Equação 1 descreve os objetivos de otimização do algoritmo de evasão descritos nesta seção.

$$\begin{aligned} \min_{\tilde{x}} \quad & (\mathbb{P}(\tilde{x} \text{ é classificado como tráfego malicioso}), \\ & \|x - \tilde{x}\|_2, N_{\text{req}}, N_{\text{mal}}) \end{aligned} \tag{1}$$

3.2. Descrição do método

É conhecido que as redes neurais baseadas na função de ativação ReLU (*Rectified linear unit*) dividem o espaço de entrada em múltiplos politopos convexos [Sattelberg et al. 2020] com uma complexidade limitada, mesmo em redes profundas [Fan et al. 2024]. Esse comportamento de divisão do espaço não é limitado a redes neurais, uma vez que outras classes de métodos de aprendizado de máquina apresentam comportamentos semelhantes [Gajer and Ravel 2025]. Diante dessas observações, o

método desenvolvido neste trabalho busca explorar tal propriedade geométrica no contexto de modelos de classificação de tráfego de rede para mapear seu espaço de decisão e evadir a detecção de amostras maliciosas.

Para isso, o método desenvolvido consiste nas fases de treinamento e execução. Durante a primeira fase, são enviadas amostras de tráfego de rede para o modelo de detecção de ameaças com o objetivo de mapear seu espaço de decisão e representá-lo por um conjunto de politopos. Após essa etapa, amostras maliciosas correspondentes a ataques de rede são modificadas para que se encontrem dentro de um dos politopos mapeados e enviadas na rede. As subseções seguintes descrevem em detalhes o funcionamento das etapas do método.

3.3. Mapeamento do espaço de decisão

A primeira etapa do método consiste em encontrar um conjunto de politopos convexos que representa o espaço de decisão utilizado pelo modelo de detecção de ameaças de rede para classificar o tráfego normal. Durante essa etapa, amostras de tráfego normal são utilizadas como pontos iniciais no mapeamento dos politopos. Inicialmente, é aplicado um algoritmo de *Min-Max Scaler* em cada *feature* numérica dessas amostras para limitar seus valores entre 0 e 1, com o objetivo de reduzir o espaço de busca.

Em seguida, as amostras são divididas em grupos com base nos valores de suas *features* categóricas, uma vez que estas podem assumir apenas valores específicos. Para cada grupo, aplica-se um algoritmo de clusterização em suas características numéricas, com o objetivo de eliminar amostras muito parecidas que provavelmente pertencem ao mesmo politopo. A partir do ponto central de cada um dos *clusters* encontrados, é mapeado um politopo $\mathcal{P}$ que representa parte do espaço de decisão de tráfego normal do modelo de detecção de ameaças. Por fim, todos os politopos encontrados são retornados para serem utilizados durante o processo de evasão. Esse procedimento é descrito pelo Algoritmo 1.

Algoritmo 1 Treinamento do modelo de evasão

Entrada: Conjunto de amostras de tráfego normal A
Saída: Conjunto de politopos P

```
Aplicar normalização Min–Max ao conjunto A
Inicializar o conjunto de politopos P ← ∅
para todo combinações possíveis de features categóricas em A faça
    B ← subconjunto de A que compartilha a mesma combinação de features categóricas
    Identificar clusters no subespaço das features numéricas de B
    para todo clusters identificados faça
        Construir politopos a partir do centróide do cluster (Algoritmo 2)
        Adicionar os politopos gerados ao conjunto P
    fim para
fim para
retorne P
```

O Algoritmo 2 é utilizado para realizar o mapeamento de um politopo a partir de uma amostra inicial. Para isso, são gerados diversos raios espaçados igualmente pelo

espaço, tendo como origem essa amostra. Em seguida, a amostra inicial é modificada ao longo da direção de cada raio, até que seja classificada como tráfego malicioso ou até que uma de suas *features* atinja os limites do espaço de busca $[0, 1]$, definidos pelo *Min-Max Scaler*. As últimas amostras modificadas que ainda foram classificadas como tráfego normal são utilizadas como as bordas durante a construção de um politopo, o qual é retornado para uso no processo de evasão.

Algoritmo 2 Mapeamento de Politopos

Entrada: Ponto inicial x_0, número de raios n_r, tamanho do passo Δ
Saída: Politopo mapeado P

Inicializar conjunto de pontos de fronteira $S \leftarrow \emptyset$
Gerar n_r direções radiais uniformemente distribuídas no espaço
para todo direções radiais geradas **faça**
    Inicializar deslocamento $\delta \leftarrow \Delta$
    enquanto verdadeiro faça
        Calcular ponto candidato $x \leftarrow x_0 + \delta \cdot \mathbf{r}$
        se $x \in [0, 1]^d \ \wedge$ VERIFICAAMOSTRA$(x) =$ TRÁFEGONORMAL **então**
            Atualizar deslocamento $\delta \leftarrow \delta + \Delta$
        senão
            Determinar último ponto válido $x \leftarrow x_0 + (\delta - \Delta) \cdot \mathbf{r}$
            Atualizar conjunto de fronteira $S \leftarrow S \cup \{x\}$
            interromper
        fim se
    fim enquanto
fim para
Construir politopo P a partir do conjunto de pontos S
retorne P

3.4. Uso do algoritmo para evasão

Após o mapeamento do espaço de decisão do modelo de detecção de ameaças de rede, os politopos encontrados podem ser utilizados para modificar uma amostra de tráfego malicioso de forma a evitar sua detecção. Primeiramente, o mesmo algoritmo de *Min-Max Scaler* utilizado durante o treinamento é aplicado às *features* numéricas da amostra maliciosa. Em seguida, são calculados os pontos dentro de cada politopo $\mathcal{P}$ encontrado durante o mapeamento que estão mais próximos da amostra, considerando apenas suas características numéricas. Esse processo é realizado pela resolução de um problema de mínimos quadrados descrito pela Equação 2, com possíveis restrições adicionais para limitar as perturbações à apenas algumas *features* consideradas não essenciais para o ataque. Os pontos encontrados são reconstruídos em amostras de tráfego ao restaurar os valores das *features* categóricas utilizadas durante a busca do politopo ao qual pertencem. Em sequência, é escolhida a amostra gerada que apresenta a menor distância $L2$ em relação à original e é aplicada a operação inversa do *Min-Max Scaler*. Esse processo é descrito pelo Algoritmo 3.

$$\begin{aligned} \tilde{\mathbf{x}} = \arg\min_{\tilde{\mathbf{x}} \in \mathbb{R}^n} \ & \|\tilde{\mathbf{x}} - \mathbf{x}\|_2 \\ \text{sujeito a } \ & 0 \leq \tilde{\mathbf{x}} \leq 1, \tilde{\mathbf{x}} \in \mathcal{P}. \end{aligned} \tag{2}$$

Algoritmo 3 Modificação de Amostras de Tráfego

Entrada: Amostra de tráfego malicioso a, conjunto de politopos P
Saída: Amostra de tráfego modificada m

Aplicar normalização *Min–Max* à amostra a
Inicializar conjunto de candidatos $C \leftarrow \emptyset$
para todo politopos $p \in P$ **faça**
    Remover as *features* categóricas associadas da amostra a
    Calcular $t \leftarrow$ projeção de a no politopo p via mínimos quadrados (Equação 2)
    Restaurar em t as *features* categóricas correspondentes ao politopo p
    Atualizar conjunto de candidatos $C \leftarrow C \cup \{c\}$
fim para
Selecionar $m \leftarrow \arg\min_{g \in C} \|a - g\|_2$
Aplicar a normalização inversa *Min–Max* à amostra m
retorne m

Para reduzir o tempo de execução desse algoritmo, é possível calcular a projeção de uma amostra apenas nos politopos que foram construídos a partir do mesmo conjunto de *features* categóricas. Esse processo reduz drasticamente a quantidade de problemas de otimização a serem resolvidos e torna o método mais viável para aplicação em ataques que exigem o envio de um grande número de requisições na rede, como ataques de negação de serviço. Caso não existam politopos mapeados que satisfaçam essa restrição, a busca é feita em todos os politopos, conforme descrito no Algoritmo 3.

4. Avaliação

Com o objetivo de avaliar e comparar os métodos de evasão, foram realizados testes experimentais em diferentes modelos de classificação de ameaças e conjuntos de dados. Nesta seção, descreve-se a preparação e implementação dos modelos de classificação de tráfego de rede que foram utilizados para a avaliação dos ataques de evasão. Em seguida, é detalhado o processo de avaliação dos métodos de evasão nesses modelos.

4.1. Preparação de modelos de detecção de ameaças

Para servir como alvos dos ataques na avaliação dos algoritmos de evasão, foram desenvolvidos e treinados modelos de detecção de ameaças. O trabalho apresentado em [Ibitoye et al. 2019] demonstra a efetividade de redes neurais do tipo *Feedforward Neural Network* (FNN) e *Spiking Neural Network* (SNN) para a tarefa de classificação de tráfego de rede e uma maior resiliência das redes SNN contra ataques do tipo *Fast Gradient Sign Method* (FGSM). Dessa forma, esses tipos de rede foram escolhidos para avaliação e comparação dos métodos de evasão.

Seguindo o trabalho de referência, foi implementada a rede do tipo FNN com a função de ativação *Rectified Linear Unit* (ReLU), *dropout* convencional e função de inicialização *Xavier* uniforme. Em seguida, foi implementada a rede SNN com a função de ativação *Scaled Exponential Linear Unit* (SELU), *Alpha dropout* e função de inicialização *Lecun* uniforme. Os seguintes hiperparâmetros foram escolhidos para os modelos, com base em resultados obtidos com a ferramenta Optuna [Akiba et al. 2019] após execuções no dataset TON_IoT:

- Quantidade de camadas escondidas: 3
- Quantidade de neurônios por camada escondida: 32
- Tamanho do *batch*: 2048
- Taxa de aprendizado: 0.004 (FNN), 0.008 (SNN)
- Probabilidade de *dropout*: 0.3 (FNN), 0.1 (SNN)
- Número de épocas: 1000

Os conjuntos de dados Bot-IoT [Koroniotis et al. 2020] (amostra de 5% disponibilizada pelos autores), TON_IoT [Moustafa 2021, Booij et al. 2022] (amostra de 2% disponibilizada pelos autores), CTU-13 [García et al. 2014] (cenários 51 e 52) e NSL-KDD [Tavallaee et al. 2009] (*dataset* completo) foram utilizados para treinar e avaliar os modelos das redes implementadas, devido aos seus diferentes níveis de dificuldade de classificação. Para cada *dataset*, foram escolhidas colunas que contém informações da conexão (como tamanho do pacote enviado e portas de origem e destino) e dados estatísticos (como a quantidade de pacotes enviados por ou para um endereço de IP).Sempre que possível, foram escolhidas *features* similares entre os *datasets*.

Em cada conjunto de dados, as *features* numéricas foram dimensionadas com base em suas médias e desvios padrão (*Standard Scaler*), e foi aplicado às características categóricas um método de *label encoder*. As amostras de cada conjunto foram particionadas em dados de treino, teste e validação, seguindo uma proporção de $70/20/10\%$. Em seguida, para cada *dataset*, foi criado um modelo FNN e um modelo SNN, utilizando os dados de treino e um *weighted sampler* para minimizar os efeitos do desbalanceamento de classes nos dados. Após essa etapa, os dados de teste foram utilizados para avaliar a capacidade do modelo treinado em classificar novos dados de tráfego de rede. Esse processo de divisão do *dataset* e treinamento de modelos foi repetido 8 vezes para cada um dos conjuntos de dados utilizados, resultando em um total de 32 redes FNN e 32 redes SNN. Todos os modelos treinados obtiveram resultados de acurácia e valor F1 acima de 90% para o conjunto TON_IoT, e acima de 96% para os demais conjuntos de dados, conforme representado na Figura 1.

4.2. Evasão da Detecção

Após a preparação dos modelos de detecção de ameaças, foram implementados os métodos de evasão para comparação. Os métodos *IDSGAN* [Lin et al. 2022] e *Gen-AAL* [Shu et al. 2020] foram implementados seguindo seus artigos de referência. A estratégia proposta por este artigo foi implementada conforme a descrição na Seção 3, utilizando os parâmetros número de raios $n_r = 50$ e tamanho do passo $\Delta = 0.01$.

Em seguida, foi realizada a execução e a avaliação de cada técnica de evasão nos modelos de detecção, utilizando uma fração de seus dados para treinamento e avaliação, dimensionadas conforme a quantidade de dados disponíveis em cada *dataset* (20% para TON_IoT e NSL-KDD, 5% para CTU-13 e 3% para Bot-IoT), utilizando uma semente diferente para cada execução. No caso dos métodos *IDSGAN* e *Gen-AAL*, foram utilizadas amostras de tráfego normal e malicioso durante o treinamento, enquanto o método proposto neste artigo utiliza apenas amostras de tráfego normal para mapear o espaço de decisão. Para cada modelo de detecção de ameaças, foram utilizados 60% dos dados separados para treinar um modelo com o método *PolyMap* uma vez e quatro modelos com cada um dos métodos *IDSGAN* e *Gen-AAL*. Durante esse processo, para cada modelo, fo-

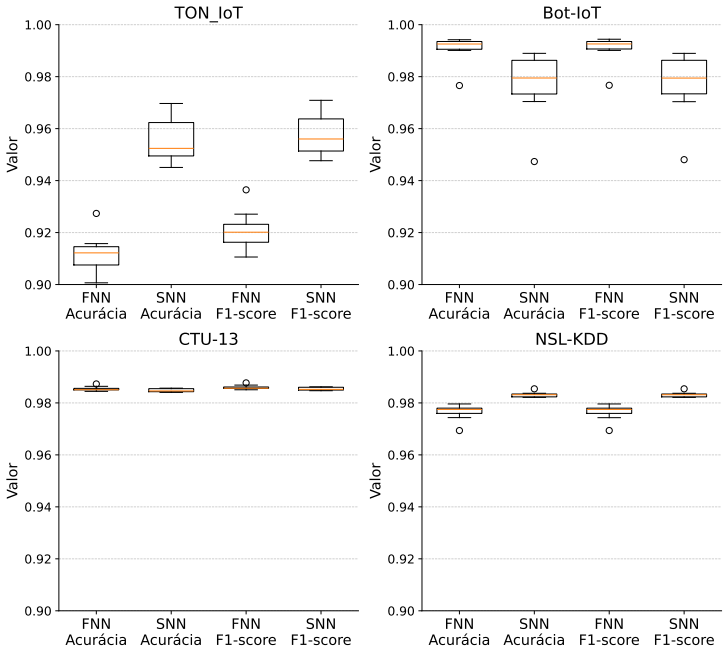

Figura 1. Acurácia e *F1-Score* obtidos durante a classificação de cada *dataset*

ram coletadas as quantidades de requisições realizadas e quantas delas foram classificadas como maliciosas.

Depois de finalizada a etapa de treinamento, os restantes 40% dos dados foram utilizados para avaliar seu desempenho durante a evasão de ataques. Nesse processo, foram utilizadas como base as amostras inicialmente classificadas como maliciosas pelo modelo de detecção para calcular a taxa de sucesso, avaliando a fração das amostras adversariais geradas por cada modelo de evasão classificadas como tráfego normal, conforme a Equação 3. Para cada amostra adversarial, também foi calculada sua distância L2 em relação à amostra original, para comparação da magnitude das perturbações adicionadas por cada modelo de evasão.

$$\text{Taxa de sucesso de evasão} = 1 - \frac{\text{quantidade de } \tilde{x} \text{ classificado como malicioso}}{\text{quantidade de } x \text{ classificado como malicioso}} \tag{3}$$

Durante a etapa de treinamento do *PolyMap*, o espaço de entrada utilizado durante o mapeamento de politopos é representado por um hipercubo com um número de dimensões $n_{features}$ e tamanho 1 (devido ao uso do *Min-Max Scaler*). Dessa forma, cada raio utilizado para o mapeamento do politopo pode percorrer uma distância máxima de $\sqrt{n_{features}}$, correspondente ao diâmetro do hipercubo. Portanto, seu custo computacional máximo durante o treinamento é de $\frac{n_r\sqrt{n_{features}}}{\Delta} \cdot n_{clusters}$ iterações. As operações executadas no dispositivo do atacante em cada iteração são simples do ponto de vista computacional, mas o processo de enviar uma amostra para a rede e observar sua classificação (comportamento compartilhado por todos os métodos) é significativamente mais custoso. Dessa forma, o custo do treinamento de um método pode ser aproximado pela quantidade de requisições enviadas na rede. Os custos computacionais durante o processo de evasão não foram comparados devido às diferenças significativas na arquitetura, funcionamento

e utilização de recursos de *hardware* de cada método de evasão.

5. Resultados

Após a execução dos ataques de evasão nos conjuntos de dados, observou-se que as taxas de sucesso de evasão obtidas nas melhores execuções do método *PolyMap* foram comparáveis às obtidas nas melhores execuções dos métodos *IDSGAN* e *Gen-AAL*, conforme demonstrado nas Figuras 2, 3, 4 e 5, que se referem aos *datasets* NSL-KDD, TON_IoT, Bot-IoT e CTU-13, respectivamente. Ao comparar os resultados obtidos nas três melhores execuções de cada tipo de ataque para cada *dataset*, resumidos na Tabela 1, também foi possível observar que o *PolyMap* obteve uma menor distância média entre as amostras originais e adversariais, o que indica que ele foi capaz de alterar o resultado de classificação dos modelos alvo, adicionando menores perturbações. Também foi observado que, ao longo de diversas execuções, o método proposto neste trabalho obteve resultados mais estáveis em termos da taxa de sucesso de evasão e da distância média do que os demais métodos analisados, conforme demonstrado na Figura 6.

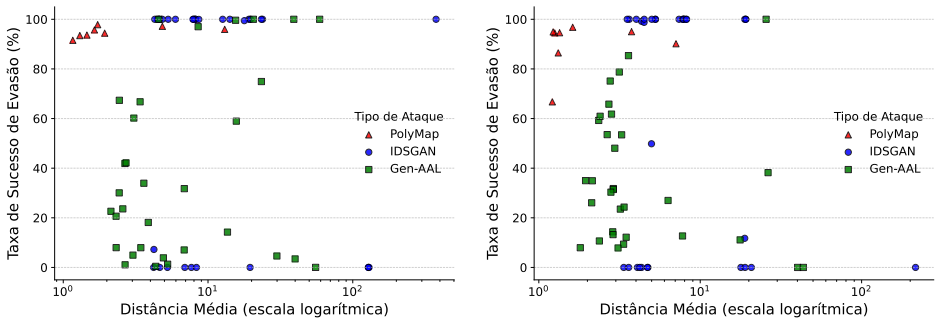

Figura 2. Comparação entre taxa de sucesso de evasão e distância média das amostras nos modelos FNN (esquerda) e SNN (direita) treinados no *dataset* NSL-KDD.

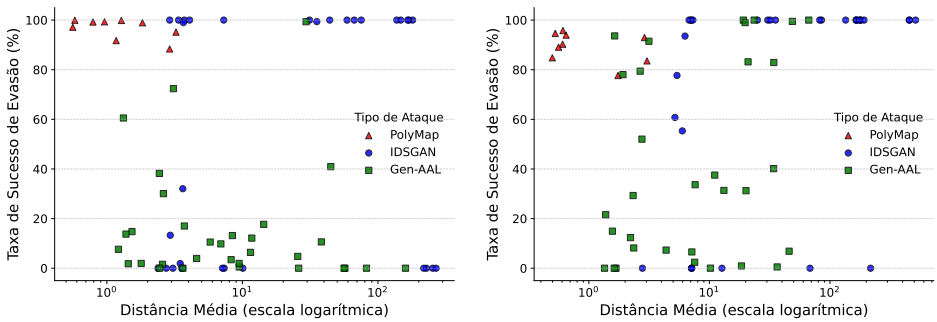

Figura 3. Comparação entre taxa de sucesso de evasão e distância média das amostras nos modelos FNN (esquerda) e SNN (direita) treinados no *dataset* TON_IoT.

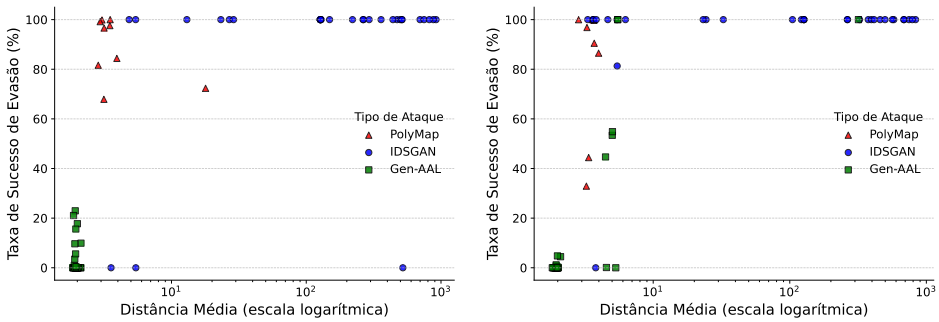

Figura 4. Comparação entre taxa de sucesso de evasão e distância média das amostras nos modelos FNN (esquerda) e SNN (direita) treinados no *dataset* BoT-IoT.

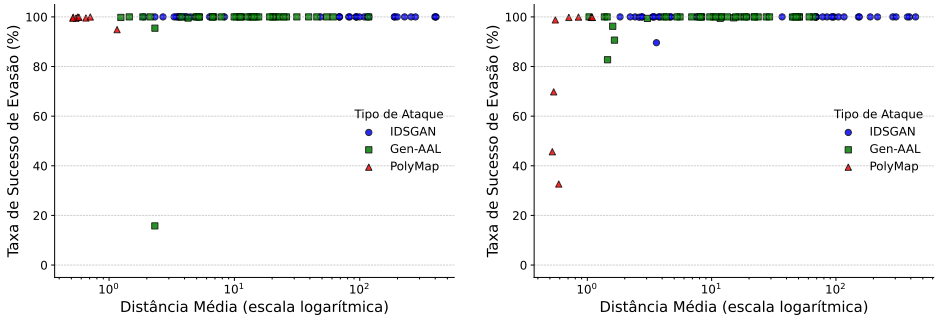

Figura 5. Taxa de sucesso de evasão vs. distância média das amostras nos modelos FNN (esquerda) e SNN (direita) treinados no *dataset* CTU-13

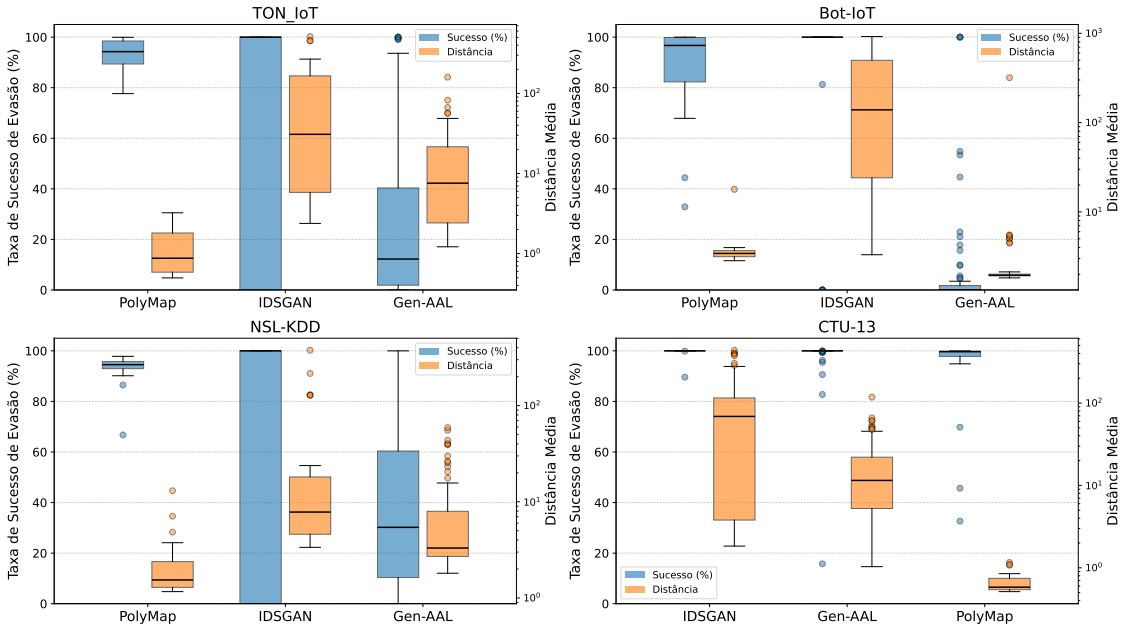

Figura 6. Resultados de taxa de sucesso de evasão e distância média entre as amostras originais e adversariais obtidos durante a execução de cada método de evasão em cada *dataset*.

Tabela 1. Média e desvio padrão das taxas de sucesso de evasão e distâncias obtidas nas três melhores execuções de cada método em cada conjunto de dados

Ataque	Tipo de Rede	Taxa de Sucesso de Evasão (%)	Distância Média
PolyMap	FNN	99.07 ± 1.35	2.81 ± 3.52
	SNN	97.55 ± 2.57	1.76 ± 1.32
IDSGAN	FNN	100.0 ± 0.00	4.46 ± 2.90
	SNN	100.00 ± 0.00	4.09 ± 1.85
Gen-AAL	FNN	74.50 ± 34.97	9.12 ± 13.07
	SNN	97.01 ± 7.12	39.56 ± 89.75

Ao comparar os resultados obtidos para cada conjunto de dados, observou-se que os métodos de evasão obtiveram sucesso de forma mais consistente no CTU-13, no qual a maioria das execuções apresentou uma taxa de sucesso superior a 95%. Por outro lado, os modelos de detecção treinados nos demais *datasets* demonstraram uma maior resiliência contra os ataques adversariais. Esse comportamento indica que as diferenças nos dados utilizados, como a similaridade entre amostras de tráfego normal e malicioso, podem influenciar na execução de ataques adversariais.

Em outro ponto de análise, observou-se que o método *PolyMap* enviou, durante sua fase de treinamento, mais requisições do que o método *Gen-AAL* e menos do que o método *IDSGAN*, conforme indicado na Figura 7. Ao se comparar as taxas de sucesso de evasão obtidas nas redes do tipo FNN e SNN, não foram observadas diferenças significativas para os métodos *PolyMap* e *Gen-AAL*, enquanto o método *IDSGAN* apresentou taxas de sucesso melhores nas redes SNN, conforme indicado na Tabela 1. Esse comportamento indica que as redes do tipo SNN, apesar de serem capazes de impedir ataques do tipo FGSM, não são muito eficazes contra outros tipos de ataques adversariais.

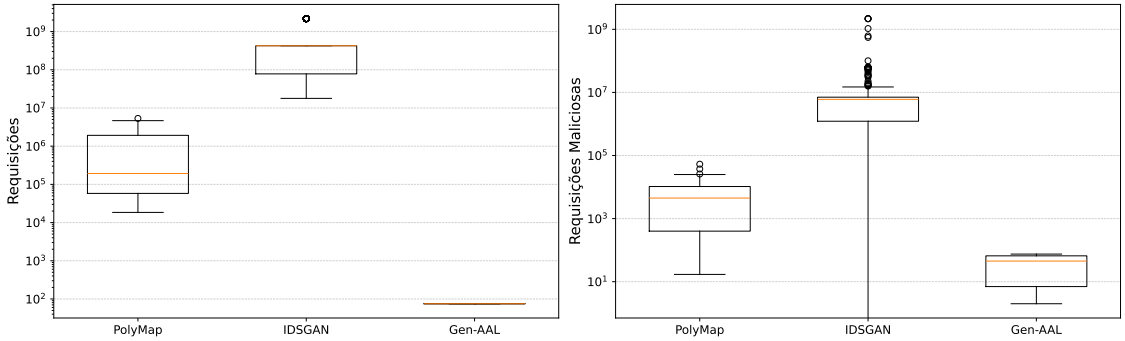

Figura 7. Métodos de ataque adversariais vs. número de requisições (esquerda) e de requisições maliciosas (direita) durante treinamento

6. Conclusão

Este trabalho apresentou o método *PolyMap* para modificar amostras de rede maliciosas com o objetivo de evitar sua detecção por modelos de classificação de tráfego de rede. O método proposto consiste inicialmente no mapeamento do espaço de decisão do modelo

de classificação, representando-o por um conjunto de politopos convexos. Em seguida, são calculadas as projeções de cada amostra nos politopos encontrados e selecionada a que resulta em menores perturbações adicionadas à amostra original.

Os resultados experimentais nos *datasets* Bot-IoT, TON_IoT, CTU-13 e NSL-KDD demonstraram que, apesar da variação entre os resultados obtidos por todos os métodos, o *PolyMap* apresentou maior estabilidade entre diversas execuções, facilitando seu uso para realizar ataques. O método também obteve uma taxa de evasão média similar a do *IDSGAN* e superior a do *Gen-AAL*, adicionando menores perturbações nas amostras, de forma a maximizar sua utilidade para a realização de ataques. Durante seu treinamento, o método proposto envia um número inferior de requisições do que o *IDSGAN*, reduzindo a chance de detecção durante essa etapa. Ele permite a escolha de diferentes objetivos para a geração de amostras adversariais pela modificação da função objetivo, limitando as perturbações às *features* consideradas não funcionais, que não influenciam o objetivo do ataque, por meio da inclusão de novas restrições na equação. Alternativamente, diferentes pesos podem ser atribuídos às perturbações em cada *feature*, de forma a priorizar alterações em características menos importantes das amostras de tráfego. Esse comportamento torna o método mais versátil do que os demais comparados, possibilitando um maior controle na geração de amostras adversariais.

Em trabalhos futuros, o método *PolyMap* pode ser aplicado a outras classes de métodos de aprendizado de máquina, como árvores de decisão, *Support Vector Machines* (SVM) e *K-nearest Neighbors* (KNN), para avaliar seu desempenho. Também podem ser desenvolvidos e comparados diferentes algoritmos para o mapeamento do espaço de decisão do modelo de detecção de ameaças de rede, com o objetivo de reduzir o número de consultas realizadas. O mecanismo utilizado pelo método *Gen-AAL* para realizar consultas em um modelo de classificação que reproduz o comportamento do modelo original também pode ser explorado no contexto do método proposto neste trabalho.

Outro ponto de estudo em trabalhos futuros corresponde às limitações do método apresentado e aos mecanismos de defesa efetivos para limitar seu funcionamento. Devido ao mapeamento do espaço como um conjunto de politopos convexos, o *PolyMap* pode apresentar dificuldades ao realizar ataques em modelos de classificação de tráfego que não possuem um espaço de decisão representável dessa forma, como sistemas baseados em reputação ou que utilizam mecanismos estocásticos para classificar uma amostra. Nesses casos, as amostras adversariais podem ser classificadas como tráfego malicioso e/ou apresentar uma grande distância entre as amostras originais, reduzindo a efetividade do ataque.

Disponibilidade de Artefatos

Os códigos utilizados para a execução e avaliação dos modelos de detecção de ameaças de rede e métodos de ataque, assim como os resultados obtidos, estão disponibilizados no repositório do GitHub `https://github.com/RafaelDiasCampos/PolyMap-SBRC`.

Referências

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM*

SIGKDD International Conference on Knowledge Discovery and Data Mining.

Booij, T. M., Chiscop, I., Meeuwissen, E., Moustafa, N., and Hartog, F. T. H. d. (2022). Ton_iot: The role of heterogeneity and the need for standardization of features and attack types in iot network intrusion data sets. *IEEE IoT-J*, 9(1):485–496.

Ducoffe, M. and Precioso, F. (2018). Adversarial active learning for deep networks: a margin based approach.

Enea (2025). Enea qosmos threat detection sdk - ids-based threat detection. `https://www.enea.com/solutions/deep-packet-inspection-traffic-intelligence/threat-detection-sdk/`. Acessado em: 14/10/2025.

Fan, F.-L., Huang, W., Zhong, X., Ruan, L., Zeng, T., Xiong, H., and Wang, F. (2024). Deep relu networks have surprisingly simple polytopes.

Fortinet (2025). Next Generation Firewall (NGFW) - See top products. `https://www.fortinet.com/products/next-generation-firewall`. Acessado em: 21/01/2026.

Gajer, P. and Ravel, J. (2025). The geometry of machine learning models.

García, S., Grill, M., Stiborek, J., and Zunino, A. (2014). An empirical comparison of botnet detection methods. *Computers & Security*, 45:100–123.

Ibitoye, O., Shafiq, O., and Matrawy, A. (2019). Analyzing adversarial attacks against deep learning for intrusion detection in iot networks. In *GLOBECOM*, pages 1–6.

Koroniotis, N., Moustafa, N., Schiliro, F., Gauravaram, P., and Janicke, H. (2020). A holistic review of cybersecurity and reliability perspectives in smart airports. *IEEE Access*, 8:209802–209834.

Lin, Z., Shi, Y., and Xue, Z. (2022). *IDSGAN: Generative Adversarial Networks for Attack Generation Against Intrusion Detection*, page 79–91. Springer International Publishing.

Moustafa, N. (2021). A new distributed architecture for evaluating ai-based security systems at the edge: Network ton_iot datasets. *Sustainable Cities and Society*, 72:102994.

Sattelberg, B., Cavalieri, R., Kirby, M., Peterson, C., and Beveridge, R. (2020). Locally linear attributes of relu neural networks.

Shu, D., Leslie, N. O., Kamhoua, C. A., and Tucker, C. S. (2020). Generative adversarial attacks against intrusion detection systems using active learning. In *Proceedings of the 2nd ACM Workshop on Wireless Security and Machine Learning*, WiseML '20, page 1–6.

Tavallaee, M., Bagheri, E., Lu, W., and Ghorbani, A. A. (2009). A detailed analysis of the kdd cup 99 data set. In *IEEE CISDA*, pages 1–6.

Progress Software Corporation (2025). Flowmon - trusted solution for network and security operations - progress flowmon. `https://www.progress.com/flowmon`. Acessado em: 14/10/2025.