ARQUITETURA DE DETECÇÃO DE INTRUSÃO POR ANOMALIAS COM FEDERATED LEARNING EM REDES IOT

Samuel Carlos Menêses Soares, Francisco Lopes de Caldas Filho, Maria Flávia Soares, Fábio Lúcio Lopes de Mendonça, Edna Dias Canedo e Rafael Timóteo de Sousa Jr
Departamento de Engenharia Elétrica, Universidade de Brasília (UnB), Brasília-DF, Brasil

RESUMO

O mercado tecnológico de Internet das Coisas (IoT) teve um avanço nos últimos anos, possuindo cada vez mais dispositivos capazes de realizar tarefas de forma inteligente tanto em ambientes industriais como ambientes domésticos. A composição desses dispositivos apresenta uma grande simplicidade em termos de arquitetura por questões de custo-benefício e praticidade, porém, essa questão traz problemas de segurança e privacidade para as redes IoT devido à quantidade de vulnerabilidades que vão sendo identificadas e exploradas por conseguinte. Com isso, métodos foram sendo aplicados para solucionar esses problemas de exploração de vulnerabilidades, como a utilização de sistemas de detecção e prevenção de intrusão para identificar possíveis ataques na rede e realizar ações preventivas em tempo real. Um dos métodos utilizado para aplicar esse tipo de sistema é a detecção por anomalias, consistindo na observação de padrões de comunicação para distinguir os tráfegos por sua natureza, sendo altamente utilizado com técnicas de Aprendizado de Máquina e Inteligência Artificial (AI) por questões de inovação e automatização. Então, será apresentada neste trabalho uma solução de sistema de detecção de intrusão por anomalias para redes IoT com o intuito de identificar tráfegos maliciosos e distingui-los de tráfegos benignos, contendo a utilização de métodos de aprendizado de máquina, como o *Federated Learning*.

PALAVRAS-CHAVE

Internet das Coisas, Aprendizado de Máquina, Inteligência Artificial, Detecção de Intrusão, *Federated Learning*

1. INTRODUÇÃO

A Internet das Coisas (IoT) é uma aplicação de infraestrutura que consiste na conexão de dispositivos em uma rede capazes de realizar tarefas com alto grau de automação e interatividade com a finalidade de aplicar soluções inteligentes para diversos tipos de rede, como *smart homes*, geradores de energia, áreas de saúde, área automotiva, dentre outros. O seu avanço no mercado vem ocorrendo de forma intensa e acelerada nos últimos anos, considerado cada vez mais uma opção viável e vantajosa para ser utilizado em sistemas inteligentes. Com essa grande difusão e necessidade de rápido aprimoramento, diversos tipos de dispositivos de micro-arquitetura que apresentam uma estrutura de modelo simples surgiram no mercado como sensores, NVIDIA Jetson, Raspberry Pi, câmeras IP. Pelo fato desses dispositivos possuírem implementação de arquiteturas simples, há a presença de diversas vulnerabilidades e há a ocorrência de maior custo computacional para processos complexos.

A presença de diversas vulnerabilidades nos dispositivos IoT permitiu a utilização de *malwares* e *botnets* como o Mirai, apresentado no artigo (Margolis, 2017) de forma aprofundada quanto a ataques de DDoS, para a aplicação de ataques como envenenamento de dados, ataques distribuídos, ataques de inferências, reconstrução de dados, dentre outras. Então, com a finalidade de identificar esses ataques sendo realizados na rede e aplicar ações preventivas de segurança, evitando possíveis danos aos dispositivos, é possível estruturar um sistema de detecção de intrusão (IDS) (Liao, 2013) e sistema de prevenção de intrusão (IPS) (Zhang, 2004). Estes sistemas são importantes para introduzirem uma camada de segurança nas redes e reduzir as chances das vulnerabilidades serem exploradas por *softwares* maliciosos. Os tipos de detecção de uma IDS podem ser baseadas em assinaturas, que consiste em avaliar o tráfego com base em um conjunto de regras pré-estabelecidas e armazenadas em um banco de dados, e podem ser baseadas em anomalias (Chatterjee,

2020), consistindo em avaliar o tráfego com base em padrões anteriores de dados previamente coletados referentes a comunicações benignas e também maliciosas para então realizar a classificação desejada acerca da natureza do fluxo de pacotes identificado na rede.

A utilização de Inteligência Artificial (AI) para soluções inteligentes é amplamente difundida nas áreas de pesquisa de redes IoT. Uma aplicação de aprendizado de máquina que traz aprimoramentos de segurança e privacidade para os dispositivos durante o treinamento é o *Federated Learning*, assim como é explicado no artigo (Li, 2020). Essa implementação consiste em realizar o aprendizado de máquina de *datasets* ou modelos de forma descentralizada em uma rede, tal que cada dispositivo será selecionado a cada rodada do procedimento para fazer o treinamento, apresentando então resultados de forma coletiva a partir de vários clientes distintos. Para com que cada cliente participe das etapas, um servidor previamente preparado coordena e recebe esses treinamentos em formatos de pesos (resultados com base em parâmetros pré-definidos) por parte de cada cliente, para realizar uma agregação e obter o modelo final com as classificações desejadas. O *Federated Learning* é uma opção de AI para descentralizar e garantir maior privacidade durante os treinamentos, envolvendo a participação de múltiplos dispositivos e clientes que utilizarão os seus próprios dados locais para realizar a aprendizagem.

Neste projeto, o objetivo é desenvolver uma arquitetura de detecção de anomalias de tráfegos identificados e obtidos em uma rede IoT a partir de um componente de captura. De tal forma que esses dados serão estruturados em modelos de aprendizado de máquina a partir de ferramentas de processamento de fluxo de dados, sendo então utilizados para realizar vários treinamentos através da técnica de *Federated Learning* com redes neurais por múltiplas camadas em cada dispositivo IoT selecionado para executar a tarefa, feito então o treinamento do modelo obtido com base em padrões gerados a partir de amostras feitas na rede utilizando um modelo adversário contendo uma *botnet* ativa realizando ataques de negação de serviço distribuído em uma rede local para fins de estudo.

2. TRABALHOS RELACIONADOS

Os autores do trabalho (Kfouri, 2019) fizeram a implementação de um HIDS (*Host-based Intrusion Detection System*) de forma distribuída para avaliar a detecção de intrusão no *backbone* de uma rede IoT, focando os esforços nos componentes intermediários de uma rede isolada com a presença de uma controladora. A análise das vulnerabilidades ocorre a partir da aplicação de protocolos de monitoramento como o Syslog e SNMP. Em comparação, este projeto em questão tem o objetivo de estruturar uma NIDS (*Network Intrusion Detection System*) para avaliar o tráfego desde o *backbone* até os próprios dispositivos IoT, tendo a análise por meio de aprendizado de máquina após a devida captura dos dados da rede.

O paper (Fares, 2019) aborda a implementação de um IPS com uma *honeynet* em redes SDN para identificar e mitigar ataques de DoS (*Denial of Service*) a partir da utilização de OpenFlow para encaminhar os pacotes suspeitos e tratá-los como uma possível ameaça. Este projeto, em comparação com o paper, aborda a implementação de uma IDS sem a aplicação de ações preventivas, tendo o foco em capturar o tráfego em uma rede IoT para identificar possíveis ameaças por ML distribuído, de forma que não apenas ataques de DoS serão detectados como atividades recorrentes de *malwares* e *botnets*, como comunicações periódicas entre os dispositivos infectados.

O paper (Nguyen, 2019) aborda a implementação de uma IDS com *Federated Learning* a partir da detecção por tipos específicos de dispositivos (*device-type-specific detection models*), identificando anomalias pela forma de comunicação dos dispositivos em relação aos seus modelos e forma de comportamento, para então fazer um treinamento federado a partir do que é identificado. Este projeto, em comparação, aborda uma IDS com FL e Redes Neurais, mas centralizando a detecção de anomalias por padrões identificadas em toda a rede pelo comportamento do fluxo e dos pacotes, não focando no padrão de modelo dos dispositivos, por uma captura constante da propagação dos ataques da *botnet* na rede isolada.

No trabalho (Fotiadou, 2021), é feita a implementação de uma arquitetura de NIDS utilizando os tráfegos da rede para realizar um aprendizado centralizado por 1D-CNN e LSTM, com a captura de pfSense e Suricata, tendo a geração de um modelo de treinamento posterior. O projeto deste artigo também implementa um NIDS, porém com aprendizado federado (distribuído entre vários clientes) a partir de uma aplicação de Rede Neural, de tal forma que a captura dos dados da rede são feitas apenas por Suricata com a utilização de espelhamento de porta no *switch* da infraestrutura local.

Os autores do artigo (Gonçalves, 2019) projetaram uma IPS em uma rede de Internet das Coisas utilizando o *software* Snort para monitorar a rede e realizar a mitigação dos ataques próximos aos dispositivos a partir de uma rede SDN, isolando-os da rede. Em comparação, este artigo aborda um IDS sem

a utilização de redes SDN com a captura por Suricata para gerar alertas de assinatura em formato JSON e utilizá-los em um modelo de treinamento por aprendizado de máquina após o processamento dos dados em uma aplicação de transformação e filtragem.

No trabalho (Dapeng, 2021), é estruturado um IDS com *Federated Learning* com *Edge Computing* para realizar os treinamentos nos dispositivos IoT com menos processamento e necessidade de largura de banda, com a injeção de modelos de treinamento públicos visando fazer um aprendizado por CNN. Neste projeto, em comparação, ocorre a detecção ativa de uma captura em tempo real para gerar um modelo de treinamento com amostras a serem treinadas pelo *Federated Learning*, de tal forma que a detecção faz parte da estruturação e da preparação da etapa de aprendizado federado.

3. ARQUITETURA DO PROJETO

O modelo lógico do sistema a ser implementado, tendo o objetivo de se construir um sistema de detecção de intrusão por anomalias através de treinamento distribuído por *Federated Learning*, é apresentado na Figura 1 contendo cada componente da arquitetura a ser definida e apresentada posteriormente. Na arquitetura proposta, inicialmente terá a implementação de uma Mirai *botnet* constituída de um servidor de controle e de um conjunto de dispositivos previamente infectados agindo como atacantes, podendo ser identificados pela Figura 2, de tal forma que a *botnet* irá realizar atividades mal intencionadas a um alvo específico dentro da rede local. A infraestrutura de testes será apresentada no tópico Modelo adversário.

Em sequência, por uma máquina que estará monitorando os tráfegos de toda a rede, nomeada de controladora do IDS, ocorrerá a captura desses dados maliciosos através de alertas pré-definidos, de tal forma que o fluxo irá passar por um processamento em tempo real por meio de um sistema de mensagens com o intuito de ser transformado em um modelo de treinamento por uma aplicação de estruturação de dados. Na etapa de detecção, são utilizados o Suricata, Apache Kafka e Apache Spark para esse fim.

Após a transformação da captura em um *dataset* de ML, este será enviado para um servidor em uma máquina virtual com o intuito de utilizá-lo em um *framework* de aprendizado federado (FL) *open-source* chamado de Flower. O treinamento será coordenado ativamente pelo servidor, de tal forma que os clientes treinarão o *dataset* individualmente e irão enviar os resultados em forma de pesos para com que ocorra a agregação, finalização da classificação do modelo e análise de desempenho por métricas de acurácia e precisão. O treinamento será realizado por Redes Neurais Convolucionais (Indolia, 2018) em múltiplas camadas e a agregação será feita por um algoritmo nomeado de *Federated Averaging*, sendo este nativo do *framework* escolhido.

Com a definição da arquitetura estabelecida, foi feita a implementação de cada componente em um ambiente real para realizar ataques isolados da rede pública, visando fazer os estudos propostos neste artigo. O procedimento da implementação está dividido nos tópicos: Modelo adversário, Modelo de detecção e *Federated Learning.*

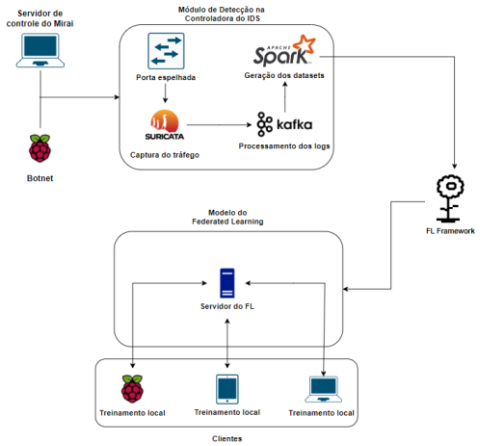

Figura 1. Modelo da arquitetura lógica

3.1 Modelo Adversário

Para realizar a captura de tráfegos maliciosos e realizar o estudo de detecção de anomalias, foi implementada, em um ambiente isolado para fins educacionais, uma *botnet* nomeada de Mirai. O Mirai, também explicado em (Zhang, 2020), consiste em um *malware* que surgiu nos últimos anos destinado a explorar as vulnerabilidades das arquiteturas de dispositivos IoT para infectá-los e realizar ataques em massa por DDoS (*Distributed Denial of Service*). O Mirai utiliza um servidor de Comando e Controle (C&C) capaz de gerenciar toda a *botnet*, podendo visualizar quantos bots estão conectados e conseguindo executar comandos de ataque a um determinado alvo. Os dispositivos infectados se comunicam com o Mirai pela porta 23.

Então, a partir da Figura 2, foi implementado em uma rede IoT um ambiente isolado contendo diversos componentes necessários para a implementação da arquitetura, principalmente com o Mirai e os dispositivos que foram infectados. Os componentes principais são: um *switch* interligando todos os dispositivos, uma máquina alvo contendo um servidor Apache, uma máquina com Ubuntu tendo o papel de controladora do IDS, alguns Raspberry Pi contendo o *malware*, e por último a controladora do Mirai. Após definir toda a rede por DHCP e gerar a comunicação entre os envolvidos, foi levantado o servidor do Mirai seguindo a distribuição Cosmic. De tal forma que é executado nele: o servidor C&C sendo acessado por *telnet*, um servidor SQL (*Structured Query Language*) com um banco de dados contendo todos os usuários e senhas dos bots pertencentes à rede, um servidor *Web* contendo todos os binários de infecção para diversos tipos de arquitetura e um servidor de escuta para identificar os dispositivos na rede.

Com o servidor em execução, basta realizar as infecções nos dispositivos. Esse procedimento foi feito de forma manual para fins de maior controle e gerenciamento de todo o processo de infecção, de tal forma que em cada dispositivo foi feito o download de um binário chamado de “sora” no formato “arm7” (específico para Raspberry Pi) através do comando: “wget http://164.72.15.130/bins/sora.arm7”. Com a execução do arquivo no dispositivo, este cria um túnel com a controladora do Mirai a partir do servidor de escuta para ser registrado na *botnet* e pelo banco de dados, confirmando então a infecção e a comunicação com o servidor C&C. Então, a partir desse momento, a controladora tem domínio sobre os bots recém infectados. Os ataques são realizados pelo terminal do servidor de controle, sendo possível especificar o tipo de ataque, a duração do ataque, em qual porta ocorrerá o *flood* de pacotes, além de diversos outros parâmetros que dependem do tipo de protocolo utilizado.

Formada a *botnet*, são feitos ataques de *flood* no servidor *Web* em execução no alvo e também no servidor DNS presente na controladora do IDS. O Mirai apresenta uma coleção de vários ataques possíveis de serem realizados. Para fins de posterior captura e estudo, foram feitos os ataques de UDP, SYN e TCP *flood* de forma simples tanto nas portas 80 (HTTP) e na porta 53 (DNS) tendo como origem os Raspberry Pi. O resultado do ataque será analisado posteriormente na controladora do IDS através do modelo de detecção implementado.

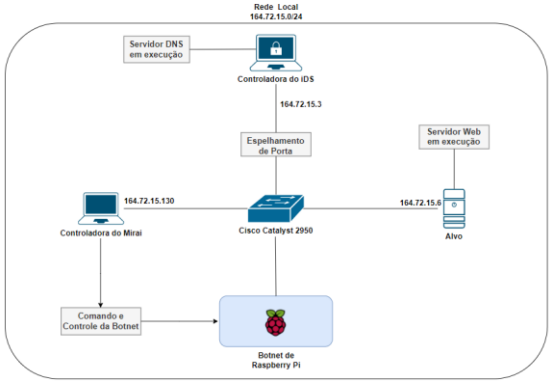

Figura 2. Rede local

3.2 Modelo de Detecção

A forma de captura dos ataques realizados pelo modelo adversário ocorre através da controladora do IDS, também presente na Figura 2. Esta máquina, com sistema operacional Ubuntu instalado, visualiza todo o

tráfego da rede local montada por *Port Mirroring* (espelhamento de porta), assim como foi feito em (Bul'ajoul, 2015) realizada através do *switch* Cisco Catalyst 2950. De tal forma que todas as outras portas que estão sendo utilizadas pelo *switch* irão ser espelhadas na interface da controladora, trazendo a capacidade de se monitorar todo o ambiente. Com essa condição satisfeita, é preparado então a forma de captura.

O objetivo inicial é gerar alertas na controladora a partir de uma biblioteca de regras para possuir campos importantes para a análise do comportamento de um tráfego. Inicialmente, foi instalado um software *open-source* de suporte a IDS/IPS chamado de Suricata. O Suricata, sendo estudado em (White, 2013), é uma ferramenta que possui uma biblioteca de regras a partir de diferentes fornecedores como o *Emerging Threats*. Então, sendo instalado o conjunto de regras necessários para gerar os alertas de captura das atividades anômalas dos dispositivos na rede, é configurado o formato de saída do log em JSON (*JavaScript Object Notation*). A justificativa de sua utilização se deve pelo fato da estruturação de cada informação ser por campos bem definidos, facilitando a visualização e a sua transformação. Então, durante a realização dos ataques, o Suricata é executado na interface da controladora que está com o espelhamento de porta ativo, captando o fluxo de tráfego em tempo real e gerando o log chamado de Eve JSON.

Por conseguinte, foi implementado um sistema de fluxo de mensagens para realizar o processamento dos dados, chamado de Apache Kafka. Com essa ferramenta, é levantado um *cluster* em formato de tópicos que receberá ativamente os dados presentes no log, de tal forma que o Suricata produz os dados no *cluster*. Então, é criado um tópico de inserção pelo Kafka que será executado, e este lerá o *log* em tempo real, assim inserindo os dados no *cluster* através do comando `tail` com o objetivo de receber os dados assim que eles vão surgindo pelo Suricata. Enquanto o Suricata produz as informações, a aplicação que irá recebê-las e consumi-las a partir do tópico é o Apache Spark. O Kafka é executado no *localhost* e na porta 9092.

O Apache Spark, estudado em (Petrov, 2018), é uma aplicação de processamento e estruturação de dados capaz de filtrar, modificar e transformá-los em um novo tipo de estrutura, conseguindo criar *datasets* em arquivos no formato CSV (*Comma-separated Values*) pelo fato de utilizar linguagem SQL. Então, pela integração entre o Kafka e o Apache Spark pelo módulo de Structured Streaming nativo da aplicação, é configurado um projeto no Maven na linguagem de programação Scala para desenvolver o código que irá coletar os dados do JSON e gerar o modelo. Na composição do projeto, são definidas as dependências da integração do Spark com o Kafka e das versões utilizadas de cada componente. O programa na linguagem Scala lê o *streaming* em JSON proveniente do Kafka e realiza funções do SQL para identificar os campos do log e atribuí-los a colunas juntamente de seus valores em um formato de tabela. Após definir todas as colunas e valores, ocorre a saída no formato de um arquivo CSV, de tal forma que cada linha é um evento capturado pelo Suricata.

As colunas filtradas a partir do log são: `flow id`, `source IP`, `destination IP`, `source port`, `destination port`, `protocol`, `hour`, `minute`, `seconds`, `severity`, `pkts to server`, `pkts to client`, `bytes to server` e `bytes to client`. Com o *dataset* formado, são feitos procedimentos de aprendizado de máquina por *Federated Learning*, de tal forma que o servidor receberá diretamente os arquivos de treinamento. A Figura 3 demonstra um exemplo de captura pelo Suricata em JSON contendo os campos apresentados anteriormente e também outros que foram descartados no processo de transformação.


```
{
  "timestamp": "2022-09-02T17:52:10.262086-0300",
  "flow_id": 1662504211621876,
  "in_iface": "enp0s25",
  "event_type": "alert",
  "src_ip": "164.72.15.12",
  "src_port": 60577,
  "dest_ip": "164.72.15.3",
  "dest_port": 53,
  "proto": "UDP",
  "community_id": "1:LEz3HXjQpXkGOSYE080n+IM/Pc4=",
  "alert": {
    "action": "allowed",
    "gid": 1,
    "signature_id": 2101948,
    "rev": 8,
    "signature": "GPL DNS zone transfer UDP",
    "category": "Attempted Information Leak",
    "severity": 2,
    "metadata": {
      "created_at": [
        "2010_09_23"
      ],
      "updated_at": [
        "2010_09_23"
      ]
    }
  },
  "app_proto": "failed",
  "flow": {
    "pkts_toserver": 623,
    "pkts_toclient": 511,
    "bytes_toserver": 933082,
    "bytes_toclient": 27910,
    "start": "2022-09-02T17:52:08.239604-0300"
  }
}
```


Figura 3. *Log* do Suricata

3.3 Federated Learning

Após a etapa de detecção, é feito o procedimento de treinamento por aprendizado federado em um ambiente pré-definido com o intuito de avaliar cada um dos eventos alertados durante os testes de ataque. A estrutura básica do *Federated Learning* envolve a presença de um servidor que irá coordenar todo o processo através de treinamento por múltiplas rodadas. Nas rodadas, cada cliente será selecionado para participar do processo e irá realizar o treinamento do modelo em questão de forma individual. No fim de cada rodada, o servidor receberá os resultados por meio de pesos e parâmetros pelos clientes com o intuito de realizar duas tarefas: juntar todos os resultados, agregá-los para formar uma avaliação final e realizar análises de desempenho a partir de métricas como acurácia e perda. A classificação final após o término das rodadas indica o comportamento dos eventos no conjunto de dados relacionados ao tráfego.

Primeiramente, para definir o ambiente de treinamento, foi levantada uma máquina virtual Ubuntu 20.04 para ser o servidor de FL. Então, para montar a estrutura, foram instaladas as dependências do *framework* a ser utilizado, chamado de Flower. Este *framework* foi escolhido pela sua facilidade de implementação, também por uma fácil integração com dispositivos IoT como Raspberry Pi. Então, foi instalado o repositório do Github de um exemplo em Tensorflow/Keras com o intuito de facilitar o processo de montagem do treinamento. No código, é definido a forma de treinamento sendo por CNN (Redes Neurais Convolucionais) de múltiplas camadas em 1 dimensão e também o método de agregação, sendo o *Federated Averaging* (FedAvg) estudado em (Nilsson, 2018), além de apresentar as formas de avaliação e recebimento de pesos na parte do servidor.

Então, para realizar o treinamento do modelo capturado, primeiramente ocorreu o tratamento dos dados fazendo pequenas alterações no CSV que foi gerado. Foi adicionada a coluna de classificação separada em "*Normal*", "*Bruteforce*", "*Botnet*", "DoS" ou "*Scan*" atribuídos a cada linha presente no *dataset*. Após o tratamento dos dados, o modelo resultante foi utilizado no treinamento, tal que uma amostra de sua estrutura está apresentada na Figura 4.

O treinamento federado, por aprendizado supervisionado, foi feito em uma máquina, de tal forma que, apenas por questões de experimentação e validação, os clientes foram executados em dois Raspberry Pi 3 Modelo B de arquitetura ARM64. Através do código modificado em TensorFlow, o modelo é lido e dividido em um conjunto de treinamento e teste. O conjunto de treinamento é constituído por linhas sem a presença da classificação, enquanto o teste é constituído por 20% de todo modelo contendo a classificação para cada evento, sendo utilizado como uma amostra para validar o modelo de treinamento e obter dados de acurácia.

3.3.1 Resultados do Experimento

Quanto ao experimento, foi utilizado o *dataset* da Figura 5 como resultado da captura e da estruturação, com uma máquina virtual contendo o código do servidor e outros dois dispositivos agindo como clientes. Com a execução do projeto em Python com a biblioteca do Flower *framework*, tanto o servidor quanto os 2 clientes são inicializados realizando uma comunicação entre eles por gRPC (*Remote Procedure Call*) no endereço `0.0.0.0:5050` em uma rede local, através de um túnel por SSL (*Secure Socket Layer*). Com a execução, o servidor espera os clientes ficarem ativos para então começar os treinamentos. O procedimento foi realizado em três rodadas com treinamentos de 5 épocas cada e com *batch size* igual a 16, tal que inicialmente ocorre a inicialização dos parâmetros de treino pelo servidor para com que os clientes possam iniciar seus aprendizados locais.

```
0,164.72.15.3,164.72.15.5,0,0,ICMP,18,5,40,3,0,0,0,0,Botnet
13144473755082,164.72.15.12,164.72.15.6,173,80,UDP,18,3,44,3,3,0,3198,0,DoS
1422554963674900,164.72.15.6,164.72.15.3,0,0,ICMP,17,10,54,3,9,16,882,1568,Normal
2177543804866060,164.72.15.11,164.72.15.3,39916,53,UDP,17,52,8,2,336,278,504000,15228,DoS
95536978984890,164.72.15.12,164.72.15.6,64610,80,UDP,18,3,45,3,1,0,1066,0,DoS
214320741940090,164.72.15.11,164.72.15.6,22600,80,UDP,18,3,46,3,2,0,2132,0,DoS
123106328354852,164.72.15.12,164.72.15.3,48629,53,UDP,17,52,10,2,598,530,897000,29004,DoS
```

Figura 4. Amostra do modelo utilizado no treinamento

Na Figura 5 está o fim da execução de um teste de aprendizado federado contendo três rodadas, sendo possível observar os valores de acurácia (taxa de proximidade do valor real) e perda (penalidade de uma má predição) para cada modelo agregado. Cada um dos dois clientes em dois Raspberry Pi diferentes realizaram seus treinamentos por 1D-CNN (Redes Neurais Convolucionais de 1 dimensão para modelos sequenciais), de

tal forma que quando o servidor recebeu os parâmetros de resultado, foi feito o procedimento de agregação para formar o modelo total a partir das contribuições dos clientes. A Tabela 1 mostra os resultados de acurácia e perda para cada uma das rodadas e a Figura 6 evidencia um gráfico contendo a análise de desempenho do modelo realizado por aprendizado federado. É possível observar que a acurácia do modelo aumentou a cada nova rodada enquanto a taxa de perda teve diminuições, conseguindo atingir 88% de acurácia, indicando um bom desempenho de detecção de eventos a partir de um conjunto de treinamento e de testes.

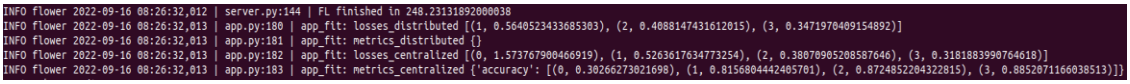

Figura 5. Treinamento de FL finalizado

Tabela 1. Resultados do aprendizado federado

Treinamento	Acurácia	Perda
Rodada 1	0.815	0.526
Rodada 2	0.872	0.380
Rodada 3	0.885	0.318

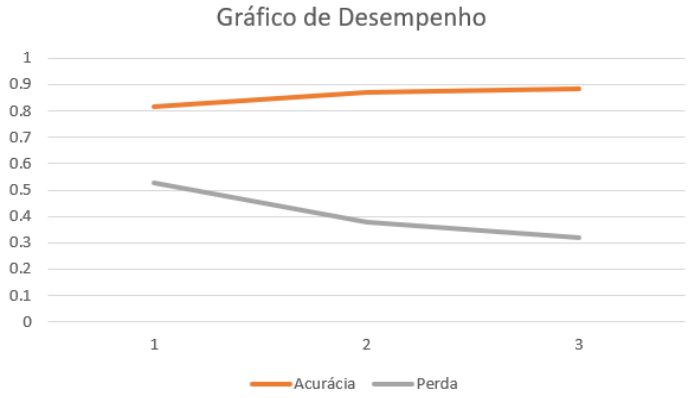

Figura 6. Gráfico de desempenho do treinamento.

4. CONCLUSÃO

Através do estudo realizado neste artigo, foi observado que é de extrema importância idealizar sistemas visando monitorar e inspecionar os tráfegos de uma rede com o intuito de identificar possíveis intrusos e ameaças que possam comprometer o ambiente. A quantidade de *malwares* presentes na rede pública é alta, e cada vez mais surgem formas de invadir sistemas que apresentam vulnerabilidades não ainda descobertas, principalmente em redes IoT fundamentais para a manutenção de diversos sistemas. Então, este projeto teve o intuito de montar uma arquitetura, a partir de uma implementação com dispositivos reais, de detecção de tráfegos maliciosos que possam indicar ameaças. O projeto em questão traz vantagens ao estudo de detecção de intrusão pelo fato de envolver uma estrutura de aprendizado federado, tendo a aplicação de uma *botnet* real para fins de estudo.

Os testes por aprendizado de federado, realizados para validar a arquitetura proposta e implementada, foram realizados através de uma máquina virtual agindo como servidor, e clientes como Raspberry Pi 3 tendo a utilização do *framework* Flower para estruturar os códigos. Após o tratamento do modelo em CSV, este foi utilizado no aprendizado por 3 rodadas a fim de treiná-lo, de tal forma que foi observado resultados decentes quanto a acurácia e perda, evidenciando a necessidade de uma melhor parametrização dos dados e até mesmo da utilização de mais dispositivos para agirem como clientes durante todo o procedimento. O *framework* do

Flower apresentou uma vantagem no aprendizado devido à facilidade da preparação do ambiente e também da alta capacidade de integração com dispositivos IoT.

O projeto apresenta várias possíveis melhorias futuras que podem ser aplicadas principalmente em termos de segurança, como a implementação de uma plataforma privada de *Blockchain* em IoT (Novo, 2018) para realizar os registros do modelo final e de cada treinamento feito, além de aplicar um método de consenso para filtrar os nós que possam estar comprometidos ou apresentando falhas antes de terminarem sua contribuição no treinamento. Algoritmos de encriptação como o SMC no artigo (Truex, 2019) ou de privacidade como o *Differential Privacy* no trabalho (Wei, 2020) são implementações futuras interessantes para manter o lado dos *endpoints* em segurança. Além disso, o projeto pode escalar para agir também como uma IPS além de uma IDS, utilizando do modelo final classificado para realizar ações preventivas logo após a finalização do treinamento com a presença de redes do tipo SDN (*Software-defined Networking*).

REFERÊNCIAS

A. A. Y. R. Fares, et al., "DoS Attack Prevention on IPS SDN Networks," 2019 *Workshop on Communication Networks and Power Systems (WCNPS)*, 2019, pp. 1-7

Adrian Nilsson, et al., 2018, A Performance Evaluation of Federated Learning Algorithms. *In Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning (DIDL '18)*, pp. 1–8

Chatterjee, Ayan & Ahmed, Bestoun, 2022. IoT Anomaly Detection Methods and Applications: A Survey. *Internet of Things*.

D. G. V. Gonçalves et al., "IPS architecture for IoT networks overlapped in SDN", 2019, *Workshop on Communication Networks and Power Systems (WCNPS)*, 2019, pp. 1-6

Dapeng Man. et al., 2021, Intelligent Intrusion Detection Based on Federated Learning for Edge-Assisted Internet of Things. *In Huawei Security and Privacy for Edge-Assisted Internet of Things*, Vol 2021, pp. 1-11.

Fotiadou, K. et al., 2021, Network Traffic Anomaly Detection via Deep Learning. *In MPDI Information and Communications Technology*, Vol. 12, No. 15, pp. 1-3.

Hung-Jen Liao, et al., Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, Volume 36, Issue 1, 2013, pp 16-24

J. Margolis, et al., "An In-Depth Analysis of the Mirai Botnet," 2017. *International Conference on Software Security and Assurance (ICSSA)*, 2017, pp. 6-12

K. Wei, et al., 2020, "Federated Learning With Differential Privacy: Algorithms and Performance Analysis". *In IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454-3469

Kfouri, et al., 2019, Design of a Distributed HIDS for IoT Backbone Components. *Federated Conference on Computer Science and Information Systems*

Max Petrov, et al., 2018, Adaptive performance model for dynamic scaling Apache Spark Streaming. *Procedia Computer Science*, Vol 136, pp. 109-117

O. Novo, "Blockchain Meets IoT: An Architecture for Scalable Access Management in IoT." *In IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1184-1195, April 2018,

Sakshi Indolia, et al., 2018, Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach. *Procedia Computer Science*, Vol 132, pp. 679-688

Stacey Truex, et al., 2019, A Hybrid Approach to Privacy-Preserving Federated Learning. *In Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security (AISec'19)*, USA, pp 1–11.

T. Li, et al., 2020, "Federated Learning: Challenges, Methods, and Future Directions," *in IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50-60

Thien Duc Nguyen. et al., 2019, DÏoT: A Federated Self-learning Anomaly Detection System for IoT. *Proceedings of the 39th IEEE International Conference on Distributed Computing Systems (ICDCS), 2019*, United States, pp. 1-8.

Waleed Bul'ajoul, et al., 2015, Improving network intrusion detection system performance through quality of service configuration and parallel technology. *Journal of Computer and System Sciences*, Vol 81, pp. 981-999

White, et al., 2013, Quantitative Analysis of Intrusion Detection Systems: Snort and Suricata. *Proceedings of SPIE - The International Society for Optical Engineering*, Vol 1

Xiaolu Zhang, et al., IoT Botnet Forensics: A Comprehensive Digital Forensic Case Study on Mirai Botnet Servers, *Forensic Science International: Digital Investigation*, Vol 32, 2020

Xinyou Zhang, Chengzhong Li and Wenbin Zheng, "Intrusion prevention system design" *The Fourth International Conference on Computer and Information Technology*, 2004, pp. 386-390