DFedForest: Floresta Federada Descentralizada

Lucas Airam C. de Souza, Gabriel Antonio F. Rebello, Gustavo F. Camilo, Lucas C. B. Guimarães, Otto Carlos M. B. Duarte

Grupo de Teleinformática e Automação (GTA)
Universidade Federal do Rio de Janeiro (UFRJ)

Resumo. *A eficácia dos sistemas de aprendizado de máquina depende fortemente da relevância dos dados que são empregados no treino. Usualmente, os dados são sensíveis e privados, pois são coletados de dispositivos e sensores usados no dia a dia das pessoas. A Lei Geral de Proteção de Dados (LGPD) coloca em risco a abordagem atual, pois regula o processamento remoto centralizado de dados sensíveis coletados de forma distribuída. Este artigo propõe um sistema de aprendizado de máquina distribuído baseado em algoritmos locais de floresta aleatória criados com árvores de decisão compartilhadas através da corrente de blocos. Os resultados mostram que a abordagem proposta iguala ou supera os resultados obtidos com o emprego de florestas aleatórias apenas com dados locais. Além disso, a proposta aumenta a detecção de novos ataques quando os domínios possuem distribuições de ameaça diferentes.*

1. Introdução

A quantidade e a qualidade dos dados utilizados para treinar o modelo são cruciais para os sistemas de aprendizado de máquina. Logo, os ambientes tradicionais de aprendizado de máquina centralizam o armazenamento de dados de diversas fontes em um domínio para o treino e a execução de algoritmos. No entanto, o armazenamento centralizado apresenta problemas de privacidade, pois requer que os "proprietários dos dados" confiem em quem centraliza os dados e na sua capacidade de evitar vazamentos de informações privadas [Bao et al. 2019]. Além disso, a centralização requer um grande poder computacional para processar o alto volume de dados gerados pelos pontos de coleta, pois o número de dispositivos conectados e quantidade de informações crescem a cada dia.

Uma forma eficiente de se garantir a privacidade é criptografar os dados. No entanto, a criptografia tradicional não permite o processamento dos dados, o que impede um aprendizado global mais eficaz através de dados encriptados. Recentemente, foi proposta a criptografia homomórfica que permite algumas operações de soma e produto em dados encriptados. Porém, criptografia homomórfica é extremamente complexa além requerer um volume de dados muito grande para o dado criptografado e para a chave criptográfica [Pisa et al. 2012].

A centralização e a crescente preocupação com a privacidade e a Lei Geral de Proteção de Dados fizeram surgir medidas alternativas distribuídas para a criação e armazenamento de modelos de aprendizado de máquina.

Este trabalho foi realizado com recursos do CNPq, CAPES, FAPERJ e FAPESP (18/23292-0, 2015/24514-9, 2015/24485-9 2014/50937-1).

Uma versão em inglês baseada neste artigo intitulada "DFedForest: Decentralized Federated Forest"deve ser submetida a congressos ou revistas internacionais.

A proposta de aprendizado federado [McMahan et al. 2016] substitui o envio de amostras privadas pelo compartilhamento de modelos de aprendizado de máquina criados localmente a partir das amostras. A proposta utiliza a arquitetura cliente-servidor, na qual os clientes utilizam dados locais para atualizar um modelo global e enviam o resultado da sua computação ao servidor. Dessa forma, os domínios compartilham apenas o resultado da computação local, que é menos sensível do que os dados privados.

As soluções distribuídas que procuram resolver a centralização e a privacidade devem se preocupar com a veracidade dos dados, uma vez que os proprietários dos dados locais podem se comportar maliciosamente, compartilhando resultados incorretos com a finalidade de prejudicar o modelo resultante. Assim, monitorar o comportamento dos colaboradores é essencial para identificar participantes maliciosos, e com isso, criar um modelo final acurado.

Este artigo propõe o DFedForest, um sistema de aprendizado federado para a criação distribuída de florestas aleatórias que possui as seguintes características: i) criação de modelos de aprendizado de máquina simples, baseados em conjuntos de árvores de decisão, de forma colaborativa e mais acurados que os modelos locais, ii) não transmissão de dados locais, evitando perda de privacidade, iii) transmissão de modelos, em vez dos dados, desidentificados para garantir um certo nível de segurança e evitar a complexa criptografia homomórfica e iv) utilização da tecnologia de corrente de blocos para garantir confiança mútua e para registrar de forma distribuída as referências dos endereços dos modelos locais, evitando que participantes maliciosos prejudiquem a acurácia do modelo.

O artigo avalia o desempenho da proposta de aprendizado de máquina distribuído através de um protótipo de detecção de intrusão em redes de computadores. O protótipo foca o problema de detecção de ataques por intermédio de dispositivos infectados por código de redes de robôs (*botnets*). O protótipo utiliza a plataforma Hyperledger Fabric e a biblioteca scikit-learning. A experimentação utiliza o conjunto de dados CTU-13 [Garcia et al. 2014] que possui amostras de tráfego normal e malicioso geradas por redes de robôs.

Os resultados da avaliação de desempenho mostram que a acurácia de uma floresta aleatória utilizando o sistema proposto supera os resultados obtidos quando o modelo de aprendizado é apenas local, sem levar em consideração dados de modelos colaborativos. Quando comparada ao modelo centralizado, a proposta distribuída limita o volume de informações transmitidas, pois os modelos de aprendizado de máquina sintetizam a grande quantidade de amostras empregadas para o treino na estrutura de dados do algoritmo utilizado.

O restante do artigo é organizado da seguinte forma. A Seção 2 exibe o estado da arte em sistemas de aprendizado de máquina e técnicas para preservação da privacidade dos dados utilizados. A Seção 3 apresenta a arquitetura proposta, analisando a segurança do sistema e a complexidade computacional. A Seção 4 detalha o desenvolvimento do protótipo, além de uma análise dos resultados obtidos. Por fim, a Seção 5 conclui o artigo e apresenta as direções futuras.

2. Trabalhos Relacionados

A coleta de dados de todo tipo e em diversos lugares tem sido responsável por grande parte do sucesso de diversas aplicações modernas relacionadas a cidades inteli-

gentes, edifícios inteligentes, saúde eletrônica entre outras. Estes dados são coletados, enviados para grandes centros de dados e processados por algoritmos de aprendizado de máquina. Mas a crescente preocupação com a privacidade e os frequentes escândalos de vazamentos de dados estão ameaçando esta prática de sucesso. Hoje, há um forte redirecionamento das pesquisas para o aprendizado de máquina distribuído e para soluções que mantenham os dados na sua origem em vez de transmiti-los para grandes centros de dados centralizados.

Giacomelli *et al.* propõem a criação de florestas aleatórias de modo distribuído [Giacomelli et al. 2019]. Vaidya *et al.* [Vaidya et al. 2013] pesquisam como criar e compartilhar modelos baseados em árvores de decisão aleatórias [Fan et al. 2003] de forma colaborativa. Em ambos os casos, os autores utilizam técnicas de criptografia para garantir a privacidade e a segurança dos modelos resultantes. A criptografia é bem complexa e introduz latência. O uso de criptografia homomórfica [Giacomelli et al. 2019] ou diferencial [Rana et al. 2015] garantem a privacidade dos dados em ambientes distribuídos ao mesmo tempo que permitem operações simples de soma e produto. Entretanto, a criptografia homomórfica gera grandes chaves criptográficas [Pisa et al. 2012] e sobrecarga computacional. Além disso, a privacidade diferencial dificulta a criação de modelos acurados devido à introdução de ruídos nos dados utilizados.

A Floresta Federada [Liu et al. 2020] é um algoritmo de aprendizado federado que centraliza o gerenciamento da construção de modelos e a classificação de amostras em um servidor. Gossip Learning é uma proposta de aprendizado federado descentralizado através da consulta de colaboradores conhecidos [Hegedűs et al. 2019]. A proposta apresenta resultados de classificação melhores do que os obtidos pelo Google [McMahan et al. 2016]. Entretanto, a proposta é suscetível ao ataque de Sybil [Douceur 2002], pois permite um atacante criar diversos perfis falsos com a finalidade de divulgar resultados incorretos e comprometer o modelo final.

A maioria das propostas de aprendizado distribuído negligenciam a segurança e assumem que todos os colaboradores são honestos. No entanto, existe a possibilidade de ataques de colaboradores maliciosos que tenham a intenção de prejudicar o modelo final. Uma forma de se contrapor a este tipo de ataque é usar uma corrente de blocos. Uma forma de garantir confiança mútua em ambientes distribuídos é usando a corrente de blocos, comprovada por diversos trabalhos, como por exemplo em cidades inteligentes [Michelin et al. 2018], controle de acesso em IoT [Pinno et al. 2017, Camilo et al. 2020], bases de dados distribuídas [Palma et al. 2019], fatias de rede [Rebello et al. 2019] e assistência médica [de Oliveira et al. 2019].

Li *et al.* propõem o uso de corrente de blocos para implementar o aprendizado federado distribuído [Li et al. 2020]. A corrente de blocos armazena o modelo global, que um conjunto de domínios atualiza. Apesar do aprendizado ocorrer de forma distribuída, os nós da rede possuem um modelo global, portanto um participante malicioso pode explorar o modelo para encontrar brechas e atacar outros nós. Outro problema é a necessidade de realizar consenso para chegar ao modelo global, o que apresenta aumento no tempo de criação do modelo final [McMahan et al. 2016].

FLChain [Bao et al. 2019] é um arcabouço de aprendizado federado com o uso de

corrente de blocos para a criação dos modelos de aprendizado de máquina. A corrente de blocos audita e rastreia os comportamentos maliciosos dos participantes, armazena o modelo e remunera os participantes honestos por sua contribuição através de um contrato inteligente. Porém, Bao *et al.* utilizam apenas algoritmos baseados em gradiente na proposta, e há múltiplas rodadas de comunicação para estabelecer o modelo final.

A detecção de intrusão é uma área de pesquisa que estuda a mitigação de ameaças É importante mitigar ameaças de rede em ambientes de Internet das Coisas (*Internet of Things* - IoT), pois geralmente são constituídos por dispositivos e sensores com poucos recursos computacionais. Devido ao baixo poder de processamento, um atacante pode facilmente comprometer os dispositivos e sensores para fazerem parte de uma rede de robôs (*botnet*) [Bezerra et al. 2018]. Assim, uma rede de robôs grande pode iniciar um ataques de negação de serviço distribuído (Distributed Denial of Service - DDoS) [Pelloso et al. 2018]. Viegas *et al*. e Guimarães *et al*. propõem duas plataformas para a classificação de dados em tempo real [Viegas et al. 2019, de Brito Guimarães et al. 2020]. Porém, poucas as propostas adotam medidas para manter a privacidade dos dados utilizados para o treinamento de modelos capazes de detectar os ataques [Nguyen et al. 2019].

Ao contrário dos artigos citados, a proposta deste artigo, o DFedForest, permite o compartilhamento seguro de modelos completos de árvore de decisão para a criação de florestas aleatórias locais, protegendo a privacidade do usuário através da desidentificação das amostras utilizadas no treino dos modelos sem introduzir a latência devido a trocas de mensagens para classificação ou uso de criptografia. Além disso, para detectar ameaças de intrusão os participantes testam os modelos de árvore de decisão antes de agregá-los, o que permite identificar comportamentos maliciosos ou modelos incorretos.

3. O Sistema DFedForest Proposto

O DFedForest[2] é um sistema distribuído que cria florestas aleatórias de forma federada, através de participantes que treinam modelos de árvore de decisão com dados locais de modo independente. A ideia principal é manter os dados localmente, evitando a transferência de dados, privados ou sensíveis. Para se obter um modelo mais robusto, em vez de se usar os conjuntos de dados locais, utilizam-se os modelos locais de aprendizado de máquina de cada domínio. Para criar os modelos de árvore de decisão e classificar amostras, os participantes utilizam o método de *bagging* [Breiman 1996] que possui duas fases: criação (*bootstrap*) e agregação (*aggregating*). A fase de criação consiste na geração de conjuntos de dados de tamanhos iguais através de amostragens aleatórias de um conjunto de dados original e gerar modelos de aprendizado de máquina a partir de cada conjunto de dados amostrado. O objetivo é treinar modelos de aprendizados que possuam estruturas diferentes capazes de generalizar o comportamento ao classificar novas amostras quando combinados. O algoritmo utiliza a fase de agregação para descobrir o rótulo de um novo dado, onde todos os modelos de aprendizado de máquina do algoritmo classificam a amostra e o resultado final da classificação é a moda dos votos de cada modelo. O sistema proposto possui uma arquitetura com quatro componentes principais: os domínios, os modelos de árvore de decisão, a corrente de blocos e a floresta aleatória local criada de forma federada. Os domínios criam os modelos, os compartilham através

[2]Decentralized Federated Forest. Disponível em https://github.com/GTA-UFRJ-team/DFedForest

da corrente de blocos, e consultam por novas árvores de decisão para evoluir a floresta aleatória local. O DFedForest utiliza o método de criação (*bootstrap*) para gerar conjuntos de dados para treinar os modelos de árvores de decisão dentro de um domínio. Antes de adicionar árvores à floresta local, o domínio usa parte de seu conjunto de dados para testar os novos modelos. Caso o resultado do teste seja maior do que o limiar definido, o sistema adiciona o novo modelo à floresta local.

3.1. Arquitetura Proposta

A floresta aleatória federada de cada domínio possui árvores de decisão criadas localmente e por outros participantes. A Figura 1 exibe a arquitetura da proposta com os componentes, exemplificando a execução do sistema para um cenário com 4 domínios. Os Domínios 1 e 2 extraem informações de suas redes para a criação das árvores de decisão. Após a criação dos modelos, os domínios compartilham através da corrente de blocos uma referência, o localizador de recurso uniforme (*Uniform Resource Locator* - URL), do modelo armazenado no servidor de armazenamento. O Domínio 3 consulta a corrente de blocos por novas referências de modelos de árvore de decisão e baixa os modelos do servidor de armazenamento dos demais domínios. Após selecionar os melhores modelos, as árvores dos Domínios 1 e 2 são agregadas à floresta local do Domínio 3.

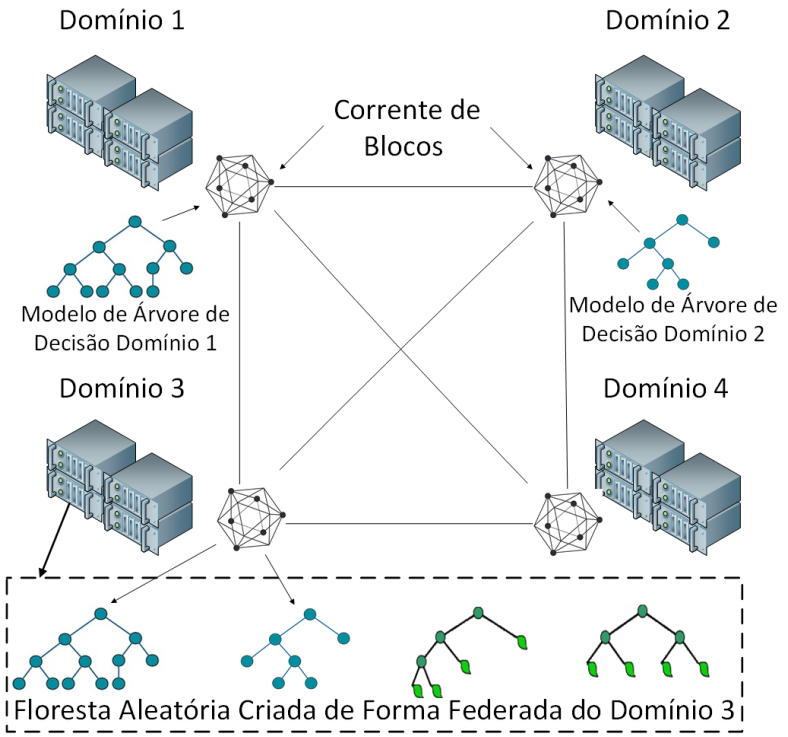

Figura 1. Arquitetura do sistema proposto. Os Domínios 1 e 2 criam modelos de árvores de decisão e compartilham através da corrente de blocos com transações de divulgação. O Domínio 3 emite uma transação de consulta a corrente de blocos, recebe os modelos criados, seleciona e evolui a floresta local.

Duas etapas dividem o diagrama completo de execução do sistema que a Figura 2 exibe. Um domínio possui acesso às capturas de tráfego, e as processa para extrair características. A hipótese da proposta é que os domínios possuem uma distribuição de tráfego

similar e o compartilhamento de informações locais ajuda a melhorar os modelos de cada domínio. Porém, é inviável compartilhar os dados para o treino dos modelos, devido a restrições quanto à privacidade, além do enorme volume e velocidade de informações produzidas em cenários de *Big Data* e dispositivos IoT. Os domínios extraem as mesmas características dos dados de rede para criação de um modelo consistente para outros participantes. A arquitetura proposta é agnóstica quanto à extração de características e rotulação dos dados, que estão fora do escopo deste artigo. O conjunto de dados é utilizado para a criação de modelos de árvore de decisão, e o domínio compartilha a referência dos modelos desejados através de transações de publicação na corrente de blocos. O tamanho dos modelos de árvore de decisão depende do total de nós da árvore. Assim, o sistema limita a profundidade das árvores criadas para reduzir o tamanho dos modelos armazenados e evitar o sobre-ajuste ao conjunto de dados de treino. A parte à esquerda do servidor de armazenamento na Figura 2 exibe todas as etapas descritas para a publicação de novos modelos.

Para adquirir modelos de outros domínios, os participantes emitem transações de consulta. A corrente de blocos armazena o local de armazenamento dos modelos de árvore de decisão através de um URL, para evitar sobrecarga na estrutura. Após obter o local de armazenamento dos modelos novos, o domínio faz requisições através do protocolo de transferência de hipertexto (*Hypertext Transfer Protocol* - HTTP) para o servidor de armazenamento referenciado. O servidor responde a requisição com os modelos de árvore solicitados. Ao receber a resposta do servidor de armazenamento, o servidor do domínio testa os modelos obtidos para verificar as métricas como acurácia e precisão em parte do seu conjunto de dados.

A parte a direita do servidor de armazenamento na Figura 2 exibe o processo descrito para evolução da floresta local com árvores criadas em outros domínios. O administrador do domínio define os critérios de parada do crescimento da floresta aleatória local e os limiares de métricas no teste.

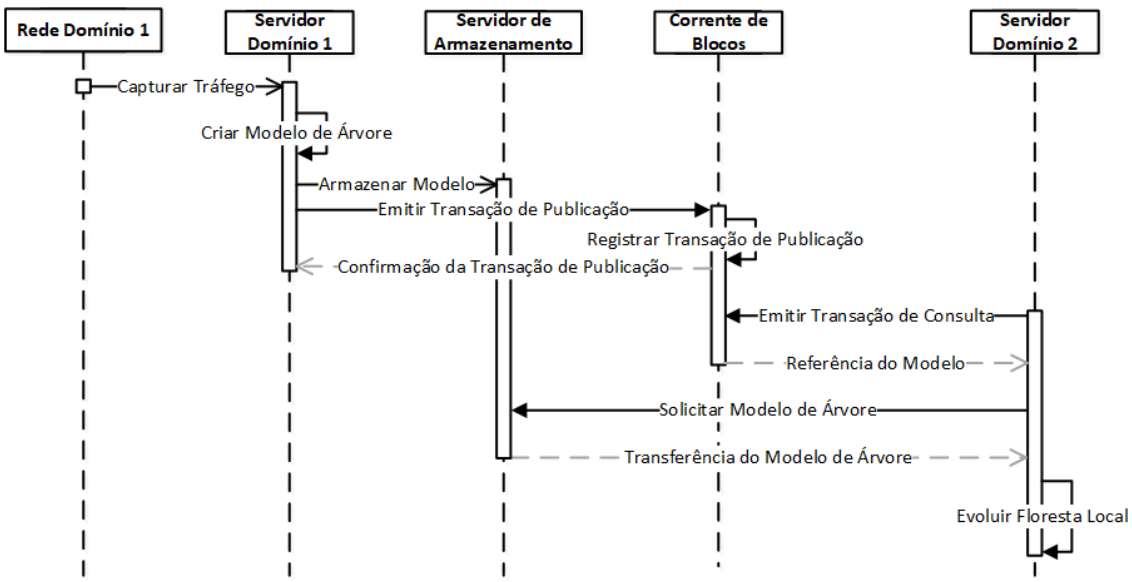

Figura 2. Diagrama de publicação e consulta por modelos de árvore de decisão.

A estrutura da corrente de blocos é responsável por armazenar a referência dos modelos de árvore de decisão compartilhados e associar os domínios aos modelos compartilhados. Dessa forma, caso um domínio publique modelos incorretos, os demais domínios facilmente identificam o responsável pelo comportamento malicioso. A Equação 1 apresenta o formato dos blocos e das transações presentes na corrente de blocos:

$$TX_{pub} = [TX_{ID}|A_{ref}|Org_{src}], \tag{1}$$

onde TX_{ID} é o identificador da transação, A_{ref} é a referência do local de armazenamento do modelo de árvore de decisão e Org_{src} é a assinatura da organização que na corrente de blocos representa o domínio responsável por publicar a transação.

Os blocos são adicionados sequencialmente na corrente de blocos, e contêm as transações divulgadas na rede. O cabeçalho dos blocos possui informações como o *hash* criptográfico das suas informações e do bloco anterior, que garantem a integridade de seu conteúdo e a ordem do bloco na rede.

3.2. Análise de Segurança

O ambiente admite ações maliciosas de domínios, como: (i) compartilhar modelos incorretos; (ii) estimar informações sobre as florestas dos demais domínios; (iii) explorar os modelos compartilhados na corrente de blocos; e (iv) emitir transações rapidamente para produzir negação de serviço na corrente de blocos.

Os modelos de árvore de decisão obtidos através da corrente de blocos são testados com dados locais antes de serem agregados à floresta aleatória. Este método é similar ao *out-of-bag* [Breiman 2001], uma vez que os dados utilizados para o teste são desconhecidos dos modelos treinados em outros domínios. Assim, o sistema detecta modelos incorretos de forma eficiente, evitando possíveis danos na floresta aleatória final dos domínios por um atacante. Devido a imutabilidade e o não-repúdio proporcionados pela corrente de blocos, os domínios podem identificar os modelos que possuem métricas de classificação baixas e rastrear os responsáveis pela divulgação. Como a corrente de blocos é permissionada, as credenciais de um participante malicioso podem ser revogadas pelos membros honestos da rede, se a maioria dos domínios concordar.

Os administradores dos domínios são responsáveis por decidir quais modelos de árvore de decisão da corrente de blocos serão adicionadas a floresta para aumentar o desempenho, sem divulgar o resultado local final. Logo, o atacante desconhece a floresta aleatória federada de cada domínio. Mesmo que um domínio consulte uma árvore, ela pode ser reprovada no teste para agregação à floresta. Como os dados de teste são mantidos em sigilo, é difícil estimar quais árvores foram selecionadas. Além disso, o atacante desconhece o tamanho das florestas e quanto mais árvores compartilhadas na corrente de blocos, mais difícil é descobrir a floresta aleatória federada de um domínio.

Os participantes da rede têm acesso aos modelos de árvore de decisão completos. Por isso, a estrutura de árvore de decisão exclui informações privadas que possam identificar um usuário da rede. Assim, a árvore é definida com características numéricas para os limiares de divisão de cada nó.

Outro comportamento malicioso é a negação de serviço através da corrente de blocos. Um domínio malicioso pode divulgar diversas transações válidas com o objetivo de

inundar os nós honestos da rede. Porém, este comportamento é facilmente identificável, uma vez que os domínios assinam as transações e, como a corrente de blocos é permissionada, o atacante pode ter suas credenciais revogadas e ser expulso da rede.

3.3. Análise de Complexidade do Sistema

A complexidade do sistema pode ser dividida em quatro partes: criação das árvores, compartilhamento, consulta e classificação. O artigo supõe que as árvores de decisão são binárias e balanceadas para simplificar o cálculo das complexidades computacionais.

A complexidade de criação das árvores é a mesma complexidade do algoritmo de árvore de decisão tradicional que é dada por $O(m.n.log(n))$, onde n representa a quantidade de amostras de treino e m a quantidade de características da amostra. Como o número de características, m, neste trabalho é igual a quatro, a complexidade de criação da árvore é descrita assintoticamente por $O(n.log(n))$.

A publicação de uma árvore possui complexidade $O(1)$, pois após a árvore ser criada e armazenada no servidor, a transação contém a referência da árvore, que é limitada e independente do tamanho do modelo. Assim, para o compartilhamento de r transações, a complexidade é linear $O(r)$.

A complexidade da consulta tem duas fases: a consulta à corrente de blocos e a requisição do modelo ao servidor de armazenamento. Na primeira fase, a consulta à corrente de blocos possui complexidade $O(1)$ caso a corrente de blocos armazene as transações em uma tabela *hash*, ou $O(t)$, caso haja busca em toda a corrente de blocos. Na segunda fase, a requisição do modelo possui complexidade $O(b)$, onde b é o número de bytes do modelo. Porém, o tamanho dos modelos de árvore de decisão utilizados no sistema é limitado para evitar o sobre-ajuste aos dados de treino e sobrecarga no armazenamento dos modelos. Para o caso em que o servidor consulta à corrente de blocos por h modelos de árvore de decisão diferentes e faz a requisição dos modelos, a complexidade total da consulta no pior caso é $O(h.t)$.

A classe binária predita $\hat{y}$ de um fluxo x por uma floresta aleatória com J árvores de decisão h é dada pela função $f(x)$:

$$\hat{y} = f(x) = arg\max_{y \in Y} \sum_{j=1}^{J} I(y = h_j(x)), \tag{2}$$

onde $h_j(x)$ retorna a classe binária do fluxo x pela árvore h_j. O termo $I(.)$ é a função indicadora. O conjunto Y representa as classes existentes, neste caso 0 para fluxos normais ou 1 para fluxos maliciosos. Em caso de empate entre as duas classes o sistema utiliza o critério de desempate escolhido pelo domínio.

A classificação binária representada pela Equação 2 é similar à classificação de uma amostra por uma floresta aleatória tradicional [Breiman 2001].

A j-ésima árvore de decisão h_j possui complexidade $O(log(d))$ para a classificação de uma amostra, onde d é a profundidade da árvore. Portanto, a complexidade da classificação da floresta aleatória criada de maneira federada é igual à floresta aleatória tradicional, dada por: $O(J.n.log(d))$, onde J representa o número de árvores da

floresta e n o número de amostras a serem classificadas. Assintoticamente o sistema possui complexidade igual à $O(J.n.log(d))$, pois após criar a floresta aleatória local o sistema suspende o treino, compartilhamento e consulta por modelos de árvores de decisão.

4. Desenvolvimento do Protótipo e Resultados

O ambiente distribuído foi desenvolvido com a plataforma de código aberto Hyperledger Fabric 2.0 [Androulaki et al. 2018] para a corrente de blocos, com nós virtualizados através de contêineres Docker v19.03.5[3]. A corrente de blocos utiliza o protocolo de consenso Raft e cada bloco possui 100 transações, conforme usado em trabalho anterior da avaliação de desempenho da plataforma Hyperledger Fabric [Gorenflo et al. 2019]. O Hyperledger Fabric é uma plataforma gratuita, facilmente programável e sem qualquer moeda ou custo associado à implementação de correntes de blocos permissionadas para organizações. O aspecto organizacional do Hyperledger atende ao cenário da proposta multi-domínios, no qual as empresas compartilham modelos de árvore de decisão. A corrente de blocos possui 2 tipos de nós: ordenadores e pares. Os nós são associados às organizações da rede, que representam os domínios. Os pares são responsáveis por interagir com a corrente de blocos, emitindo, validando e verificando transações. Os ordenadores são responsáveis por verificar as assinaturas dos pares, estabelecer a ordem das transações dentro de blocos e divulgar blocos para a rede. O sistema considera inválidas as transações com assinaturas desconhecidas ou revogadas. Um protótipo do sistema foi desenvolvido em Python v2.7.13 utilizando a biblioteca scikit-learn v0.20.4[4] para a manutenção dos modelos parciais de árvore de decisão. Os experimentos deste trabalho foram realizados em um servidor Intel i7-8700 CPU 3.20 GHz com 6 núcleos de processamento e 32 GB de RAM e apresentam os valores médios com intervalos de confiança de 98%.

O protótipo utiliza o conjunto de dados CTU-13 [Garcia et al. 2014] que possui amostras de tráfego normal e malicioso geradas com máquinas infectadas por redes de robôs. A escolha desse conjunto de dados permite a avaliação do desempenho do sistema utilizando um problema atual que afeta milhares de dispositivos vulneráveis nas redes de computadores. O conjunto de dados CTU-13 possui 13 cenários descritos na Tabela 1 com 14 características extraídas do cabeçalho dos pacotes, nas quais 8 identificam os fluxos e outras duas são nulas em todo o conjunto de dados, ToS destino e ToS origem. Como o sistema proposto compartilha os modelos de árvore de decisão sem o uso de criptografia, os domínios excluem as características que associam um fluxo a um usuário[5] para desidentificar a quem pertence os dados, pois a estrutura da árvore contém apenas informações sobre as distribuições dos fluxos. Portanto, para os treinos e testes, 4 características são utilizadas: número de pacotes, quantidade de bytes enviados pela origem, total de bytes do fluxo e duração.

Montovani *et al.* afirmam que o algoritmos de árvore de decisão não sofrem muita variação em relação aos valores padrão na otimização hiperparamétrica e convergem rapidamente [Mantovani and othes 2016]. Assim, são realizadas quatro buscas em grade para

[3]Disponível em https://docs.docker.com/

[4]O scikit-learn [Pedregosa et al. 2011] é uma biblioteca de código aberto, bem documentada, para a criação de modelos de aprendizado de máquina, que possui um grande número de desenvolvedores. Disponível em https://scikit-learn.org/0.20/

[5]IP origem, Porta origem, IP destino, Porta destino e protocolo da camada de aplicação.

Cenário	Fluxos Normais	Fluxos Maliciosos	Rede de Robôs	Nós Infectados
1	30387	4959	Neris	1
2	20941	9120	Neris	1
3	116887	26822	Rbot	1
4	25268	1768	Rbot	1
5	4679	901	Virut	1
6	7494	4630	Menti	1
7	1677	63	Sogou	1
8	72822	6126	Murio	1
9	43340	184979	Neris	10
10	15847	106352	Rbot	10
11	2718	8164	Rbot	3
12	7628	2168	NSIS.ay	3
13	31939	39993	Virut	1

Tabela 1. Divisão das amostras e tipos de redes de robôs (*botnets*) dos cenários presentes no conjunto de dados CTU-13.

ajustar os hiperparâmetros de número de árvores e a profundidade máxima da floresta aleatória. Os outros hiperparâmetros permanecem com os valores padrão. A primeira busca utilizou os valores [10, 100, 500] para o número de árvores e [5, 30, 100, 1000] para a profundidade das árvores, com o melhor caso 100 árvores e profundidade 30. Com os resultados da primeira busca é perceptível que a partir de 100 árvores os valores das métricas possuem baixa variância, o mesmo resultado verificado em trabalhos anteriores [Oshiro et al. 2012]. Dessa forma, o número de árvores é fixado em 100 e a busca em grade é limitada à profundidade das árvores, para buscar modelos menores com os mesmos níveis de acurácia e precisão. O espaço de busca é divido em valores próximos do melhor resultado de cada busca anterior, e por fim o valor de profundidade 23 é o valor encontrado para o melhor caso da sintonização. Os experimentos seguintes, tanto para a floresta aleatória quanto para o DFedForest utilizam os hiperparâmetros encontrados.

O Experimento 1 tem como objetivo verificar e comparar o desempenho do sistema DFedForest e da floresta aleatória quando os dados estão particionados na rede. O Experimento 1 combina todos os 13 cenários da Tabela 1 para produzir um conjunto de dados balanceado, equilibrando o número de fluxos normais e maliciosos. Este conjunto de dados completo com os 13 cenários é divido em subconjuntos distintos e balanceados do conjunto de dados completo através de amostras aleatórias para obter o subconjunto de cada domínio. Duas situações são consideradas para a abordagem tradicional: (i) os domínios possuem acesso ao conjunto de dados completo para criar o modelo de floresta aleatória, sem manter a privacidade dos dados e (ii) os domínios criam a floresta aleatória com seus dados. A floresta no caso (iii) é do sistema DFedForest, onde cada domínio é responsável pela criação da mesma quantidade de árvores, sendo 128 árvores criadas em cada cenário para comparação dos resultados. Os valores exibidos na Figura 3 são medidos para a fração de 30% de teste do conjunto de dados do Domínio 1 para equivalência na comparação. A quantidade inicial de domínios é 2 e variada exponencialmente até atingir o número de 128 domínios.

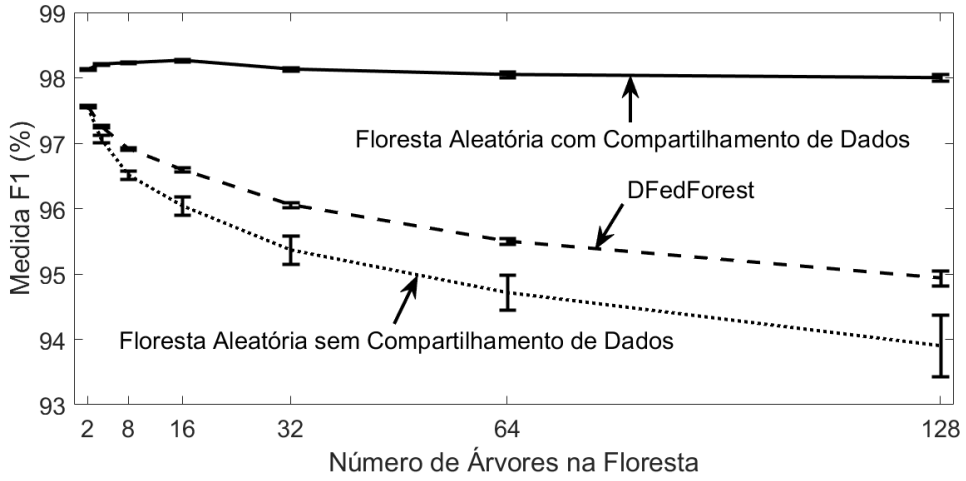

Figura 3. Comparação da medida F1 para diferentes quantidades de domínios com partes iguais do conjunto de dados. As linhas ilustram os resultados das abordagens de classificação: (i) floresta aleatória com compartilhamento de dados dos domínios para o treino dos modelos; (ii) floresta aleatória treinada com dados apenas do domínio; e (iii) floresta aleatória criada com o sistema DFedForest.

Os resultados do Experimento 1 mostram que o caso (i) produz o melhor resultado em todos os cenários, quando a floresta aleatória possui acesso aos conjuntos de treino dos domínios. Porém, como essa solução é inviável para manter a privacidade dos dados, os domínios optam por medidas que garantem a privacidade dos dados, como nos casos (ii) e (iii). O DFedForest é superior à floresta aleatória sem compartilhamento de dados na maioria dos casos, e quanto mais os dados estão particionados entre os domínios maior é a vantagem obtida. Os resultados obtidos são compatíveis com os encontrados por Giacomelli *et al.* [Giacomelli et al. 2019],

O Experimento 2 tem como objetivo avaliar o efeito da adição de árvores de dois domínios distintos para a detecção de novas ameaças. Para que os dois domínios possuam diferentes tipos de ameaça o Domínio 1 combina os cenários 1, 2, 6, 8 e 9 e o Domínio 2 combina os cenários 3, 4, 5, 7, 10, 11, 12 e 13, conforme recomendado pelos autores do conjunto de dados [Garcia et al. 2014]. Cada conjunto é dividido em 30% de dados para teste e o restante para treino. A Figura 4 exibe os resultados para a acurácia na classificação do Domínio 1 com relação aos dados de teste do Domínio 2. O Domínio 1 possui 64 árvores criadas com seu conjunto de dados, representadas na Figura 4 pela linha sólida e adiciona árvores do Domínio 2, representadas pela linha pontilhada. A adição de modelos criados no Domínio 2 à floresta do Domínio 1 possui um impacto positivo de até 3% de aumento na acurácia ao classificar ameaças desconhecidas presentes no conjunto de teste do Domínio 2, resultado representado pela linha pontilhada na Figura 4. Como a sintonização de hiperparâmetros exibe que o tamanho da floresta não impacta nas métricas de classificação após um número de 100 árvores, pode-se afirmar que o ganho na precisão foi devido ao sistema DFedForest. O compartilhamento de modelos melhora a detecção de novas ameaças desconhecidas pelo Domínio 1.

O Experimento 3 tem como objetivo verificar o tempo de acesso à corrente de

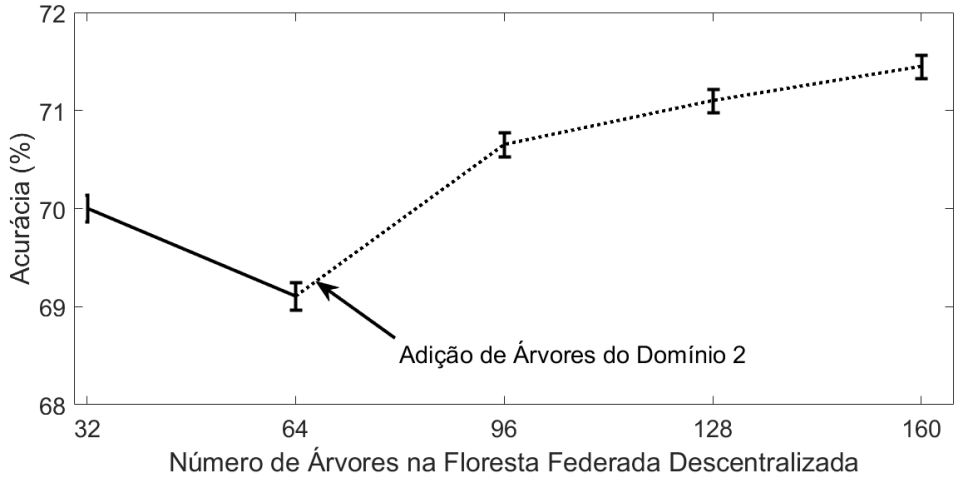

Figura 4. Impacto da adição de árvores do Domínio 2 na acurácia da floresta aleatória do Domínio 1 ao classificar o conjunto de teste do Domínio 2. A linha sólida representa o resultado da classificação do DFedForest no Domínio 1 com árvores criadas no próprio domínio. A linha pontilhada exibe os resultados após a adição de árvores do Domínio 2.

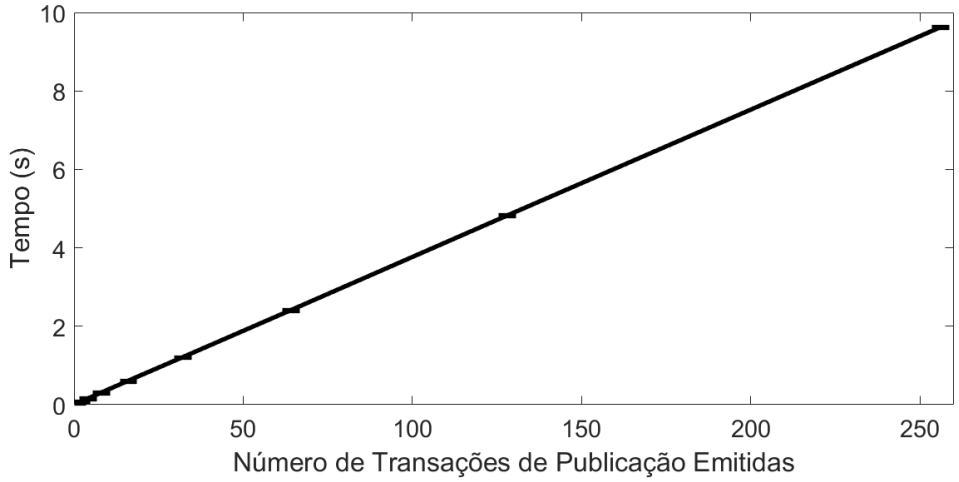

Figura 5. Comportamento assintótico para emitir transações de publicação que referenciam modelos de árvore de decisão armazenados fora da corrente de blocos.

blocos para o compartilhamento das referências dos modelos de árvores de decisão. A Figura 5 exibe o tempo necessário para a emissão de diferentes quantidades de transações de publicação por um domínio. Como analisado na Seção 3.3, o tempo para emitir t modelos varia linearmente em relação ao número de modelos compartilhados. Observa-se o mesmo comportamento em relação ao tempo necessário para a consulta por novos modelos. Após construir a floresta aleatória de acordo com as políticas do domínio, não há troca de mensagens para a classificação de novas amostras. Essa característica é essencial para aplicação do sistema no problema de classificação em tempo real. Outras propostas, que necessitam de troca de mensagens ou utilizam criptografia dos dados, produzem alta latência, atingindo minutos para obter classes de novas amostras [Giacomelli et al. 2019,

Liu et al. 2020, Fan et al. 2003].

5. Conclusão

O sistema DFedForest proposto classifica o tráfego de rede sem exigir sobrecarga e processamento e de mensagens trocadas quando comparado ao modelo de aprendizado federado que usa criptografia homomórfica e requer o envio das novas amostras. O sistema possui baixa complexidade computacional, assintoticamente descrita por $O(J.n.log(d))$, igual à floresta aleatória tradicional com J árvores de profundidade d para classificar n amostras. O DFedForest apresenta um desempenho melhor do que os resultados obtidos apenas com dados locais do mesmo domínio. Esta melhora se deve ao compartilhamento dos modelos locais entre todos os domínios. O sistema proposto utilizou uma corrente de blocos para compartilhar de forma distribuída a referência na qual os modelos locais são obtidos. As propriedades de imutabilidade e transparência permitem auditar domínios maliciosos que pretendam prejudicar a detecção de ameaças. A corrente de blocos não afeta o desempenho do sistema, pois o protótipo do sistema desenvolvido requer meio segundo para compartilhar um modelo e varia linearmente com a quantidade de modelos compartilhados. Para trabalhos futuros o objetivo é integrar o DFedForest com um sistema de arquivos distribuído, como o Sistema de Arquivos Interplanetário (*InterPlanetary File System* - IPFS), e avaliar a proposta com diferentes conjuntos de dados.

Referências

Androulaki et al. (2018). Hyperledger Fabric: A distributed operating system for permissioned blockchains. In *Proceedings of the 13th EuroSys Conference*, page 30.

Bao, X. et al. (2019). FLChain: A blockchain for auditable federated learning with trust and incentive. In *2019 5th BIGCOM*, pages 151–159.

Bezerra, V. et al. (2018). Providing IoT host-based datasets for intrusion detection research. In *Anais do XVIII SBSeg*, pages 15–28.

Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2):123–140.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Camilo, G. F. et al. (2020). AutAvailChain: Disponibilização segura, controlada e automática de dados IoT usando corrente de blocos. *SBRC*.

de Brito Guimarães, L. C. et al. (2020). TeMIA-NT: Monitoramento e análise inteligente de ameaças de tráfego de rede. In *Salão de Ferramentas SBRC 2020*.

de Oliveira, M. T. et al. (2019). Towards a blockchain-based secure electronic medical record for healthcare applications. In *IEEE ICC*, pages 1–6.

Douceur, J. R. (2002). The sybil attack. In *IPTPS*, pages 251–260. Springer.

Fan, W., Wang, H., Yu, P. S., and Ma, S. (2003). Is random model better? On its accuracy and efficiency. In *Third IEEE International Conference on Data Mining*, pages 51–58.

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

Giacomelli, I. et al. (2019). Privacy-preserving collaborative prediction using random forests. *AMIA Summits on Translational Science Proceedings*, 2019:248.

Gorenflo, C., Lee, S., Golab, L., and Keshav, S. (2019). FastFabric: Scaling Hyperledger Fabric to 20,000 transactions per second. In *2019 IEEE ICBC*, pages 455–463.

Hegedűs, I., Danner, G., and Jelasity, M. (2019). Gossip learning as a decentralized alternative to federated learning. In *IFIP DAIS*, pages 74–90. Springer.

Li, Y. et al. (2020). A blockchain-based decentralized federated learning framework with committee consensus. *arXiv preprint arXiv:2004.00773*.

Liu, Y., Liu, Y., Liu, Z., Liang, Y., Meng, C., Zhang, J., and Zheng, Y. (2020). Federated Forest. *IEEE Transactions on Big Data*, pages 1–1.

Mantovani, R. G. and othes (2016). Hyper-parameter tuning of a decision tree induction algorithm. In *2016 5th BRACIS*, pages 37–42.

McMahan, B. et al. (2016). Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*.

Michelin, R. A., Dorri, A., Lunardi, R. C., Steger, M., Kanhere, S. S., Jurdak, R., and Zorzo, A. F. (2018). SpeedyChain: A framework for decoupling data from blockchain for smart cities. In *MobiQuitous*, pages 145–154.

Nguyen, T. D., Marchal, S., Miettinen, M., Fereidooni, H., Asokan, N., and Sadeghi, A.-R. (2019). DÏoT: A federated self-learning anomaly detection system for IoT. In *2019 IEEE 39th ICDCS*, pages 756–767.

Oshiro, T. M., Perez, P. S., and Baranauskas, J. A. (2012). How many trees in a random forest? In *MLDM*, pages 154–168. Springer.

Palma, L. M., Vigil, M. A., Pereira, F. L., and Martina, J. E. (2019). Blockchain and smart contracts for higher education registry in brazil. *IJNM*, 29(3):e2061.

Pedregosa, F. et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pelloso, M. et al. (2018). A self-adaptable system for DDoS attack prediction based on the metastability theory. In *IEEE GLOBECOM*, pages 1–6.

Pinno, O. J. A., Gregio, A. R. A., and De Bona, L. C. (2017). Controlchain: Blockchain as a central enabler for access control authorizations in the iot. In *IEEE GLOBECOM*, pages 1–6.

Pisa, P. S., Abdalla, M., and Duarte, O. C. M. B. (2012). Somewhat homomorphic encryption scheme for arithmetic operations on large integers. In *IEEE GIIS*, pages 1–8.

Rana, S., Gupta, S. K., and Venkatesh, S. (2015). Differentially private random forest with high utility. In *2015 IEEE International Conference on Data Mining*, pages 955–960.

Rebello, G. A. F. et al. (2019). Providing a sliced, secure, and isolated software infrastructure of virtual functions through blockchain technology. In *IEEE HPSR*, pages 1–6.

Vaidya, J., Shafiq, B., Fan, W., Mehmood, D., and Lorenzi, D. (2013). A random decision tree framework for privacy-preserving data mining. *IEEE TDSC*, 11(5):399–411.

Viegas, E., Santin, A., Bessani, A., and Neves, N. (2019). BigFlow: Real-time and reliable anomaly-based intrusion detection for high-speed networks. *FGCS*, 93:473–485.