T800: ferramenta de firewall e benchmark para IoT

Pedro H Borges Monici[1,2], Gabriel Victor C Fernandes[1,2], César H de Araujo Guibo[1,2]
Gustavo de Carvalho Bertoli[1], Aldri Santos[3], Lourenço Alves Pereira Jr.[1]

[1] Divisão de Ciência da Computação
Instituto Tecnológico de Aeronáutica (ITA) – São José dos Campos, SP – Brazil

[2]Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo (USP) – São Carlos, SP – Brazil

[3]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte, MG – Brazil

{gabriel.fernandes,pedroh.monici}@ga.ita.br, {bertoli,ljr}@ita.br

cesarguibo@usp.br, aldri@dcc.ufmg.br

Abstract. *Cybersecurity is crucial for digital transformation as many computing assets are exposed on the network. In addition, there are few solutions to protect Internet of Things devices. Thus, this article presents the T800 packet filter to provide packet filtering using advanced algorithms with low resource consumption. The results show the efficiency of the T800 via implementation and experimentation through the ESP32 microcontroller. Furthermore, T800 increased the device's computational availability by excluding malicious traffic from the processing pipeline during attacks.*

Resumo. *Segurança Cibernética é crucial para a transformação digital, pois muitos ativos computacionais ficam expostos na rede. Além disso, observam-se poucas soluções para proteção de dispositivos de Internet das Coisas. Assim, este artigo apresenta o filtro de pacotes T800, capaz de filtrar pacotes com baixo consumo computacional. Os resultados evidenciam a eficiência do T800 por meio de implementação e experimentação através do microcontrolador ESP32. Mais ainda, o T800 foi capaz de aumentar a disponibilidade computacional do dispositivo tendo em vista que o tráfego malicioso é excluído do processamento durante os ataques.*

1. Introdução

Os avanços tecnológicos como a expansão das formas de conectividade, miniaturização dos componentes eletrônicos e aumento da capacidade de processamento dos processadores, pavimentaram o caminho para o que hoje é conhecido como Internet das Coisas (IoT). No contexto de IoT, cada vez mais objetos se encontram disponíveis na internet como eletrodomésticos, equipamentos de infraestrutura crítica, de saúde, entre outros. Dessa forma, novos modelos de negócios e aumento de eficiência operacional são possibilitados em diversos domínios [Shafique et al. 2020]. Em adição aos benefícios e vantagens da IoT, o aumento da conectividade resulta também em desafios de segurança devido ao aumento da superfície de ataque à esses dispositivos.

No contexto de segurança IoT, um caso típico é o comprometimento de diversos dispositivos IoT fazendo com que componham uma rede sob o controle de um agente malicioso externo, que normalmente a direciona para a realização de ataques coordenados contra alvos específicos. Essas redes são denominadas *botnets* [Bertino and Islam 2017]. Dispositivos de Internet das Coisas (IoT) são alvos comuns pelo fato de possuírem ciclo de atualização e manutenção precários, serviços não seguros (ex. telnet) ou acessos para configuração remota através de credenciais não seguras. Tais características podem ser observadas como parte das estratégias de ataque em *malwares* como o Mirai [Antonakakis et al. 2017]. Em adição a esses riscos e características dos dispositivos IoT, a maioria dos ataques se inicia pelas atividades de *scanning*, que buscam identificar dispositivos IoT visíveis e/ou vulneráveis por toda internet, principalmente quando novas vulnerabilidades são encontradas [Durumeric et al. 2014, Durumeric et al. 2015].

Neste artigo apresentamos uma implementação pontual do *T800*, um filtro de pacotes inteligente com o objetivo de se tornar um componente padrão na proteção de dispositivos IoT contra ataques de *scanning*. Tal implementação se encontra no contexto do arcabouço ESP-IDF baseado no microcontrolador ESP32, com uso do sistema operacional de tempo real FreeRTOS e da pilha TCP/IP LwIP. Em adição à ferramenta de filtragem, apresentamos também uma aplicação externa ao sistema como forma de avaliação (*benchmark*) do filtro. O *T800* busca endereçar as limitações das ferramentas atuais para segurança de dispositivos IoT. Essas ferramentas possuem como uma das limitações a pouca generalização de regras, sem a possibilidade de implementação de técnicas mais avançadas de detecção como o uso de algoritmos de aprendizado de máquina [Gupta et al. 2017, Ben Achballah et al. 2018]. A forma de realização do *benchmark* consiste de uma aplicação que utiliza as ferramentas *Iperf* e *Nmap* para a geração de tráfego benigno e malicioso, respectivamente. Ao executar esse *benchmark* obtemos a simulação da operação do dispositivo IoT com o *T800* em uma rede comprometida. Nesse cenário de simulação e através da aplicação disponibilizada é possível a extração de métricas referentes aos recursos computacionais do dispositivo IoT sob avaliação. Menciona-se que *T800* corresponde à ferramenta utilizada no artigo publicado na trilha principal do SBRC 2022 [Fernandes et al. 2022].

O artigo está organizado da seguinte maneira: Na Seção 2, descrevemos a arquitetura do *T800*. Na Seção 3 descrevemos a implementação do *T800* e sua integração com a Stack TCP/IPv4. Na Seção 4 descrevemos como se deu a demonstração do filtro. E por fim, na Seção 5 conclui-se este estudo.

2. Arquitetura do T800

O filtro *T800* contempla uma arquitetura para implementação de um firewall destinado a dispositivos de Internet das Coisas. Como ilustrado na Figura 1, trata-se de um componente a ser acoplado à pilha de protocolos de rede existente no sistema operacional. Sua funcionalidade principal é interceptar a entrada e a saída de pacotes e tomar decisões sobre permitir ou negar o processamento. Neste artigo tratamos da implementação do *T800* junto ao FreeRTOS, com a pilha de protocolos lwIP (lightweight IP) e SDK ESP-IDF, configurados especificamente para a plataforma de desenvolvimento Espressif ESP32. Assim, a abordagem de filtragem de pacotes consiste da avaliação dos cabeçalhos de pacotes do protocolo TCP/IP. Desse modo, o *T800* apresenta-se como uma ferramenta e possibilita a avaliação de diferentes cargas de trabalho. Testamos o desempenho por meio

de um estudo de caso que permite diferenciar o tráfego como maligno ou benigno.

O *T800* foi construído como um novo componente do arcabouço ESP-IDF, se tornando parte da biblioteca padrão. Com isso, introduzimos duas novas dependências: *esp-nn*[1], uma biblioteca oficial da Espressif (fabricante do ESP32) que implementa funções comuns para aprendizado de máquina otimizadas em Assembly; e a *tflite-lib*[2], biblioteca da Google que possibilita implementar modelos de Aprendizado de Máquina desenvolvidos no arcabouço *Tensorflow* com otimizações para transpor esses modelos para dispositivos de baixos recursos computacionais, por exemplo, na quantização dos pesos dos modelos finais.

Os resultados que o T800 apresenta já foram analisados anteriormente [Fernandes et al. 2022]. Dessa forma, sua implementação, descrita na Seção 3 e a arquitetura desenvolvida se mostraram eficientes e extensíveis, uma vez que a implantação do T800 na pilha TCP/IP do ESP-IDF proporcionou um baixo consumo computacional e uma alta generalização de regras, o que permite o uso de diferentes políticas de filtragem de pacote sem afetar de forma significativa os recursos computacionais do dispositivo.

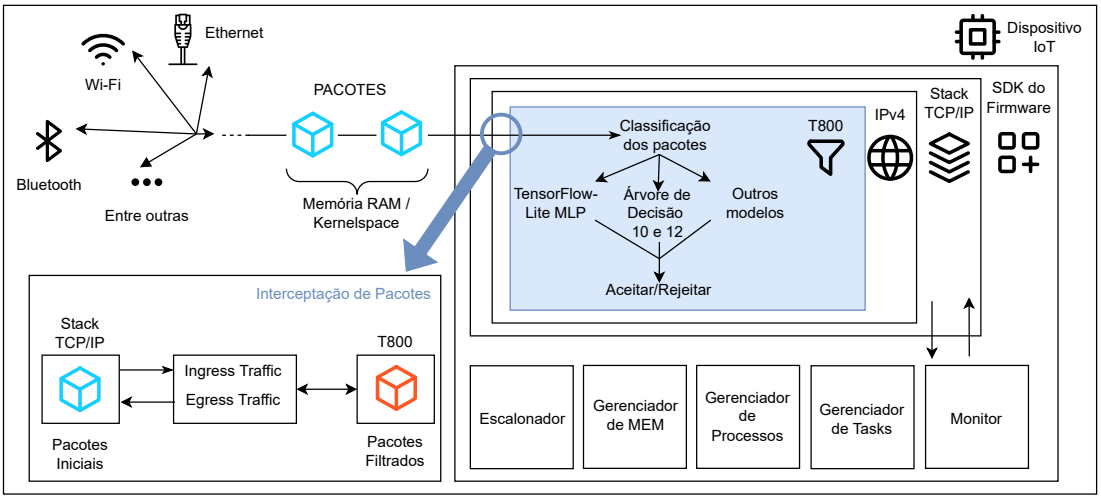

Figura 1. Arquitetura e contexto do filtro de pacotes T800.

Em um primeiro momento, o *T800* passa por uma etapa de inicialização. Essa etapa de inicialização acontece pela disponibilização de uma estrutura de configuração inicial, incluindo tanto uma função que classifica pacotes, quanto a informação sobre o contexto (estático ou dinâmico) em qual essa função é executada. Esse tipo de contexto é necessário pois o *T800* poderá operar de duas maneiras diferentes, sendo elas abordagens com ou sem a necessidade de guardar os estados anteriores do fluxo ao qual o pacote pertence (*stateless* ou *stateful*). Por outro lado, caso essa inicialização não seja concluída, o *T800* não exerce nenhuma influência sobre o sistema. Após a etapa de inicialização, o *T800* é executado, passando a atuar diretamente nas camadas de Rede (L3) e de Transporte (L4). A função que utiliza um modelo de aprendizado de máquina para realizar a classificação dos pacotes é escolhida e definida previamente na estrutura de configuração inicial. Sua entrada consiste em dados dos cabeçalhos TCP e IP, que se encontram presentes nos pacotes recebidos pela ESP32. A estrutura de dados responsável

[1]esp-nn: `https://github.com/espressif/esp-nn`
[2]Tensorflow Lite: `https://github.com/tensorflow/tflite-micro`

por representar um pacote é denominada `pbuf`, um tipo abstrato de dados definido pela lwIP [Dunkels 2001] que contém ponteiros para o *payload* da aplicação (L7) e os *headers* das camadas 2 a 4 (enlace, IP, TCP). Inclui também encadeamento para outros pacotes (fragmentação) quando necessário.

A execução do *T800* é orientada a eventos (chegada ou saída de pacotes) e ocorre de forma concorrente com o sistema. Assim, após a função de filtragem ser acionada, ocorre a classificação dos pacotes ou fluxos com base nos atributos disponíveis no `pbuf`. Portanto, se ocorre a marcação de tráfego malicioso, a estrutura `pbuf` é descartada e sua memória liberada. Caso contrário, o pacote segue seu processamento usual dentro do funcionamento do ESP-IDF.

2.1. Políticas de filtragem de pacotes

Uma política de filtragem de pacotes pode ser definida como um conjunto de regras que especificam se um pacote deve receber permissão para continuar o seu percurso natural dentro de uma camada de protocolos ou ser rejeitado. Assim, considerando o quão geral pode ser essa definição, existem inúmeras maneiras de se construir tal política. O *T800* utiliza o arcabouço denominado **AB-TRAP** (**A**ttack, **B**onafide, **T**rain, **R**ealization and **P**erformance) [Bertoli et al. 2021] que busca a concepção de mecanismos de proteção capazes de contemplar toda a cadeia de desenvolvimento, desde a caracterização de operação normal *versus* maliciosa até a implementação e avaliação de desempenho da solução proposta. Logo, as soluções para filtragem avançada de pacote testadas no *T800* seguiram essa metodologia, correspondendo ao processo de geração de dados dos ataques, treinamento dos modelos até a implementação e avaliação na placa baseada no ESP32.

Utilizamos três políticas de filtragem diferentes, pois dessa forma foi possível verificar a viabilidade de diferentes tipos de algoritmos de aprendizado de máquina. Elas foram selecionadas para testar a viabilidade operacional em uma plataforma de baixa capacidade computacional. Verificou-se também a aplicabilidade do *T800* em preservar a capacidade de generalização das regras. Duas políticas foram feitas com estruturas de Árvores de Decisão com profundidades 10 (DT-10) e 12 (DT-12). Já a última foi construída com uma *Multilayer Perceptron* (MLP) que possui duas camadas escondidas com 16 neurônios cada e uma camada de saída com dois neurônios. As árvores de decisão foram implementadas através de um encadeamento de estruturas condicionais, que se baseiam nos testes de atributos realizados pelos modelos originais. Essa simplicidade de interpretação e transposição dos modelos treinados para um encadeamento de estruturas condicionais as tornam propícias para o contexto de embarcados, pois elas incorrem em pouco uso de recursos computacionais. A *Multilayer Perceptron* (MLP), por outro lado, é especificada com uma função de ativação sigmóide para as camadas escondidas, e com uma função de ativação *softmax* para a camada de saída. Tanto sua implementação quanto seu treinamento foram feitos através do *Tensorflow* e empregamos o *Tensorflow-Lite* para poder ser utilizado na ESP32. Essa abordagem permitiu que o modelo tivesse otimizações de espaço e processamento, além de um desempenho satisfatório de inferência em dispositivos embarcados.

2.2. Conjunto de dados e treinamento dos modelos

Para a etapa de treinamento, as árvores de decisão empregaram a métrica de entropia como critério de divisão. Já a MLP foi treinada por 2000 épocas com o otimizador Adam,

uma taxa de aprendizado de 1.10^{-5} e um tamanho de lote de 260. O conjunto de dados aplicado para todas essas etapas de treinamento foi obtido pela metodologia AB-TRAP com uma pequena alteração que consistiu na remoção do atributo `tcp.window_size`. A partir desse processo foram obtidos os resultados de *F1-score* 0.93, 0.93 e 0.91 para a Árvore de Decisão com profundidade 10, profundidade 12 e a MLP, respectivamente.

3. Implementação do *T800*

A implementação do *T800* teve os seguintes objetivos principais: baixo custo computacional e capacidade de atualização das políticas de filtragem. Nesse contexto, a implementação do filtro estabelece uma interface padronizada que, com o uso de poucos recursos, facilita o desenvolvimento e a implantação de novas lógicas de filtragem. Tal interface é disponibilizada pela estrutura demonstrada na Figura 2. Ela possui apenas duas entidades essenciais: um valor responsável por codificar seu modo de funcionamento e uma função de classificação que impõe uma política de filtragem de pacotes. O modo de funcionamento define não apenas como o sistema irá agir para interceptar os pacotes, mas também quando ele deve executar a função de classificação e quais argumentos ela deve receber. Já a função de classificação recebe o contexto necessário para realizar a classificação e retorna a classificação do pacote. Dessa forma, levando em consideração critérios como tempo de execução e memória utilizada pelo dispositivo, o usuário consegue não somente escolher o modo mais adequado para sua aplicação, mas também selecionar a melhor política de filtragem para seu caso de uso através da função de classificação. Um detalhe importante é que essa abordagem permite a alteração de ambas as entidades em tempo de execução e, assim, atribui uma maior adaptabilidade ao filtro.

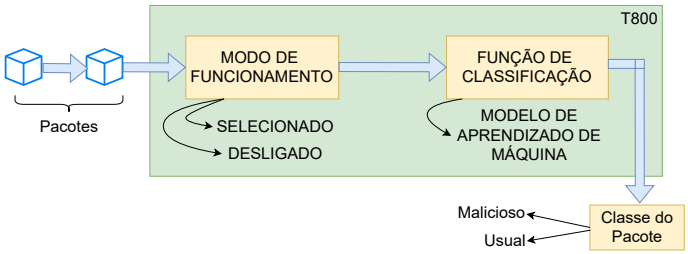

Figura 2. Estrutura da implementação do *T800*.

3.1. Extensibilidade do *T800*

Considerando toda a implementação do T800, percebe-se que a interface provida pela ferramenta no ESP-IDF confere extensibilidade ao filtro de pacotes. Através dela, políticas de filtragem como as descritas na Seção 2.1 que empregam a assinatura de um pacote podem ser implementadas. Portanto, diferentes modelos podem ser utilizados para classificar pacotes, uma vez que o *Tensorflow* está presente no *T800*, introduzindo assim modularidade e facilidades evidenciada nos mais diversos contextos de aplicação.

Os modelos de aprendizado de máquina que serão utilizados pelo *T800* devem contemplar apenas os cabeçalhos dos pacotes TCP, otimizando, assim, a quantidade de dados que serão processados dentro do dispositivo, uma vez que os *payloads* dos pacotes não precisam ser levados em consideração para a classificação. Ademais, o fato da grande maioria dos ataques de *scanning* não levarem em conta o fluxo dos pacotes de

uma certa comunicação permite que esta implementação inicial do *T800* não necessite considerar o contexto da comunicação TCP/IP sob avaliação (fluxo). Ainda assim, alguns modelos não podem ser implementados pelo fato de que o dispositivo possui restrições computacionais, então utilizar um modelo muito demandante desses recursos (e.g. KNN) pode comprometer o sistema. Portanto, o *T800* é uma alternativa personalizada para a defesa de dispositivos embarcados contra ataques cibernéticos de forma que independe do sistema computacional, podendo ser utilizado em qualquer sistema que utiliza a pilha TCP/IP LwIP, e que permite a coleta das métricas de desempenho de tais soluções com baixo custo computacional.

3.2. Coleta de métricas computacionais

A implementação do *T800* permite a aferição de diversas métricas de desempenho para o funcionamento da ferramenta. Como ilustrado na Figura 3, o filtro de pacotes integra-se com a pilha TCP/IP base do sistema, no caso, a LwIP. Primeiramente, os pacotes chegam pela pilha e logo são interceptados pela parte de instrumentação antes de chegarem ao *T800*. Em seguida, o *T800* é configurado com o seu modo de funcionamento para em seguida realizar a filtragem. Tal filtragem pode ser tradicional ou avançada. A primeira realiza regras estáticas de filtragem de pacotes, sem a necessidade de algoritmos muito complexos. Já a avançada, segue a abordagem realizada em nossos experimentos, com o auxílio de modelos de aprendizado de máquina. Finalmente, o pacote pode ser classificado ou não como malicioso, podendo seguir seu processamento no LwIP ou ser descartado. Essa integração permite que o *T800* tenha uma visão profunda do sistema, que permite a coleta de medidas de custo computacional.

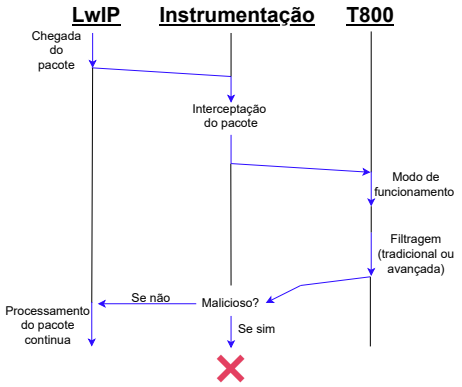

Figura 3. Interação do *T800* com a *stack* de protocolos do ESP-IDF, LwIP.

Para facilitar a análise de desempenho de diferentes configurações para o *T800* foi utilizado um aparato externo ao sistema que auxilia na instrumentação. Tal aparato consiste em um servidor UDP que realiza a coleta de várias métricas de desempenho computacional. Esse servidor utiliza o protocolo UDP pelo fato do *T800* agir apenas em pacotes TCP, logo, os dados de coleta nunca poderão ser perdidos ou bloqueados pelo *T800*. A Figura 4 ilustra o funcionamento desse serviço. A partir de uma ESP32 e uma máquina atacante na mesma rede para avaliação, elas se comunicam via protocolo UDP para gerenciar as configurações de uma conexão TCP que ficará ativa por 180 segundos. No passo (1) da execução, a ESP32 manda uma mensagem para a máquina atacante sinalizando o inicio do experimento. Em (2), o atacante envia um código que pode especificar

a política de filtragem do *T800* ou sinalizar que o componente não deve ser habilitado. No passo (3), a ESP32 responde comunicando que recebeu as informações necessárias e já realizou a sua configuração inicial. Nesse processo, dois servidores são inicializados na ESP32. Um é responsável por receber todo o tráfego TCP da simulação, enquanto o outro coleta as métricas de desempenho e as envia para a máquina atacante via UDP em períodos de 1 segundo. No quarto passo (4), o tráfego de rede é gerado pela máquina atacante. Por fim, depois da coleta de todas as métricas, no quinto passo (5), a máquina atacante manda um mensagem para a ESP32 que sinaliza o fim do experimento e, no sexto passo (6), a ESP32 envia uma resposta confirmando a finalização da simulação. Essa abordagem permite que a política de filtragem de pacotes seja alterada em tempo de execução – passo (2) – possibilitando a adaptabilidade da solução.

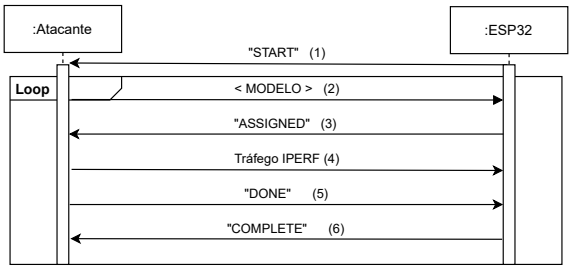

Figura 4. Diagrama de Sequência para a coleta de métricas da ESP32 durante a fase de homologação

4. Demonstração

A demonstração do filtro de pacotes ocorrerá utilizando um cenário de rede vulnerável com tráfego malicioso direcionado ao dispositivo embarcado que possui o *T800*. O dispositivo exercitará sua capacidade de tráfego nessa rede ao receber uma quantidade alta e fixa de dados de um host que executa uma ferramenta para geração de tráfego. Assim, atuarão na demonstração o dispositivo embarcado, no papel de exercitar sua capacidade de processar tráfego em uma rede vulnerável, e um host que envia alta quantidade de dados, enquanto recebe as métricas coletadas no dispositivo. Essas duas máquinas devem estar na mesma rede e serem capazes de estabelecer conexão. A execução desse ambiente pode ser reproduzida de acordo com os passos do manual[3], de acordo com o vídeo[4] , e o material está disponível no repositório online[5]. Para a reprodução ser realizada com êxito, deve ser possível atualizar o firmware da ESP32 com a implantação do *T800* no ESP-IDF, pela sua compilação e *flash* no dispositivo.

5. Conclusão

Os projetos construídos no arcabouço ESP-IDF, possuem uma fraca desenvoltura contra ataques cibernéticos. Assim, o *T800* foi construído para suprir tal escassez de ferramentas e métodos capazes de prevenir tais ataques. O fato do filtro permitir diferentes políticas de filtragem de pacotes auxilia na proteção das mais variadas aplicações IoT. Isso se torna evidente ao permitir uma grande generalização de regras de filtragem e por disponibilizar

[3]Manual: github.com/c2dc/t800-sbrc2022/blob/main/manual/manual.md
[4]Vídeo demonstração: `https://bit.ly/3LcPz7S`
[5]T800: `https://github.com/c2dc/t800-sbrc2022`

a implantação de praticamente qualquer modelo de aprendizado de máquina através do *Tensorflow Lite*. Além disso, o *T800* visa facilitar trabalhos na área de segurança de dispositivos embarcados que utilizam a pilha LwIP, permitindo a coleta de métricas de recursos computacionais e a generalização de políticas de filtragem de pacotes.

Agradecimentos

Este trabalho tem apoio financeiro do Programa de Pós-graduação em Aplicações Operacionais—PPGAO/ITA, da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) processos #2020/09850-0 e #2021/09416-1, e do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) processo n.o 157021/2021-1. Agradecemos os revisores anônimos pelas ótimas sugestões de melhoria deste artigo.

Referências

Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J. A., Invernizzi, L., Kallitsis, M., et al. (2017). Understanding the mirai botnet. In *26th USENIX security symposium (USENIX Security 17)*, pages 1093–1110.

Ben Achballah, A., Ben Othman, S., and Ben Saoud, S. (2018). Fw_ip: A flexible and lightweight hardware firewall for noc-based systems. In *2018 International Conference on Advanced Systems and Electric Technologies (IC_ASET)*, pages 261–265.

Bertino, E. and Islam, N. (2017). Botnets and internet of things security. *Computer*, 50(2):76–79.

Bertoli, G. D. C., Júnior, L. A. P., Saotome, O., Dos Santos, A. L., Verri, F. A. N., Marcondes, C. A. C., Barbieri, S., Rodrigues, M. S., and De Oliveira, J. M. P. (2021). An end-to-end framework for machine learning-based network intrusion detection system. *IEEE Access*, 9:106790–106805.

Dunkels, A. (2001). Design and implementation of the lwip tcp/ip stack. *Swedish Institute of Computer Science*, 2(77).

Durumeric, Z., Adrian, D., Mirian, A., Bailey, M., and Halderman, J. A. (2015). A search engine backed by internet-wide scanning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 542–553.

Durumeric, Z., Li, F., Kasten, J., Amann, J., Beekman, J., Payer, M., Weaver, N., Adrian, D., Paxson, V., Bailey, M., et al. (2014). The matter of heartbleed. In *Proceedings of the 2014 conference on internet measurement conference*, pages 475–488.

Fernandes, G. V. C., Monici, P. H. B., de Araujo Guibo, C. H., de Carvalho Bertoli, G., dos Santos, A., and Pereira, L. A. (2022). Implementação de um filtro de pacotes inteligente para dispositivos de internet das coisas. In *SBRC 2022 (to appear)*.

Gupta, N., Naik, V., and Sengupta, S. (2017). A firewall for internet of things. In *2017 9th International Conference on Communication Systems and Networks (COMSNETS)*.

Shafique, K., Khawaja, B. A., Sabir, F., Qazi, S., and Mustaqim, M. (2020). Internet of things (iot) for next-generation smart systems: A review of current challenges, future trends and prospects for emerging 5g-iot scenarios. *Ieee Access*, 8:23022–23040.