F-NIDS – Sistema de Detecção de Intrusão descentralizado com base em Aprendizado Federado

Jonathas A. de Oliveira[1], Rodolfo I. Meneguette[2], Vinícius P. Gonçalves[1], Rafael T. de Sousa Jr.[1], Daniel L. Guidoni[3], José C. M. Oliveira[4], Geraldo P. Rocha Filho[4]

[1]Universidade de Brasília – UnB – Brasília – DF – Brasil

[2]Universidade de São Paulo – USP – São Carlos – SP – Brasil

[3]Universidade Federal de Ouro Preto – UFOP, Ouro Preto – MG – Brasil

[4]Universidade Estadual Do Sudoeste Da Bahia – UESB – Vitória da Conquista – BA – Brasil

jonathas.oliveira@aluno.unb.br, {vpgvinicius, desousa}@unb.br,

meneguette@icmc.usp.br, guidoni@ufop.edu.br,

{jcarlos, geraldo.rocha}@uesb.edu.br,

Abstract. *The coming of IoT networks introduced new scalability and security challenges due to the massive number of connections and higher data transferring rate in these networks. Although there have been efforts in recent years to mitigate these effects, there are still questions to be investigated, such as data privacy and scalability in distributed IoT scenarios. This work proposes the F-NIDS is an intrusion detector that uses federated artificial intelligence and differential privacy techniques, combined with asynchronous communication between system entities, aiming for scalability and data confidentiality. F-NIDS has an architecture proposal to allow usage in cloud or fog IoT environments. Results have shown that: the confidential detection model, used on F-NIDS, keeps satisfactory performance metrics and, in the event of an attack, predicts and determines the nature.*

Resumo. *O advento das redes de IoT introduziu novos desafios de escalabilidade e segurança devido ao grande número de conexões e maior taxa de transferência de dados nessas redes. Embora tenha havido esforços nos últimos anos para mitigar esses efeitos, ainda há perguntas a serem investigadas, como privacidade de dados e escalabilidade em cenários de IoT distribuídos. Este trabalho propõe que o F-NIDS é um detector de intrusão que usa a inteligência artificial federada e técnicas de privacidade diferencial, combinadas com a comunicação assíncrona entre entidades do sistema, visando escalabilidade e confidencialidade dos dados. O F-NIDS possui uma proposta de arquitetura para permitir o uso em ambientes de IoT em nuvem ou em fog. Os resultados mostraram que: o modelo de detecção confidencial do F-NIDS apresenta métricas satisfatórias de desempenho e, no caso de um ataque, prever e determinar satisfatoriamente a sua natureza.*

1. Introdução

Desde a década passada vislumbra-se um aumento considerável da interconexão entre humanos, máquinas e serviços. O que resultou no paradigma de comunicação da *IoT - Internet of Things* [Rahman and Asyhari 2019, Geraldo Filho et al. 2019]. Preocupações, tais como escalabilidade, latência e privacidade das informações, têm sido levantadas dentro do contexto da IoT [Habibzadeh et al. 2019]. Arquiteturas descentralizadas podem trazer algumas soluções para essas questões, oferecendo maior disponibilidade e uma escalabilidade superior [Roman et al. 2013, Cavalcante et al. 2022]. No quesito segurança, as estratégias mais populares usadas em redes de IoT compreendem a utilização dos NIDS – *Network Intrusion Detection Systems*. Esses sistemas são responsáveis por determinar e alertar se uma atividade em particular é normal ou maliciosa na rede [Chaabouni et al. 2019].

Embora exista uma vasta quantidade de trabalhos sobre a questão da privacidade no contexto de ML – *Machine Learning* –, ainda há poucas aplicações em um cenário real de utilização [Chen et al. 2020]. No aspecto da descentralização, FL – *Federated Learning* – é um paradigma recentemente proposto para permitir a distribuição das tarefas de ML com maior preservação da privacidade dos dados de treino, essa técnica tem demonstrado uma ampla gama de aplicações, principalmente em que confidencialidade é um aspecto importante [Zhu et al. 2021].

Nesse contexto, FL é uma técnica de ML que permite criar arquiteturas descentralizadas de aprendizado. Possibilita a integração com mecanismos de DP – *Differential Privacy* –, mitigando algumas das limitações de NIDS já mencionadas. Nessa técnica, muitos clientes colaborativamente treinam um modelo separadamente, sob a orquestração de um servidor central. O FL possibilita que os dados de treinamento permaneçam descentralizados e restritos na origem, de maneira que não haja qualquer compartilhamento dos mesmos [McMahan et al. 2017]. Nessa técnica, o servidor central pode receber somente os pesos e gradientes dos modelos treinados pelos clientes [Zhu et al. 2021]. A DP garante que qualquer versão de um conjunto de dados estatísticos permanece igualmente verossímil independente de conter ou não um item em particular [Dwork 2006], minimizando a possibilidade de inferência de informações individuais sensíveis, mas mantendo as propriedades estatísticas do conjunto de dados.

Diferentes abordagens para *NIDS's* descentralizados foram propostos com avanços no campo [Al-Yaseen et al. 2017] [M. et al. 2019]. Essas soluções abordam o uso de multiagentes com aprendizado de máquina e focam principalmente em questões de performance e escalabilidade. Outras propostas de *NIDS* descentralizado, que exploram com mais ênfase a questão da privacidade e confiabilidade dos dados, foram investigados em [Tabassum et al. 2021]. Apesar de serem arquiteturas voltadas para a privacidade dos dados, o primeiro trabalho admite um ponto central de vulnerabilidade. Já em [Tabassum et al. 2021] utiliza-se *Distributed Machine Learning - DMS* para agregação e comunicação entre os agentes e permite transferência de dados brutos para o agente central [Qian et al. 2020], sendo, portanto, uma limitação na proteção da privacidade. A utilização de aprendizado federado também vem sendo avaliado para a construção de detectores de intrusão [Zhao et al. 2020], porém sem o foco nas exigências específicas das redes *IoT*.

Este trabalho vai além, superando as limitações citadas anteriormente e propõe o

F-NIDS, um sistema de detecção de intrusão descentralizado. O sistema é baseado em FL, e tem como objetivo oferecer uma arquitetura distribuída, isenta a necessidade de troca de dados dos clientes, na medida que dispensa transmissão de informações individuais dos mesmos. E, para garantir a proteção dos modelos, utiliza técnicas de DP, possibilitando uma transmissão mais confidencial de modelos entre os clientes. O F-NIDS contém as seguintes características principais: (i) apresenta uma alta acurácia de classificação de tráfego de dados num cenário de IoT distribuído; (ii) dotado de uma arquitetura descentralizada que permita escalar facilmente; e (iii) garante um maior sigilo tanto dos dados de treinamento como também dos modelos treinados.

O restante do artigo está organizado da seguinte forma. A Seção II apresenta os artigos relacionados. A Seção III descreve o F-NIDS. A Seção IV apresenta os resultados por meio da avaliação do desempenho. A Seção V finaliza o artigo com a conclusão e os trabalhos futuros.

2. Trabalhos relacionados

No campo dos *NIDS's* descentralizados, diferentes soluções foram propostas [Al-Yaseen et al. 2017] [M. et al. 2019]. Essas soluções abordam o uso de multiagentes com aprendizado de máquina e focam principalmente em questões de performance e escalabilidade. A utilização de aprendizado federado também vem sendo avaliado para a construção de detectores de intrusão [Zhao et al. 2020], porém se limitam em analisar dados provenientes de redes convencionais, sem o enfoque nas características específicas das redes *IoT* e na confidencialidade dos modelos. No entanto, além da falta de mecanismos de garantia da confidencialidade dos modelos resultantes, os trabalhos citados possuem limitações no que diz respeito a sua escalabilidade para atender redes altamente distribuídas, como explorado nesta pesquisa [Chaabouni et al. 2019].

Outras soluções de NIDS descentralizados, que exploram com mais ênfase a questão da privacidade e confiabilidade dos dados, foram investigadas. Em [Tabassum et al. 2021] foi utilizado o método DMS – *Distributed Machine Learning* – para agregação e transmissão dos modelos. Apesar dar uma ênfase para a proteção dos dados de treinamento, tal proposta admite que haja a armazenamento de dados de treinamento em um agente central, o que, de acordo com [Qian et al. 2020], pode representar ameaças à privacidade das informações. Além disso, para garantir a privacidade dos modelos, redes neurais generativas foram utilizadas. No entanto, como demonstrado em [Salem et al. 2020], tal técnica de ML também pode também ser vulnerável às violação de confidencialidade, especificamente a ataques de inferência dos dados de treinamento. Portanto outras técnicas de NIDS descentralizados podem resultar em uma melhor proteção de dados de treino e dos modelos.

Em [Ruzafa-Alcazar et al. 2021], é proposto um *NIDS* federado que privilegia a confidencialidade, utilizando técnicas de DP. Essa é a solução que mais se assemelha ao F-NIDS. Os autores adotaram o algoritmo de agregação dos modelos `Fed+`. Esse algoritmo é apontando como sendo capaz de prover uma maior acurácia do modelo global, reduzindo os efeitos da perturbação causada pela DP, e na mitigação de percas causadas pela própria agregação em si. Embora proponha um NIDS descentralizado, o cenário trata apenas de um subconjunto de aplicações de IoT (i.e., aplicações industriais), não avaliando a sua generalização para outras aplicações de IoT, tais como, por exemplo,

carros autônomos, cidades inteligentes e em *datacenters* no geral. Além disso, algumas limitações do algoritmo de agregação são levantados, no que tange a questão da capacidade de personalização dos agentes e também na robustez dos modelos gerados [Kundu 2021]. Vale ressaltar que [Ruzafa-Alcazar et al. 2021] utilizaram, na avaliação dos resultados, uma quantidade demasiadamente reduzida de clientes, o que pode ter impactado nos números obtidos, sendo uma estratégia que pode ser inadequada para testar um cenário real de aprendizado federado, em que se prevê algumas dezenas ou até mesmo milhões de clientes. Um estudo mais robusto, com maior número de clientes, pode ser realizado para validar NIDS utilizando aprendizado federado.

A avaliação desses trabalhos permitiu concluir que ainda há problemas em aberto nesse campo. Dentre esses problemas, está a ausência de uma proposta arquitetural distribuída capaz de atender diversas cargas de trabalho. O outro problema, e não menos importante, diz respeito a confidencialidade das informações, dessa maneira, há espaço para uma proposta que atenda com maior ênfase a questão da privacidade dos dados, além dos modelos treinados colaborativamente. Nesse sentido, o F-NIDS foi proposto visando oferecer soluções para os problemas mencionados.

3. F-NIDS - Federated Network Intrusion Detection System

Esta seção apresenta o F-NIDS, um sistema de detecção de intrusão federado com mecanismos de DP e *publish/subscribe*. O F-NIDS protege a confidencialidade dos dados de treino e do modelo contra ataques de inferência de membros. A seguir será apresentado a visão geral, o funcionamento e ferramentas usadas para a modelagem do F-NIDS.

3.1. Visão geral

A arquitetura geral do F-NIDS é demonstrada na Figura 1. Na figura, o agente central (label AC) gera os pesos globais (label G) iniciais, fazendo a sua propagação para os demais membros do sistema. O AC tem a função de agregar pesos e produzir um modelo global por meio de uma técnica de agregação de pesos locais e também realizar a orquestração, propagando os pesos obtidos nessa agregação. As tarefas de detecção, recepção dos pesos globais e treinamento de um modelo local é feito feito pelos Agentes de detecção (label DA). Esses DA's, além de treinar um modelo local usando um *dataset* individual, recebem as requisições de detecção por meio de um sub-sistema auxiliar denominado *Message Broker –MB–* , que recebe mensagens, armazena e roteia para um ou mais destinatários de maneira assíncrona. Os clientes enviam as requisições de detecção por meio desse mecanismo, publicando em filas internas do MB, permitindo assim ganhos de escalabilidade e confiabilidade. Para tanto, a tarefa de captura de pacotes é feita pelos próprios clientes individualmente. Para proporcionar maior robustez contra ataques de inferência nos modelos, cada DA vem dotado de uma camada adicional de segurança utilizando DP. Ao final, o AC produz um modelo global que herda algumas propriedades de DP dos demais modelos locais. Essa arquitetura se baseia em três mecanismos: (i) mecanismo descentralizado de treinamento usando FL; (ii) mecanismo de treino com DP e (iii) mecanismo de detecção descentralizado e distribuído.

3.2. Mecanismo de aprendizado federado com privacidade

Embora o F-NIDS seja distribuído, o agente central gera o modelo inicial com parâmetros aleatórios e realiza a orquestração do treinamento do modelo e a transmissão

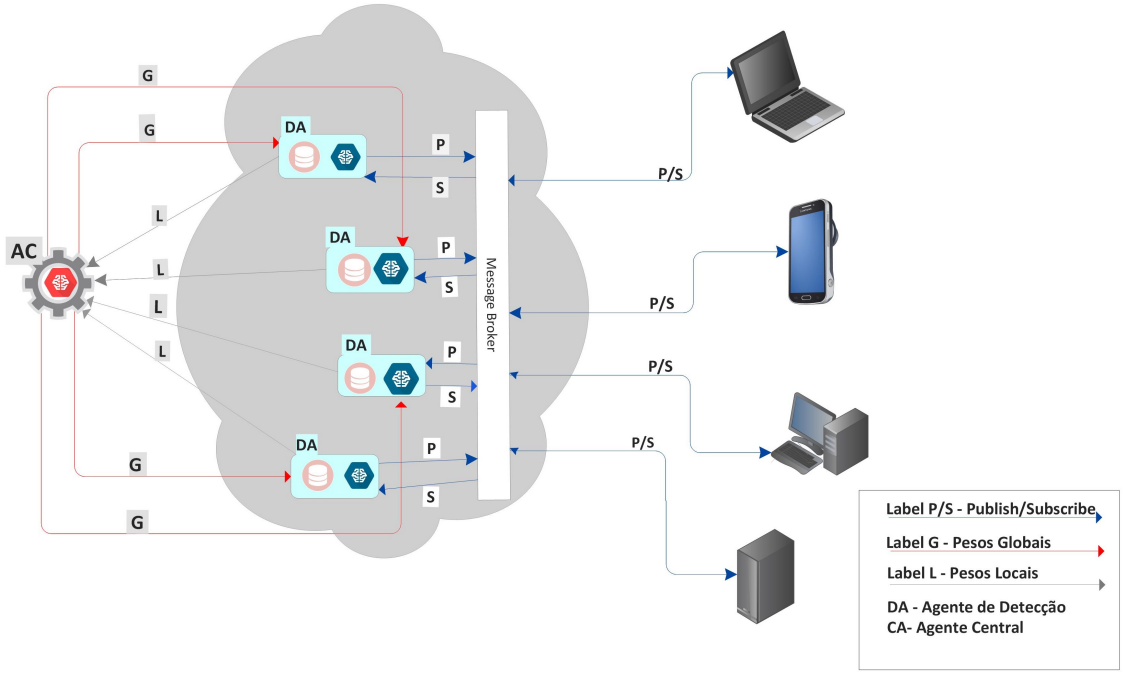


Figure 1. Arquitetura geral de funcionamento do F-NIDS.

desses parâmetros. Essa orquestração é realizada por meio de variáveis denominadas hiperparâmetros federados. Tais hiperparâmetros (dispostos na Tabela 1) definem as configurações usadas nos classificadores treinados. Quando todas as condições definidas nos hiperparâmetros forem atendidas, é iniciada uma rodada federada em que é aplicado o algoritmo de agregação de pesos, denominado `FedAvg` – Federated Averaging –. A rodada federada conclui quando encerra-se a propagação dos parâmeros do modelo federado para os agentes de detecção.

No `FedAvg` [McMahan et al. 2017] o agente $k_t \in K$ (sendo $n_k = |\mathcal{P}_k|$, com $\mathcal{P}_k$ os índices do *dataset* contido no agente k_t e com um $C = 1$ correspondendo ao dataset completo e η a taxa de aprendizagem) computa o vetor de gradientes $g_k = \nabla F_k(w_t)$ correspondente ao treinamento local do modelo w_t. Então atribui-se aos próprios agentes a tarefa de atualizar os pesos localmente $k_t \leftarrow k_t - \eta \nabla F_k(k_t)$ várias vezes antes da etapa de agregação, que continua sendo realizada pelo AC. Nessa estratégia, o custo computacional é controlado por três hiperparametros: C, sendo a fração de agentes que realizam os cálculos em cada rodada; E o número de épocas de cada agente no seu *dataset* local; e B o tamanho no *minibatch* usado por cada cliente. Dessa maneira o `FedAvg` possui o valor de $B = \infty$ (tamanho do minibatch igual ao tamanho do dataset local) e $E = 1$. O AC agrega esses gradientes e aplica a atualização $w_{t+1} \leftarrow w_t - \eta \nabla f(w_t)$, pois $\nabla f(w_t) = \sum_{k=1}^{K} \frac{n_k}{n} g_{t+1}^k$.

O Algoritmo 1 descreve esse processo de treinamento dos modelos de classificação do F-NIDS. Consiste de duas etapas, a etapa global (das linhas 1 a 9) e a etapa local (das linhas 10 a 16). Na linha 1 é inicializado um vetor de pesos globais. Na linha seguinte, itera-se sobre cada rodada e internamente calcula-se a quantidade de

Algorithm 1 `FederatedAveraging`. Os *K* agentes são indexados por *k*; *B* é o tamanho do minibatch local, *E* é o número de épocas locais, e η é a taxa de aprendizagem.

O agente central executa:

1: initialize w_0
2: **for** each round t = 1, 2, ... **do**
3: $m \leftarrow \max(C \cdot K, 1)$
4: $S_t \leftarrow$(random set of m clients)
5: **for** each client $k \in S_t$ **in parallel do**
6: $w_{t+1}^k \leftarrow$DetectionAgentUpdates(k, w_t)
7: **end for**
8: $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
9: **end for**

DetectionAgentUpdates(k, w)**:** ▷ //Run on detection agent k
10: $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
11: **for** each local epoch i from 1 to E **do**
12: **for** batch $b \in \mathcal{B}$ **do**
13: $w \leftarrow w - \eta \nabla \ell(w; b)$
14: **end for**
15: **end for**
16: return w to central agent

clientes que vão participar do treinamento em uma determinada rodada. Na quinta linha é feita a iteração sobre alguns dos DA's registrados no F-NIDS, passando como argumento os pesos do modelo global (linha 6) e obtendo um vetor de pesos locais de cada agente. Ao se obter os pesos locais de um determinado agente, o resultado é indexado em um vetor contendo cada um desses pesos. Na linha 8 é feita a agregação propriamente dita de todos os pesos dos modelos treinados localmente, por cada DA. Daí então os pesos globais são atualizados para serem usados na rodada seguinte. Já a etapa local começa na linha 10 em diante, com o treino dos modelos locais nos agentes. A etapa se inicia com dados de treino locais sendo divididos em um conjunto B de mini-lotes de exemplos. Nas linhas 11 e 12, itera-se sobre cada uma das épocas i dos agentes. Para cada uma dessas épocas, todos os mini-lotes $b \in B$ são utilizados como argumento para se computar os pesos locais do agente atual (linha 13), além dos pesos globais e a taxa de aprendizagem do agente em questão. A etapa local finaliza quando um conjunto de pesos locais w do modelo são obtidos e repassados para o agente central.

3.3. Mecanismo de privacidade diferencial do classificador

Para assegurar a maior confidencialidade dos modelos treinados, é usado o algoritmo `DP-SGD` – *Differentially Private - Stochastic Gradient Descent* –. Essa abordagem é a versão clássica do algoritmo de otimização de modelos SGD, mas com a inclusão de inclusão da DP. Esse algoritmo limita a sensibilidade de cada gradiente. Consegue esse resultado, por meio do *clipping* e perturbação do gradiente durante o treino, visando amplificar e monitorar o custo da privacidade [Bu et al. 2020]. Seja $clip_c : g_t(x_i) \in \mathbb{R}^p \rightarrow g_t(x_i)/max(1, \frac{\|g_t(x_i)\|_2}{C}) \in \mathbb{R}^p$ a função

de *clipping* que re-escala os valores de entrada de maneira que o resultado tem a máxima norma ℓ_2 de C. Assim, o passo de atualização pelo algoritmo DP-SGD é dado por: $w^{(t+1)} = w^{(t)} - \eta_t\{\frac{1}{B}\sum_{i\in\mathbb{B}_t} clip_c(\nabla_{w_t}\mathcal{L}(w_t, x_i)) + \delta\}$, e $\delta \sim \mathcal{N}(0, \sigma^2C^2I)$ é a variável aleatória correspondente a DP gaussiana e σ^2 o desvio padrão da perturbação [Abadi et al. 2016].

O pseudocódigo completo do `DP-SGD` é apresentado no Algoritmo 2. A linha 1 contém a lista de hiperpârametros utilizados no treinamento que são os exemplos de treino (i.e., taxa de aprendizagem, escala de pertubação σ, tamanho de L e o limite da norma C). Na linha 2 um conjunto de pesos iniciais são inicializados randomicamente. A partir da linha 3 o algoritmo itera sobre cada época. Na linha 4, em cada época, uma amostra L_t é selecionada e para cada subconjunto i de L_t computa o vetor de gradientes $g_t(x_i)$. Na linha 8 é feito o *clipping* do vetor de gradientes obtido, em seguida acrescentando a pertubação gaussiana na linha 9 e por fim ajustando os pesos sinápticos do modelo em função do gradiente obtido e da taxa de aprendizagem.

Algorithm 2 `DP-SGD`. Differentially private SGD

1: **Input:** Exemplos $\{x_1, ..., x_N\}$, função de custo $\mathcal{L}(w) = \frac{1}{N}\sum_i \mathcal{L}(w, x_i)$. Parametros: taxa de aprendizagem η_t, escala de perturbação σ, tamanho do sub-grupo L, limite da norma do gradiente C e T a quantidade de épocas.
2: **Inicialize** w_0 randomicamente
3: **for** $t \in [T]$ **do**
4: Escolha um amostra randômica L_t com probabilidade amostral $\frac{L}{N}$
5: **for** $i \in L_t$ **do** ▷ `//Obtenção do gradiente`
6: $g_t(x_i) \leftarrow \nabla_{w_t}\mathcal{L}(w_t, x_i)$
7: **end for**
8: $\bar{g}_t(x_i) \leftarrow clip_c(g_t(x_i))$ ▷ `//Clipping`
9: $\tilde{g}_t \leftarrow \frac{1}{L}\sum_i(\bar{g}_t(x_i) + \mathcal{N}(0, \sigma^2C^2I))$ ▷ `//Pertubação`
10: $w_{t+1} \leftarrow w_t - \eta_t\tilde{g}_t$ ▷ `//Ajuste`
11: **end for**
12: **Retorna** w_T e computa o custo geral de privacidade (ϵ, δ) usando o método de contabilização.

3.4. Mecanismo de detecção

O F-NIDS divide em três etapas o processo de detecção de intrusão: A captura; a classificação e as contramedidas. A captura é feita diretamente pelos clientes, utilizando mecanismo interno de captura. A detecção é realizada pelos DA's, mediante publicação e subscrição em filas do MB. Contramedidas podem ser realizadas em conjunto pelos agentes e pelos dispositivos baseados em regras próprias, permitindo que cada um dos membros do sistema possam implementar sua política individual de repúdio contra agentes maliciosos.

A interação proposta pelo F-NIDS para a interação entre um cliente, interessado em detectar um determinado pacote, e um DA é ilustrado na Figura 2. O cliente inicia o processo capturando um pacote e verificando se o emissor está na lista de repúdio. Caso o emissor não esteja previamente bloqueado, o cliente envia uma requisição de classificação para o DA. É importante ressaltar que o cliente não sabe qual será o DA responsável por

detectar a intrusão, nem sua localização, pois o MB desacopla as partes. No entanto, o MB garante que a comunicação terá uma resposta assíncrona, através do mecanismo *publish/subscribe*. O classificador, ao predizer como benigno, notifica o cliente interessado. Caso o pacote seja categorizado como malicioso, o F-NIDS irá notificar o cliente inicial e emitirá um alerta a todos os demais clientes que subscreveram na fila de alertas do MB. Essa notificação contém a probabilidade da classificação feita, qual classe de ataque foi detectado e a origem que emitiu o pacote. O clientes, ao receberem notificações de ataque, ou um alerta, inclui a origem na lista de repúdio e encerra quaisquer conexões com o emissor do pacote malicioso.

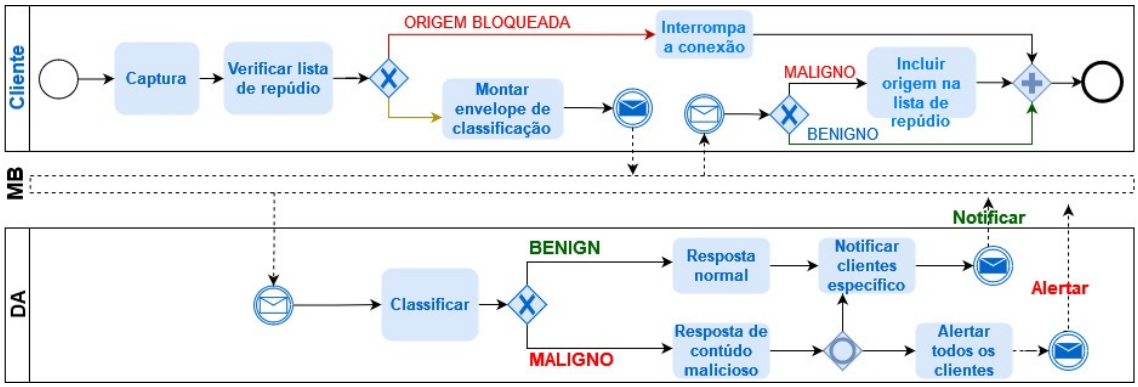


Figure 2. Operação básica do mecanismo de detecção.

4. Metodologia e resultados

Nesta seção é apresentada a avaliação de desempenho do F-NIDS. O F-NIDS é avaliado comparando-se as métricas de performance de predição multi-classe e na predição binária, frente aos outros três métodos estudados. Tal estratégia tem como objetivo atestar que o F-NIDS possui viabilidade, em contraposição aos outros métodos dispostos no presente trabalho, em termos de capacidade de detecção satisfatória de ataques, a despeito de aplicar dois algoritmos que penalizam a acurácia. Ou seja, que a aplicação dos dois métodos incluídos no F-NIDS resultam num impacto pouco significativo nas métricas de predição observadas.

A performance dos métodos foi avaliada utilizando, além da acurácia, a precisão e o *recall*. A precisão (P_c) é descrita na Equação 1 e é a medida de performance do classificador na detecção de uma determinada classe c. Sendo (TP_c) os exemplos corretamente classificados como sendo c. Já (FP_c) são os exemplos incorretamente classificados como sendo da classe c. Enquanto que o recall R_c é a razão descrita na Equação 2, em que FN_c são os exemplos classificados incorretamente como sendo de outra classe que não a classe c. Ambas as métricas são computadas classe a classe.

$$P_c = \frac{TP_c}{TP_c + FP_c} \tag{1}$$

$$R_c = \frac{TP_c}{TP_c + FN_c} \tag{2}$$

Quatro métodos de treinamento de classificadores foram avaliados comparativamente: O método centralizado –ANN–; o centralizado com DP –ANN-DP–; o método de treino aplicando somente o algoritmo federado – FED – e o método federado com DP –F-NIDS–. Foram treinados dez modelos individuais para cada método, sendo cada modelo avaliado sobre uma fração randômica de 10% do dataset de teste. Na Seção 4.1, com o objetivo de obter o desempenho geral multi-classe e binária nos quatro métodos, a acurácia global será analisada no decorrer das rodadas de treinamento. Essa estratégia tem por objetivo observar em qual rodada cada método atinge os níveis relativos de convergência. A seção 4.2 trata avaliação dos resultados de precisão e *recall* obtidos, visando verificar o desempenho de classificação em cada uma das classes individualmente.

O dataset usado é baseado no NF-ToN-IoT-v2 [1]. Esses dados foram produzidos a partir dos arquivos `.pcap` do dataset ToN-IoT e então processados para gerar dados no padrão da ferramenta NetFlow [2] [Sarhan et al. 2022]. Produzindo uma base de dados com 43 *features* relevantes. Foram removidos as informações de endereço do emissor e do destinatário. O dataset resultante é desbalanceado e possui 2.5×10^6 exemplos extraídos randomicamente, a partir dos dados originais, o que corresponde a aproximadamente 14,75% do volume de dados contidos no NF-ToN-IoT-v2. Desses dados resultantes, 80% foram separados para o conjunto de treinamento e 20% destinado ao conjunto de testes. O experimento utilizou as bibliotecas Tensorflow [3] para treinamento da rede neural, TF Privacy [4] para privacidade diferencial e Flower [5] para o aprendizado federado. A Tabela 1 apresenta a lista de hiperpârametros utilizados nos treinamentos dos modelo. Os hiperpârametros do perceptron multicamadas foram obtidos usando o método de *hiperparameter tuning*, o tamanho do *minibatch* foi o maior suportado pela GPU utilizada (NVIDIA V100 com 16GB de GDDR5). Os hiperparâmetros relacionados com o aprendizado federado foram escolhidos levando-se em consideração a CPU e RAM disponíveis (8 núcleos e 24GB de RAM). Outros valores de hiperparâmetros foram tentados, porém resultaram em baixa performance ou em um tempo excessivo para que o classificador demonstrasse sinais de convergência.

4.1. Resultados de acurácia do F-NIDS

A Figura 3 apresenta os resultados de acurácia obtido nos quatro métodos em estudo (ANN, ANN-DP, FED e F-NIDS), em função das rodadas de treinamento. Analisando comparativamente os resultados da Figura 3a, o método ANN possui em média 7,7% a mais em acurácia que o método F-NIDS. Quando comparada a acurácia do método ANN e os demais, as diferenças foram, respectivamente, 0,06% sobre o FED e de 3% sobre o ANN-DP. Tais resultados estão dentro do esperado, pois o algoritmo de *FedAvg* exige um custo adicional em termos de acurácia do classificador, visto que precisa agregar os pesos obtidos por modelos locais treinados com um número significativamente reduzido de exemplos. Há que se considerar também o efeito do algoritmo *DP-SGD*, que impacta significativamente a acurácia do classificador por incluir as perturbações no modelo. Com base nos números apresentados, é possível concluir que as diferenças de performance

[1] `https://staff.itee.uq.edu.au/marius/NIDS_datasets/`

[2] `https://www.cisco.com/c/pt_br/tech/quality-of-service-qos/netflow/index.html`

[3] `https://www.tensorflow.org`

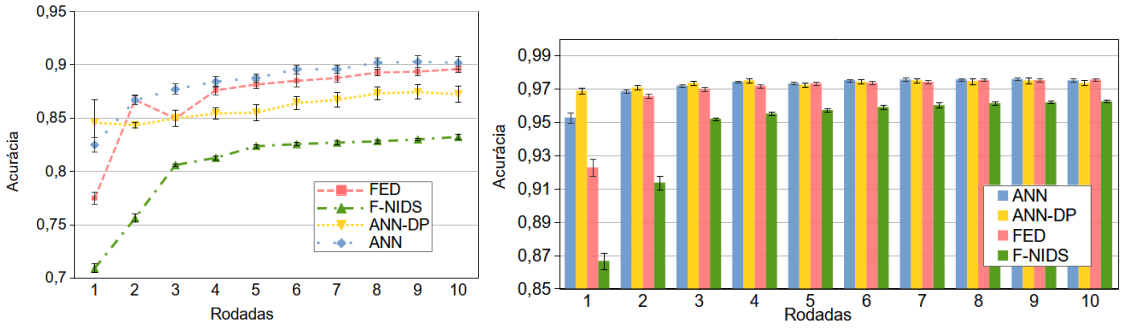
[4] `https://www.tensorflow.org/responsible_ai/privacy/guide`

[5] `https://flower.dev/`

Table 1. Relação de hiperparâmetros utilizados no treinamento dos modelos

Parâmetro	Valor padrão
Neurônios na camada oculta	160
Taxa de aprendizagem	0,02
Épocas	10
Rodadas	10
Tamanho do mini-lote	1000
Conjunto de validação	20%
Norma L_2	1.5
Perturbação σ	0.5
Fração miníma de DA's de treino	0,1
Fração miníma de DA's de avaliação	0,1
Mínimo de DA's de treino	10
Mínimo de DA's disponíveis	75

multi-classe são pouco significantes entre o método F-NIDS e todos os demais métodos estudados.



(a) Acurácia multi-classe (b) Acurácia binária

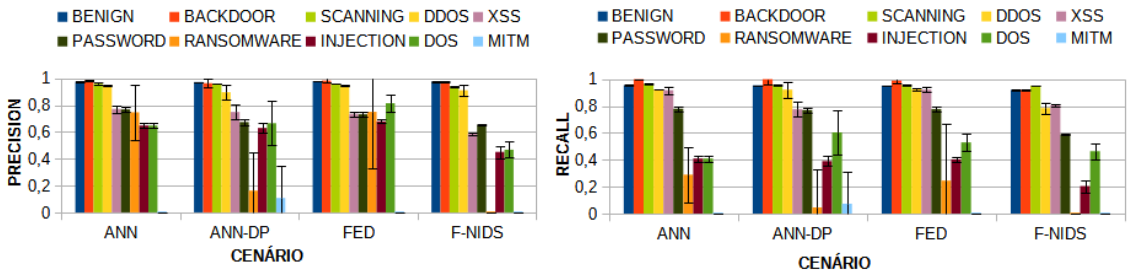
Figure 3. Resultados globais dos métodos por rodada

A Figura 3b apresenta os resultados da acurácia binária do classificador durante as dez rodadas de treinamento. Tais resultados são obtidos agrupando-se as classes de ataque em uma única classe. Nesse caso, é possível notar que a acurácia de detecção entre trafego normal ou malicioso apresenta diferenças ainda mais reduzidas. O impacto na acurácia do F-NIDS, em relação aos demais métodos, pode ser medida pela diferença entre as acurácias dos demais métodos e a acurácia do F-NIDS. Nesse caso, verificou-se que o impacto na acurácia binária média dos métodos ANN, ANN-DP e FED, em relação ao método F-NIDS é de apenas 1,2%, 1,1% e 1,3%, respectivamente. Portanto, não foram observadas alterações relevantes de performance entre os quatro métodos, dentro desse contexto. Tais resultados permitem concluir que as perdas de acurácia binária, decorrentes da aplicação algoritmos *FedAvg* e *DP-SGD* em conjunto, resultam em impactos pouco significativos de acurácia binária dos modelos resultante.

4.2. Avaliação da performance do F-NIDS por classe

Na Figura 4, é apresentado a precisão e *recall* obtidos em cada classe usando o método de classificação do F-NIDS, comparado os métodos ANN, ANN-DP e FED. A Figura 4a

apresenta especificamente a precisão, ou proporção de acertos dentre as previsões feitas, demonstrando a capacidade do modelo classificar corretamente uma classe. Na Figura 4b é apresentado o *recall* de cada classe, separada pelos quatro métodos em estudo. Essa métrica é a proporção de acertos dentre as classes reais. A precisão de predições *Benign, Backdoor, Sacanning e DDOS* não apresentam diferenças significativas do F-NIDS com os demais métodos. Nas demais classes (XSS, Password, Ransomware, Injection, DOS e MITM), diferenças perceptíveis e significativas podem ser observadas. Vale ressaltar a baixa precisão obtida na classificação de *Ramsomware e MITMT*. Aparentemente esse comportamento está associado à quantidade de exemplos, que é inferior ao tamanho do mini-lote escolhido como parâmetro L para o Algoritmo *DP-SGD*. No método F-NIDS, as classes DOS e Injection performaram significativamente abaixo dos demais métodos, embora contassem com um número de exemplos de treino comparativamente próximo aos de classes com melhor precisão. Esses resultados permitem concluir que a aplicação do algoritmo *FedAvg*, em conjunto com o *DP-SGD*, apresentou impactos pouco significativos na precisão e *recall* multi-classe.

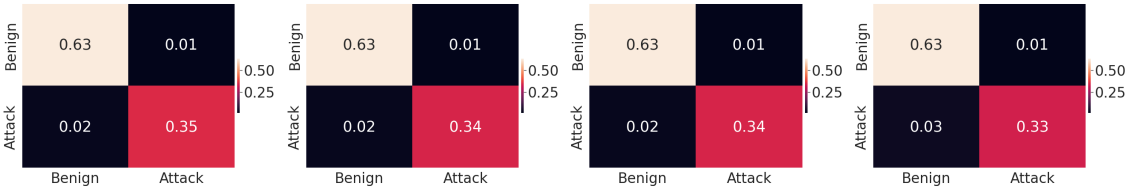


(a) Precisão (b) Recall

Figure 4. Resultados de precisão e recall multi-classe

4.3. Eficiência binária do F-NIDS

A avaliação dos resultados por classe permite registrar a eficiência dos classificadores em detectar cada uma das classes individualmente, porém predizer se um determinado pacote é passível de ser corretamente classificado como benigno ou malicioso é um atributo primordial para o F-NIDS. Isto posto, as classes de ataque foram agrupadas em apenas uma classe, denominada *Attack*, resultando duas classificações possíveis (i.e: benign; attack).



(a) ANN (b) ANN-DP (c) FED (d) F-NIDS

Figure 5. Matrizes de confusão binária dos classificadores.

A Figura 5 apresenta as matrizes de confusão de cada um dos métodos, mostrando o cruzamento entre os exemplos classificados como benignos ou maliciosos e os seus

valores reais correspondentes. O método F-NIDS (Figura 5d) obteve um resultado semelhante aos demais nas detecções corretas de tráfego benigno, o que também contribuiu para manter uma quantidade reduzida de falsos positivos. Já nas detecções de ataques, o F-NIDS apresentou uma quantidade de detecções apenas ligeiramente inferior aos demais métodos, representados pelas Figuras 5a, 5b e 5c. Além disso, os ataques verdadeiros e incorretamente classificados como benignos pelo método F-NIDS tiveram pouca diferença sobre os demais métodos. Tais resultados demonstram que o método F-NIDS é praticamente tão eficiente quanto os métodos ANN (Figura 5a), ANN-DP (Figura 5b) e FED (Figura 5c) para detectar pacotes benignos e quase tão eficiente para detectar corretamente os ataques que as outras abordagens mencionadas. A Tabela 2 apresenta numericamente os resultados das métricas obtidas, para cada método estudado. Contém as médias das dez observações e os erros representados pelo seu desvio padrão.

Table 2. Avaliação das métricas de classificação binária dos métodos

Método	**Métricas**		
	Precisão	**Recall**	**Acurácia**
ANN	$0.975 \pm 0{,}005$	$0{,}955 \pm 0{,}005$	$0{,}974 \pm 0$
ANN-DP	$0.971 \pm 0{,}002$	$0{,}954 \pm 0{,}003$	$0{,}973 \pm 0{,}001$
FED	0.981 ± 0	$0{,}980 \pm 0$	$0{,}974 \pm 0$
F-NIDS	$0{,}885 \pm 0{,}28$	$0{,}970 \pm 0{,}264$	$0{,}962 \pm 0$

Quando comparado os efeitos do algoritmo *DP-SGD* sobre performance de classificação de ataques, por meio da comparação do método ANN (Figura 5a) com o ANN-DP (Figura 5b), é possível notar que esse algoritmo influencia levemente na capacidade de detectar ataques verdadeiros. Considerando todas as possibilidades, pode-se concluir que o algoritmo *DP-SGD*, com os hiperparâmetros de treinos utilizados, resulta em penalizações pouco significativas na eficácia binária do classificador. Comportamento semelhante foi verificado na avaliação do efeito do algoritmo *FedAvg* sobre o classificador, por meio da comparação dos métodos ANN (Figura 5a) e FED (Figura 5c). Nesses casos, verificou-se a vantajem dos métodos centralizados, sobre os federados, somente na identificação dos ataques verdadeiros.

As métricas obtidas nos experimentos apresentaram resultados condizentes com as hipóteses estabelecidas. Ainda sim poderia se obter métricas de performance ainda mais elevadas, através de um menor nível de perturbação gaussiana, ou ainda pela diminuição do número de DA's utilizados. Embora seja tentador realizar tais diminuições, essa abordagem pode reduzir a eficiência dos modelos na defesa contra ataques de inferência de membros ou inversão de modelos. Portanto torna-se importante estabelecer um equilíbrio entre performance e robustez, equilíbrio esse que é ponderado principalmente pelas necessidades da organização que irá utilizar o F-NIDS como sua ferramenta de detecção de intrusão.

5. Conclusão

Essa pesquisa mostrou que um dos principais desafios para a IoT é a detecção descentralizada e escalável de intrusão nessas redes. Apesar das tentativas para a realização de detecções precisas em um ambiente de IoT, ainda haviam pontos passiveis de serem

melhorados no que diz respeito aos aspectos de escalabilidade, confidencialidade e confiabilidade aos que já havia sido proposto. Os resultados mostraram que o F-NIDS possui métricas de acurácia, precisão e recall semelhantes às métricas obtidas em estratégias tradicionais que utilizam apenas o aprendizado centralizado. Considerando os resultados obtidos, o F-NIDS se mostra efetivo para: (i) detectar eficientemente eventuais ataques e determinar de maneira satisfatória qual dos ataque está sendo realizado, dentre uma lista de ataques mais comuns; (ii) treinar modelos de maneira distribuída mantendo a privacidade dos dados de treinamento e (iii) ser dotado de uma arquitetura descentralizada de classificação que permita sua utilização em redes IoT das mais diversas aplicações, assegurando a escalabilidade e confiabilidade do sistema. Nos trabalhos futuros, serão analisados os melhores hiper-parâmetros de treino com DP que maximizam a segurança ao passo que mantém bons níveis de performance de classificação. Os modelos serão submetidos a ataques de inferência de membros e os resultados analisados visando estabelecer qual configuração de DP é a mais vantajosa dentro de cenários de IoT a serem previamente estabelecidos nos estudos subsequentes.

Agradecimentos: Geraldo P. Rocha Filho agradece à FAPESP (processo 2021/06210-3) e ao CNPq (304264/2022-9, PQ-2) por financiarem seus projetos de pesquisas. Rafael T. de Sousa Jr. tem apoio do CNPq Outorga 310941/2022-9 PQ-1D, da FAPDF Outorga 625/2022 SISTeR City e da Universidade de Brasília Outorga FUB/COPEI 7129. Daniel L. Guidoni agradece à FAPESP (2020/05126-6) e FAPEMIG (APQ-02675-21) por financiarem seus projetos de pesquisas.

References

[Abadi et al. 2016] Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep learning with differential privacy. page 308–318.

[Al-Yaseen et al. 2017] Al-Yaseen, W. L., Othman, Z. A., and Nazri, M. Z. A. (2017). Real-time multi-agent system for an adaptive intrusion detection system. *Pattern Recognition Letters*, 85:56–64.

[Bu et al. 2020] Bu, Z., Dong, J., Long, Q., and Su, W. (2020). Deep Learning With Gaussian Differential Privacy. *Harvard Data Science Review*, 2(3). https://hdsr.mitpress.mit.edu/pub/u24wj42y.

[Cavalcante et al. 2022] Cavalcante, I. C., Meneguette, R. I., Torres, R. H., Mano, L. Y., Gonçalves, V. P., Ueyama, J., Pessin, G., Amvame Nze, G. D., and Rocha Filho, G. P. (2022). Federated system for transport mode detection. *Energies*, 15(23):9256.

[Chaabouni et al. 2019] Chaabouni, N., Mosbah, M., Zemmari, A., Sauvignac, C., and Faruki, P. (2019). Network intrusion detection for iot security based on learning techniques. *IEEE Communications Surveys Tutorials*, 21(3):2671–2701.

[Chen et al. 2020] Chen, H., Hussain, S. U., Boemer, F., Stapf, E., Sadeghi, A. R., Koushanfar, F., and Cammarota, R. (2020). Developing privacy-preserving ai systems: The lessons learned. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–4.

[Dwork 2006] Dwork, C. (2006). Differential privacy. In Bugliesi, M., Preneel, B., Sassone, V., and Wegener, I., editors, *Automata, Languages and Programming*, pages 1–12, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Geraldo Filho et al. 2019] Geraldo Filho, P., Villas, L. A., Gonçalves, V. P., Pessin, G., Loureiro, A. A., and Ueyama, J. (2019). Energy-efficient smart home systems: Infrastructure and decision-making process. *Internet of Things*, 5:153–167.

[Habibzadeh et al. 2019] Habibzadeh, H., Nussbaum, B. H., Anjomshoa, F., Kantarci, B., and Soyata, T. (2019). A survey on cybersecurity, data privacy, and policy issues in cyber-physical system deployments in smart cities. *Sustainable Cities and Society*, 50:101660.

[Kundu 2021] Kundu, A. (2021). Fed+: A unified approach to robust personalized federated learning.

[M. et al. 2019] M., R., Ahmed, M., and Khan, R. (2019). An adaptive distributed intrusion detection system architecture using multi agents. *International Journal of Electrical and Computer Engineering (IJECE)*, 9:4951.

[McMahan et al. 2017] McMahan, B., Moore, E., Ramage, D., Hampson, S., and Arcas, B. A. y. (2017). Communication-Efficient Learning of Deep Networks from Decentralized Data. In Singh, A. and Zhu, J., editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR.

[Qian et al. 2020] Qian, B., Su, J., Wen, Z., Jha, D. N., Li, Y., Guan, Y., Puthal, D., James, P., Yang, R., Zomaya, A. Y., Rana, O., Wang, L., Koutny, M., and Ranjan, R. (2020). Orchestrating the development lifecycle of machine learning-based iot applications: A taxonomy and survey. *ACM Comput. Surv.*, 53(4).

[Rahman and Asyhari 2019] Rahman, M. A. and Asyhari, A. T. (2019). The emergence of internet of things (iot): Connecting anything, anywhere. *Computers*, 8(2).

[Roman et al. 2013] Roman, R., Zhou, J., and Lopez, J. (2013). On the features and challenges of security and privacy in distributed internet of things. *Computer Networks*, 57(10):2266–2279. Towards a Science of Cyber Security Security and Identity Architecture for the Future Internet.

[Ruzafa-Alcazar et al. 2021] Ruzafa-Alcazar, P., Fernandez-Saura, P., Marmol-Campos, E., Gonzalez-Vidal, A., Hernandez Ramos, J. L., Bernal, J., and Skarmeta, A. F. (2021). Intrusion detection based on privacy-preserving federated learning for the industrial iot. *IEEE Transactions on Industrial Informatics*, pages 1–1.

[Salem et al. 2020] Salem, A., Sautter, Y., Backes, M., Humbert, M., and Zhang, Y. (2020). Baaan: Backdoor attacks against autoencoder and gan-based machine learning models.

[Sarhan et al. 2022] Sarhan, M., Layeghy, S., and Portmann, M. (2022). Towards a standard feature set for network intrusion detection system datasets. *Mobile Networks and Applications*, 27(1):357–370.

[Tabassum et al. 2021] Tabassum, A., Erbad, A., Mohamed, A., and Guizani, M. (2021). Privacy-preserving distributed ids using incremental learning for iot health systems. *IEEE Access*, 9:14271–14283.

[Zhao et al. 2020] Zhao, R., Yin, Y., Shi, Y., and Xue, Z. (2020). Intelligent intrusion detection based on federated learning aided long short-term memory. *Physical Communication*, 42:101157.

[Zhu et al. 2021] Zhu, H., Zhang, H., and Jin, Y. (2021). From federated learning to federated neural architecture search: a survey. *Complex & Intelligent Systems*, 7.