ORDEM
DOS
ENGENHEIROS

ISEP INSTITUTO SUPERIOR
DE ENGENHARIA DO PORTO

Melhoria da Segurança e Robustez dos Modelos de Aprendizagem Automática

3º Prémio Inovação Jovem Engenheiro 2023

Atribuído por:

Ordem dos Engenheiros – Região Sul

Trabalho desenvolvido por:

João Pedro Machado Vitorino

Membro Estagiário nº 88453, Colégio de Engenharia Informática

Professor Assistente Convidado, Instituto Superior de Engenharia do Porto

30 de Dezembro de 2023

Resumo

As organizações podem beneficiar do uso da Inteligência Artificial, e mais especificamente da Aprendizagem Automática (ML), para enfrentar os ciberataques direcionados aos seus processos de negócio. No entanto, apesar dos benefícios de utilizar ML em soluções de cibersegurança inteligentes, estes também têm vulnerabilidades que não devem ser negligenciadas. A falta de robustez é um dos principais desafios que prejudicam a confiabilidade de ML, pois os modelos são suscetíveis a exemplos adversos: dados com pequenas perturbações que causam comportamentos inesperados.

Este trabalho apresenta uma metodologia para uma análise e comparação da robustez de modelos ML para o domínio da Deteção de Intrusões de Rede, e uma ferramenta inteligente para a geração de exemplos adversos realistas: Método de Perturbação com Padrões Adaptativos (A2PM). É demonstrado que um ataque adverso bem-sucedido não é garantidamente um ciberataque bem-sucedido, e que as perturbações adversas só são realistas se cumprirem as restrições de uma rede de computadores e de cada classe de ciberataque.

A2PM pode ser usado para ataques adversos, ao causar erros de classificação de forma iterativa, e para treino adverso, ao melhorar a variedade de um conjunto de dados. Foram efetuadas duas aplicações práticas em redes de computadores empresariais e em redes de dispositivos IoT, considerando diferentes tipos de modelos ML e de ciberataques. Foi verificado que o treino adverso com perturbações simples permitiu aos modelos reter uma boa generalização a fluxos de tráfego de rede normais, para além de serem mais robustos contra fluxos perturbados.

A conclusão mais importante deste trabalho é: os modelos ML podem ser incrivelmente valiosos para melhorar um sistema de cibersegurança, mas as suas próprias vulnerabilidades não devem ser negligenciadas. É essencial continuar os esforços de investigação para melhorar a segurança e a robustez de ML e dos sistemas inteligentes que utilizam ML.

Índice

1 Introdução

Esta primeira secção contextualiza o desafio abordado por este trabalho, apresenta os objetivos que foram estabelecidos e descreve as principais contribuições.

1.1 Contexto e Motivação

As organizações podem beneficiar dos avanços tecnológicos dos últimos anos para transformar os seus processos de negócio, integrar os sistemas de informação e automatizar a tomada de decisão. Contudo, com as organizações a tornarem-se mais dependentes dos sistemas digitais, os riscos de um ciberataque ficam mais graves [1]. Cada nova tecnologia introduz novas vulnerabilidades que podem ser exploradas por atacantes para prejudicar um sistema. Isto torna-se particularmente preocupante quando uma organização processa informação confidencial e dados pessoais, ou gere infraestrutura crítica, como nos setores de saúde e de energia [2].

As disrupções causadas por um ciberataque podem ser extremamente caras para uma organização. Em 2022, foi reportado que o custo médio de uma quebra de segurança foi de 4.35 milhões de dólares americanos, um aumento de 12.7% desde 2020 [3]. Este aumento tanto do número de ciberataques como dos seus custos em vários setores denota que as organizações enfrentam muitos desafios de segurança. Como a mitigação dos ataques não é um processo trivial e as pequenas empresas tendem a não cumprir as boas-práticas de segurança, a maioria acaba por fechar o negócio dentro de 6 meses depois de sofrer uma quebra de segurança [4].

Com a continuidade de negócio em risco, é essencial que as organizações de todos os tamanhos monitorizem as suas redes de computadores, para detetarem atividade suspeita e conseguirem mitigar possíveis ameaças. Isto é onde a Inteligência Artificial, e mais especificamente a Aprendizagem Automática (ML, do inglês *Machine Learning*), podem ser incrivelmente valiosos [5]. Os modelos ML podem ser criados a partir de diversos algoritmos, incluindo algoritmos baseados em árvores de decisão e algoritmos de aprendizagem profunda baseados em redes neuronais artificiais. Estes modelos podem ser treinados para automatizar várias tarefas, desde o reconhecimento de padrões e anomalias em tráfego de rede até à classificação de várias classes de ciberataques.

No entanto, apesar dos benefícios de utilizar ML em soluções de cibersegurança inteligentes, estes também têm vulnerabilidades que não devem ser negligenciadas. A falta de robustez é um dos principais desafios que prejudicam a confiabilidade de ML, pois os modelos são suscetíveis a exemplos adversos: dados com pequenas perturbações que causam comportamentos inesperados [6]. Caso um fluxo de tráfego de rede benigno tenha informação em falta, isto pode levar a um erro de classificação e a falsos alarmes. Por outro lado, caso um fluxo de tráfego de um ciberataque seja modificado ligeiramente, esse fluxo pode conseguir enganar um modelo e evitar a deteção, prejudicando os sistemas e processos de negócio de uma organização [7].

Os engenheiros de ML e os profissionais de cibersegurança ainda não têm o conhecimento nem as ferramentas necessárias para prevenir as disrupções dos modelos, por isso os exemplos adversos representam uma grande ameaça para os sistemas inteligentes que utilizam ML [8], [9]. Uma das principais dificuldades no desenvolvimento de estratégias de defesa é encontrar um equilibro entre a robustez de um modelo contra exemplos adversos e a generalização desse modelo para dados normais que irá encontrar regularmente. Portanto, a melhoria da segurança dos modelos ML é um desafio pertinente que requer mais esforços de investigação [10].

Este trabalho apresenta os esforços de investigação e desenvolvimento realizados para melhorar a segurança e a robustez dos modelos ML, com o foco no domínio da Deteção de Intrusões de Rede (NID, do inglês *Network Intrusion Detection*), sob a orientação de Dra. Isabel Cecília Correia da Silva Praça Gomes Pereira e Dra. Eva Catarina Gomes Maia. O trabalho desenvolvido esteve alinhado com a participação do Grupo de Investigação em Engenharia do Conhecimento e Apoio à Decisão (GECAD) [11], do Instituto Superior de Engenharia do Porto, em dois projetos do programa Horizonte 2020 da União Europeia: *Secure Collaborative Intelligent Industrial Assets* (SeCoIIA) [12] e *Verification and Validation of Automated Systems' Safety and Security* (VALU3S) [13]. O primeiro focou-se na segurança da transição digital da indústria, abordando as redes de dispositivos de Internet-das-Coisas (IoT, do inglês *Internet-of-Things*). O segundo focou-se na melhoria dos processos de verificação e validação dos sistemas inteligentes, para garantir a sua segurança.

1.2 Objetivos e Contribuições

O principal propósito deste trabalho foi a criação de uma solução para treinar modelos ML mais robustos, com recurso à geração de exemplos adversos realistas de ciberataques. Como os fluxos de tráfego são representados num formato de dados tabulares, foi necessário garantir que os exemplos gerados são adequados e aplicáveis a uma rede de computadores real. Com base neste propósito, foram estabelecidos três objetivos mais específicos:

- **OB1**: Formular uma abordagem para analisar a robustez de modelos ML que considere as restrições complexas do domínio NID.
- **OB2**: Desenvolver uma ferramenta capaz de gerar exemplos adversos realistas de fluxos de tráfego de rede pertencentes a ciberataques.
- **OB3**: Validar e testar a ferramenta desenvolvida em cenários do domínio NID adequados para os projetos SeCoIIA e VALU3S.

Durante o desenvolvimento, foram examinadas as restrições complexas das redes de computadores, dos protocolos de comunicação utilizados, e dos ciberataques que enfrentam. Foi desenvolvido o Método de Perturbação com Padrões Adaptativos (A2PM, do inglês *Adaptative Perturbation Pattern Method*) e foram realizadas várias experiências, considerando algoritmos de aprendizagem profunda baseados em redes neuronais artificiais e algoritmos baseados em conjuntos de árvores de decisão, nomeadamente *Multilayer Perceptron* (MLP) e *Random Forest* (RF).

As principais contribuições deste trabalho podem ser sumarizadas em quatro pontos:

- Uma metodologia para uma análise confiável da robustez de múltiplos modelos ML, com um vetor de ataque de evasão adequado para uma rede de computadores real.
- Uma ferramenta inteligente, A2PM, para a geração de exemplos adversos realistas para ataques adversos e treino adverso em domínios de dados tabulares complexos.
- Duas aplicações práticas da ferramenta em redes de computadores empresariais e em redes de dispositivos IoT, considerando diferentes tipos de modelos ML e de ciberataques.
- Duas análises do realismo dos exemplos gerados, dos seus efeitos nos modelos, e do consumo de tempo da ferramenta, com base nos resultados das duas aplicações práticas.

O trabalho de investigação e desenvolvimento apresentado neste documento contribuiu para a dissertação do Mestrado em Engenharia de Inteligência Artificial do autor. Foram também publicados vários artigos científicos em conferências e revistas científicas, com experiências adicionais que não constam neste documento, incluindo outros modelos e outros conjuntos de dados.

Para facilitar a leitura, este documento está divido em várias secções. Esta primeira secção contextualizou o desafio abordado e apresentou os principais objetivos e contribuições. A Secção 2 apresenta o estado atual da literatura sobre exemplos adversos e descreve as ferramentas existentes. A Secção 3 apresenta a metodologia de análise de robustez que foi formulada e descreve a ferramenta de geração de exemplos adversos que foi desenvolvida. A Secção 4 apresenta a aplicação da ferramenta a dois cenários de redes de computadores empresariais e redes de dispositivos IoT, analisando e discutindo os resultados obtidos. Finalmente, a Secção 5 apresenta as principais conclusões, indicando possíveis melhorias a serem exploradas no futuro.

2 Estado da Arte

Esta secção apresenta o estado atual da literatura sobre exemplos adversos e descreve as ferramentas existentes para a realização de ataques de evasão contra modelos ML.

2.1 Exemplos Adversos

ML tem sido cada vez mais utilizado para tornar os sistemas digitais mais inteligentes, mas os modelos não são perfeitos. Num conjunto de dados todos da mesma classe, um modelo pode classificar corretamente todos exceto um. Esse pode ser atribuído a uma classe completamente diferente porque o modelo considera-o diferente dos restantes. A razão por detrás deste comportamento inesperado é a existência de falhas na generalização do modelo que não foram corrigidas durante o seu processo de treino [14]. Como um conjunto de dados de treino não cobre todos os casos que um modelo poderá encontrar quando estiver a ser utilizado num sistema real, esse modelo irá inevitavelmente aprender algumas simplificações dos dados que levam a incorreções nos seus mecanismos internos [15], [16]. Estas incorreções podem ser difíceis de notar porque os modelos ML só falham em dados bastante específicos, que são designados como exemplos adversos.

Um exemplo adverso pode ter pequenas perturbações que são impercetíveis para engenheiros de ML, mas que o tornam muito diferente para um modelo. Essas perturbações podem ocorrer naturalmente em capturas de dados incorretas ou com informação em falta, mas também podem ser especificamente criadas por atacantes para explorar alguma falha na generalização de um modelo [17], [18]. Apesar de todos serem suscetíveis a exemplos adversos, diferentes modelos aprendem diferentes simplificações dos dados. Por isso, alguns serão mais vulneráveis a perturbações em certas variáveis do que outros, apresentando falhas que são específicas para cada modelo e, consequentemente, são difíceis de detetar e corrigir [19], [20].

Devido aos avanços nas tecnologias de visão por computador, os principais desenvolvimentos na área dos exemplos adversos têm sido focados no domínio da classificação de imagens [21], [22]. Apesar de as variáveis serem pixéis com qualquer valor entre 0 e 255, o que é substancialmente diferente dos fluxos de tráfego de rede, é importante perceber como esses desenvolvimentos podem ser aplicados nas soluções de cibersegurança e se existem conceitos que podem ser transferidos para o domínio NID. Na literatura atual, as perturbações que transformam dados normais em exemplos adversos podem ser criadas a partir de dois principais conceitos: uma perturbação de mancha que modifica muito um número reduzido de pixéis (Figura 1) [23]; e uma perturbação de máscara que modifica subtilmente muitos ou até mesmo todos os pixéis de uma imagem (Figura 2) [24].

Figura 1. Perturbação através de uma mancha, baseada em [23].

Figura 2. Perturbação através de uma máscara, baseada em [24].

Apesar de a maior parte dos desenvolvimentos ser na classificação de imagens, a suscetibilidade dos modelos ML a estes exemplos também tem sido notada noutros domínios com diferentes tipos de dados, como áudio, texto, dados tabulares e séries temporais [19], [25]. No domínio NID, as perturbações têm de ser criadas num formato tabular, em que as variáveis contêm dados categóricos ou numéricos que representam características dos fluxos de tráfego de rede [26], [27]. Isto requer perturbações mais complexas, mas os conceitos utilizados nas imagens poderiam ser adaptados. Uma perturbação de mancha poderia substituir os valores das variáveis categóricas, como o estado da conexão, enquanto uma perturbação por máscara poderia aumentar ou diminuir os valores das variáveis numéricas, como a quantidade de pacotes enviados [28], [29].

No entanto, nem todas as perturbações são adequadas para o domínio NID. Em contraste com os pixéis de uma imagem, diferentes variáveis podem apresentar valores completamente distintos, de acordo com a característica que representam e com as correlações entre várias variáveis [28]. Portanto, para garantir que um exemplo pode ser transmitidos através de uma rede de computadores real, é necessário ter em consideração os protocolos de comunicação e os propósitos dos ciberataques ao gerar perturbações [30], [31]. Apesar de ainda ser difícil criar exemplos adversos realistas, a crescente popularidade desta área está a levar ao desenvolvimento de novas ferramentas para atacar diversos tipos de modelos, o que é preocupante para a segurança de ML e dos sistemas inteligentes que utilizam ML [8], [26], [32].

2.2 Ataques de Evasão

Nos últimos anos, várias ferramentas foram desenvolvidas para automatizar os ataques a modelos ML e melhorar as tentativas de causar erros de classificação de forma a evitar a deteção. Estes ataques de evasão podem precisar de diferentes níveis de acesso a um modelo: apenas dos valores das suas previsões para um certo conjunto de dados; de conhecimento da arquitetura do modelo e das variáveis utilizadas; e de acesso total aos seus mecanismos e parâmetros internos [33], [34]. Estas características afetam a escolha de uma ferramenta, visto que um atacante tem de cumprir os seus propósitos maliciosos muitas vezes sem acesso a informação do modelo nem das variáveis.

Vários métodos inicialmente desenvolvidos para perturbar imagens foram adaptados para perturbar fluxos de tráfego de rede. Contudo, a maioria não tem em consideração as restrições destes fluxos nem as funcionalidades pretendidas dos diferentes ciberataques, por isso não têm conseguido gerar exemplos adversos realistas [35], [36]. As ferramentas que poderiam potencialmente ser utilizadas no domínio NID estão descritas em baixo.

O ataque polimórfico [37] foi desenvolvido para o domínio NID, por isso tenta abordar a preservação das características de cada classe de ciberataque. Um algoritmo de seleção de variáveis é aplicado para obter as que são mais relevantes para a distinção entre fluxos benignos e de ciberataques. Depois, as restantes variáveis, que são consideradas como não-relevantes, são perturbadas por uma versão de uma rede adversária generativa [38]: uma rede generativa *Wasserstein* [39]. Ao não modificar as variáveis mais importantes, as características necessárias para um ciberataque poderão ser preservadas. Contudo, as perturbações aleatórias geradas pela rede generativa *Wasserstein* nas restantes variáveis ignora toda a estrutura dos fluxos de tráfego, o que pode levar a exemplos adversos inválidos que não poderiam ser transmitidos numa rede de computadores real.

O ataque de limites [29] otimiza iterativamente as perturbações de acordo com uma grande configuração manual de todas as restrições que se pretender utilizar. Apesar de poder preservar as correlações entre as diferentes variáveis de um fluxo de tráfego e poder potencialmente gerar exemplos realistas, esta ferramenta requer conhecimento de perito para configurar manualmente valores específicos para cada variável e para cada perturbação. Esta análise de perito não escala bem para cenários com muitas classes de ciberataques em redes de computadores complexas.

Apesar de terem sido desenvolvidos para imagens, as perturbações do *Jacobian-based Saliency Map Attack* (JSMA) [40] e do ataque *OnePixel* [41] podem potencialmente preservar a estrutura de um fluxo de tráfego. O primeiro minimiza o número de variáveis perturbadas através de um acesso total aos mecanismos e parâmetros internos de um modelo, e o segundo escolhe apenas a variável mais relevante para modificar. Estas ferramentas apenas perturbam as variáveis mais adequadas sem afetar as restantes, apesar de não considerarem qualquer restrição nos valores das perturbações, o que pode levar a valores incompatíveis entre as diferentes variáveis de um fluxo de tráfego.

Como não foi encontrada nenhuma ferramenta capaz de cumprir todas as restrições do domínio NID, este trabalho focou-se em desenvolver uma solução para gerar exemplos adversos realistas e usá-los para melhorar as estratégias de defesa dos sistemas inteligentes que utilizam ML.

3 Solução Proposta

Esta secção apresenta a metodologia de análise da robustez de modelos ML que foi formulada e descreve a ferramenta de geração de exemplos adversos que foi desenvolvida.

3.1 Metodologia de Análise

O domínio NID é um domínio de dados tabulares complexo, em que os exemplos adversos devem representar fluxos de tráfego de rede reais que conseguiriam evitar a deteção numa rede de computadores. Estes exemplos adversos deveriam enganar um modelo ML para que os classificasse como tráfego benigno enquanto na realidade continuariam a ser parte de um ciberataque. Senão, os exemplos não seriam úteis para um atacante e poderiam mesmo ser contraproducentes para uma estratégia de defesa porque o modelo aprenderia com base em informação incorreta que levaria a uma perda de desempenho.

Para uma análise de robustez ser confiável e dar resultados úteis, deve ser incluída uma avaliação e comparação de múltiplos modelos ML. Devem ser utilizados exemplos adversos realistas num vetor de ataque adequado e aplicável a uma rede de computadores real. Portanto, é importante examinar as restrições dos dados utilizados e estabelecer uma metodologia de análise adequada para domínios de dados tabulares complexos como o domínio NID.

3.1.1 Restrições dos Dados

Para gerar exemplos adversos realistas no domínio NID, é necessário examinar as restrições complexas das redes de computadores, dos protocolos de comunicação utilizados, e dos ciberataques que enfrentam. Um fluxo de tráfego, representado num formato tabular, contém várias variáveis que representam as características que esse tráfego exibe ao ser transmitido numa rede de computadores. Como essas variáveis têm de seguir distribuições específicas de acordo com os protocolos de comunicação, a presença de um determinado valor numa variável pode restringir os valores de outras variáveis. Conceptualmente, isto leva a dois principais tipos de restrições:

- **Restrições intra-variável**: Limitam os valores possíveis para uma variável;
- **Restrições inter-variável**: Limitam os valores possíveis para uma ou mais variáveis, de acordo com os valores presentes em uma ou mais outras variáveis.

Os dois tipos de restrições podem ser observados no *Inter-Arrival Time* (IAT), uma das características dos fluxos de tráfego de rede mais relevantes para o domínio NID. O valor de IAT representa o tempo entre a chegada de dois pacotes subsequentes de um fluxo, podendo ser representado como duas variáveis: o seu valor mínimo ao longo do fluxo (MinIAT) e o seu valor máximo (MaxIAT). Estas variáveis são importantes para a deteção de várias classes de ciberataques *Denial-of-Service* (DoS). Um valor baixo de MinIAT pode indicar um ataque DoS curto que tenta sobrecarregar rapidamente

um servidor com muitos pedidos, enquanto que um valor alto de MaxIAT pode indicar um ataque DoS prolongado que tenta sobrecarregar um servidor ao manter várias conexões abertas.

Estas duas variáveis, MinIAT e MaxIAT, apresentam uma restrição intra-variável bastante direta: só podem ter valores positivos porque decorre sempre algum tempo entre dois pacotes. No entanto, também podem ter outras restrições intra e inter-variável mais complexas, de acordo com as especificidades de uma rede de computadores e da funcionalidade pretendida para um ciberataque.

Algumas restrições mais complexas podem ser observadas num ciberataque Slowloris, um ataque DoS prolongado que tenta sobrecarregar um servidor web. Um fluxo de tráfego utilizado neste ciberataque deve usar o protocolo *Transmission Control Protocol* (TCP) e a bandeira *Push* (PSH) para manter a conexão aberta no porto número 80, onde o tráfego é recebido [42]. O fluxo pode ter um IAT que varia entre 20 e 30 segundos para aparentar ser tráfego normal, em vez de pacotes agendados. Um valor mais alto de IAT não pode ser utilizado porque uma medida de segurança comum para servidores web é terminar qualquer conexão após 30 segundos de inatividade [43].

Devido à falta de restrições das ferramentas de geração de perturbações do domínio da classificação de imagens, é difícil transferi-los para o domínio NID. O protocolo de comunicação, a bandeira da conexão e o porto devem permanecer os mesmos, senão o exemplo adverso deixaria de ser realmente um fluxo Slowloris. Os valores de MinIAT e MaxIAT podem ser modificados, mas nem todas as perturbações são adequadas para uma rede de computadores real.

Partindo dos valores originais de MinIAT e MaxIAT, 20 e 30 segundos, três diferentes exemplos podem ser gerados: o primeiro com um MinIAT de 22 e MaxIAT de 28, o segundo com 18 e 32, e o terceiro com 26 e 24. Apesar de os três exemplos poderem enganar um modelo ML e ser erradamente classificados como tráfego benigno, só o primeiro continua a ser realmente um fluxo Slowloris. O segundo exemplo é na verdade inofensivo porque o servidor web considerado terminará a conexão ao 31º segundo de inatividade, antes de o pacote seguinte ser recebido ao 32º segundo, prevenindo a funcionalidade deste ciberataque. Relativamente ao terceiro exemplo, esse não seria sequer possível numa rede de computadores real porque um fluxo com pacotes a pelo menos cada 26 segundos não pode ter também pacotes até ao máximo de 24 (Figura 3).

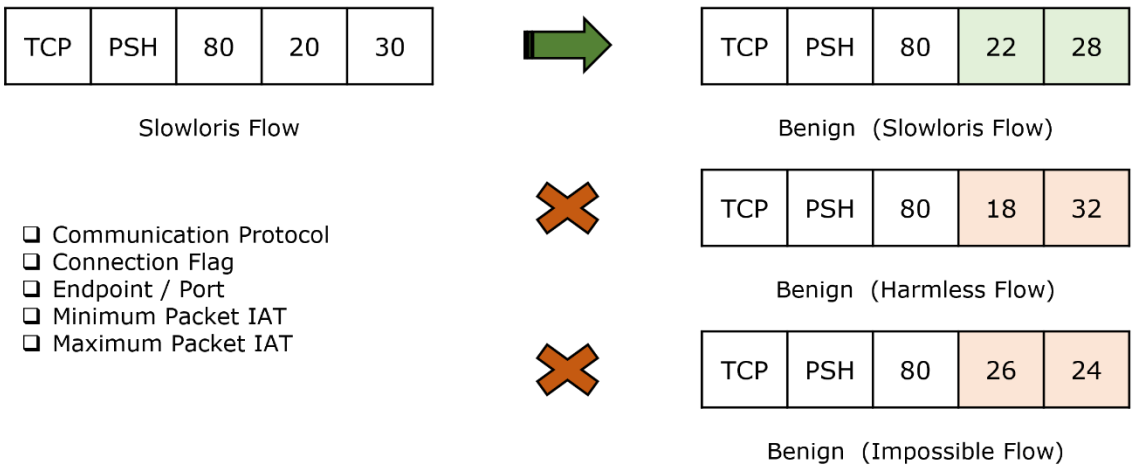

Figura 3. Perturbações num fluxo de tráfego da classe Slowloris.

Apesar de os três exemplos terem sido criados com perturbações semelhantes, levariam a Resultados bastante diferentes numa rede de computadores, e apenas um poderia ser usado contra um sistema NID real. Portanto, um ataque adverso bem-sucedido a um modelo ML não é garantidamente um ciberataque bem-sucedido numa rede de computadores.

Ao inspecionar o terceiro exemplo, pode ser observado que a razão de ser impossível é porque não cumpre a estrutura de dados tabulares inerente dos fluxos de tráfego. Por outro lado, a razão de o segundo exemplo ser inofensivo é porque não cumpre a funcionalidade pretendida para a classe Slowloris. Portanto, é possível definir duas propriedades fundamentais que são necessárias para que os exemplos adversos se assemelhem a um conjunto de dados reais:

- **Validade**: Conformidade com as restrições de um domínio, com a sua estrutura de dados;
- **Coerência**: Conformidade com as restrições de uma classe específica, com as características que a distinguem de outras classes.

O conceito de realismo torna-se um conceito unificante que combina estas duas propriedades. Um exemplo adverso só pode ser totalmente realista se for simultaneamente válido dentro da estrutura de dados do seu domínio e coerente com as caraterísticas da sua classe. Uma exemplificação destas propriedades é fornecida em baixo, descrevendo restrições de domínio e restrições específicas de uma classe que é necessário cumprir no domínio NID. As variáveis MinIAT e MaxIAT são consideradas na classe de um ataque DoS curto.

Durante a geração de perturbações, se MinIAT for aumentado para um valor maior do que MaxIAT, esse fluxo nunca seria encontrado por um modelo numa rede de computadores. Para preservar a validade, uma restrição inter-variável de domínio deve ser aplicada: o valor de MinIAT perturbado não pode ser maior do que o valor de MaxIAT perturbado. Abordagens para cumprir este tipo de restrições, incluindo intervalos de valores e pertença a múltiplas categorias, têm começado a ser investigadas recentemente [28], [29]. Estas restrições conseguem melhorar a validade de um ataque adverso para o domínio NID, mas a validade por si só não é suficiente para um ataque adverso serem um ciberataque bem-sucedido.

Se a restrição anterior for cumprida, fluxos válidos com MinIATs aumentados moderadamente podem ser erradamente classificados como benignos, mas não seriam rápidos o suficiente para sobrecarregar um servidor web. Consequentemente, esses supostos exemplos adversos não seriam realmente fluxos de um ataque DoS curto, representariam apenas tráfego benigno que não seria útil para um ciberataque. Para também preservar a coerência, é necessário aplicar uma restrição intra-variável específica dessa classe: o valor de MinIAT perturbado não pode ser maior do que o maior valor de MinIAT original para a classe de ataque DoS curto. Esta restrição é substancialmente mais complexa, mas é necessário que uma ferramenta a cumpra para garantir que um ataque adverso pode ser utilizado num sistema NID real.

Apesar de a validade ter sido abordada previamente, é essencial abordá-la em conjunto com a coerência para obter exemplos adversos realistas. Portanto, um exemplo adverso de um ciberataque deve representar tráfego de rede capaz de ser transmitido através de uma rede de computadores

real, e representar também um fluxo de ciberataque coerente capaz de cumprir a funcionalidade pretendida e os propósitos maliciosos de um atacante.

3.1.2 Comparação da Robustez

Para além de utilizar exemplos adversos realistas, uma comparação da robustez de modelos ML deve também considerar um cenário de ataque que possa ser realizado contra um sistema NID real. Para estabelecer um cenário de ataque adequado, é importante considerar como são criados os modelos ML e como são desenvolvidos os sistemas NID.

Uma boa prática para a criação de modelos ML para tarefas de NID e de classificação de ciberataques é utilizar os métodos de *Holdout* e *Cross-validation* em conjunto. Um conjunto de dados de fluxos de tráfego de rede pode ser dividido em subconjuntos de treino e teste, com 70% e 30% dos dados, e estratificação pode ser aplicada para garantir que as proporções das classes são preservadas. Depois, uma validação cruzada com 5 subconjuntos pode ser realizada, criando subconjuntos com 20% dos dados de treino cada, o que corresponde a 14% do conjunto de dados total. Neste processo de validação, 5 iterações são efetuadas, cada uma treinando um modelo com 4 subconjuntos e validando-o com o restante. Isto permite fazer uma otimização dos hiperparâmetros do modelo ao treiná-lo e validá-lo com 70% dos dados, e depois realizar uma avaliação final independente da generalização do modelo com os restantes 30% dos dados (Figura 4).

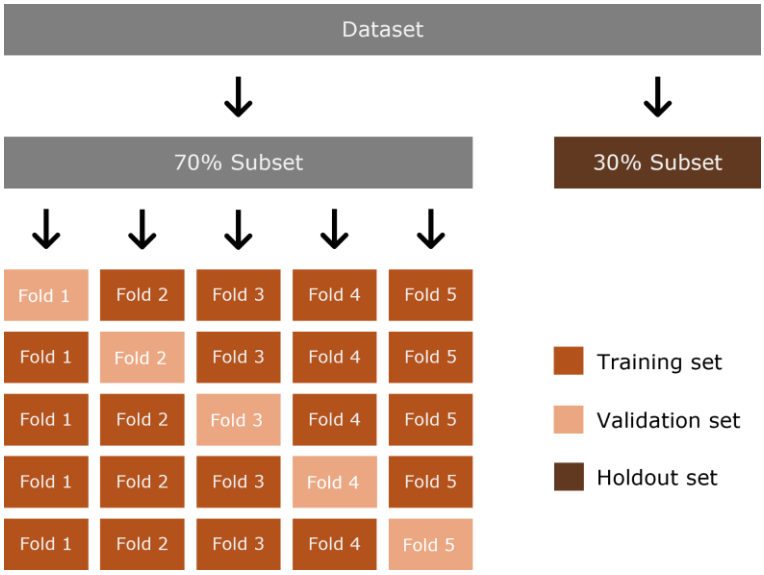

Figura 4. Método de Holdout combinado com Cross-validation.

Para além da sua generalização, para também avaliar a robustez de um modelo, um conjunto de exemplos adversos pode ser utilizado. Considerando que os sistemas NID são desenvolvidos num ambiente seguro com dados validados e verificados, é difícil um atacante ter acesso ou poder modificar os dados de treino do modelo, por isso terá de realizar um ataque de evasão utilizando novos dados que o modelo ainda não processou. Considerando também que os sistemas NID são colocados em produção com medidas de segurança para encapsular os modelos ML utilizados, um

atacante não terá acesso aos parâmetros internos do modelo, e só saberá se um exemplo adverso consegue evitar deteção se o ciberataque inteiro for completado com sucesso.

Este vetor de ataque de evasão pode ser simulado ao utilizar uma ferramenta de ataque com apenas acesso ao subconjunto de teste normal e à resposta de sim ou não de um modelo, caso o ataque tenha sido detetado ou não. Desta forma, é possível criar um conjunto de dados de teste adverso para avaliar os erros de classificação de um modelo específico sem causar o enviesamento dos resultados. Portanto, uma avaliação da robustez de um modelo ML pode ser realizada em 4 passos:

1. Preparar um conjunto de dados, dividindo-o em subconjuntos de treino e de teste;
2. Treinar e validar um modelo ML, utilizando o subconjunto de treino;
3. Realizar um ataque de evasão para criar um subconjunto de teste adverso específico para esse modelo, usando o subconjunto de teste normal e as previsões do modelo;
4. Avaliar o desempenho do modelo nos subconjuntos de teste normal e adverso, analisando a sua generalização a dados normais e a sua robustez a dados perturbados.

Como a introdução de alguns exemplos adversos nos dados de treino de um modelo pode melhorar a sua aprendizagem e torná-lo mais robusto, também é importante avaliar se essa abordagem é eficaz para um modelo específico. Assim, pode ser criado um segundo modelo com uma abordagem de treino adverso para analisar se existe perda de desempenho em dados normais e se essa perda é compensada por uma melhoria de desempenho em dados perturbados. Ao criar este par de modelos para vários algoritmos, desde algoritmos baseados em conjuntos de árvores de decisão até algoritmos de aprendizagem profunda baseados em redes neuronais artificiais, é possível realizar uma análise confiável de múltiplos modelos ML. Portanto, a metodologia de análise e comparação completa tem 5 passos, dos quais os passos 3 e 4 devem ser replicados para cada algoritmo:

1. Preparar um conjunto de dados, dividindo-o em subconjuntos de treino e de teste;
2. Criar uma perturbação simples numa cópia do subconjunto de dados de treino normais, criando um subconjunto de dados de treino adverso que contém mais variedade de dados;
3. Treinar e validar dois modelos ML, utilizando o subconjunto de treino normal para o primeiro e o subconjunto de treino adverso para o segundo;
4. Realizar dois ataques de evasão para criar um subconjunto de teste adverso específico para cada modelo, usando o subconjunto de teste normal e as previsões de cada modelo;
5. Avaliar o desempenho de cada modelo nos subconjuntos de teste normal e adverso, comparando a sua generalização a dados normais e a sua robustez a dados perturbados.

Relativamente às métricas a usar para a avaliação dos modelos, a métrica *accuracy* é a mais utilizada na literatura atual, por medir a proporção de fluxos corretamente classificados. Contudo, esta métrica é enviesada para as classes maioritárias, dando menos relevância às classes minoritárias, que são as classes dos ciberataques no domínio NID. Como o passo 4 irá gerar exemplos adversos apenas para os ciberataques, é necessário calcular esta métrica utilizando todos os fluxos exceto os benignos, para os efeitos do ataque serem exibidos corretamente. Outra métrica de avaliação bastante relevante é

o *F1-Score*, que calcula a média harmónica entre *precision* e *recall*, considerando os falsos positivos e os falsos negativos. Ao utilizar o *macro-averaged F1-Score*, é atribuída a mesma relevância a todas as classes minoritárias, por isso um valor alto nesta métrica indica que as diferentes classes de ciberataques estão a ser corretamente identificadas.

Conforme mais erros de classificação são feitos em fluxos de tráfego normais, mais baixo será o resultado de um modelo no conjunto de dados de teste normal. Da mesma forma, conforme mais erros de classificação são feitos em fluxos de tráfego perturbados, mais baixo será o resultado de um modelo no conjunto de dados de teste adverso. Portanto, um modelo ML deverá obter um bom equilíbrio entre os resultados obtidos nos conjuntos de dados de teste normal e adverso. Dos múltiplos modelos a ser avaliados, o que apresentar o melhor equilíbrio é considerado como o que tem a generalização mais robusta. No domínio NID, esse modelo será o mais confiável para a tarefa de deteção e classificação de ciberataques num sistema de segurança inteligente.

A metodologia proposta permite uma abordagem de segurança-por-design durante o durante o ciclo de vida de modelos ML. Da comparação realizada no último passo, o melhor modelo pode ser escolhido para ser colocado em produção, mas se novos conjuntos de dados forem obtidos, a análise e comparação deve ser repetida para garantir que o modelo continua a ser robusto. Esta metodologia deve ser regularmente replicada com dados atualizados para antecipar possíveis ameaças e usar esse conhecimento para melhorar a estratégia de defesa de um sistema inteligente (Figura 5).

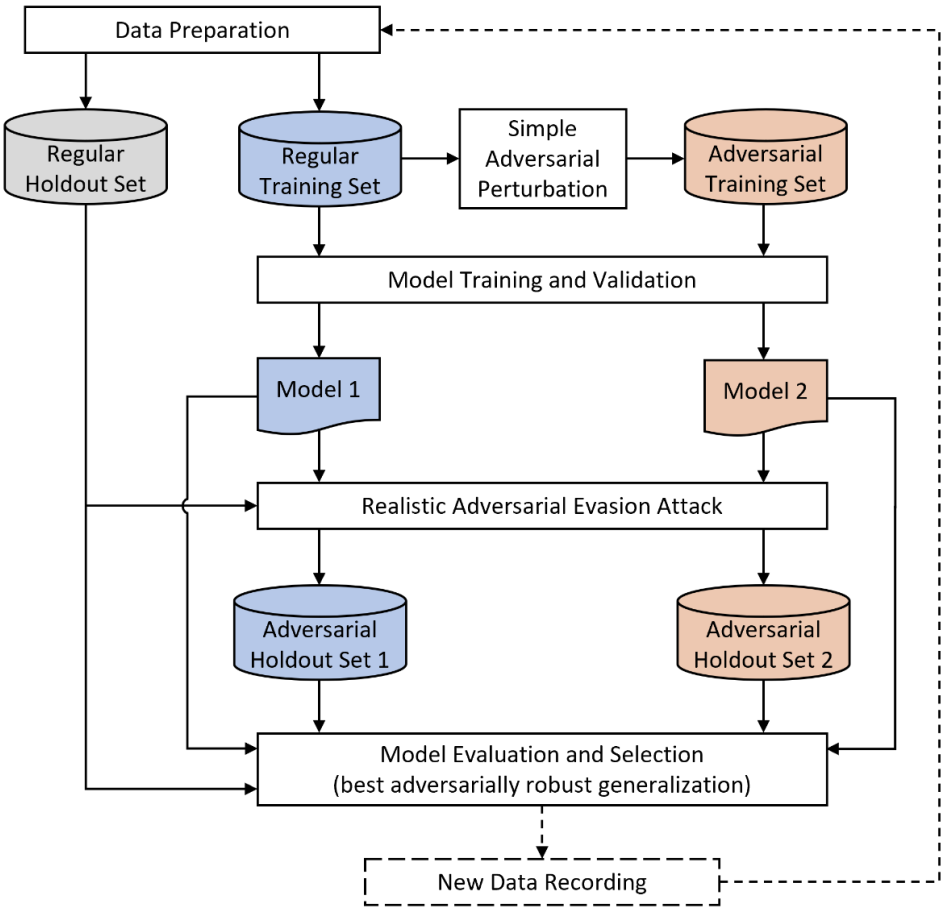

Figura 5. Metodologia de análise e comparação de robustez.

3.2 Ferramenta Desenvolvida

Considerando as duas propriedades necessárias para um exemplo adverso ser realista, validade e coerência, foi desenvolvida uma ferramenta capaz de gerar perturbações que cumprem as restrições das redes de computadores e das classes de ciberataques que enfrentam.

A ferramenta A2PM analisa as características de cada classe para gerar exemplos adversos realistas para domínios com dados tabulares (Figura 6). Foi implementado na linguagem de programação *Python 3* e utiliza a biblioteca *numpy* para otimizar várias operações matemáticas, conseguindo consumir poucos recursos computacionais e gerar perturbações rapidamente através de estruturas de dados vetorizadas. O código-fonte está disponível publicamente no *GitHub*[1] e a documentação pode ser consultada no *Read the Docs*[2], com os detalhes de cada função e um guia de utilização.

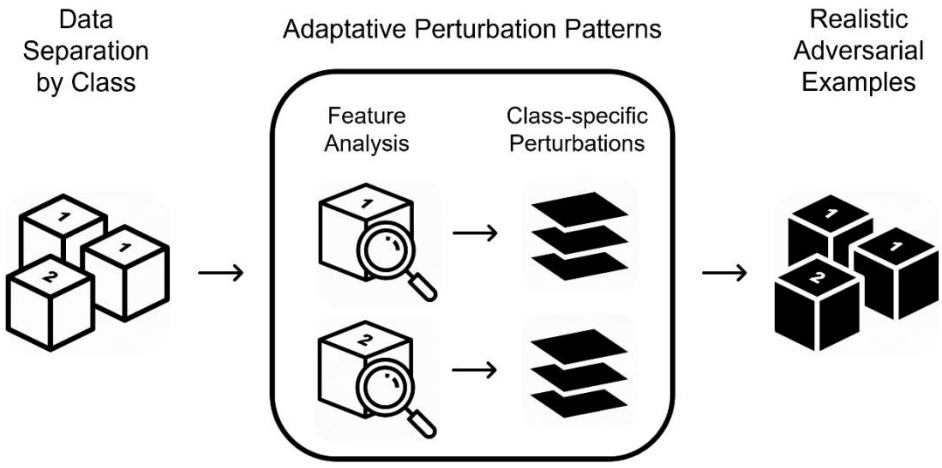

Figura 6. Método de Perturbação com Padrões Adaptativos.

Foi adotada uma arquitetura modular que atribui uma sequência independente de padrões adaptativos a cada classe, que analisa diferentes variáveis para aprender as suas características. Para treinar A2PM e o ajustar a um domínio específico, só é necessária uma configuração base para a sequência de padrões. Podem ser criadas perturbações simples num subconjunto de dados para melhorar a variedade dos dados e realizar treino adverso, ou pode ser efetuado um ataque adverso completo para iterativamente criar exemplos adversos e os testar contra um modelo ML.

Para usar A2PM para treino adverso no domínio NID, basta treiná-lo com um subconjunto de dados de treino. É providenciada uma função para criar uma perturbação simples em cada fluxo, criando um subconjunto de treino adverso. Isto permite a um modelo aprender não só do tráfego capturado numa rede, mas também de pequenas variações desse mesmo tráfego. Estas perturbações poderiam ser criadas manualmente através de uma análise extensa de todos os dados, mas é preferível usar a função automatizada de A2PM para prevenir o enviesamento da seleção de variáveis.

[1] Código-fonte de A2PM: https://github.com/vitorinojoao/a2pm

[2] Documentação de A2PM: https://a2pm.readthedocs.io/en/latest

Para usar A2PM para ataques adversos no domínio NID, basta treiná-lo com um subconjunto de dados de teste. São providenciadas funções para realizar ataques de evasão a modelos ML, gerando múltiplas perturbações até cada fluxo ter sido classificado erradamente ou o número máximo de iterações de ataque ser atingido. Isto resulta em subconjuntos de teste adversos específicos para cada modelo, com as perturbações necessárias para causar o maior número de erros nesse modelo. É importante destacar que apesar de esta ferramenta poder ser usada tanto para ataque como para defesa, são realizados diferentes procedimentos na função de uma perturbação simples e nos ataques completos, por isso pode ser realizada uma análise confiável de robustez.

Os ataques de evasão podem ser direcionados, tentando alterar um fluxo de um ciberataque para este passar despercebido como um fluxo benigno, ou não direcionados, tentando causar qualquer erro de classificação, como por exemplo, de Slowloris para outra classe de ciberataque. Para prevenir o uso de demasiados recursos computacionais, é realizada uma paragem antecipada do ataque quando as últimas iterações não estão a conseguir causar mais erros de classificação.

3.2.1 Padrão de Intervalos

Um dos principais aspetos da análise de variáveis realizada por A2PM é o intervalo de valores possíveis para uma variável numérica em diferentes classes. Cada um destes intervalos representa uma restrição intra-variável que deve ser cumprida ao limitar os valores mínimo e máximo que uma variável pode ter após ser perturbada, de acordo com a classe considerada.

O padrão de Intervalos foi criado com base neste conceito. Este padrão regista automaticamente os intervalos de valores das variáveis para as diferentes classes do seu conjunto de dados de treino, ficando preparado para gerar perturbações válidas e coerentes em cada variável de um novo conjunto de dados (Figura 7). A implementação pode ser configurada com uma 'probabilidade de ser aplicada', no intervalo $(0, 1]$, que determina se uma variável individual vai ser perturbada ou não, numa iteração de A2PM. Para variáveis que só devem conter valores inteiros, é também possível especificar 'perturbação inteira' para que o padrão garanta que as perturbações não resultam em valores contínuos nessas variáveis específicas.

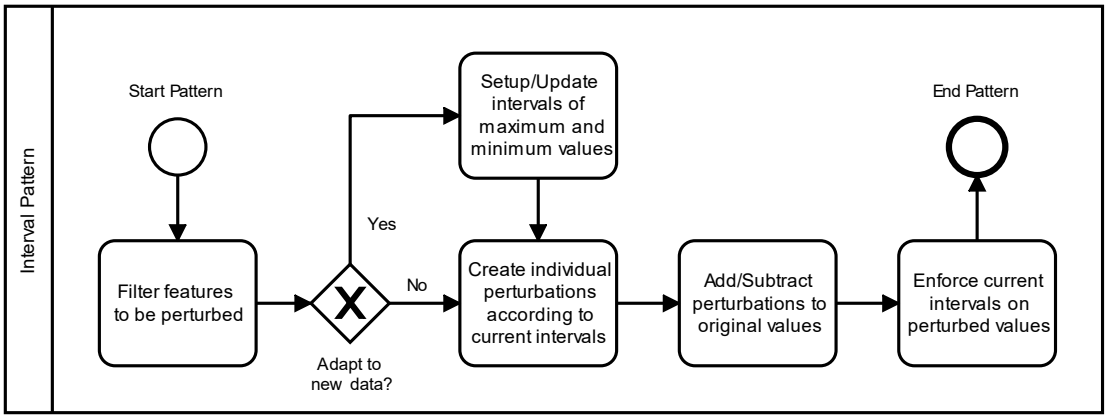

Figura 7. Vista geral do padrão de Intervalos.

Para além de um cenário estático em que todo o conjunto de dados está disponível, o padrão também pode ser treinado ao longo do tempo, para ter em consideração as mudanças nas distribuições dos dados. Em vez de um intervalo estático, é possível utilizar intervalos dinâmicos que permitem uma adaptação incremental a novos dados, de acordo com a configuração do valor de 'inércia'. Para uma dada variável e um valor de inércia $k \in [0, 1]$, os valores mínimo m_i e máximo M_i do intervalo atualizado após treinar com um conjunto de dados i são definidos como:

$$m_i = m_{i-1} * k + \min(x_i) * (1 - k) \quad (1)$$

$$M_i = M_{i-1} * k + \max(x_i) * (1 - k) \quad (2)$$

onde $\min(x_i)$ e $\max(x_i)$ são os valores mínimo e máximo originais das amostras x_i do conjunto i.

Cada perturbação é gerada de acordo com um número gerado aleatoriamente e é afetada pelo intervalo atualizado. O número aleatório $\varepsilon \in (0, 1]$ é utilizado para obter uma proporção da diferença entre os valores mínimo e máximo do intervalo. Por predefinição, de acordo com os valores de ε utilizados noutras ferramentas da literatura atual, este é restringido a $[0.1, 0.3]$, mas outros valores podem ser configurados. Para uma dada variável, uma perturbação P_i para um conjunto de dados i pode ser representada como:

$$P_i = (M_i - m_i) * \varepsilon \quad (3)$$

Após uma perturbação ser gerada, esta é aleatoriamente adicionada ou subtraída ao valor original de uma variável. Excecionalmente, se o valor original for maior que o intervalo atualizado, a perturbação é sempre subtraída, e vice-versa. O valor resultante desta operação é limitado aos valores mínimo e máximo do intervalo para garantir que todas as perturbações são realistas.

3.2.2 Padrão de Combinações

Em relação às variáveis categóricas não correlacionadas entre si, ou seja, que representam características distintas dos dados, a principal restrição intra-variável é manter a sua lista limitada de valores qualitativos. Como a abordagem do padrão de Intervalos não pode ser replicada mesmo que estas variáveis estejam representadas numa forma numérica ou binária, uma solução simples seria registar cada valor qualitativo que possa estar presente numa variável em diferentes classes.

No entanto, o aspeto mais pertinente da geração de perturbações em dados tabulares é a correlação intrínseca entre várias variáveis. Como o valor presente numa variável pode influenciar os valores possíveis noutras variáveis, podem existir várias restrições inter-variável tanto nas numéricas como nas categóricas que estão correlacionadas entre si, ou seja, que representam a mesma característica dos dados. Para melhorar a solução anterior e cumprir os dois tipos de restrições, foi necessário permitir a análise conjunta de várias variáveis, agregando-as de forma otimizada num registo comum de vários valores de várias variáveis que podem estar presentes em diferentes classes.

O padrão das Combinações foi criado com base neste conceito. Este padrão regista automaticamente as combinações de valores de múltiplas variáveis para as diferentes classes do seu conjunto de dados

de treino, ficando preparado para gerar perturbações de forma simultânea em várias variáveis de um novo conjunto de dados (Figura 8). A implementação pode ser configurada com 'variáveis bloqueadas', cujos valores são registados e utilizados para encontrar combinações para outras variáveis, mas sem serem elas próprias perturbadas. Neste padrão, a 'probabilidade de ser aplicada' no intervalo $(0, 1]$ afeta simultaneamente várias variáveis.

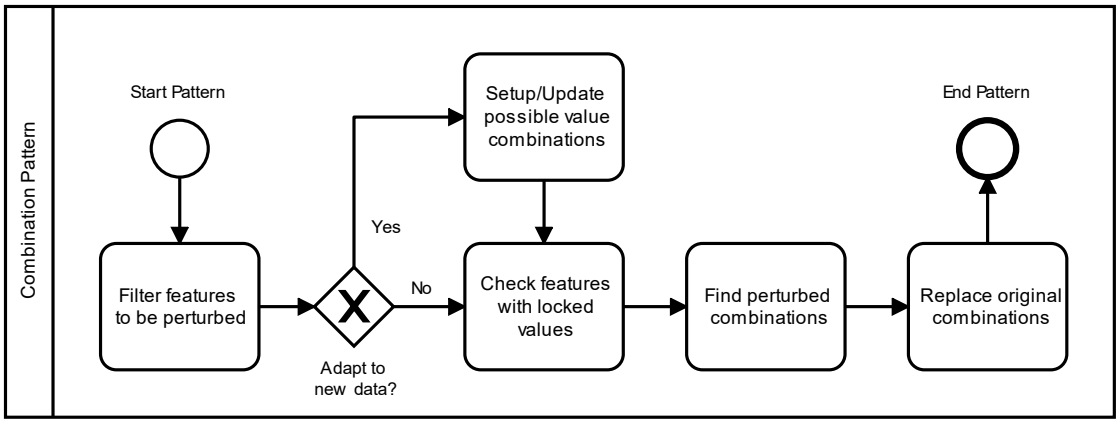

Figura 8. Vista geral do padrão de Combinações.

Para além das combinações registadas inicialmente, novos conjuntos de dados providenciados ao longo do tempo poderão conter novas possibilidades. Estas podem ser acrescentadas às combinações anteriores, ou podem ser utilizadas em atualizações incrementais que retiram as combinações mais antigas e menos relevantes. Para uma dada variável e um valor de inércia $k \in [0, 1]$, o número de combinações atualizadas C_i após treinar com um conjunto de dados i pode ser definido como:

$$C_i = C_{i-1} * k + \text{unique}(x_i) \quad (4)$$

onde $\text{unique}(x_i)$ é o número de combinações únicas originais das amostras x_i do conjunto i.

Cada perturbação criada por este padrão consiste numa escolha aleatória de uma combinação da lista de possibilidades atualizada, considerando as variáveis no estado bloqueado. Esta nova combinação substitui diretamente os valores originais das várias variáveis, garantindo que as perturbações se mantêm sempre realistas.

3.2.3 Sequências de Padrões

Os padrões de Intervalos e de Combinações podem ser agregados de várias formas, sempre seguindo uma ordem sequencial. A principal vantagem de aplicar perturbações consecutivas através de uma sequência de padrões é que esta abordagem permite cumprir restrições inter-variável de maior complexidade, para as quais um único padrão não seria suficiente para preservar validade e coerência. É importante destacar que todos os padrões numa sequência são treinados de forma independente com o seu subconjunto de dados de treino, para prevenir qualquer viés ao analisar as diferentes variáveis. Após o treino, ao aplicar essa sequência durante a geração de exemplos adversos, perturbações cumulativas são criadas automaticamente em novos dados.

Uma exemplificação de como estas sequências são utilizadas é fornecida em baixo, considerando um domínio pequeno, mas com restrições relativamente complexas. O domínio contém três variáveis categóricas nominais, F0, F1 e F2, e duas variáveis numéricas inteiras, F3 e F4. Para um exemplo adverso ser realista neste domínio, deve cumprir as seguintes restrições:

- F0 deve manter sempre o seu valor original;
- F1 e F4 podem ser modificados mas têm valores específicos para cada classe;
- F2 e F3 podem ser modificados mas têm valores específicos para cada classe, que podem ser influenciados por F0 e por F1.

A configuração base de A2PM correspondente a estas restrições contém uma sequência de padrões, onde são especificadas quais as variáveis que cada padrão irá analisar e posteriormente perturbar:

- Padrão de Combinações – Modificar {F1};
- Padrão de Combinações – Modificar {F2, F3}, Bloquear {F0, F1};
- Padrão de Intervalos – Modificar {F3, F4}, Inteiro {F3, F4}.

Esta configuração é aplicada separadamente aos subconjuntos de dados de cada classe, o que leva a que sejam treinadas sequências de padrões diferentes para cada classe. Ao gerar exemplos adversos para uma dada classe, se for utilizada uma 'probabilidade de ser aplicada' de 1.0 para todos os padrões, são efetuadas três perturbações cumulativas (Figura 9). A primeira perturbação criada é trocar F1 com outro valor qualitativo coerente com a classe considerada, de 'B' para 'C'. Depois, sem modificar o F0 original nem o novo F1, uma combinação adequada é encontrada para F0, F1, F2 e F3. Como os valores originais de F2 e F3 eram influenciados por 'A' e 'B', novos valores são encontrados para corresponder a 'A' e 'C'. Finalmente, as variáveis inteiras F3 e F4 são perturbadas de acordo com os seus intervalos de valores possíveis para a classe considerada. Em relação a F3, a perturbação é criada no valor da nova combinação, para se manter coerente com F0 e F1.

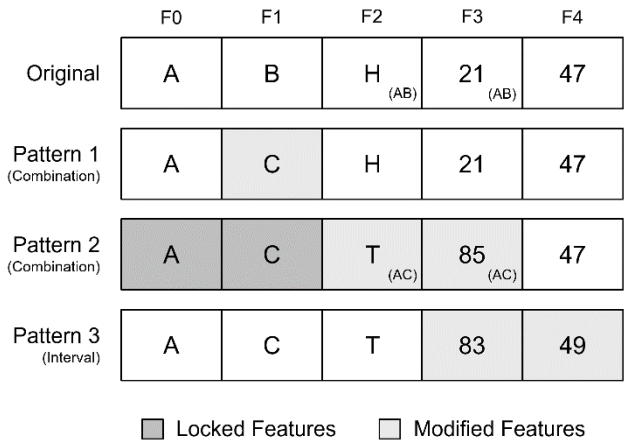

Figura 9. Perturbações cumulativas de uma sequência de padrões.

4 Aplicação Prática

Esta secção apresenta a aplicação da metodologia e da ferramenta desenvolvida a dois cenários de redes de computadores empresariais e redes de dispositivos IoT. É descrita a configuração utilizada, e os resultados obtidos são analisados e discutidos.

4.1 Configuração

Para aplicar a ferramenta desenvolvida e analisar metodicamente a robustez de modelos ML, dois cenários foram considerados: redes de computadores utilizadas em ambientes empresariais e redes de dispositivos inteligentes utilizados em ambientes IoT. Para estes cenários, exemplos adversos foram criados com A2PM utilizando os fluxos de tráfego originais dos conjuntos de dados CIC-IDS2017 [44] e IoT-23 [45], respetivamente. Foi analisado o realismo das perturbações efetuadas ao comparar esses exemplos com os fluxos originais, e também o consumo de tempo da ferramenta para cada iteração de ataque. Para garantir uma análise fidedigna das perturbações efetuadas, as potenciais ferramentas alternativas da literatura atual foram incluídas: JSMA e OnePixel.

Como os mecanismos de um algoritmo de aprendizagem profunda baseado em redes neuronais artificiais e de um algoritmo baseado num conjunto de árvores de decisão são consideravelmente diferentes, os dois tipos de algoritmos foram analisados. A sua suscetibilidade foi avaliada através da realização de ataques de evasão direcionados e não direcionados contra modelos de MLP [46] e RF [47], durante o máximo de 50 iterações. Para além da avaliação de modelos criados através de um treino normal, estes também foram comparados com modelos criados através de treino adverso.

Os ataques foram realizados num computador com equipamento relativamente comum: uma máquina com 16 gigabytes de memória de acesso aleatório, uma unidade de processamento central de 8 núcleos, e uma unidade de processamento gráfico de 6 gigabytes. Foi utilizada sempre a linguagem de programação *Python 3* e as seguintes bibliotecas: *numpy* e *pandas* para preparação e manipulação dos dados, *tensorflow* para os modelos de MLP, *scikit-learn* para os modelos de RF, e *adversarial-robustness-toolbox* para as ferramentas de ataque alternativas.

4.1.1 Preparação dos Dados

Os fluxos de tráfego de rede presentes em CIC-IDS2017 e IoT-23 são bastante úteis para comparar modelos ML por representarem a atividade normal de uma rede e também incluírem várias classes de ciberataques. O primeiro foi disponibilizado pelo Instituto Canadiano para a Cibersegurança, e contém 7 capturas de tráfego num ambiente heterogéneo com a interação de 12 computadores [44]. Por sua vez, o segundo foi criado no *Stratosphere Research Laboratory* e contém 23 capturas de tráfego de dispositivos IoT reais com uma conexão de internet [45]. A Tabela 1 contém as principais caraterísticas das capturas selecionadas, com o nome da classe a ser benigno ou um ciberataque.

Tabela 1. Características dos conjuntos de dados.

Cenário	Conjunto de dados (Capturas)	Total de fluxos	Total por classe	Nome da classe
Rede de computadores empresariais	CIC-IDS2017 (Tuesday, Wednesday)	1.138.612	873.066	Benigno
			230.124	Hulk
			10.293	GoldenEye
			7.926	FTP-Patator
			5.897	SSH-Patator
			5.796	Slowloris
			5.499	Slowhttptest
			11	Heartbleed
Rede de dispositivos IoT	IoT-23 (Capture-1-1, Capture-34-1)	1.031.893	539.587	PortScan
			471.198	Benigno
			14.394	DDoS
			6.714	C&C

Antes de os dados poderem ser utilizados, foram necessários alguns passos de pré-processamento. Primeiro, as variáveis que não tinham informação útil para a classificação do ciberataque, como a data da captura e os endereços de origem e destino, foram removidas. Depois, as variáveis categóricas foram convertidas para valores numéricos através de *One-hot Encoding*. Devido a elevada cardinalidade dessas variáveis, as categorias com uma frequência muito baixa foram agregadas numa única categoria designada como 'outra', para evitar a codificação de valores qualitativos que não estavam presentes em quase nenhuma amostra e, portanto, tinham uma baixa relevância.

A metodologia de análise foi seguida e os conjuntos de dados foram divididos em subconjuntos de treino e teste com 70% e 30% dos fluxos, com estratificação para preservar as proporções das diferentes classes. Uma análise das restrições dos dados foi efetuada para cada cenário, para identificar qual a configuração base mais adequada para A2PM.

Foram identificadas 58 variáveis numéricas e 25 categóricas em CIC-IDS2017. A maior parte das numéricas eram valores contínuos, mas algumas eram variáveis discretas, só podendo ter valores inteiros. Devido à correlação entre as variáveis categóricas codificadas, estas requeriam perturbações combinadas. Adicionalmente, as variáveis relativas ao protocolo de comunicação e ao porto não podiam ser modificadas, para garantir a coerência de cada fluxo com a sua classe de ciberataque. Portanto, a seguinte configuração foi utilizada para o cenário de rede de computadores empresariais, sendo que cada conjunto de variáveis foi convertido para os seus respetivos índices no código:

1. Padrão de Intervalos: Modificar {*variáveis numéricas*}, Inteiro {*variáveis discretas*};
2. Padrão de Combinações: Modificar {*variáveis categóricas*}, Bloquear {porto, protocolo}.

Foram identificadas 8 variáveis numéricas e 34 categóricas em IoT-23, aproximadamente metade do número de variáveis do anterior. Apesar das diferentes variáveis presentes neste conjunto de dados, as restrições foram semelhantes. A principal diferença foi que, para além do protocolo de comunicação, um fluxo também tinha de ser coerente com as variáveis codificadas que representavam o protocolo aplicacional, que foi designado como 'serviço'. A configuração base para o cenário de rede de dispositivos IoT foi:

1. Padrão de Intervalos: Modificar {*variáveis numéricas*}, Inteiro {*variáveis discretas*};
2. Padrão de Combinações: Modificar {*variáveis categóricas*}, Bloquear {porto, protocolo, serviço}.

É importante destacar que as configurações só foram aplicadas às classes de ciberataques, para gerar exemplos compatíveis com os seus propósitos maliciosos. Caso a ferramenta fosse aplicada ao tráfego benigno, apenas geraria novos fluxos benignos que poderiam parecer um ciberataque, o que não seria útil. Nas duas configurações, a 'probabilidade de ser aplicada' foi de 0.6 e 0.4 para os padrões de Intervalos e Combinações, respetivamente. Estes valores foram usados para priorizar um pouco mais as perturbações pequenas de variáveis numéricas individuais em vez das perturbações maiores de várias variáveis categóricas combinadas.

4.1.2 Otimização dos Modelos

Um total de 4 modelos MLP e 4 modelos RF foram criados para classificação multiclasse, um por cenários e por abordagem de treino: treino normal e treino adverso. De acordo com a metodologia de análise proposta, os modelos foram treinados e uma otimização de hiperparâmetros foi realizada utilizando o subconjunto de dados de treino com 70% dos fluxos.

Um modelo MLP [46] é uma rede neuronal artificial com uma camada de entrada, uma camada de saída, e uma ou mais camadas escondidas pelo meio. Cada camada contém múltiplos nós com conexões para os nós da camada seguinte. Quando utilizado para classificação, o número de nós de entrada e de saída são o número de variáveis e de classes, respetivamente. Devido ao elevado custo computacional de treinar um MLP, foi utilizada uma técnica de otimização bayesiana [48], e um subconjunto de validação com 20% do subconjunto de treino. Para prevenir sobreajuste aos dados, foi utilizada uma paragem antecipada do treino quando este estabilizou.

A otimização levou a uma arquitetura com 4 camadas e um número decrescente de nós. As camadas escondidas utilizaram a função de ativação *Rectified Linear Unit* (ReLU) e na técnica de regularização *Dropout*, que ajuda a prevenir o sobreajuste ao ignorar uma certa percentagem dos nós de uma camada durante o treino do modelo. Para a classificação multiclasse, foi utilizada a função de ativação *Softmax* na camada de saída. A arquitetura de MLP para o cenário empresarial foi:

1. Camada de entrada: 83 nós, 512 tamanho de lote;
2. Camada escondida: 64 nós, ativação *ReLU*, 10% *Dropout*;
3. Camada escondida: 32 nós, ativação *ReLU*, 10% *Dropout*;
4. Camada de saída: 8 nós, ativação *Softmax*.

Uma arquitetura semelhante foi utilizada para o cenário IoT, com um número decrescente de nós, apesar de ter apresentado um tamanho de lote menor e uma percentagem de *Dropout* maior:

1. Camada de entrada: 42 nós, 128 tamanho de lote;
2. Camada escondida: 32 nós, ativação *ReLU*, 20% *Dropout*;
3. Camada escondida: 16 nós, ativação *ReLU*, 20% *Dropout*;
4. Camada de saída: 4 nós, ativação *Softmax*.

O resto da configuração de MLP foi comum aos dois cenários por serem tarefas semelhantes de classificação multiclasse de ciberataques. A Tabela 2 sumariza a configuração otimizada.

Tabela 2. Configuração utilizada para Multilayer Perceptron.

Hiperparâmetro	Valor
Objective Loss	Categorical Cross-Entropy
Optimizer	Adam Algorithm
Learning Rate	0.001
Early Stopping	Enabled
Maximum Epochs	50
Class Weights	Balanced

Por outro lado, um modelo RF [47] baseia-se num conjunto de árvores de decisão. Cada árvore individual realiza uma previsão, e a classe mais votada por todo o conjunto é a escolhida como a previsão final do modelo. Este processo é baseado no conceito de que as previsões coletivas de múltiplos modelos de classificação serão melhores do que as previsões de um só.

Como treinar um RF tem um custo computacional bastante menor, o método de *Cross-validation* foi utilizado em conjunto com *Grid Search* para encontrar os melhores hiperparâmetros, a partir de valores normalmente utilizados no domínio NID. Portanto, 5 iterações de treino e validação foram realizadas para cada combinação de hiperparâmetros. A Tabela 3 sumariza a configuração de RF otimizada para os dois cenários.

Tabela 3. Configuração utilizada para Random Forest.

Hiperparâmetro	Valor
Splitting Criterion	Gini Impurity
Number of Trees	100
Maximum Depth of a Tree	32
Minimum Samples in a Leaf	2
Maximum Features	$\sqrt{\text{Número de variáveis}}$

4.2 Resultados e Discussão

Os resultados obtidos em cada cenário foram consolidados, e foi analisado o realismo dos exemplos adversos, os erros de classificação causados pelos ataques direcionados e não direcionados, a robustez dos modelos com uma abordagem de treino normal e adverso, e o consumo de tempo de cada iteração de ataque.

4.2.1 Rede de Computadores Empresariais

No cenário de rede de computadores empresariais, exemplos adversos foram gerados a partir dos fluxos originais do conjunto de dados CIC-IDS2017. Para analisar o realismo dos exemplos gerados por A2PM e pelas ferramentas alternativas, estes foram comparados com os fluxos originais, considerando a funcionalidade pretendida para cada ciberataque. Um fluxo foi escolhido para ser detalhado, através de um número gerado aleatoriamente para prevenir qualquer viés.

O fluxo escolhido era da classe Slowloris, e as perturbações criadas por A2PM aumentaram a duração total do fluxo e o seu IAT, reduzindo o número de pacotes transmitidos por segundo e o seu tamanho. Foi observado que essas modificações foram principalmente focadas na alteração de características temporais do ciberataque para prevenir a sua deteção, sem alterar a funcionalidade pretendida. Portanto, para além de ser tráfego válido que poderia ser transmitido numa rede de computadores real, o exemplo adverso permaneceu coerente com a sua classe.

No entanto, JSMA não conseguiu gerar um exemplo adverso realista para este fluxo. Esta ferramenta criou uma grande inconsistência nas variáveis categóricas codificadas ao atribuiu um único fluxo a dois portos distintos: 80 e 88. Devido às perturbações não restringidas, o valor da variável a representar o porto 88 foi aumentado, sem ter em consideração a sua correlação com a variável do porto 80. Para além da bandeira PSH original para manter a conexão aberta, a bandeira de Finished (FIN) também foi assinalada, o que sinaliza para a conexão terminar e, portanto, contradiz o propósito malicioso do ciberataque. Apesar de também terem sido modificadas duas variáveis numéricas, este exemplo adverso só conseguiu enganar o modelo porque utilizou variáveis categóricas que são incompatíveis com um fluxo de tráfego real.

Da mesma forma, OnePixel também gerou um exemplo que contradiz a classe Slowloris. A variável perturbada foi a que representa a bandeira de Reset (RST), que também sinaliza o término da conexão atual. Como esta ferramenta só perturba uma única variável, escolheu uma que o modelo não aprendeu a detetar, por não ser coerente com as caraterísticas deste ciberataque numa rede de computadores real. Por isso, em dados tabulares, nem JSMA nem OnePixel são alternativas adequadas a A2PM. A Tabela 4 contém as modificações realizadas pelas ferramentas, com '--' a indicar que uma variável manteve o seu valor original.

Tabela 4. Variáveis modificadas num exemplo adverso da classe Slowloris.

Variável	Valor original	Valor de A2PM	Valor de JSMA	Valor de OnePixel
Flow duration	109,034,141	119,046,064	109,034,140	--
Mean flow IAT	13,600,000	19,374,259	--	--
Flow packets per second	0.0825	0.0429	0.0824	--
Mean forward packet length	49.4	48.1	--	--
Min forward segment size	40	36	--	--
Connection flags	'PSH'	--	'PSH' + 'FIN'	'PSH' + 'RST'
Destination port	'80'	--	'80' + '88'	--

Em relação aos ataques direcionados de A2PM, os modelos criados com uma abordagem de treino normal exibiram grandes declínios de desempenho. Apesar de tanto MLP como RF obterem acima de 99% na métrica de classificação *accuracy* no subconjunto de dados de teste normais, uma única iteração de ataque baixou esses resultados em aproximadamente 15% e 33%. Nas iterações seguintes, os fluxos perturbados foram evitando gradualmente a deteção do modelo MLP, mas o modelo RF foi rapidamente enganado e chegou a 0% de *accuracy*. Após 50 iterações, os resultados baixos de ambos os modelos evidenciaram a sua elevada suscetibilidade a exemplos adversos.

Por outro lado, os modelos criados com treino adverso mantiveram resultados substancialmente mais elevados, com muito menos fluxos de ciberataques a serem erradamente classificados como benignos. Esta abordagem de treino com uma perturbação simples em cada fluxo de ciberataque permitiu aos dois modelos aprender a maior parte das possíveis variações desses fluxos. O segundo modelo RF destacou-se por preservar o valor de 99.91% de *accuracy* durante todo o ataque, o que evidenciou a sua excelente melhoria de robustez com treino adverso (Figura 10).

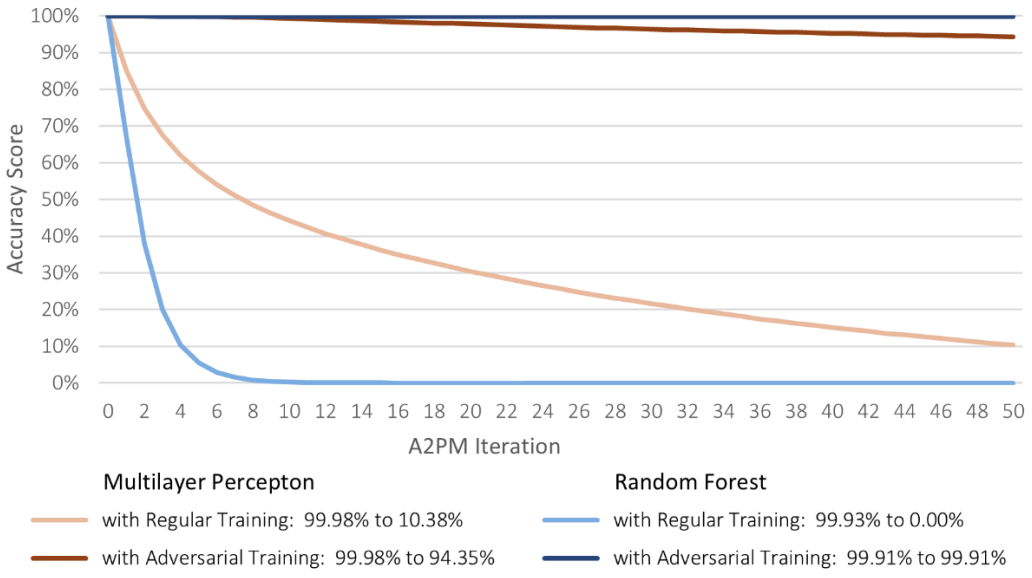

Figura 10. Accuracy num ataque direcionado numa rede Empresarial.

Os ataques não direcionados de A2PM baixaram bastante tanto a métrica de *accuracy* como a de *macro-averaged F1-Score*, com declínios de aproximadamente 99% e 79%, apesar de o modelo RF ter sido na mesma mais afetado nas iterações iniciais. A falta de capacidade dos dois modelos de distinguir entre as diferentes classes de ciberataques corrobora a sua elevada suscetibilidade a exemplos adversos. Contudo, treino adverso permitiu aos modelos manter resultados bastante mais altos, com uma diminuição gradual de menos de 2% por iteração. Apesar de alguns exemplos adversos ainda os enganarem e levarem a previsões incorretas, os dois modelos foram capazes de distinguir cada tipo de ciberataque, mitigando o impacto das perturbações. O modelo RF conseguiu atingir resultados mais altos do que o modelo MLP tanto nos ataques direcionados como nos não direcionados, o que indica que consegue obter uma generalização mais robusta nas redes de computadores empresariais (Figuras 11 e 12).

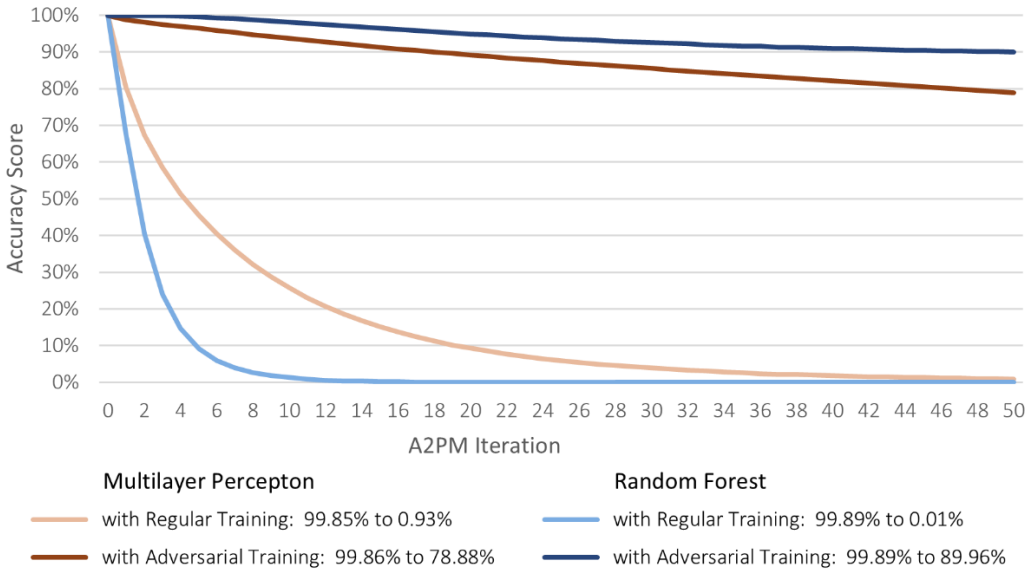

Figura 11. Accuracy num ataque não direcionado numa rede Empresarial.

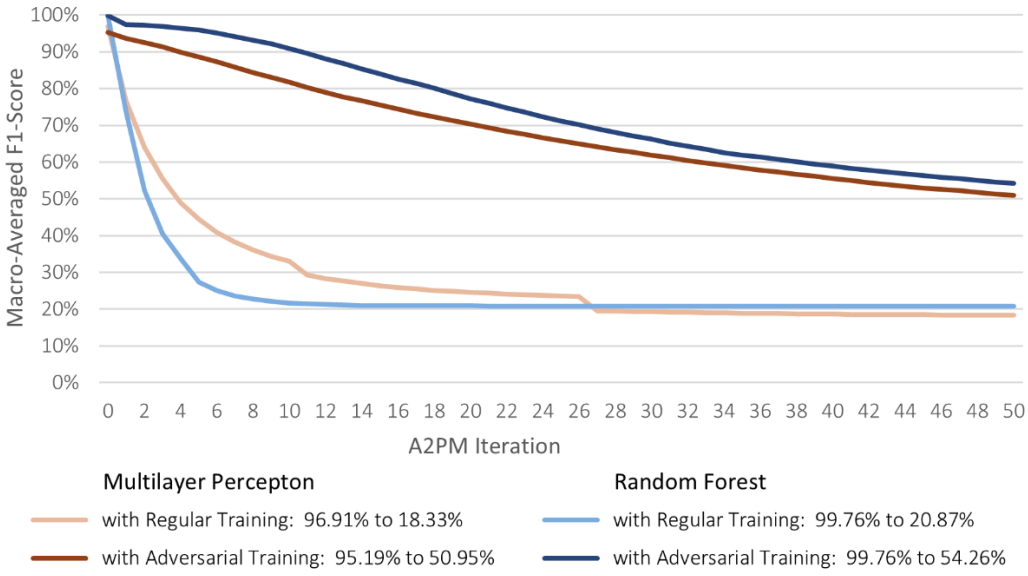

Figura 12. F1-Score num ataque não direcionado numa rede Empresarial.

Para analisar o consumo de tempo de A2PM, o número de milissegundos necessários para cada iteração de ataque foi registado e a sua média foi calculada. Foi tida em consideração a quantidade decrescente de exemplos gerados em cada iteração, conforme o ataque progredia, pelo que o tempo de cada iteração foi dividido pelo número de exemplos adversos gerado. Os ataques foram realizados a um ritmo médio de 10 exemplos adversos por cada 1.7 milissegundos no computador utilizado, o que evidenciou a rápida execução e escalabilidade da ferramenta desenvolvida quando utilizada para ataques de evasão em redes de computadores empresariais.

4.2.2 Rede de Dispositivos IoT

No cenário de rede de dispositivos IoT, exemplos adversos foram gerados a partir dos fluxos originais do conjunto de dados IoT-23. A análise efetuada no cenário anterior foi replicada, começando pelo realismo dos exemplos gerados por A2PM e pelas ferramentas alternativas. Um fluxo foi escolhido para ser detalhado através de outro número gerado aleatoriamente.

O fluxo escolhido era da classe DDoS, o que corresponde a um ciberataque *Distributed Denial-of-Service* realizado por software malicioso capturado no conjunto de dados IoT-23. A2PM substituiu as variáveis categóricas codificadas do estado e histórico da conexão com outra combinação válida, também utilizada por outros fluxos da classe DDoS. Em vez de uma conexão incompleta (OTH) com uma soma de verificação incorreta (BC), tornou-se numa tentativa de conexão (S0) com uma bandeira de sincronização (SYN). As perturbações realizadas neste exemplo adverso permitiram que permanecesse um fluxo de tráfego válido e compatível com a sua classe.

Tal como no cenário anterior, nem JSMA nem OnePixel geraram exemplos realistas. Para além do original OTH, as duas ferramentas aumentaram o valor da variável que representa uma conexão estabelecida com uma tentativa de término (S3). Como um fluxo com estados OTH e S3 simultaneamente não é tráfego válido nem é coerente com o ciberataque, as ferramentas permanecem alternativas inadequadas para A2PM em domínios de dados tabulares. Para além dos estados, JSMA também atribuiu o mesmo fluxo a dois protocolos de comunicação distintos, o que evidenciou a inconsistência das suas perturbações. A Tabela 5 contém as modificações realizadas pelas ferramentas, com '--' a indicar que uma variável manteve o seu valor original.

Tabela 5. Variáveis modificadas num exemplo adverso da classe DDoS.

Variável	Valor original	Valor de A2PM	Valor de JSMA	Valor de OnePixel
Connection state	'OTH'	'S0'	'OTH' + 'S3'	'OTH' + 'S3'
Connection history	'BC'	'SYN'	--	--
Communication protocol	'TCP'	--	'TCP' + 'ICMP'	--

Em relação aos ataques direcionados de A2PM, os declínios de desempenho foram muito mais lentos do que no cenário anterior. O valor de *accuracy* do modelo MLP só começou a ser mais baixo do que 50% na iteração 43, e o do modelo RF estabilizou em aproximadamente 86%. Estes resultados evidenciaram que os dois modelos, e em especial RF, são menos suscetíveis a exemplos adversos direcionados à classe benigna em redes de dispositivos IoT.

Com uma abordagem de treino adverso com uma perturbação simples em cada fluxo de ciberataque, os modelos conseguiram manter resultados ainda mais altos durante os ataques direcionados. Apesar de muitos fluxos perturbados ainda conseguirem evitar a deteção do modelo MLP, o número de fluxos erradamente classificados como benignos pelo modelo RF foi substancialmente mais baixo, o que lhe permitiu manter um valor na métrica de *accuracy* acima de 99%. Portanto, este modelo conseguiu detetar com sucesso a maior parte das variações dos fluxos de ciberataques incluídos neste conjunto de dados (Figura 13).

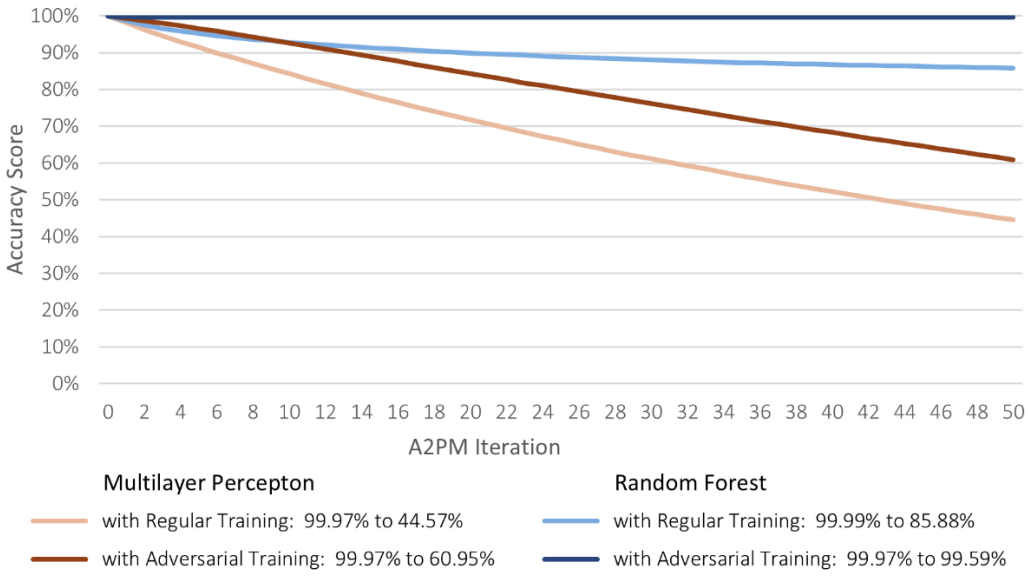

Figura 13. Accuracy num ataque direcionado numa rede IoT.

Os ataques não direcionados de A2PM causaram pequenas diminuições nas duas métricas utilizadas. Apesar de o modelo RF começar a estabilizar a partir da iteração 5, o modelo MLP continuou a decrescer aproximadamente mais 48% na métrica de *accuracy* e 17% na de *macro-averaged F1-Score*. Esta diferença, observada tanto nos ataques direcionados como nos ataques não direcionados, sugere que os modelos RF, e possivelmente os algoritmos baseados em conjuntos de árvores de decisão em geral, são inerentemente mais robustos na deteção de ciberataques em fluxos de tráfego de redes de dispositivos IoT. Ao contrário do cenário anterior, treino adverso não levou a muitas melhorias nos resultados. Não obstante, o subconjunto de treino adverso com maior variedade dos dados contribuiu na mesma para a criação de modelos com uma generalização mais robusta porque estes exibiram menos erros de classificação ao longo dos ataques (Figuras 14 e 15).

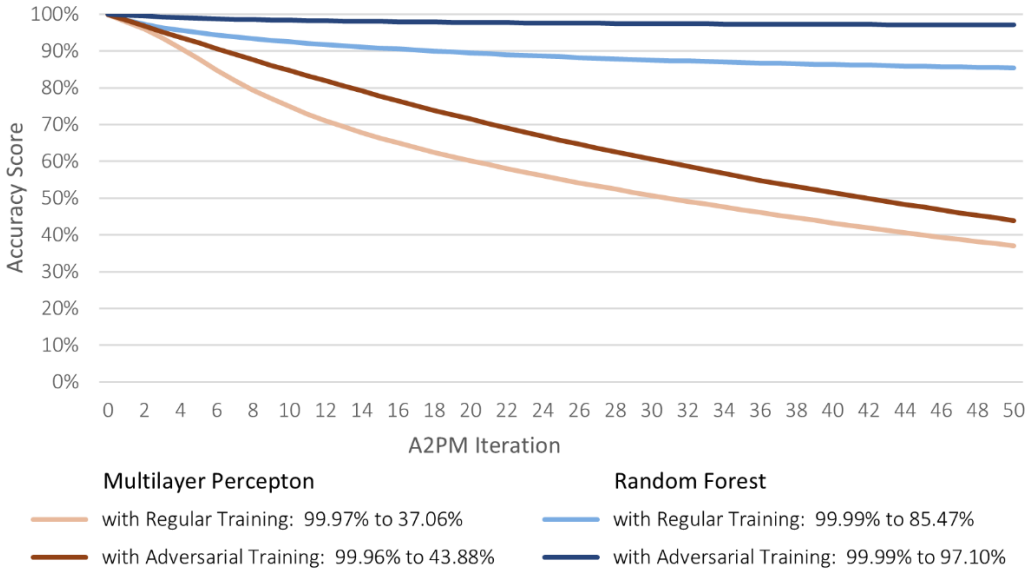

Figura 14. Accuracy num ataque não direcionado numa rede IoT.

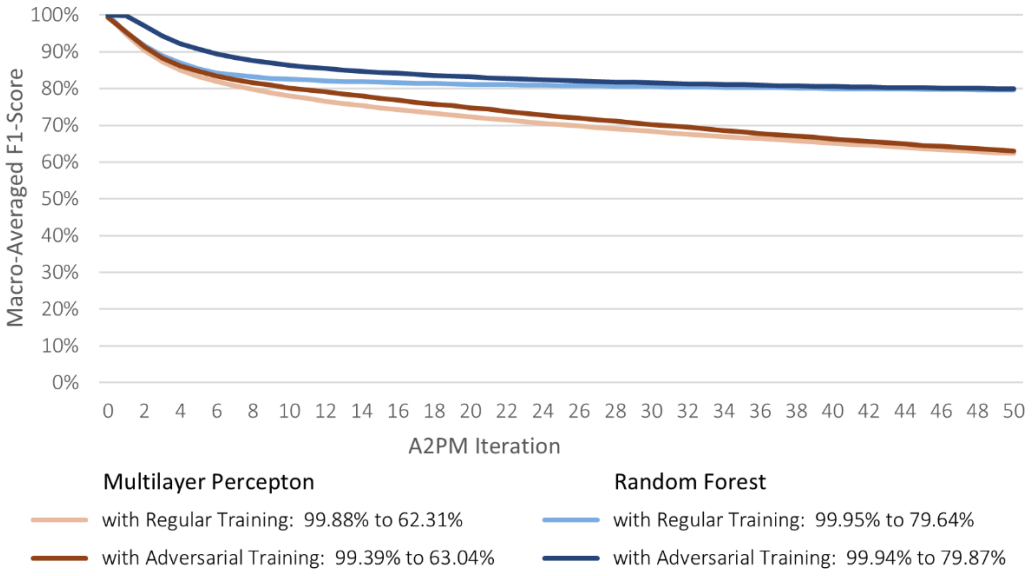

Figura 15. F1-Score num ataque não direcionado numa rede IoT.

Uma análise do consumo de tempo também foi realizada, para verificar a escalabilidade da ferramenta desenvolvida. O número de milissegundos necessários para cada iteração de ataque foi registado e a sua média foi calculada, dividindo pelo número de exemplos gerados em cada iteração. Neste cenário, os ataques foram realizados a um ritmo médio de 10 exemplos adversos por cada 2.4 milissegundos. Ao comparar com o cenário anterior, é possível observar que o tempo foi 41% maior para os fluxos de IoT-23 do que para os de CIC-IDS2017. Apesar de ter aproximadamente metade do número de variáveis, o padrão de Combinações foi configurado para bloquear mais variáveis do protocolo aplicacional 'serviço'. Isto sugere que uma maior complexidade de restrições inter-variável levará a um maior tempo necessário para cada iteração. Mesmo assim, o consumo de tempo foi razoavelmente baixo, o que comprova a rápida execução de A2PM.

5 Conclusões

Este trabalho abordou a falta de robustez dos modelos ML e explorou o realismo dos exemplos adversos no domínio NID, na tarefa de classificação multiclasse de ciberataques. Os três objetivos inicialmente estabelecidos foram cumpridos a 100%. Uma solução foi desenvolvida para melhorar a segurança dos modelos ML nos domínios de dados tabulares complexos, com foco no domínio NID, o que resultou em várias contribuições. Os principais resultados para cada objetivo foram:

- **OB1**: Uma metodologia para uma análise confiável da robustez de múltiplos modelos ML, com um vetor de ataque de evasão adequado para uma rede de computadores real. Foi demonstrado que um ataque adverso bem-sucedido não é garantidamente um ciberataque bem-sucedido, e que é necessário ter em consideração as características e funcionalidades pretendidas de cada classe de ciberataque.
- **OB2**: Uma ferramenta inteligente, A2PM, para a geração de exemplos adversos realistas em domínios de dados tabulares complexos como o domínio NID. A ferramenta tem uma arquitetura modular e pode ser utilizada para ataques adversos, ao causar erros de classificação nos modelos ML de forma iterativa, e para treino adverso, ao melhorar a variedade de um conjunto de dados.
- **OB3**: Duas aplicações práticas da ferramenta a redes de computadores empresariais e a redes de dispositivos IoT, considerando diferentes tipos de modelos ML e de classes de ciberataques. Foram realizadas análises do realismo dos exemplos adversos gerados, dos efeitos dos ataques adversos e do treino adverso no desempenho dos modelos, e do consumo de tempo médio de cada iteração de ataque.

Os resultados obtidos evidenciam as vulnerabilidades dos algoritmos de aprendizagem profunda baseados em redes neuronais artificiais e dos algoritmos baseados em árvores de decisão, e demonstram que esses algoritmos podem beneficiar substancialmente de uma abordagem de treino adverso com perturbações simples. A metodologia e a ferramenta descritas neste trabalho podem ajudar os engenheiros de ML e os profissionais de cibersegurança a melhorar a robustez dos seus modelos com exemplos adversos gerados de forma realista. Desta forma, é possível introduzir uma abordagem de segurança-por-design durante o ciclo de vida de modelos ML.

A ferramenta desenvolvida, A2PM, só gera exemplos completamente realistas quando é providenciada informação sobre a quantidade e os tipos de variáveis utilizadas por um modelo. Esta restrição é uma limitação inerente à forma como são criadas as sequências de padrões de Intervalos e de Combinações, mas também funciona como uma salvaguarda para que A2PM apenas possa ser utilizado por quem tem acesso à informação confidencial sobre o modelo e as features, para evitar ao máximo um uso ilegítimo por verdadeiros atacantes.

Como trabalho futuro, é importante replicar a metodologia proposta em diferentes tipos de redes de comunicação para começar a categorizar as diversas restrições que poderão estar presentes nas redes de diferentes organizações. Ao realizar novas aplicações práticas com mais modelos ML e mais

classes de ciberataques, será possível começar a reduzir o conhecimento necessário para desenvolver uma boa estratégia de defesa, e assim melhorar o desempenho e a confiabilidade dos sistemas de cibersegurança que utilizam modelos ML.

Em relação à capacidade da ferramenta de ser incrementalmente adaptada ao treinar com novos conjuntos de dados, isto só é possível após aprender as características de um conjunto de dados inicial. Se esse conjunto de dados inicial tiver dados não-representativos ou com informação enviesada, a ferramenta não conseguirá gerar exemplos realistas. Portanto, é necessário ter em atenção o primeiro conjunto de dados fornecido, e será relevante continuar a trabalhar na adaptabilidade das sequências de padrões de A2PM para melhorar a transferibilidade dos exemplos para diferentes redes de computadores.

Por último, a conclusão mais importante deste trabalho é: os modelos ML podem ser incrivelmente valiosos para melhorar um sistema de cibersegurança, mas as suas próprias vulnerabilidades não devem ser negligenciadas. É essencial continuar os esforços de investigação para melhorar a segurança e a robustez de ML e dos sistemas inteligentes que utilizam ML.

Referências

[1] European Union Agency for Cybersecurity, N. Christoforatos, I. Lella, E. Rekleitis, C. Van Heurck, and A. Zacharis, "Cyber Europe 2022: After Action Report," 2022. doi: 10.2824/397622.

[2] Verizon, "Data Breach Investigations Report 2022," 2022. Accessed: Dec. 13, 2023. Available: https://www.verizon.com/business/resources/reports/dbir/

[3] IBM Security and Ponemon Institute, "Cost of a Data Breach Report 2022," 2022. Accessed: Dec. 13, 2023. Available: https://www.ibm.com/reports/data-breach

[4] S. Mansfield-Devine, "Sophos: The State of Ransomware 2022," *Comput. Fraud Secur.*, vol. 2022, no. 5, 2022, doi: 10.12968/s1361-3723(22)70573-8.

[5] European Union Agency for Cybersecurity *et al.*, "ENISA Threat Landscape 2022," 2022. doi: 10.2824/764318.

[6] C. Szegedy *et al.*, "Intriguing properties of neural networks," in *Proceedings of the 2nd International Conference on Learning Representations, ICLR 2014*, 2014, pp. 1–10. doi: 10.48550/ARXIV.1312.6199.

[7] European Union Agency for Cybersecurity, A. Malatras, and G. Dede, "AI Cybersecurity Challenges: Threat Landscape for Artificial Intelligence," 2020. doi: 10.2824/238222.

[8] R. S. Siva Kumar *et al.*, "Adversarial Machine Learning-Industry Perspectives," in *2020 IEEE Security and Privacy Workshops (SPW)*, 2020, pp. 69–75. doi: 10.1109/SPW50608.2020.00028.

[9] European Union Agency for Cybersecurity, A. Malatras, I. Agrafiotis, and M. Adamczyk, "Securing Machine Learning Algorithms," 2022. doi: 10.2824/874249.

[10] European Commission and Directorate-General for Communications Networks Content and Technology, "The Assessment List for Trustworthy Artificial Intelligence (ALTAI) for self assessment," Publications Office of the European Union, 2020. doi: 10.2759/002360.

[11] "GECAD Research Group." https://www.gecad.isep.ipp.pt/ (accessed Dec. 09, 2023).

[12] "SeCoIIA 871967 Project." doi: 10.3030/871967.

[13] "VALU3S 876852 Project." doi: 10.3030/876852.

[14] D. Stutz, M. Hein, and B. Schiele, "Disentangling Adversarial Robustness and Generalization," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 6969–6980. doi: 10.1109/CVPR.2019.00714.

[15] A. Fawzi, O. Fawzi, and P. Frossard, "Fundamental limits on adversarial robustness," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015*, 2015.

[16] S. Sabour, Y. Cao, F. Faghri, and D. J. Fleet, "Adversarial Manipulation of Deep Representations," in *Proceedings of the 4th International Conference on Learning Representations, ICLR 2016*, 2016. doi: 10.48550/ARXIV.1511.05122.

[17] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015*, 2015, pp. 1–11. doi: 10.48550/ARXIV.1412.6572.

[18] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Proceedings of the 5th International Conference on Learning Representations, ICLR 2017*, 2017, pp. 1–14. doi: 10.48550/ARXIV.1607.02533.

[19] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples," *arXiv*, 2016, doi: 10.48550/ARXIV.1605.07277.

[20] P. Tabacof and E. Valle, "Exploring the Space of Adversarial Images," in *2016 IEEE International Joint Conference on Neural Networks (IJCNN)*, 2016. doi: 10.48550/ARXIV.1510.05328.

[21] K. Ren, T. Zheng, Z. Qin, and X. Liu, "Adversarial Attacks and Defenses in Deep Learning," *Engineering*, vol. 6, no. 3, pp. 346–360, 2020, doi: 10.1016/j.eng.2019.12.012.

[22] J. Zhang and C. Li, "Adversarial Examples: Opportunities and Challenges," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 31, no. 7, pp. 2578–2593, 2020, doi: 10.1109/TNNLS.2019.2933524.

[23] K. Eykholt *et al.*, “Robust Physical-World Attacks on Deep Learning Visual Classification,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1625–1634. doi: 10.1109/CVPR.2018.00175.
[24] D. Edwards and D. B. Rawat, “Study of Adversarial Machine Learning with Infrared Examples for Surveillance Applications,” *Electronics*, vol. 9, no. 8, 2020, doi: 10.3390/electronics9081284.
[25] X. Yuan, P. He, Q. Zhu, and X. Li, “Adversarial Examples: Attacks and Defenses for Deep Learning,” *IEEE Trans. neural networks Learn. Syst.*, vol. 30, no. 9, pp. 2805–2824, 2019, doi: 10.1109/TNNLS.2018.2886017.
[26] P. Papadopoulos, O. Thornewill von Essen, N. Pitropakis, C. Chrysoulas, A. Mylonas, and W. J. Buchanan, “Launching Adversarial Attacks against Network Intrusion Detection Systems for IoT,” *J. Cybersecurity Priv.*, vol. 1, no. 2, pp. 252–273, 2021, doi: 10.3390/jcp1020014.
[27] M. J. Hashemi, G. Cusack, and E. Keller, “Towards Evaluation of NIDSs in Adversarial Setting,” in *Proceedings of the 3rd ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks*, 2019, pp. 14–21. doi: 10.1145/3359992.3366642.
[28] M. A. Merzouk, F. Cuppens, N. Boulahia-Cuppens, and R. Yaich, “Investigating the practicality of adversarial evasion attacks on network intrusion detection,” *Ann. Telecommun.*, 2022, doi: 10.1007/s12243-022-00910-1.
[29] X. Peng, W. Huang, and Z. Shi, “Adversarial Attack Against DoS Intrusion Detection: An Improved Boundary-Based Method,” in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, 2019, pp. 1288–1295. doi: 10.1109/ICTAI.2019.00179.
[30] A. McCarthy, P. Andriotis, E. Ghadafi, and P. Legg, “Feature Vulnerability and Robustness Assessment against Adversarial Machine Learning Attacks,” in *Proceedings of the 2021 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, 2021, pp. 1–8. doi: 10.1109/CyberSA52016.2021.9478199.
[31] G. Apruzzese, M. Andreolini, L. Ferretti, M. Marchetti, and M. Colajanni, “Modeling Realistic Adversarial Attacks against Network Intrusion Detection Systems,” *Digit. Threat. Res. Pract.*, vol. 1, no. 1, 2021, doi: 10.1145/3469659.
[32] N. Martins, J. M. Cruz, T. Cruz, and P. Henriques Abreu, “Adversarial Machine Learning Applied to Intrusion and Malware Scenarios: A Systematic Review,” *IEEE Access*, vol. 8, pp. 35403–35419, 2020, doi: 10.1109/ACCESS.2020.2974752.
[33] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, “A survey on adversarial attacks and defences,” *CAAI Trans. Intell. Technol.*, vol. 6, no. 1, pp. 25–45, Mar. 2021, doi: 10.1049/cit2.12028.
[34] J. Li, Y. Liu, T. Chen, Z. Xiao, Z. Li, and J. Wang, “Adversarial Attacks and Defenses on Cyber-Physical Systems: A Survey,” *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5103–5115, 2020, doi: 10.1109/JIOT.2020.2975654.
[35] K. He, D. D. Kim, and M. R. Asghar, “Adversarial Machine Learning for Network Intrusion Detection Systems: A Comprehensive Survey,” *IEEE Commun. Surv. Tutorials*, vol. 25, no. 1, pp. 538–566, 2023, doi: 10.1109/COMST.2022.3233793.
[36] M. Pujari, Y. Pacheco, B. Cherukuri, and W. Sun, “A Comparative Study on the Impact of Adversarial Machine Learning Attacks on Contemporary Intrusion Detection Datasets,” *SN Comput. Sci.*, vol. 3, no. 5, p. 412, 2022, doi: 10.1007/s42979-022-01321-8.
[37] R. Chauhan, U. Sabeel, A. Izaddoost, and S. Shah Heydari, “Polymorphic Adversarial Cyberattacks Using WGAN,” *J. Cybersecurity Priv.*, vol. 1, no. 4, pp. 767–792, 2021, doi: 10.3390/jcp1040037.
[38] I. Goodfellow *et al.*, “Generative Adversarial Nets,” in *Advances in Neural Information Processing Systems*, 2014, vol. 27. Available: https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf
[39] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein Generative Adversarial Networks,” in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*, 2017, vol. 70, pp. 214–223. Available: https://proceedings.mlr.press/v70/arjovsky17a.html
[40] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, “The Limitations of Deep Learning in Adversarial Settings,” in *2016 IEEE European Symposium on Security and Privacy*, 2016, pp. 372–387. doi: 10.1109/EuroSP.2016.36.

[41] J. Su, D. V. Vargas, and K. Sakurai, “One Pixel Attack for Fooling Deep Neural Networks,” *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 828–841, 2019, doi: 10.1109/TEVC.2019.2890858.
[42] T. Shorey, D. Subbaiah, A. Goyal, A. Sakxena, and A. K. Mishra, “Performance Comparison and Analysis of Slowloris, GoldenEye and Xerxes DDoS Attack Tools,” *2018 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI 2018*, pp. 318–322, 2018, doi: 10.1109/ICACCI.2018.8554590.
[43] Z. Al-Qudah, M. Rabinovich, and M. Allman, “Web Timeouts and Their Implications,” in *Passive and Active Measurement*, 2010, pp. 211–221.
[44] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization,” in *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, 2018, pp. 108–116. doi: 10.5220/0006639801080116.
[45] S. Garcia, A. Parmisano, and M. J. Erquiaga, “IoT-23: A labeled dataset with malicious and benign IoT network traffic.” Zenodo, Jan. 2020. doi: 10.5281/zenodo.4743746.
[46] F. Murtagh, “Multilayer perceptrons for classification and regression,” *Neurocomputing*, vol. 2, no. 5, pp. 183–197, 1991, doi: 10.1016/0925-2312(91)90023-5.
[47] L. Breiman, “Random forests,” *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.
[48] J. Snoek, H. Larochelle, and R. P. Adams, “Practical Bayesian Optimization of Machine Learning Algorithms,” in *Advances in Neural Information Processing Systems*, 2012.