UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA - CIn
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

RICARDO HELISSON BEZERRA AMORIM

MONITORANDO REDES EM INTERNET DAS COISAS (IoT): segurança de baixo custo com Suricata

Recife-PE
2024

RICARDO HELISSON BEZERRA AMORIM

MONITORANDO REDES EM INTERNET DAS COISAS (IoT): segurança de baixo custo com Suricata

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de mestre(a) em Ciência da Computação. Área de concentração: Redes de Computadores.

Orientador: Prof. Dr. José Augusto Suruagy Monteiro

Recife-PE

2024

.Catalogação de Publicação na Fonte. UFPE - Biblioteca Central

Amorim, Ricardo Helisson Bezerra.
MONITORANDO REDES EM INTERNET DAS COISAS (IoT): segurança de baixo custo com Suricata / Ricardo Helisson Bezerra Amorim. - Recife, 2024.
107f.: il.

UNIVERSIDADE FEDERAL DE PERNAMBUCO, CENTRO DE INFORMÁTICA - CIn, PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO.
Orientação: José Augusto Suruagy Monteiro.

1. IoT; 2. DDoS; 3. Suricata; 4. Segurança; 5. Internet das Coisas; 6. Casa Inteligente. I. Monteiro, José Augusto Suruagy. II. Título.

UFPE-Biblioteca Central CDD 004

Ricardo Helisson Bezerra Amorim

“MONITORANDO REDES EM INTERNET DAS COISAS (IoT): segurança de baixo custo com Suricata”

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação. Área de Concentração: Redes de Computadores e Sistemas Distribuídos.

Aprovado em: 27/03/2024.

BANCA EXAMINADORA

_____

Prof. Dr. Carlos André Guimarães Ferraz
Centro de Informática / UFPE

_____

Prof. Dr. Frederico Araújo da Silva Lopes
Instituto Metrópole Digital / UFRN

_____

Prof. Dr. José Augusto Suruagy Monteiro
Centro de Informática / UFPE

(**orientador**)

Dedico todo o esforço deste trabalho a minha esposa, minha filha e minha mãe.

AGRADECIMENTOS

A Deus, por me fazer acreditar que seria possível cursar o mestrado e nunca me deixou perder as esperanças, mesmo diante de tantos obstáculos;
À minha esposa, por estar sempre ao meu lado e pelo apoio incondicional;
À minha filha, por sempre ter o olhar motivacional diário;
Ao Prof. Suruagy, que foi o professor que me orientou neste trabalho, muito obrigado pela paciência, pela empatia que teve comigo em tantos momentos difíceis que passei durante todo esse tempo;
À Universidade Federal do Pernambuco, em especial ao Centro de Informática (Cin), por ter me acolhido de forma calorosa e acolhedora.

“A menos que modifiquemos a nossa maneira de pensar, não seremos capazes de resolver os problemas causados pela forma como nos acostumamos a ver o mundo.” (ALBERT EINSTEIN apud UNITED STATES, 1995, p. 45).

RESUMO

A Internet das Coisas trouxe inegáveis avanços científicos em várias áreas do conhecimento e, por outro lado, foi protagonista do aumento do número de ataques DDoS mundialmente nos últimos anos, principalmente envolvendo dispositivos IoT residenciais. Diante desse cenário, os dispositivos IoT residenciais têm preferência para participar de ataques DDoS como *slaves* porque possuem falhas na segurança e alta disponibilidade, ou seja, geralmente estão funcionando por longos períodos e não dispõem de mecanismos robustos de segurança, devido a limitações de *hardware* e de *software*. Nesse sentido, a nova abordagem para a segurança em IoT envolve o desafio de produzir uma solução de segurança eficiente, com baixo custo e consumo de *hardware* limitado. Notando uma necessária atenção da comunidade científica nesse aspecto, o presente trabalho propõe uma solução de segurança de baixo custo com Suricata, nomeada de Block Suricata Anti-DDoS, capaz de alertar sobre as vulnerabilidades de segurança em dispositivos IoT de redes residenciais, com foco em ataques DDoS de inundação HTTP. Trata-se de uma solução gratuita capaz de enviar notificações via aplicativo Telegram ao administrador da rede doméstica. Na avaliação de desempenho, o computador de baixo custo com o Block Suricata Anti-DDoS foi considerado como sistema, e os alertas gerados foram considerados como serviço. Foram testadas diferentes configurações de carga do sistema, em três casas inteligentes. O melhor cenário consistiu na velocidade mínima de ataques gerados pelo *software* Low Orbit Ion Cannon (LOIC), 10 *threads* e a Casa Inteligente 3. O pior cenário consistiu na velocidade máxima de ataques gerados pelo LOIC, 1000 *threads* e a Casa Inteligente 1. No melhor cenário, obteve-se um consumo de memória de 26,4% e consumo de CPU de 34,7%, impactando na acurácia da detecção em 99,9%. No pior cenário, obteve-se um consumo de memória de 87% e consumo de CPU de 96%, impactando na acurácia da detecção em 5,58%. A avaliação de desempenho utilizando tráfego real em três casas inteligentes demonstrou uma média aritmética de 86,32% de acurácia do Block Suricata Anti-DDoS para detectar ataques DDoS de inundação HTTP, comprovando o benefício da solução de segurança proposta.

Palavras-chave: IoT; DDoS; Suricata; Segurança; Internet das Coisas; Casa Inteligente.

ABSTRACT

The Internet of Things has brought undeniable scientific advances in various areas of knowledge and, otherwise, has been the protagonist of the increase in the number of DDoS attacks worldwide in recent years, mainly involving residential IoT devices. Given this scenario, residential IoT devices are preferred to participate in DDoS attacks as slaves because they have security flaws and high availability, that is, they are generally operating for long periods and do not have robust security mechanisms, due to hardware limitations and of software. In this sense, the new approach to IoT security involves the challenge of producing an efficient security solution, with low cost and limited hardware consumption. Noting a necessary attention from the scientific community in this aspect, the present work proposes a low-cost security solution with Suricata, called Block Suricata Anti-DDoS, capable of warning about security vulnerabilities in IoT devices in residential networks, focusing on HTTP flood DDoS attacks. It is a free solution, with efficient code, capable of sending notifications via the Telegram application to the home network administrator. In the performance evaluation, the low-cost computer with Block Suricata Anti-DDoS was considered as a system, and the alerts generated were considered as a service. Different system load configurations were tested in three smart homes. In the best scenario, memory consumption of 26.4% and CPU consumption of 34.7% were obtained, impacting detection accuracy by 99.9%. In the worst case scenario, memory consumption of 87% and CPU consumption of 96% were obtained, impacting detection accuracy by 5.58%. The performance evaluation using real traffic in three smart homes demonstrated an arithmetic average of 86.32% accuracy of Block Suricata Anti-DDoS to detect HTTP flood DDoS attacks, proving the benefit of the proposed security solution.

Keywords: IoT; DDoS; Suricata; Security; Internet of Things; Smart Home.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABINC	Associação Brasileira de Internet das Coisas
API	*Application Programming Interface*
BIoT	*Bio Internet of nano-things*
Bot	*Robot*
CIoT	*Consumer Internet of Things*
CSA	*Central Security Assistant*
DDoS	Ataque de Negação de Serviço
DHCP	*Dynamic Host Configuration Protocol*
DNS	*Domain Name System*
FTP	*File Transfer Protocol*
GPG	*GNU Privacy Guard*
GUI	Interface Gráfica do Usuário
HIoT	*Human Internet of Things*
HTTP	*Hypertext Transfer Protocol*
ICMP	*Internet Control Message Protocol*
IDoT	*Identity of Things*
IDRS	Sistema de Detecção e Resposta a Intrusões
IDS	*Intrusion Detection System,* Sistema de Detecção de Intrusão
IDPS	Sistemas de Prevenção e Detecção de Intrusão
IIoT	*Industrial Internet of Things*
IKEv2	*Internet Key Exchange Version 2*
IoBNT	*Internet of bio-nano-things*
IoE	*Internet of Everything*
IoMCT	*Internet of Mission Critical Things*
IoMT	*Internet of Medical Things*, Internet das Coisas Médicas
IoNT	*Internet of Nano Things*
IoP	*Internet of People*
IoRT	*Internet of Remote Things*
IoT	*Internet of Things*, Internet das Coisas
KRB5	*Kerberos*
LOIC	*Low Orbit Ion Cannon*
IP	*Internet Protocol*

IPS	*Intrusion Prevention System*, Sistema de Prevenção de Intrusão
ISN	*Initial Sequence Number*
M2M	*Machine to Machine,* máquina a máquina
MiM	*Man-in-the-Middle*
MIT	*Massachusetts Institute of Technology*
mMTC	*Massive Machine-Type Communication*
NFS	*Network File System*
NTP	*Network Time Protocol*
QoS	Qualidade de Serviço
RDP	*Remote Desktop Protocol*
RFID	*Radio Frequency Identification*
RPL	*Routing Protocol For Low Power And Lossy Networks*
SDN	Rede Definida por Software
SIEM	Gerenciamento de Eventos e Informações de Segurança
SIP	*Session Initiation Protocol*
SMB	*Server Message Block*
SMNP	*Simple Network Management Protocol*
SMTP	*Simple Mail Transfer Protocol*
SSH	*Secure Socket Shell*
SSL	*Secure Sockets Layer*
SYM	*Pacote Synchronize*
TCP	*Transmission Control Protocol*
TFTP	*Trivial File Transfer Protocol*
TLS	*Transport Layer Security*
UDP	*User Datagram Protocol*
UFW	*Uncomplicated Firewall*
URL	*Uniform Resource Locator*
UTM	Sistemas de Gerenciamento Unificado de Ameaças
VLAN	*Virtual Local Area Network*
VoIP	Voz sobre IP
WIoT	Dispositivos IoT vestíveis, *Wearable IoT*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo expõe os principais conceitos envolvendo Internet das Coisas (IoT) e a segurança de rede nesses dispositivos. Na seção de motivação e justificativa, é abordada a relevância da segurança da informação aplicada ao contexto de IoT. A próxima seção trata do problema de pesquisa, na qual são discutidos os desafios mais comuns no quesito de implementar a segurança em dispositivos IoT. Já na seção de objetivos, é proposta a solução para o problema de pesquisa apresentado. Por fim, são abordadas a metodologia e a estrutura desta dissertação, visando à melhor compreensão.

1.1 VISÃO GERAL SOBRE INTERNET DAS COISAS (IOT)

A Internet das Coisas (IoT) é um conceito revolucionário que permite uma conexão avançada máquina a máquina (M2M), incluindo sistemas, sensores inteligentes e dispositivos (DJEHAICHE et al., 2023). A Internet das Coisas (IoT) tem revolucionado o modo de ver e de fazer diversas tarefas do cotidiano. O sonho da automação deixou o plano onírico para se tornar realidade na palma da mão: drones, *smartwatches*, *smartphones* capazes de controlar casas inteligentes, dentre outros exemplos. O futuro idealizado em *Uma Odisséia no Espaço* (1968) está aqui e agora.

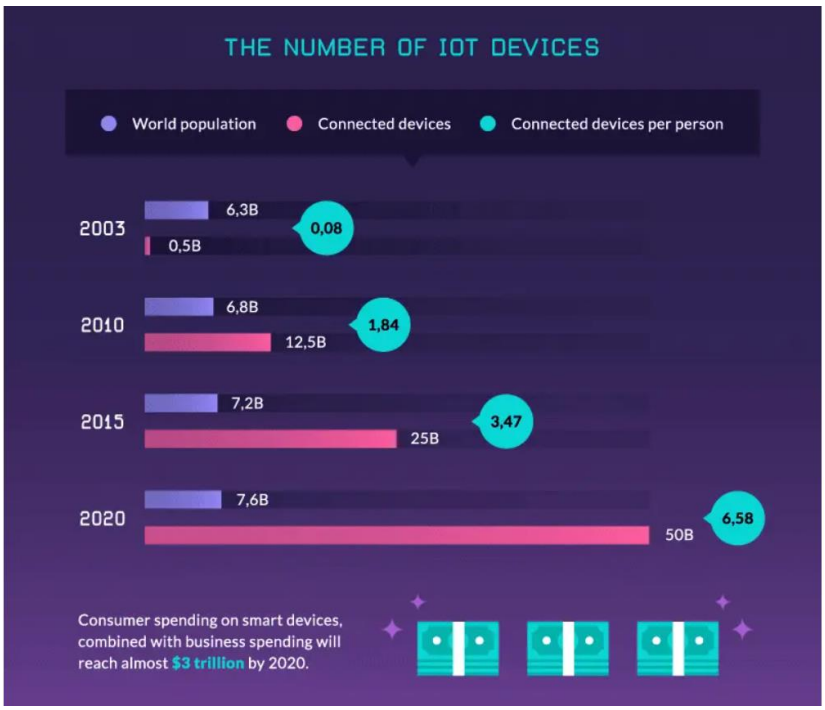
Assim, a Internet das Coisas (IoT) tem a sua origem com a utilização de dispositivos com comunicação máquina a máquina (M2M), na qual os dispositivos são capazes de se conectarem uns aos outros e de transmitir dados de forma automatizada por meio de redes sem fio, sem necessitar de interferência humana. Alguns autores afirmam que o uso de M2M data da Segunda Guerra Mundial, na qual pilotos identificavam amigos e inimigos, dessa forma evitando atingir aliados (ABDUL-QAWY; TADISETTY, 2015).

Nesse sentido, para alguns autores, M2M e IoT são conceitos bastante semelhantes, havendo atualmente uma pequena diferenciação na finalidade de ambos. Dessa forma, o conceito de M2M é mais utilizado com sentido industrial, enquanto o conceito de IoT está mais voltado aos consumidores finais. Por outro lado, IoT também significa uma tecnologia M2M que se comunica usando redes IP (ABDUL-QAWY; TADISETTY, 2015). Assim, os conceitos de M2M e IoT possuem estrita correlação, muitas vezes, se sobrepondo.

De acordo com Pataca (2021), a evolução que a Internet das Coisas proporcionou consiste na quebra de paradigma da comunicação entre computadores, para uma nova tipologia de comunicação, a qual abrange objetos do cotidiano, ou seja, *things* ou coisas. Então, as *coisas* do cotidiano passam a receber ordens, executá-las e transmitir dados.

Nas últimas décadas, a Internet das Coisas se tornou muito presente no mundo inteiro. O Gráfico 1 a seguir exemplifica essa mudança progressiva, havendo no ano de 2003, 0,5 bilhão de dispositivos IoT, para 6,3 bilhões de pessoas no mundo, totalizando 0,08 dispositivos por pessoa. Tais estatísticas foram se modificando ao longo dos anos, tanto que no ano de 2010 o número de dispositivos IoT superou o número da população mundial. Já em 2020, observa-se mais de 50 bilhões de dispositivos conectados, em contraste com uma população de apenas 7,6 bilhões de pessoas, ou seja, foi alcançado um marco de aproximadamente 6 IoTs por pessoa (SAFEATLAST, 2022).

Gráfico 1 – Dispositivos IoT e a População Mundial



Fonte: Safeatlast (2022).

Para Abdul-Qawy e Tadisetty (2015), essa grande expansão no número de dispositivos IoT é decorrente de três fatores principais. Primeiramente, a capacidade de desenvolver dispositivos de uso doméstico ou industrial com boa capacidade de processamento e baixo custo. Em segundo lugar, cita-se o advento da internet,

servindo como plataforma global de comunicação. Por último, verifica-se preços mais acessíveis das tecnologias sem fio. Tais fatos contribuíram bastante para o incremento mundial do número de dispositivos que são utilizados em Internet das Coisas.

Tal realidade, marcada por um número expressivo de dispositivos IoT, também impactou na economia dos países. O estudo realizado por Safeatlast (2022) verificou que consumidores e empresas gastaram quase 3 trilhões de dólares em 2020 apenas com dispositivos IoT. Dessa forma, além de constituir um salto tecnológico, a Internet das Coisas representa um nicho expressivo na economia mundial.

Assim, destacam-se *smartphones*, itens de automação doméstica e industrial, de monitoramento ambiental e soluções em cidades inteligentes. Tal progresso científico pode ser conferido em vários países mundialmente, desde o âmbito residencial até o âmbito empresarial. Aliado à ampla utilização da Internet das Coisas, surgiram áreas específicas nas quais a atuação dos IoT é relevante. A Tabela 1 abaixo apresenta algumas dessas áreas.

Tabela 1 – Tipos de Dispositivos IoT

Tipo	Definição	Autor (es)
IoT	*Internet of Things* – é o conceito em que objectos físicos interactuam entre sí através da Internet e podendo identificar-se mutuamente.	TSOUMANIS et al., (2020), ASHOURI et al., (2019)
BIoT	*Bio internet of nano-things* – é focada na interconexão de nano dispositivos de biotecnologia capazes de estabelecer comunicação efectiva.	MIRAZ et al., (2018)
CIoT	*Consumer Internet of Things* – é focada na interconexão de dispositivos em ambiente doméstico proporcionando a comunicação entre sí em correspondência com as necessidades do consumidor. Em 2020 a CIoT será a 3ª maior indústria em termos de IoT, e 70% de dispositivos vestíveis são dedicados ao *fitness*.	FAGAN et al. (2019), REN et al. (2019), MIRAZ et al., (2018)
HIoT	*Human Internet of Things* – é focada na utilização de dispositivos e objectos	MIRAZ et al., (2018)
	pessoais inteligentes que se comunicam por internet.	
IDoT	*Identity of things* – é focada a identidade das coisas e dispositivos interconectados à internet.	MIRAZ et al., (2018)
IIoT	*Industrial Internet of Things* - refere-se a um sistema interconectado de dispositivos inteligentes em um ambiente industrial que conecta recursos, incluindo sensores, atuadores, controladores e máquinas, como um sistema de controlo inteligente.	JANSEN e MERWE (2020), CHALAPATHI et al. (2019), KAMIENIECK et al. (2019), BOYES et al., (2018)

Tipo	Definição	Autor (es)
IoBNT	*Internet of Bio-nano-things* – é focada na interconexão de nano dispositivos de biotecnologia capazes de estabelecer comunicação efectiva, usados na medicina.	MIRAZ et al., (2018)
IoE	*Internet of Everything* – é a rede de interconexão entre objectos inteligentes (smart things), pessoas, processos e dados, em tempo real.	CHAUHAN e JAIN (2019), SHOJAFAR e SOOKHAK (2020), LIU et al., (2020)
IoMCT	*Internet of Mission Critical Things* - é motivada pela convergência da detecção, comunicação, processamento e controlo de dispositivos e sensores em missões críticas ou circunstâncias extremamente complicadas.	SRINIVASAN et al., (2019)
IoMT	*Internet of Mobile Things* – é focada na interconexão de dispositivos e sensores que estabelecem a comunicação na mobilidade e em repouso.	HAMMOOD et al. (2019), SRINIVASAN et al. (2019)
IoNT	*Internet of Nano Things* – é focada na interconexão de nano dispositivos, formando uma rede de comunicação.	MIRAZ et al., (2018)
IoP	*Internet of People* – é focada na interconexão das pessoas através da rede de Internet.	LANGLEY et al., (2019), KHARCHENKO (2019)
IoRT	*Internet of Remote Things* – é focada na interconexão de dispositivos remotos,	BACCO et al., (2017)
	através da rede de Internet. Geralmente usam a comunicação via satélite.	

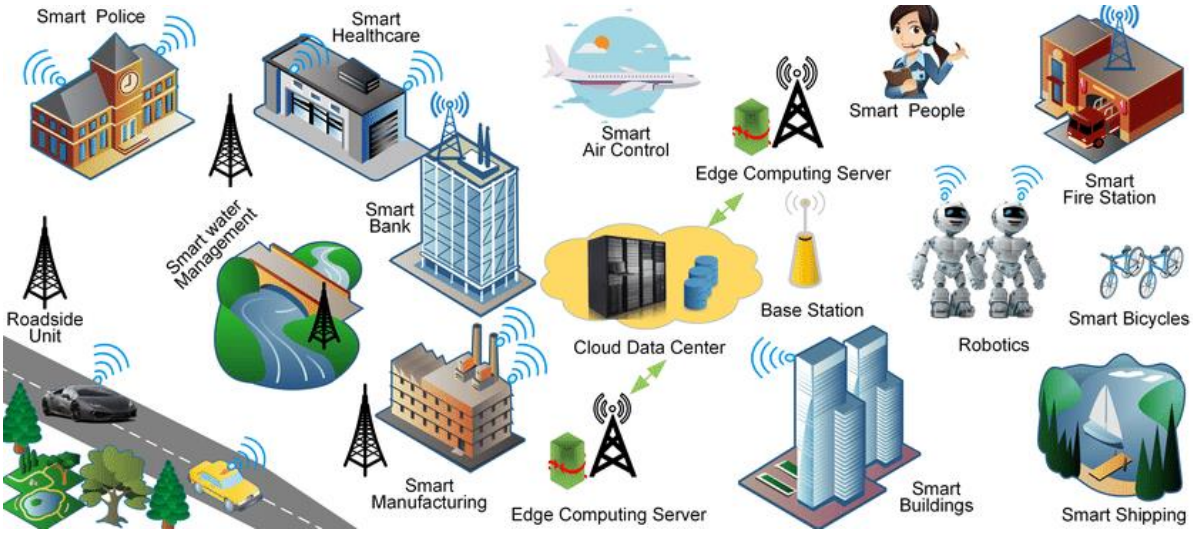
Fonte: Pataca (2021, adaptado).

Assim, foram criadas diversas tipologias de IoT, como a Internet das Coisas para o Consumidor (CIoT), que se caracteriza por dispositivos de uso doméstico; Internet das Coisas Industriais (IIoT), incluindo, por exemplo, sensores e máquinas; Internet das Coisas Nano-Biológicas (IoBNT), a qual se caracteriza por nano-sensores fabricados para coletar dados sobre o corpo humano; Internet das Coisas Remotas (IoRT), em que os dispositivos se comunicam através de satélites, dentre outras (PATACA, 2021).

Diante do exposto, nota-se a intensa repercussão que a Internet das Coisas trouxe para os diversos segmentos da sociedade, incluindo os consumidores finais, indústrias, área médica, bem como área nano-biológica. A capacidade de adaptação dos IoT a diversas finalidades é um destaque da tecnologia, além de apresentar baixo custo e conexão à internet, que por sua vez abre portas para um mundo de possibilidades. Assim, observa-se atualmente uma evolução no conceito de comunicação máquina a máquina, transformando-se em dispositivos essenciais para o cotidiano de bilhões de pessoas.

Nesse sentido, a Internet das Coisas se tornou presente na vida de milhões de pessoas, abrangendo diferentes áreas. Por exemplo, na área residencial, existe o conceito de casas conectadas, as quais são dotadas de monitoramento de energia, sensores inteligentes, dispositivos de mídia e computação, todos conectados à rede doméstica. Além disso, outras aplicações possíveis são na área automotiva, aviação, saúde, infraestrutura de rede, abastecimento de água, indústria e até mesmo em construções inteligentes, conforme pode-se observar na Figura 1 adiante.

Figura 1 – Dispositivos IoT



Fonte: RAFIQUE, et al. (2020).

Segundo Saarikko, Westergren e Blomquist (2017), a promessa de um futuro hiperconectado possui consequências, as quais devem ser avaliadas, sob pena de não se aproveitar todos os benefícios do IoT. Por esse ângulo, observa-se que há em um primeiro momento grande entusiasmo com o surgimento intenso de dispositivos IoT, mas esse conceito ainda não se encontra inacabado, pelo contrário, mais linhas de pesquisa precisam abranger e compreender sobre ferramentas de segurança envolvendo o mundo IoT.

Nesse cenário, idealizando a Internet das Coisas como a tecnologia do futuro, mas que ainda necessita de melhorias no seu desenvolvimento, surge um cenário preocupante de vulnerabilidades em torno da Internet das Coisas, demonstrando que há ainda muito a se investir e a se explorar nesse campo. Um exemplo dessa necessidade foi descrito por Tredinnick e Laybats (2018) em um ambiente de cassino, cujo tanque de peixes conectado à internet foi alvo de um ataque de rede bem-sucedido, resultando no furto de informações valiosas da empresa.

Considerando essa realidade, pesquisadores do Massachusetts Institute of Technology (MIT) estudaram utilizar chips capazes de processar criptografia baseada em treliça, um tipo de criptografia quântica, para aumentar a segurança em dispositivos (BANERJEE; PATHAK; CHANDRAKASAN, 2019). Essa nova arquitetura poderá ser integrada em dispositivos IoT na era da computação quântica, ou seja, soluções têm sido buscadas para a problemática envolvendo a Internet das Coisas e os ataques cibernéticos no futuro.

Enquanto a era quântica não ocorre, soluções mais baratas são imperativas. Hassan et. al (2021) afirmam que o alto custo é uma grande barreira para se adotar uma estratégia de segurança no âmbito da Internet das Coisas. Portanto, um mecanismo de segurança ideal para dispositivos IOTs deve ser acessível e de baixo custo.

Diante do exposto, o futuro hiperconectado proporcionado pela Internet das Coisas traz benefícios inegáveis à sociedade moderna. No entanto, também tem se tornado alvo de invasões e acessos indevidos, demonstrando que existe um risco em potencial nos dispositivos IoT utilizados por milhares de pessoas ao redor do mundo. A prevenção para tal problema requer tecnologias de baixo custo e de instalação simplificada, as quais possam minimizar as vulnerabilidades nas redes IoT, oferecendo uma experiência mais segura aos usuários.

1.2 MOTIVAÇÃO E JUSTIFICATIVA

Atualmente, a segurança em IoT vem ganhando destaque em diversas linhas de pesquisa, principalmente por não ser possível aplicar ferramentas de segurança já existentes que exijam alta capacidade de processamento (MEYER; GEMMER; ANDRADE; MELLO; NOGUEIRA; WANGHAM, 2022). Nesse sentido, há que se aprofundar as pesquisas na área, visando à concepção de técnicas de segurança que tornem o uso dos dispositivos IoT mais seguros, considerando-se as limitações de *hardware*.

Além das limitações de *hardware*, os dispositivos IoT também apresentam limitações de *software*. Woei-Kae *et al.* (2018) destacam que esse fato dificulta a criação de softwares para IoT, já que existe a possibilidade de haver incompatibilidade com o próprio dispositivo, bem como entre o dispositivo IoT e os outros dispositivos conectados a ele, por exemplo, entre sensores e o *gateway* responsável por receber e processar as informações.

Para além das dificuldades citadas, para Yadav e Yadav (2018), o quesito segurança é o maior desafio no uso dos IoT. Para chegar a essa conclusão, os autores elaboraram um levantamento dos principais riscos e desafios na implementação da Internet das Coisas na Índia. À medida em que o número de dispositivos IoT em utilização aumenta, as vulnerabilidades também se multiplicam. Outro fato que contribui para o desafio de segurança em IoT é o fato de existirem milhões de dispositivos semelhantes. Dessa forma, uma falha na segurança de um dispositivo está presente em milhões desse mesmo tipo, o que significa replicação dos riscos de segurança em vários dispositivos.

Outro desafio citado no estudo indiano é a manutenção da privacidade, já que o acesso indevido a informações coletadas pelos dispositivos IoT resultam em prejuízo ao usuário. Esse é mais um dos motivos para que as técnicas de segurança sejam otimizadas, com o objetivo de evitar o vazamento de dados sensíveis.

Diante desse cenário, o aumento no uso da Internet das Coisas é marcado também por desafios, dentre eles as limitações de *hardware* e *software*, a replicabilidade dos riscos de segurança em vários dispositivos e o acesso indevido a informações sensíveis, decorrentes das falhas na segurança. Assim, urge em se estabelecer medidas de segurança que possuam baixo custo, consumo de *hardware*

limitado, compatibilidade de *software* e efetividade em coibir ataques em dispositivos IoT.

1.3 PROBLEMA

A segurança em Internet das Coisas é motivo de preocupação no mundo inteiro. É notável a participação que a Internet das Coisas representa em ataques cibernéticos e exploração de vulnerabilidades.

O estudo de Koroniotis, Moustafa e Sitnikova (2019) revelou que 80% dos dispositivos IoT possuem problemas com a privacidade, e que 60% possuem problemas com a segurança. Além disso, um alvo comum de ataques são os *smart plugs*, os quais oferecem automação a dispositivos que não são IoT. O mesmo estudo ainda demonstrou que os dispositivos IoT têm preferência para participar de ataques DDoS como *slaves* porque possuem baixa segurança e alta disponibilidade, ou seja, geralmente estão funcionando o tempo todo e não dispõem de mecanismos robustos de segurança.

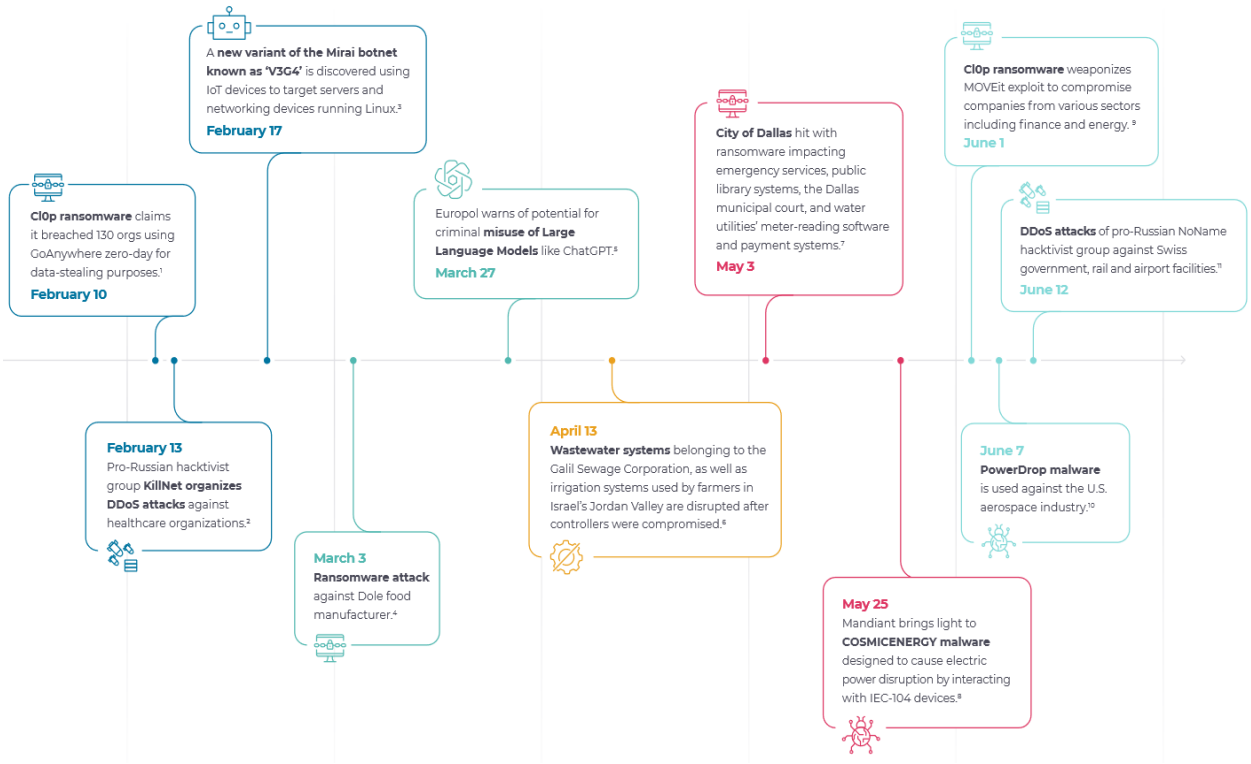
Nesse sentido, Mazhar et. al. (2023) destaca a necessidade de uma nova abordagem para a segurança em IoT, considerando ser um desafio produzir uma solução de segurança eficiente, com baixo custo e baixa demanda de processamento. Além desse desafio, soma-se o fato de que a Internet das Coisas pressupõe dispositivos que se conectam com inúmeros outros dispositivos, aumentando assim a vulnerabilidade a ataques.

Destaca-se também o assunto da segurança dos dispositivos IoT no cenário das redes de comunicação de 5ª geração, ou redes 5G. Um dos pilares das redes 5G é o *Massive Machine-Type Communication* (mMTC), o qual permite que uma quantidade massiva de dispositivos IoT se conectem diretamente na rede. Quando se considera esse cenário, com inúmeros dispositivos IoT vulneráveis com limitações de memória e processamento conectados a uma rede 5G, as possibilidades de ataques se multiplicam (VALADARES et. al, 2022). Dessa forma, o advento da rede 5G proporciona uma quantidade massiva de dispositivos vulneráveis conectados, o que representa mais pontos fracos na rede e mais ataques, pois os dispositivos IoT constituem em pontos passíveis de violação.

Outro dado preocupante é o destaque que os ataques utilizando IoT têm representado mundialmente em 2023. Os setores mais afetados foram o hospitalar, o

fabril, o de água e o de energia, ou seja, áreas críticas (NOZOMI NETWORKS, 2023). Abaixo, na Figura 2, é possível identificar ataques DDoS que culminaram na suspensão de atividades de ferrovias e de aeroportos na Suíça, indústria aeroespacial nos Estados Unidos, violação de dados em mais de 130 instituições de saúde, surgimento de uma nova variante do *botnet* Mirai V3G4 que se utiliza de dispositivos IoT para atacar servidores Linux, paralisação na fábrica de alimentos Dole, ataque a diversos sistemas de água, comprometimento da irrigação no Vale do Jordão em Israel, parada do *software* de leitura de medidores de concessionárias de água e sistemas de pagamento na cidade de Dallas, dentre outros eventos negativos.

Figura 2 – Cronograma de ataques em 2023



Fonte: Nozomi Networks (2023, p.3).

Esses dados são apenas uma demonstração do prejuízo que pode ser causado pela quebra de segurança em dispositivos IoT. Tais problemas ultrapassam a esfera financeira, impactando também grave dano humanitário ao interromper serviços essenciais. O relatório ainda aponta que os ataques DDoS são e têm a tendência de continuarem a ser o principal tipo de ataque direcionado a dispositivos IoT, posto que essa modalidade de ataque pode ser feita em escala, bem como se aproveita do fato

de os dispositivos IoT terem poder computacional baixo e estarem amplamente distribuídos mundialmente (NOZOMI NETWORKS, 2023).

Diante do exposto, revela-se imperativo dispor de ferramentas de segurança em Internet das Coisas, que apresentem baixo custo. Nesse contexto, o problema da presente pesquisa envolve as vulnerabilidades de segurança dos dispositivos IoT residenciais na sociedade moderna.

1.4 OBJETIVO

O presente trabalho visa à mitigação de vulnerabilidades de segurança em dispositivos IoT de redes residenciais, com foco em alertar sobre ataques DDoS de inundação HTTP.

Para alcançar tal objetivo, serão realizadas as seguintes atividades:

- Verificar qual sistema operacional gratuito é o mais compatível com hardwares domésticos;
- Identificar sistemas gratuitos com baixo processamento e baixo consumo de memória que sejam capazes de identificar ameaças na rede doméstica;
- Prover meios que possibilitem ao usuário comum verificar em uma interface amigável todos os eventos de alertas relacionados aos ataques DDoS na sua rede doméstica;
- Codificar uma solução capaz de enviar alertas de ataques DDoS para o administrador da rede via Telegram;
- Criar, implementar e disponibilizar gratuitamente a solução de segurança chamada de Block Suricata Anti-DDoS;
- Analisar o comportamento do *hardware* e *software* em cenários distintos (consumo de memória e de CPU) e como essas variações influenciam na taxa de detecção de ataques DDoS de inundação HTTP usando a solução proposta Block Suricata Anti-DDoS.

1.5 METODOLOGIA

Esta seção apresenta a metodologia seguida por esta dissertação. Primeiramente, quando se trata de objetivos, o presente trabalho se classifica como exploratório. Conforme Zikmund (2000), a utilização do estudo exploratório é útil quando se pretende analisar uma situação e gerar ideias novas. Nessa configuração, no presente trabalho há a criação de uma solução de segurança, submetida a um estudo de caso com a finalidade de passar por testes de segurança.

Quanto à natureza da pesquisa, pode se classificar como aplicada, posto que visa a procura de soluções para os problemas mapeados, considerando-se que tais problemas são originários de demandas da comunidade. Assim, a presente pesquisa vem com o intuito de inovar e impactar na realidade, suprindo demanda na área de segurança em IoT, ampliando ideias e soluções possíveis (FLEURY; WERLANG, 2017).

Sob a ótica da abordagem de problema, o método escolhido é o quantitativo, no qual é avaliado como as variações nas métricas dos experimentos afetam o desempenho da solução proposta.

Quanto aos procedimentos metodológicos, inicialmente foi conduzida uma pesquisa bibliográfica de artigos publicados entre os anos 2020 a 2023, visando buscar os trabalhos mais recentes envolvendo a segurança em dispositivos IoT. Foram selecionados artigos relevantes que utilizam o fluxo de dados para detectar ataques contra a segurança de dispositivos IoT. Foram utilizadas as palavras-chave “security iot flow” nas fontes IEEE, Springer Link, Elsevier, ACM, Wiley Online Library e Multidisciplinary Digital Publishing Institute (MDPI). As fontes foram acessadas pela *internet*, nos idiomas inglês e português.

Os critérios de inclusão e exclusão consistiram em artigos publicados entre 01/01/2020 a 04/06/2023; artigos que estão disponíveis *on-line*; revisados por pares; apresentar uma solução de segurança em IoT baseada no fluxo; solução aplicável no cenário residencial.

Os resultados obtidos encontram-se expostos na Tabela 2 abaixo. Ao todo, foram 268 artigos distribuídos entre as bases de pesquisa relacionadas. Após análise dos critérios de exclusão e retirada dos artigos repetidos, foram excluídos 252 artigos. A próxima etapa consistiu em analisar a adequação aos critérios de inclusão, resultando em 16 artigos adequados. Dentre esses, restaram selecionados 4 artigos

relevantes para o presente trabalho, os quais serão apresentados no Capítulo 3 como Trabalhos Relacionados.

Tabela 2 – Resultados da Pesquisa Bibliográfica

Bases de Pesquisa	Resultados obtidos	Removidos pelos critérios de exclusão ou repetidos	Adequados aos critérios de inclusão	Selecionados
IEEE	59	57	2	0
Springer Link	44	41	3	0
Elsevier	94	90	4	2
ACM	1	0	1	0
Wiley Online Library	16	14	2	0
MDPI	54	50	4	2
TOTAL	268	252	16	4

Fonte: elaborada pelo autor

Com base nesses trabalhos selecionados, foi idealizado um cenário residencial e elencadas as características relevantes para mitigação de ataques nesse cenário. Após essa etapa, houve a definição das ferramentas, *softwares* e *hardwares* a serem utilizados, descritos na seção 4.

A etapa seguinte consistiu em implementar a ferramenta de segurança, com o intuito de diminuir a vulnerabilidade de segurança dos dispositivos IoT da rede doméstica. Posteriormente, foram realizados experimentos, descritos na seção 5, visando à avaliação de desempenho da ferramenta de segurança implementada.

1.6 ESTRUTURA DA DISSERTAÇÃO

Os capítulos seguintes deste trabalho apresentam-se organizados da seguinte forma: no capítulo 2, são expostos os conteúdos que constituem o embasamento desta pesquisa. Já no capítulo 3, destacam-se os trabalhos relacionados à presente dissertação. No capítulo 4, são expostas a proposta, a arquitetura e a sua implementação. Por sua vez, o capítulo 5 tem como foco os a avaliação de

desempenho da solução proposta Block Suricata Anti-DDoS. Por último, no capítulo 6 são descritas as conclusões alcançadas com esta dissertação, bem como as contribuições e os trabalhos futuros.

2 EMBASAMENTO

O presente capítulo aborda os principais temas envolvendo esta dissertação, tais como uma visão geral sobre a segurança em Internet das Coisas, as principais ameaças em IoTs, os ataques distribuídos de negação de serviço, os Sistemas de Detecção e Prevenção de Intrusões e sua relação com os IoTs, os Sistemas de Análise de Dados e, por fim, as ferramentas de geração de Ataques de Negação de Serviço Distribuídos.

2.1 SEGURANÇA EM INTERNET DAS COISAS

2.1.1 Visão Geral

Atualmente, a Internet das Coisas (IoT) está causando uma revolução na sociedade em um nível global. Entretanto, a utilização massiva de dispositivos IoT também traz consigo uma variedade de desafios relacionados à segurança (NATH; NATH, 2022). Portanto, faz-se necessário adotar estratégias para entendimento e combate às ameaças de segurança em IoT.

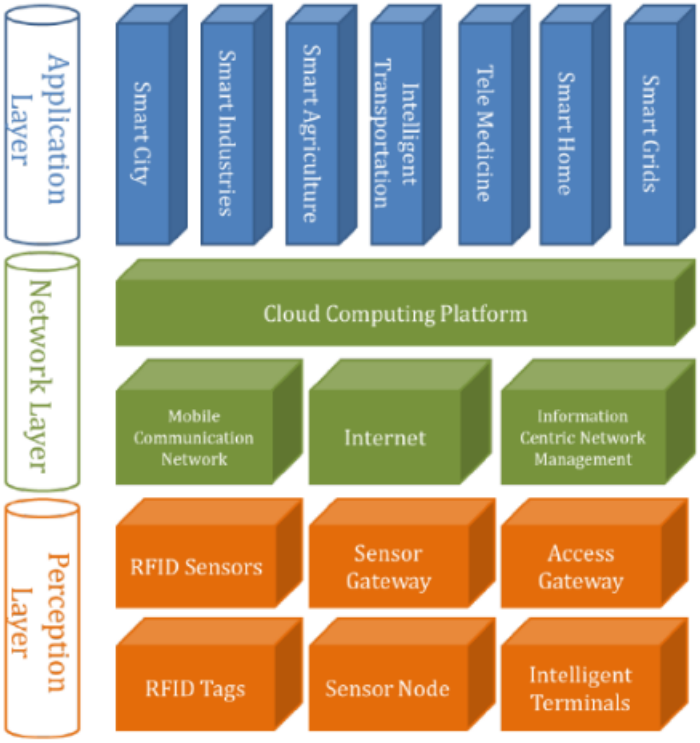
Os principais desafios na segurança de IoT descritos na literatura atual envolvem o vazamento de informações, a espionagem, autenticação, bloqueio de acesso, adulteração de dados, autorização e privacidade. Dentre os últimos ataques, destaca-se o *malware* Mirai, o qual se utilizou de um ataque DDoS com *botnet* para atacar IoTs, tais como câmeras IP, dispositivos residenciais, impressoras, dentre outros (JURCUT; RANAWEERA; XU, 2019).

2.1.2 Principais Ameaças em IoTs

Atualmente, há a tendência de estratificar as principais ameaças com base em uma classificação de camada à qual o dispositivo IoT pertence. Jurcut, Ranaweera e Xu (2019) expuseram três camadas, quais sejam, aplicação, rede e percepção. A Figura 3 abaixo representa tais camadas, que serão pormenorizadas nas subseções adiante. Em azul, está a camada de aplicação, na qual se encontram os dispositivos IoT relacionados à aplicação final, como cidades inteligentes, distribuição de energia, sistema de transportes inteligente, agricultura inteligente, dentre outros. Em cor verde,

se encontra a camada de rede, em que se classificam os IoTs que costumam fazer conexão com outras redes, como *smartphones*. Já a camada de percepção, representada na cor laranja, contempla dispositivos como sensores e *tags* RFID, responsáveis por perceber e coletar dados.

Figura 3 – Arquitetura de dispositivos IoT



Fonte: Jurcut, Ranaweera e Xu (2019)

2.1.2.1 Camada de Aplicação

A camada de aplicação consiste nas diferentes aplicações para IoT, por exemplo, *smart grids* para distribuição de energia, dispositivos vestíveis, transporte, agricultura, dentre outros. Os principais ataques nesta camada podem ser classificados em duas vertentes: ataques de *software* e ataques de criptografia (JURCUT; RANAWEERA; XU, 2019).

De acordo com Jurcut, Ranaweera e Xu (2019), os ataques de *software* incluem *phishing, worms, spyware* e cavalos de Troia, geralmente com o objetivo de capturar a senha de acesso dos usuários. Já os ataques de criptografia visam à exploração dos protocolos de criptografia, como ataques de texto cifrado e de texto

simples conhecido. Os ataques de texto cifrado ocorrem quando o atacante possui acesso às mensagens cifradas, porém não possui as mensagens originais, nem a chave de criptografia que foi utilizada. Por sua, vez, nos ataques de texto conhecido o atacante possui acesso a mensagens criptografadas e às mensagens originais, tentando decifrar qual chave foi utilizada na criptografia (UFRJ, 2022).

Como exemplo, cita-se a possibilidade de ataque em sistemas de energia inteligentes, ou *smart grids*. Tais sistemas são projetados para monitorar o consumo de energia em cidades, bem como coordenar todo o fluxo energético da produção até o consumo. O resultado de um ataque nesse tipo de sistema pode ocasionar desde um *blackout* até a sobrecarga de reatores nucleares (JURCUT; RANAWEERA; XU, 2019). Conforme Nozomi Networks (2022), no segundo semestre de 2022 houve vários ataques direcionados a *smart grids*, como a empresa italiana de energia GSE, a ucraniana DTEK Group e a indiana Tata Power, demonstrando o quão relevante é a abordagem da segurança no contexto das redes de energia inteligentes.

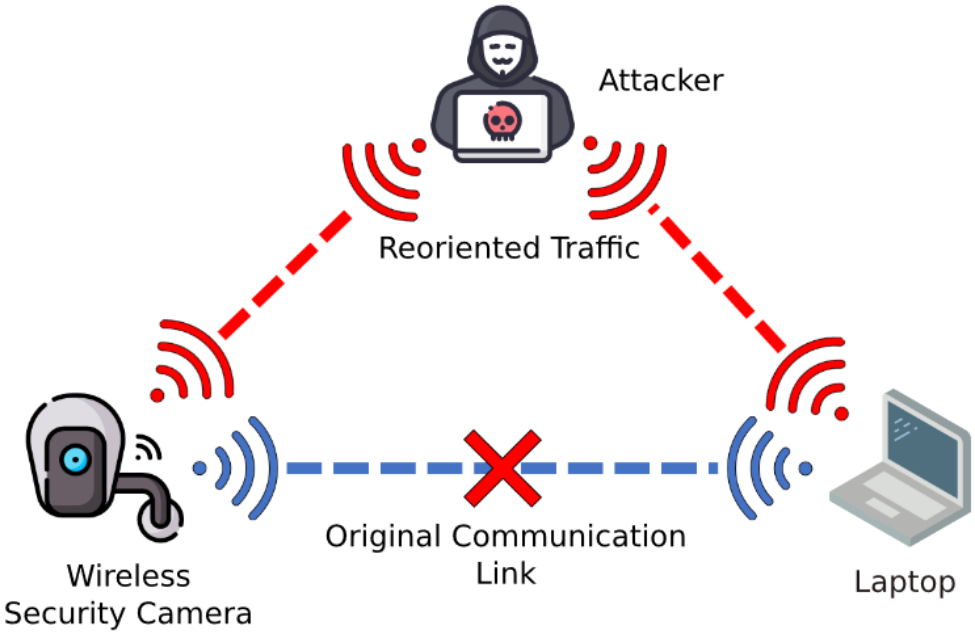
Outro alvo comum são os dispositivos vestíveis ou WIoT, os quais são classicamente usados com a finalidade de aferir temperatura corporal, pressão arterial, frequência cardíaca, dentre outros parâmetros relacionados à saúde. Os dados pessoais obtidos são repassados pela nuvem, o que torna esses dispositivos mais vulneráveis a ataques contra a privacidade (JURCUT; RANAWEERA; XU, 2019).

Além das aplicações citadas, também se destaca a agricultura inteligente, na qual se utiliza a Internet das Coisas na forma de sensores para monitorar o clima, pulverização das culturas, irrigação automatizada, dentre outros. Nesse sentido, de acordo com Jurcut, Ranaweera e Xu (2019), por possuírem limitação de recursos, geralmente repassam os dados para serem processados ou armazenados em outro local, sendo vulneráveis a ataques como *Man-in-the-Middle* (MiM) e *spoofing*. Ling *et al*. (2017) descreve ataques *spoofing* em tomadas inteligentes, em que o atacante se passa pelo dispositivo IoT, recebendo as credenciais de acesso e podendo, portanto, controlar livremente o dispositivo. Por outro lado, no ataque MiM, por exemplo, há a espionagem entre dois nós de comunicação, onde o atacante pode alterar a topologia de rede, acessar informações pessoais ou inserir informações falsas buscando o comprometimento da segurança da rede IoT (SIVASANKARI e KAMALAKKANNAN, 2022).

Logo a seguir, a Figura 4 representa um ataque MiM, no qual observa-se dois dispositivos que se conectavam originalmente, a câmera de segurança à esquerda e

o *laptop* à direita, com uma linha tracejada azul representando o fluxo original de comunicação. Entretanto, após o ataque *Man-in-the-Middle*, representado pela linha tracejada vermelha, o atacante reorienta o tráfego e torna-se capaz de ter acesso a dados sensíveis (STADDON; LOSCRI; MITTON, 2021).

Figura 4 – Ataque *Man-in-the-Middle* em um sistema de câmeras de segurança inteligentes



Fonte: Staddon, Loscri e Mitton (2021)

Outra aplicação de Internet das Coisas que tem atualmente se destacado nos ataques cibernéticos é o transporte inteligente. Nozomi Networks (2022) aferiu que apenas no ano de 2022 aconteceram inúmeros ataques a redes de transporte inteligente ao redor do mundo, tais como suspensão no sistema de transporte de trens na Dinamarca, vazamento de dados por meio de *ransomware* na empresa alemã de transportes Continental, ataque DDoS em sites de aeroportos nos Estados Unidos, dentre outros. Tais vulnerabilidades têm origem em parte em muitos pontos de entrada no sistema de transporte, por exemplo, navegação em tempo real, dados de localização e horário de acidentes, além de cada veículo autônomo conectado ao sistema. Dessa forma, Jurcut, Ranaweera e Xu (2019) defendem que esse fato torna as redes veiculares mais suscetíveis a ataques, já que possuem conexão com pontos de várias fontes diferentes.

Diante do exposto, verifica-se que para cada aplicação de IoT, existem vulnerabilidades diferentes e tipos de ataques mais suscetíveis. Entender tais

diferenças contribui para vislumbrar soluções de segurança que sejam mais adequadas a cada aplicação, visto que soluções muito abrangentes e complexas não são apropriadas para uso em IoT, devido às limitações de processamento e armazenamento desses dispositivos.

2.1.2.2 Camada de Rede

A presente camada é responsável pela conexão de dados entre os dispositivos IoT e outras redes. É considerada uma camada que exige precaução, visto que ataques envolvendo-a podem prejudicar a segurança das demais camadas. Sua análise pode ser dividida em redes de comunicação móvel e computação em nuvem (JURCUT; RANAWEERA; XU, 2019).

Quanto às redes de comunicação móvel, destaca-se a utilização de *smartphones* e minicomputadores. Jurcut, Ranaweera e Xu (2019) relatam ataques como esgotamento de bateria, exclusão de dados, bem como DDoS inundação e amplificação. Além desses, existem ataques relacionados ao uso de *Bluetooth*, como *bluesnarfing*, *bluejacking* e *bluebugging*, os quais necessitam que o atacante esteja ao alcance de 10 metros do *smartphone*, e podem resultar em acesso indevido aos contatos, podendo inclusive realizar chamadas telefônicas e escutá-las, acessar a internet, bem como acessar e enviar mensagens do dispositivo da vítima (SIDDHANT *et al*., 2018).

Paralelamente, o principal desafio da computação em nuvem é manter a privacidade, posto que os dados são armazenados por terceiros devido à limitação de armazenamento e processamento inerentes aos dispositivos IoT, conforme afirmam Jurcut, Ranaweera e Xu (2019).

2.1.2.3 Camada de Percepção

A presente camada contém os sensores e *tags* RFID (*Radio Frequency Identification*). As *tags* RFID podem receber ataques de espionagem, negação de serviço, vírus e *spoofing*, no qual o atacante se disfarça da *tag* RFID para propósitos indevidos, dentre outros. Jurcut, Ranaweera e Xu (2019) ainda destacam a possibilidade de bloqueio de sinal da *tag* RFID através de um ataque chamado de *jamming*, tal qual uma gaiola de Faraday, isolando o sinal e inutilizando a *tag* RFID

por falta de comunicação.

Quanto aos sensores, os autores exemplificam ataques em sensores ZigBee. Em 2016, por exemplo, um *malware* foi capaz de gerar iluminação em código SOS em lâmpadas Philips (JURCUT; RANAWEERA; XU, 2019). O ataque ocorreu quando um drone se aproximou do prédio da Oracle e assumiu o controle dos dispositivos fazendo com que as lâmpadas inteligentes emitissem um padrão de iluminação de código Morse SOS (FRUSTACI et al., 2018).

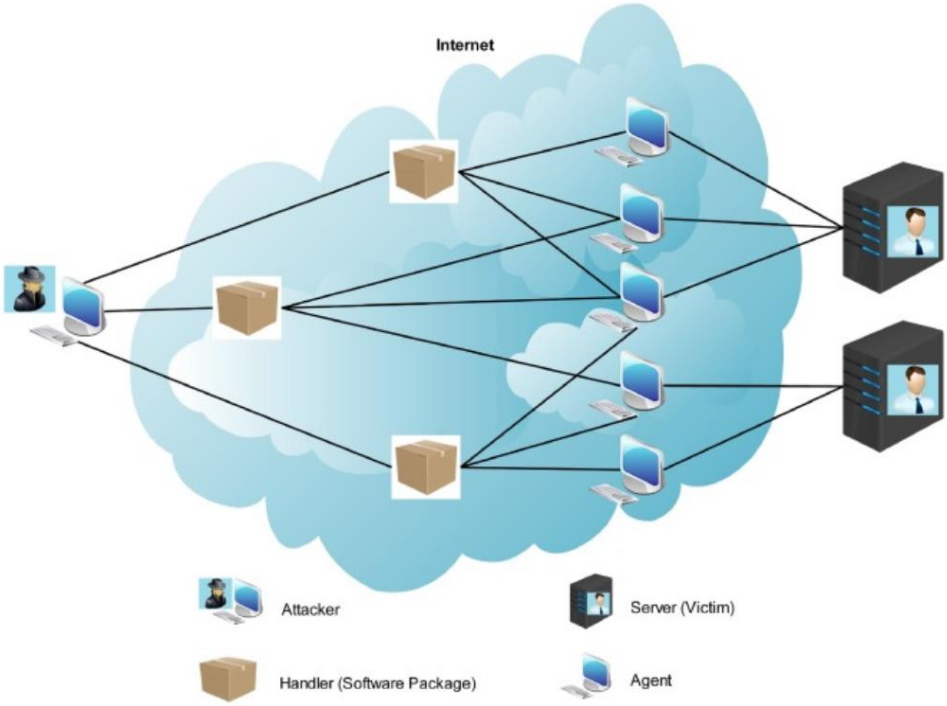
Portanto, revela-se que o ataque ao qual o dispositivo IoT está mais suscetível está relacionado à camada a qual ele pertence, seja camada de rede, de percepção ou camada de aplicação. Dessa forma, o fato de o IoT ser um *smartphone*, uma *tag* RFID ou uma lâmpada inteligente pode direcioná-lo para ataques mais específicos. Considerando o contexto apresentado, este trabalho envolve a detecção de ataques DDoS de inundação HTTP, na camada de aplicação da fonte, ou seja, dos hosts infectados, ataque que será detalhado na seção seguinte.

2.1.3 Ataque distribuído de negação de serviço

Os ataques distribuídos de negação de serviço (DDoS) são um problema grave quando se trata de segurança da informação, por terem a capacidade de gerar grandes danos financeiros a empresas, governo, provedores de *internet*, bem como usuários residenciais. Esses são ataques constituídos por um *botmaster*, um atacante que detém um *software* malicioso chamado de *bot*. Tal *software* é utilizado para infectar um escravo ou *slave,* com o objetivo de torná-lo obediente aos seus comandos, por meio de canais de comando e controle. Após essa etapa, o *botmaster* pode se utilizar dos *slaves* para realizar vários tipos de ataques, como DDoS, envio de *spam*, falsificação de identidade, dentre outros (NAZARI; DAHMARDEH; ALIABADY, 2021).

Na Figura 5 a seguir, é destacado o atacante à esquerda, o qual envia pacotes contendo *bots* para infectar os dispositivos *slaves*. Esses dispositivos são chamados na figura de agentes, por serem os responsáveis pela execução do ataque às vítimas, as quais estão representadas à direita. As linhas interligando tais atores representam as trocas de dados e de pacotes que ocorrem no ambiente da Internet (MANAVI, 2018).

Figura 5 – Estrutura geral de um ataque DDoS



Fonte: Manavi (2018).

Nesse cenário, os dispositivos IoT possuem destaque quando se trata do tema DDoS, posto que eles têm grande preferência para participar de ataques DDoS como *slaves (*KORONIOTIS; MOUSTAFA; SITNIKOVA, 2019). Portanto, os dispositivos IoT estão sendo buscados atualmente por criminosos e recrutados para tais tipos de ataques por serem mais suscetíveis.

Diante desse cenário, a necessidade de novas abordagens para os ataques DDoS se justifica também pelas constantes melhorias que os *botmasters* executam nos *bots* para que a atividade maliciosa se assemelhe à atividade normal da rede (NAZARI; DAHMARDEH; ALIABADY, 2021).

Os ataques DDoS podem ser enquadrados em algumas classificações, a saber, os ataques de camada de rede/transporte e os ataques da camada de aplicação. Na camada de rede/transporte, podem ser exemplificados os ataques de inundação e de amplificação. Os ataques de inundação ocorrem quando a largura de banda é excedida por um grande tráfego proveniente dos *hosts* infectados por *bot*, resultando em ataques inundação *User Datagram Protocol* (UDP), Internet *Control Message Protocol* (ICMP) e pacote *synchronize* (SYN), de acordo com o protocolo alvo (MANAVI, 2018). Assim, a pilha de rede é incapaz de responder às solicitações legítimas e ilegítimas, pois as filas de conexão estão ocupadas pelas inúmeras

solicitações ilegítimas (NAWROCKI, 2023).

O outro tipo de ataque DDoS presente na camada de rede/transporte é o ataque de amplificação, os quais falsificam endereços e utilizam os protocolos *Domain Name System* (DNS) e *Network Time Protocol* (NTP). Conforme Nawrocki (2023), um exemplo típico de tal situação é um *botmaster* ou atacante que envia pacotes para um servidor. Esses pacotes contêm no cabeçalho o endereço falsificado da vítima, como se fossem originários da vítima. Então, quando o servidor emite as respostas, elas não serão direcionadas ao *botmaster*, e sim à vítima, a qual não fez nenhuma solicitação, e receberá um grande tráfego. Geralmente, o pacote DNS que é enviado pelo *botmaster* possui em torno de 100 bytes, porém a resposta direcionada à vítima ultrapassa 2000 bytes, por isso o servidor funciona como amplificador do tráfego refletido para a vítima.

Já o ataque de amplificação por *Network Time Protocol* (NTP) está relacionado ao protocolo NTP, o qual é responsável por manter o horário da rede atualizado. Em tal ataque, é priorizado o uso do comando MONLIST, que consiste em envio de mensagem na porta 123 UDP, retornando uma lista com os 600 últimos hosts que realizaram conexão com o servidor NTP, IP de origem e destino, tipo de pacote e a versão do protocolo NTP utilizada. Assim, uma solicitação com 8 bytes pode resultar em uma resposta com milhares de bytes, por isso é chamado de amplificação. Um exemplo seria quando o atacante falsifica o IP de origem dos pacotes enviados ao servidor NTP, colocando o IP da vítima, com o objetivo de que toda a resposta seja direcionada à vítima, sobrecarregando-a e impedindo-a de responder às solicitações legítimas (GONDIM; ALBUQUERQUE; OROZCO, 2020).

Por outro lado, no âmbito dos ataques da camada de aplicação, são descritos os ataques de inundação do *Hypertext Transfer Protocol* (HTTP) e inundação do *Session Initiation Protocol* (SIP). Este trabalho trata especificamente do ataque inundação de HTTP, no qual o atacante envia inúmeras solicitações HTTP para a vítima, o que consome os recursos, causando transtornos à vítima. Já os ataques de inundação SIP são caracterizados por várias solicitações feitas pelo atacante ao servidor SIP. Esse servidor é responsável por receber as solicitações de chamada voz sobre IP (VoIP), para possibilitar a realização de chamadas telefônicas pela *internet* (MANAVI, 2018).

Diante do exposto, verifica-se que os ataques DDoS possuem diversas apresentações e formas diferentes de ação, podendo atuar tanto na terceira camada

– rede/transporte, quanto na sétima camada – aplicação. Nesse sentido, o presente trabalho visa colaborar na detecção de ataques DDoS que ocorrem na fonte, na camada de aplicação, mais precisamente os ataques de inundação HTTP.

2.2 SISTEMAS DE DETECÇÃO E PREVENÇÃO DE INTRUSÕES

Considerando o cenário de vulnerabilidades e ataques à segurança em Internet das Coisas, destacam-se os Sistemas de Detecção de Intrusão (IDS), os Sistemas de Prevenção de Intrusão (IPS) e os Sistemas de Detecção e Resposta a Intrusões (IDRS).

Os Sistemas de Detecção de Intrusão (IDS) consistem em sistemas capazes de monitorar uma rede, classificar os padrões de atividade como padrões normais ou padrões maliciosos, para então alertar o administrador da rede sobre tais ocorrências. Assim, os IDS devem ser capazes de manter a Integridade, Confidencialidade e Disponibilidade na rede, gerando a menor quantidade de alarmes falsos possível (GUPTA; JINDAL; BEDI, 2023).

Essas ferramentas são úteis para a prevenção de ataques, porém nem sempre é possível abranger os ataques em sua totalidade. Assim, as tentativas de invasão são registradas e avisadas para que possa ser tomada uma atitude de diminuir o dano causado ao sistema (SOBH, 2016). Dessa forma, o administrador da rede pode tomar ciência das atividades maléficas que foram detectadas para então tomar providências visando mitigá-la.

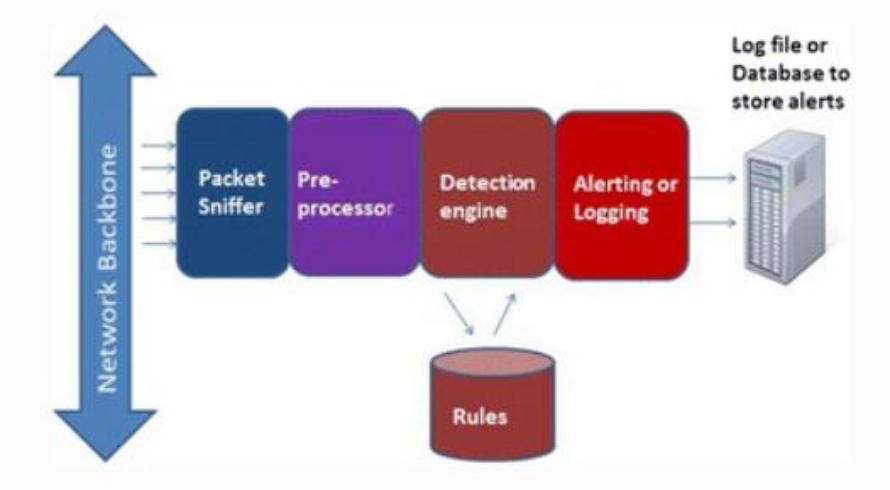
Entretanto, um IDS não possui ferramentas para bloquear um fluxo malicioso. Por esse motivo, foram criados Sistema de Detecção e Resposta a Intrusões (IDRS), os quais são IDS dotados de um módulo de resposta. Assim, o sistema se torna capaz de detectar uma invasão e de agir quando essa invasão é detectada (GUPTA; JINDAL; BEDI, 2023). Portanto, o fato de o IDS precisar de intervenção humana treinada é um dos limitadores do seu uso como mecanismo de primeira escolha na segurança de uma rede (SOBH, 2016).

Além dos IDS, existem também os Sistemas de Prevenção de Intrusão (IPS). Ao contrário dos IDS, tais sistemas são proativos, capazes de impedir de forma automática o fluxo de dados malicioso (SOBH, 2016). Logo, um IPS fornece a detecção de atividades suspeitas na rede, aliada à resposta para bloquear tal atividade.

Em uma analogia, Sobh (2016) compara o IDS a um informante, e o IPS a um policial. Essa comparação ilustra o papel do IDS em detectar ataques e notificar o responsável pela rede, enquanto o IPS age como polícia, interferindo na ação suspeita. Verifica-se, portanto, que os Sistemas de Detecção de Intrusão (IDS) e os Sistemas de Prevenção de Intrusão (IPS) possuem ampla importância na detecção e mitigação de ataques, cada qual atuando de forma distinta, mas que podem se complementar para proporcionar maior segurança à rede.

Nesse sentido, três Sistemas de Prevenção e Detecção de Intrusão (IDPS) foram analisados por Waleed, Jamali e Masood (2022): Snort, Suricata e Zeek, todos possuindo código aberto. O Snort foi criado em 1998, e tem as vantagens de ser muito utilizado mundialmente, suportar os modos IDS e IPS, registrar os ataques e gerar um arquivo de texto com as informações de detecção. Possui as desvantagens de não apresentar Interface Gráfica do Usuário (GUI), utilizar detecção de uso indevido e não utilizar detecção baseada em anomalia.

Figura 6 – Arquitetura do IDPS Snort



Fonte: Park e Ahn (2017)

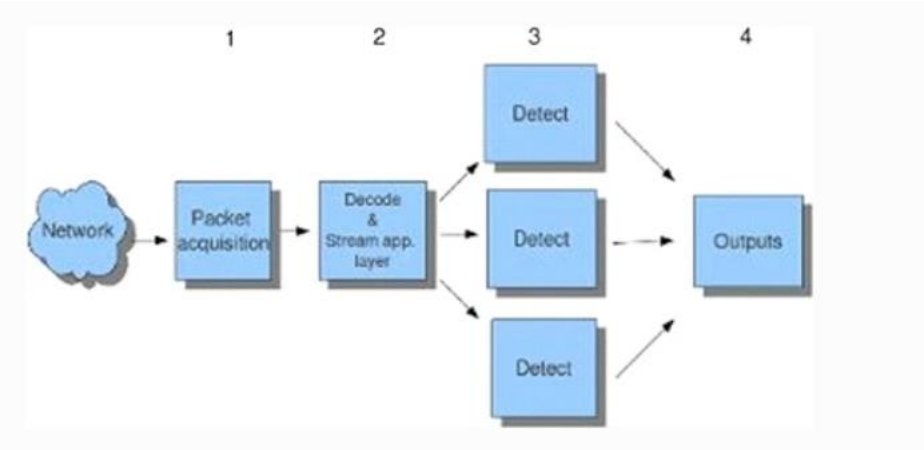
Além disso, Park e Ahn (2017) destacam que o Snort possui apenas um *thread*, e quando a capacidade dele é excedida, os pacotes restantes são ignorados. Na Figura 6 acima é possível observar essa característica, ou seja, o Snort possui apenas um mecanismo de detecção, representado na cor marrom. A figura representa a arquitetura do Snort, destacando em azul o *packet sniffer*, responsável por colher o tráfego de rede proveniente do *backbone*. A segunda etapa, representada na cor roxa, consiste em um pré-processador que faz uma pré-análise dos pacotes. A terceira

etapa é o Mecanismo de Detecção, em marrom, o qual analisa os pacotes. Então, quando as regras são quebradas, é emitido um alerta ou registro no banco de dados, representado na cor vermelha (PARK e AHN, 2017).

Quanto ao IDPS Suricata, Waleed, Jamali e Masood (2022) afirmam que possui as vantagens de ser uma ferramenta que suporta os modos IDS e IPS, possuir múltiplos mecanismos de detecção simultâneos (*multi-threading*), ser capaz de analisar uma quantidade maior de tráfego de rede quando comparado ao Snort, possuir interface gráfica, gerar arquivo de texto ou no formato JSON como saída. Ressalte-se que o Suricata apresentou melhor desempenho do que o Snort e o Zeek no trabalho citado.

Na Figura 7 abaixo, pode-se visualizar a primeira etapa da Arquitetura do Suricata, a qual consiste na aquisição de pacotes da rede. Depois, existe a etapa de decodificação de fluxo e, posteriormente, a detecção *multi-thread* com vários módulos de detecção que trabalham simultaneamente. Ao final, é gerada uma saída de dados via texto ou arquivo JSON com os alertas de tráfegos suspeitos identificados (PARK e AHN, 2017).

Figura 7 – Arquitetura do IDPS Suricata



Fonte: Park e Ahn (2017)

O Suricata ainda é caracterizado por ser um IDPS baseado em regras. Dessa forma, cada regra representa um padrão de tráfego na rede, além de indicar o método para bloquear determinado ataque. Geralmente, o Suricata é posicionado na rede atrás de um *firewall*, com o objetivo de esmiuçar e debelar o tráfego não bloqueado pelo *firewall* (ALCANTARA, 2022).

O terceiro *software* analisado foi o Zeek, cujas funcionalidades não incluem

módulo IPS, apenas IDS. Como vantagem apresenta a fácil implantação e detecção baseada em anomalias; e como desvantagens apresenta menor quantidade de regras quando comparada ao Suricata e Snort, além de possuir menos estudos a respeito (PARK e AHN, 2017).

Dessa forma, após análise dos principais *softwares* IDS e IPS de código aberto, foi escolhido o IDPS Suricata, o qual é utilizado neste trabalho e especificado na seção 4.

2.3 SISTEMAS DE ANÁLISE DE DADOS

Com o intuito de auxiliar na visualização dos dados provenientes do Suricata, este trabalho utiliza um *dashboard* com as ferramentas Elasticsearch, Logstash e Kibana. Tais sistemas trabalham juntos, formando a pilha ELK, sendo capazes de importar os dados provenientes do Suricata e gerar relatórios, métricas de desempenho, filtros, dentre outros, além de disponibilizar um *dashboard* de fácil interação (SOUZA et al., 2019).

Além disso, o ElasticSearch possui alta disponibilidade, com garantia de que o cluster principal continue em funcionamento ainda que um dos servidores secundários pare de funcionar; escalabilidade, em que novos nós podem ser adicionados caso o tráfego de dados aumente; além de alta performance no uso em buscas complexas (NEVES, 2023).

Por ser uma ferramenta *open source* que auxilia na tomada de decisão e de boa usabilidade, a pilha ELK é utilizada por inúmeras empresas e também pelo setor público, a exemplo do Instituto de Biologia da Unicamp (SOUZA et al., 2019).

Além da pilha ELK, o presente trabalho se utiliza da ferramenta Wazuh, uma plataforma de Gerenciamento de Eventos e Informações de Segurança (SIEM). O principal benefício dessa plataforma é entregar monitoramento de segurança detalhada, sem ser necessária a instalação de agente ou *script* no dispositivo IoT, ou seja, não há diminuição do desempenho do IoT. Dessa forma, como os dispositivos IoT entregam e recebem dados por meio do *gateway*, o Wazuh é capaz de inspecionar esse tráfego proveniente do *gateway*, sem necessitar passar pelo processamento do IoT (ZAHID, 2023).

Por esses motivos, a integração da plataforma Suricata com a pilha ELK se revela de suma importância, trazendo os benefícios de visualização de dados de

forma organizada, *dashboard* interativo, disponibilização de relatórios, além de alta disponibilidade. No mesmo sentido, o SIEM Wazuh obtém informações valiosas de segurança do IoT, sem comprometer o seu poder computacional.

2.4 FERRAMENTAS DE GERAÇÃO DE ATAQUES DE NEGAÇÃO DE SERVIÇO DISTRIBUÍDOS

Existem duas principais ferramentas para testar as redes no que concerne aos ataques de negação de serviço distribuídos (DDoS): *Slowloris* e *Low Orbit Ion Cannon.*

O *Low Orbit Ion Cannon* (LOIC) é escrito em linguagem C# e possui código aberto. Segundo Araújo (2017), o seu funcionamento consiste em gerar um volume grande de tráfego TCP, UDP e HTTP, direcionando-o à vítima, com o intuito de deixá-la incapaz de responder às requisições legítimas. Ele é utilizado neste trabalho com o objetivo de simular ataques DDoS tipo inundação HTTP.

Quanto ao *software Slowloris,* é usada a linguagem Perl, e a estratégia utilizada é enviar a um servidor requisições parciais, mantendo as solicitações abertas por muito tempo. Assim, as novas solicitações que chegam ao servidor não conseguem ser atendidas (ARAÚJO, 2017). O presente trabalho utiliza o *software* LOIC, por ser o mais indicado em ataques inundação HTTP.

3 TRABALHOS RELACIONADOS

Os trabalhos relacionados apresentados neste capítulo são resultado de uma revisão bibliográfica com critérios descritos na seção 1.5, referente à metodologia da dissertação. Na seção 3.1, é exposto o estado da arte; da seção 3.2 à seção 3.5 são explanados os trabalhos relacionados mais relevantes para a presente pesquisa; em seguida na seção 3.6 é apresentada uma comparação entre os trabalhos relacionados.

3.1 ESTADO DA ARTE

Os trabalhos na área de segurança em Internet das Coisas envolvem o uso de *deep learning* e de *blockchain*, dentre outros métodos. A seguir, são exemplificados alguns trabalhos para compor o estado da arte em segurança de IoT.

O trabalho de Altan (2021) ressaltou a segurança de Internet das Coisas no contexto do aprendizado de máquina, apresentando a solução de segurança SecureDeepNet-IoT. Essa ferramenta utiliza aprendizado de máquina com a técnica de redes neurais para detectar ameaças em dispositivos IoT. O modelo foi treinado com uma base de dados realista, e alcançou a precisão de 95,05% a 94,39% no resultado de classificação de ataques em IoTs. Este método apresenta a vantagem de além de detectar, também classificar o tipo de ameaça, e possui como desvantagem o alto consumo de processamento e memória, bem como alto custo.

Por outro lado, o artigo proposto por Alani (2023) está direcionado ao contexto industrial da Internet das Coisas ou IIoT. O autor propõe a solução E2I3DS, baseada em aprendizado de máquina. Nos resultados, a solução alcançou precisão de 99,97% na detecção de ataques em IoTs industriais, possuindo como vantagens a eficácia e o treinamento do modelo com dados provenientes de dispositivos IIoT, e como desvantagens o uso de processamento elevado. O trabalho ainda descreve as principais diferenças entre uma rede de Internet das Coisas habitual e uma rede de Internet das Coisas Industrial (IIoT), destacando o volume exponencial de dados gerados pelos dispositivos IIoT e a necessidade de se manter a precisão e continuidade dos dados gerados por esses dispositivos, já que qualquer interrupção ou modificação dos dados gera impacto negativo no processo de fabricação.

Em uma abordagem centrada nos ataques de roteamento em dispositivos IoT, os autores Alghamdi e Bellaiche (2023) propõem um sistema de detecção de ataques em protocolo *Routing Protocol For Low Power And Lossy Networks* (RPL). Os ataques RPL atingem o roteamento da rede IoT e acarretam na parada do tráfego de pacotes entre os nós da rede. O tipo de ataque estudado nesse artigo é o *wormhole*, ou buraco de minhoca, que consiste em um ataque capaz de interromper o roteamento em uma rede IoT, posto que os caminhos entre os nós da rede são modificados, assim um caminho entre dois nós de sensores pode ser mudado pra prejudicar o tráfego de pacotes. Visando à mitigação desse tipo de ataque, os autores utilizam uma técnica em cascata, ou seja, em duas etapas, em que a primeira etapa consiste no Fator de Confiança Dinâmico (DTF) e a segunda etapa na análise com *deep learning*, de tal forma que a segunda etapa só é executada se o valor do fator de confiança cair abaixo de um limite pré-determinado, evitando que a detecção em aprendizado de máquina seja deflagrada a todo momento. A precisão de detecção obtida foi de 96% dos ataques no estudo. As vantagens desse método foram o baixo consumo de processamento e a baixa latência.

Para além do uso de aprendizado de máquina, disposto nos três contextos acima, desponta no cenário de segurança em IoT a utilização de *blockchain* como alternativa de segurança. No artigo de Zhang et. al (2022), por exemplo, apresenta um método de segurança em Internet das Coisas utilizando *blockchain*, focado em sistemas de transporte marítimo, atuando em ambientes de programação e gerenciamento de fluxo de transporte marítimo com a finalidade de mitigar ataques de repetição. O ataque de repetição ocorre quando o invasor copia um pacote que trafega entre duas partes da rede e envia para outra parte da rede, mimetizando um tráfego legítimo e causando prejuízos ao funcionamento da rede. A técnica baseia-se em vários blocos encadeados, em que cada bloco armazena o *hash* do bloco anterior, então se houver tentativa de alterar o conteúdo do bloco anterior, o bloco atual e os blocos subsequentes serão invalidados, pois o *hash* não corresponderá ao original. Após a adoção da solução, a segurança apresentou aumento de 8% e o processamento de transações ficou 6% mais veloz.

Diante do exposto, o estado da arte em se tratando de segurança em dispositivos IoT abrange o uso de *blockchain* e deep learning. Entretanto, o contexto de dispositivos residenciais possui características peculiares, tais como a limitação de memória e de CPU, fato esse que torna o *blockchain* e o aprendizado de máquina em

um uso dificultoso, dado a exigência de *hardware* e de *software* decorrente dessas tecnologias.

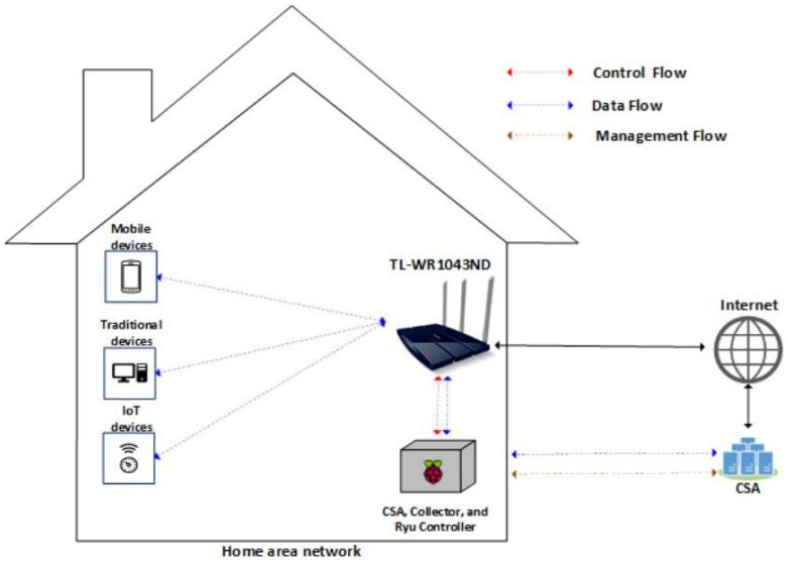
Nesse sentido, nas próximas subseções, serão apresentados os trabalhos relacionados, um em cada subseção correspondente. Tais trabalhos versam sobre segurança em IoT e, além disso, obedecem aos critérios de inclusão descritos na seção de metodologia, ou seja, são artigos aplicáveis ao cenário residencial, estão disponíveis on-line, são revisados por pares e apresentam uma solução de segurança em IoT baseada no fluxo.

3.2 SOLUÇÃO DE SEGURANÇA COM REDE DEFINIDA POR SOFTWARE

Esse trabalho foi proposto pelos autores De Melo, Miani e Rosa (2022), e tem o objetivo de diminuir o risco à segurança de dispositivos IoT utilizando rede definida por *software* (SDN) e *machine learning*. Assim, o aprendizado de máquina atua identificando fluxos maliciosos na rede doméstica, enquanto o SDN atua bloqueando o tráfego malicioso identificado na rede.

Para concretização do projeto, chamado de *FamilyGuard*, os autores utilizaram como hardware um *Raspberry Pi3 Quadcore* com 1GB de RAM e um cartão de memória de 32 Gb. O sistema operacional utilizado no projeto foi o *Raspberry Pi OS*. Além disso, utilizou-se também um roteador TPLINK com *OpenWrt* e *Open vSwitch,* representados abaixo.

Figura 8 – Esquema de montagem do dispositivo de segurança FamilyGuard



Fonte: DE MELO, MIANI e ROSA (2022)

Na Figura 8 acima, é possível verificar o esquema de fluxo de dados, representado pela linha azul, e o controle do fluxo, representado pela linha vermelha. Os dados transitam entre a internet e o roteador, entre o roteador e o controlador Raspberry Pi, e entre os dispositivos móveis, tradicionais e dispositivos IoT. O *Central Security Assistant* (CSA), estrutura externa à casa, é responsável por criar os modelos de aprendizado de máquina, ou seja, existe a terceirização desta atividade para uma empresa que transmite esses dados já prontos pela *internet* para a casa inteligente, posto que seria exigido um grande poder computacional da estrutura *FamilyGuard* caso a própria estrutura residencial ficasse responsável por treinar esses modelos.

Considerando o exposto, o projeto possui o diferencial de incluir a análise do tráfego de todos os dispositivos residenciais, e não somente os dispositivos IoT. Dessa forma, também estão inclusos na análise os computadores, por exemplo, indicados na imagem acima como dispositivos tradicionais. Por outro lado, o trabalho fica dependente de uma empresa - *Central Security Assistant* – a qual treinará os modelos e os transmitirá à casa inteligente.

3.3 SOLUÇÃO DE SEGURANÇA USANDO SNORT E SURICATA

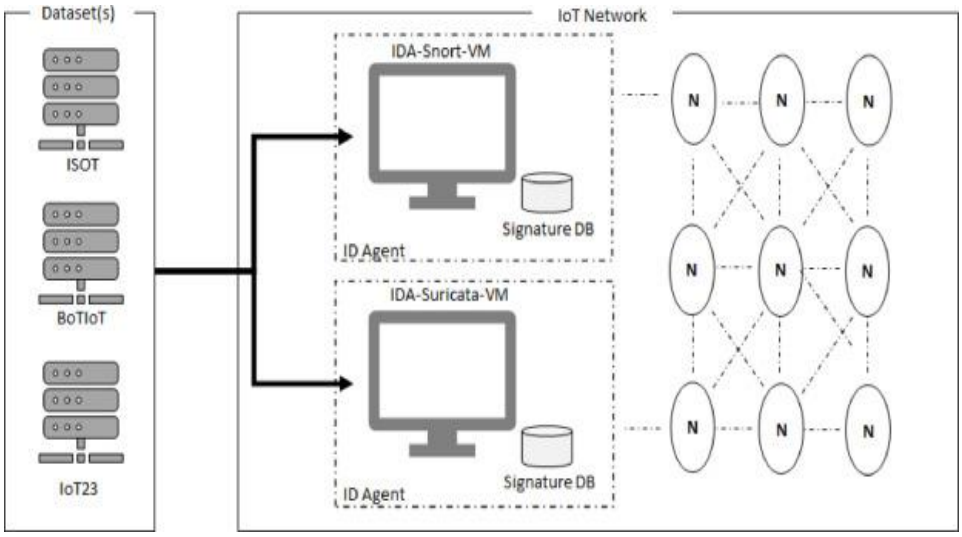
O trabalho dos autores Nasir, Arshad e Khan (2023) trata da detecção de intrusão de ataques em dispositivos de Internet das Coisas. As ferramentas utilizadas nesse estudo consistiram nos *softwares* de Detecção e Prevenção de Intrusão Snort e Suricata, reconhecidos no mercado. Ao final do estudo foi feita uma comparação entre o desempenho dos dois sistemas.

Os autores utilizaram tais sistemas IDPS instalados em uma máquina virtual utilizando o sistema operacional Kali Linux, com limitação de recursos, com o intuito de simular um ambiente IOT. As máquinas virtuais foram configuradas possuindo 2 núcleos e 2 GB de memória cada uma. Os conjuntos de dados utilizados foram voltados para detecção de *botnets*: IoT23, ISOT e BoTIoT, sendo o IoT23 o único a utilizar dados de ataques em dispositivos IoT reais, enquanto os outros se referem a dispositivos simulados.

Nesse sentido, os experimentos demonstraram os IDPS Snort e Suricata instalados nas máquinas virtuais recebem os fluxos provenientes dos bancos de dados, monitorando esse fluxo e enviando alertas quando um tráfego malicioso é

identificado. A Figura 9 abaixo ilustra a topologia de detecção de intrusão que foi utilizada nesse trabalho, sendo os nós ou dispositivos representados pela letra “N” à direita, os conjuntos de dados ilustrados à esquerda, e as duas máquinas virtuais posicionadas no centro da figura, a superior correspondendo ao Snort e a inferior correspondendo ao Suricata.

Figura 9 – Topologia de detecção de intrusão nos IDS Snort e Suricata



Fonte: NASIR, ARSHAD e KHAN (2023)

Além disso, os autores se utilizaram do poder computacional dos próprios dispositivos IoT para processar as informações de detecção de ataques. Para esse projeto ser possível, a pesquisa levou em consideração que os dispositivos IoT apresentam frequentemente limitação de recursos, como memória e processamento. Para solucionar tal problema, foi escolhida uma abordagem colaborativa, na qual a carga de detecção é compartilhada entre vários dispositivos IoT da rede. Os autores relatam que esse fato diminuiu o custo financeiro da solução de segurança proposta.

Dessa forma, trata-se de um trabalho extremamente recente e atualizado, utilizando ferramentas IDPS Snort e Suricata. Entretanto, pode-se notar algumas limitações na execução. Verifica-se que os autores optaram por fracionar a carga de processamento de detecção entre os dispositivos IoT da rede, estratégia que foi chamada de compartilhamento de carga de detecção. Porém os experimentos não previram cenários em que podem haver poucos dispositivos ligados ou funcionantes na rede, o que tornaria o processamento dos dados inviável e, por consequência,

poderia ocasionar uma falha na operação do sistema IDPS, aumentando a suscetibilidade a ataques e também diminuindo a performance dos IoT na rede.

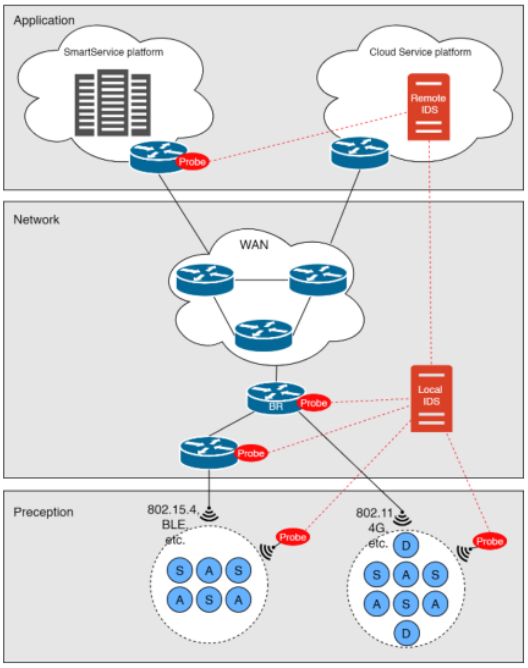
3.4 SOLUÇÃO DE SEGURANÇA COM PROCESSAMENTO LOCAL E EM NUVEM

Nesse trabalho, os autores Santos *et al.* (2023) priorizaram o uso de um IDS que não utilizasse os recursos próprios dos IoT, ou seja, foi implementada uma arquitetura de *hardware* exclusivo para realizar as tarefas de IDS, para que os recursos dos IoT não fossem consumidos com as atividades de detecção. Assim, os IoT da rede têm o seu poder computacional disponível para suas próprias atividades.

Dessa forma, foi utilizada uma abordagem baseada em fluxo, na qual várias sondas espalhadas pela rede captavam os fluxos, enquanto os dados foram analisados em dois blocos, um IDS local e outro IDS na nuvem. Dessa forma, quando o IDS identifica a ação de um fluxo malicioso, é emitido um alerta para o banco de alertas utilizando *syslog*.

Na arquitetura, foram utilizados vários *Raspberry PI* como roteadores de borda para abrigar as sondas IDS, com o sistema Ubuntu Server instalado. Tal escolha torna essa solução de segurança em um custo elevado. Os resultados deste trabalho foram comparados aos IDS Snort, Suricata e Zeek.

Figura 10 – Arquitetura com conjunto de IDS em nuvem e IDS local



Fonte: SANTOS et al. (2023)

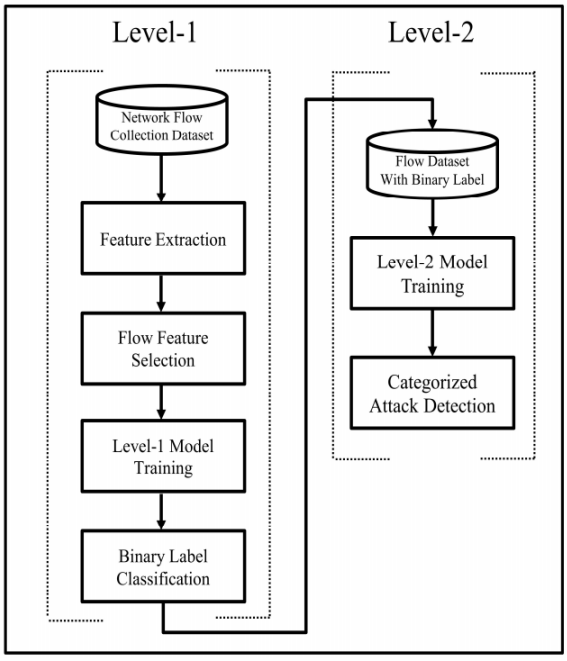
A Figura 10 demonstra a arquitetura utilizada neste trabalho, evidenciando as sondas ou *probes* em vermelho, as quais captam o fluxo nas três camadas – percepção, rede e aplicação – e direcionam os dados para análise no IDS remoto em nuvem ou para o IDS local, ambos representados na cor laranja. Dessa forma, o IDS remoto e o IDS local são utilizados conforme o seu poder computacional.

3.5 SOLUÇÃO DE SEGURANÇA COM MACHINE LEARNING

Por sua vez, o trabalho dos autores Ullah e Mahmoud (2020) é focado na utilização de *machine learning* para mitigar ataques no contexto da Internet das Coisas. Os autores consideraram um grande desafio executar a classificação dos ataques, além de ter que realizar as atividades de detecção e classificação em tempo hábil, a fim de evitar a ocorrência do ataque.

O modelo proposto é baseado em dois níveis, sendo o primeiro nível responsável por analisar e classificar o fluxo em normal ou anormal. Caso o fluxo seja considerado anormal pelo nível 1, ele será direcionado para o segundo nível, que se encontra na névoa, em *fog computing*. No nível 2, o fluxo malicioso será então classificado, para saber à qual categoria de ataque ele pertence. Portanto, a arquitetura é classificada de alto custo. Tal sequência pode ser observada na Figura 11 a seguir.

Figura 11 – Sistema de Detecção de Intrusão em Dois Níveis



Fonte: ULLAH e MAHMOUD (2020)

Foram utilizados Wireshark e TCPdump para interceptação dos pacotes no tráfego de rede, além dos métodos árvore de decisão e floresta aleatória para a classificação nos níveis 1 e 2, respectivamente.

Tal projeto apresenta algumas limitações para implantação em uma rede residencial, a saber: alto custo, latência na comunicação entre a rede local e o *fog,* além de dependência de conexão com o *fog* para garantir a segurança da residência. Dessa forma, caso ocorra alguma indisponibilidade no *fog* ou indisponibilidade na rede de *internet* que realiza a comunicação entre a rede local e *fog*, o sistema de segurança não funcionará adequadamente. Além disso, a latência da comunicação entre rede local e *fog* pode inviabilizar o uso desse mecanismo, predispondo a rede a ataques.

3.6 COMPARAÇÃO ENTRE OS TRABALHOS RELACIONADOS

Os trabalhos relacionados possuem características diferentes entre si, apesar de todos abordarem a segurança em dispositivos IoT. Contudo, vale ressaltar que algumas informações são muito relevantes para este trabalho, como: ser uma solução de baixo custo, ser aplicável ao cenário residencial, utilizar o fluxo da rede na análise, emitir alerta para notificar o usuário da rede, e a informação se a solução de segurança não diminuiu a performance dos IoT na rede. Cada uma dessas características possui a sua importância, as quais serão discutidas a seguir e sintetizadas na Tabela 3.

O motivo de o baixo custo estar entre as características importantes se baseia no fato de que uma solução de segurança residencial de alto custo poderia restringir e até impossibilitar a utilização em massa pela população. O próximo atributo, que analisa se a solução pode ser aplicada ao cenário residencial, foi incluído na análise porque muitas soluções propostas nos trabalhos analisados estavam focadas em outras áreas de aplicação de IoT, como *Industrial Internet of Things* (IIoT), *smart grids*, bem como agricultura inteligente. Dessa forma, tais áreas apresentam diferenças que incompatibilizam o uso dessas ferramentas no cenário residencial.

Outro ponto importante é que a ferramenta de segurança utilize o fluxo da rede como parte da análise, e que possa englobar uma notificação de fácil acesso para os moradores da residência. Assim, como se trata de um ambiente residencial, é importante que a solução de segurança emita um alerta que seja acessível ao usuário final.

Tabela 3 – Comparação entre os trabalhos relacionados

CARACTERÍSTICAS/AUTORES	NASIR, ARSHAD e KHAN (2023)	SANTOS et al. (2023)	ULLAH e MAHMOUD (2020)	DE MELO, MIANI e ROSA (2022)	TRABALHO PROPOSTO
BAIXO CUSTO	✓	X	X	X	✓
APLICÁVEL AO CENÁRIO RESIDENCIAL	✓	✓	✓	✓	✓
BASEADA NO FLUXO DA REDE	✓	✓	✓	✓	✓
EMITE ALERTA AO USUÁRIO DA REDE	X	X	X	X	✓
NÃO DIMINUI A PERFORMANCE DOS IOT NA REDE	X	✓	✓	✓	✓
FERRAMENTAS UTILIZADAS	IDPS SNORT E SURICATA, HARDWARE DOS IOT DA REDE	IDS LOCAL E IDS CLOUD, HARDWARE Raspberry PI	MACHINE LEARNING E FOG COMPUTING	MACHINE LEARNING E SDN	IDPS SURICATA, HARDWARE BEELINK

Fonte: elaborada pelo autor

Quanto à informação se a solução de segurança diminuiu a performance dos IoT na rede, esse ponto foi adicionado ao se perceber que alguns trabalhos utilizavam o próprio *hardware* dos dispositivos IoT para o processamento dos *softwares* de segurança, o que pode acarretar baixa performance dos dispositivos na rede, e por consequência baixa adesão dos usuários ao sistema de segurança. Essas informações estão resumidas na Tabela 3 já mencionada acima.

4 PROPOSTA, ARQUITETURA E IMPLEMENTAÇÃO

Este capítulo aborda a proposta da dissertação, detalhando a arquitetura utilizada, o cenário utilizado, além das etapas de implementação realizadas na solução de segurança Block Suricata Anti-DDoS.

4.1 PROPOSTA

O presente trabalho possui como objetivo mitigar as vulnerabilidades de segurança em dispositivos IoT. Para alcançá-lo, propõe-se o monitoramento de tráfego em rede doméstica e geração de alertas via Telegram ao ocorrerem tentativas de ataques DDoS do tipo inundação HTTP.

Para tanto, esta dissertação propõe a criação de uma solução gratuita de segurança, o Block Suricata Anti-DDoS, com código capaz de gerar alertas via Telegram sem consumir os recursos computacionais dos dispositivos IoTs da rede doméstica. Ao receber o alerta pelo Telegram, o usuário da rede doméstica poderá identificar o dispositivo que está participando do ataque e desligá-lo, evitando-se assim a propagação do ataque. Dessa forma, a solução de segurança proposta evita que os IoTs da rede tenham suas vulnerabilidades exploradas e sejam utilizados como *slaves* em ataques DDoS, agindo na fonte dos ataques DDoS, posto que os dispositivos mais utilizados atualmente para gerar esse tipo de ataque são os IoT residenciais.

Além disso, existe no mercado a carência de um sistema gratuito que esteja integrado com o Telegram em tempo real, enviando notificações via aplicativo para o administrador sobre ataques DDoS na rede. Trata-se de um código desenvolvido em Python, com pouquíssimas linhas, impactando de forma positiva na performance em notificar os ataques.

Nesse sentido, o Block Suricata Anti-DDoS consiste na criação de uma solução de segurança gratuita, com sistema operacional que já se encontra instalado e configurado com o *software* Suricata para detecção de ataques DDoS em redes domésticas. Além disso, os *softwares* Wazuh e ELK (Elasticsearch, Kibana e Logstash) atuarão de forma integrada com o Suricata, lendo os *logs* de ataques DDoS e gerando alertas, que serão em seguida prospectados pelo código em Python para gerar mensagens via Telegram.

Após a implementação e realização dos testes, a solução de segurança Block Suricata Anti-DDoS está disponível para download no *link* https://github.com/ricardohelisson/detectar_ataque_ddos_com_suricata . Trata-se de uma contribuição à comunidade, de tal forma que os usuários que desejarem poderão fazer o *download*, e em seguida, configurar os dados pessoais para envio das notificações via Telegram (*token* e *chat_id*), que são informações confidenciais.

Além disso, tais funcionalidades de *token* e *chat_id* são ferramentas importantes de segurança, posto que garantem a autenticidade dos alertas gerados, de tal forma que somente o possuidor dessas informações confidenciais terá acesso aos alertas enviados pelo Telegram. Assim, a solução valoriza o princípio da autenticidade.

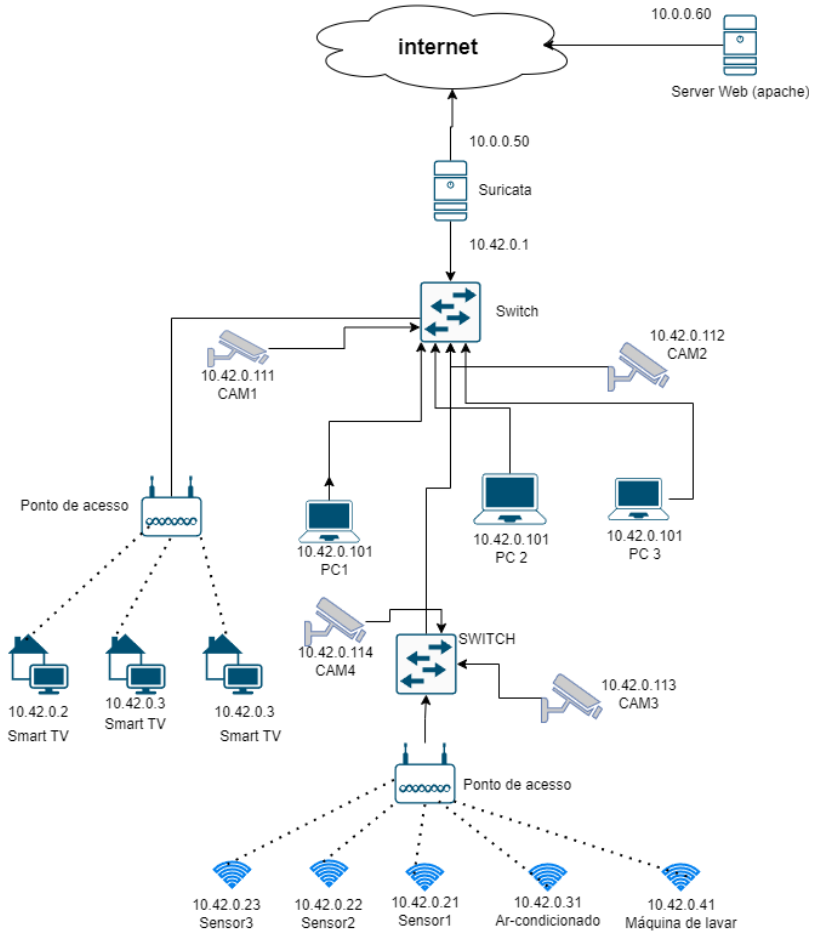
Diante do exposto, a solução de segurança proposta nesta dissertação contém os atributos de utilizar um sistema operacional gratuito compatível com *hardwares* domésticos, com baixo processamento e baixo consumo de memória, em uma interface amigável para o usuário final. Dessa forma, os dados são coletados da rede residencial, por meio do software Suricata, utilizando o *hardware* mini computador da marca BeeLink, o qual possui processador Intel Celeron n5095, memória DDR 4 e 8 Gb de memória RAM. Outras ferramentas a serem utilizadas neste trabalho são o Suricata e a pilha ELK, além do Wazuh, uma plataforma de Gerenciamento de Eventos e Informações de Segurança (SIEM).

Adicionalmente, foi utilizada a linguagem de programação Python versão 3, por ser uma linguagem gratuita e de código aberto. Os autores Khoirom et. al (2020) destacaram inúmeras vantagens do Python em relação à linguagem de programação Java. Dentre elas, pode-se citar a necessidade de um código menor para executar uma mesma tarefa quando comparado a outras linguagens, o fato de ter um grande suporte da comunidade, o fato de Python consumir menos memória e ter uma boa velocidade de execução, a disposição de muitas bibliotecas permitindo ao programador executar de forma mais simples códigos que seriam complexos em outras linguagens, e ainda o fato de ter uma curva de aprendizado menor. Ainda segundo os autores, o Java possui as desvantagens de ser pago a partir de 2019 para negócios, comerciais e produção, além de consumir mais memória e de ter execução mais lenta. Os autores ainda destacam que Python é considerada a linguagem que mais cresce nos últimos anos e, devido a todos esses fatores, foi optado por utilizar tal linguagem de programação neste trabalho.

Levando-se em conta que os casos de ataques DDoS mais comuns são do tipo HTTP inundação, no Suricata foi criada uma regra baseada nesse tipo de ataque, considerando o seu comportamento. Esses *scripts* são armazenados em um banco de dados nessa plataforma e servirão para análise e tomada de decisões em relação a cada equipamento presente na rede. Dessa forma, ao serem percebidos no monitoramento comportamentos semelhantes aos tipos de ataques citados, esse sistema armazenará as informações em um arquivo de *log* e o Wazuh buscará essas informações nesses *logs* e as exibirá na sua listagem de eventos de segurança.

Quanto à escolha do Telegram, destaca-se que é um aplicativo de mensagens instantâneas capaz de fornecer latência zero no tratamento de muitas solicitações ao mesmo tempo. Já o principal aplicativo concorrente, o Whatsapp, apresentou latência de 0,5 milissegundos (SWANDI et. al., 2021).

Figura 12 - Cenário de monitoramento de rede residencial



Fonte: elaborada pelo autor

Na Figura 12 acima, observa-se o cenário geral de uma proposta de monitoramento para uma rede doméstica que utilize a Internet das Coisas, sendo representados itens comuns em redes residenciais. A *internet* está conectada à rede interna da residência através do *hardware* Beelink. Por sua vez, o *hardware* BeeLink contendo o *software* Suricata é conectado também ao *switch* não-gerenciável e pontos de acesso, que distribuem o sinal *wifi* para os dispositivos IoT da rede.

Dessa forma, o Suricata tem acesso a duas interfaces de rede, recebendo a *internet* por meio da ens33 representada pelo IP 10.0.0.50, e distribuindo *internet* para a residência por meio da interface ens37 representada pelo IP 10.42.0.1. Nas residências de teste, existem três *smart tv*, quatro câmeras de segurança, três sensores com lâmpadas, três computadores, um ar condicionado e uma máquina de lavar *smart*. O servidor apache atuará neste trabalho como o receptor dos ataques DDoS na porta 80, ataques esses originados do LOIC instalado em um computador na residência.

Esse cenário foi idealizado considerando que é necessário fazer o monitoramento de ambientes residenciais para observar objetos físicos conectados que são dotados de tecnologia que os fazem transmitir dados. Assim, eles fazem uso da internet, e por isso podem ser controlados e executar tarefas por parte de usuários desautorizados. Embora esses dispositivos sejam importantes para muitas pessoas ou instituições, se seu tráfego não for controlado poderá ocasionar perdas significativas de dados e problemas a outras redes de computadores por conta de tráfego excessivo e não autorizado.

Diante disso, foram utilizados bibliografia atualizada e dados de monitoramento com o objetivo de minimizar o risco de ataques e tornar as redes IoT residenciais mais seguras, empregando tecnologias de baixo custo e alto rendimento.

4.2 ARQUITETURA

Com o intuito de entregar uma solução de segurança gratuita que alcance tais atributos, a presente dissertação se utilizou das ferramentas listadas abaixo:

a) Sistema operacional Ubuntu versão 22.04 LTS;
b) *Software* Suricata e pilha ELK;
c) *Software* Wazuh;
d) Código em Python 3.

Buscando minimizar ações de ataques DDoS na rede doméstica, esta pesquisa se munirá de algumas ferramentas, como o BeeLink que funciona como *hardware*. Além dele, alguns *softwares* são utilizados. O primeiro é o VMware vSphere Hypervisor 7.0, o qual tem como objetivo virtualizar as aplicações, permitindo assim que mais de uma máquina virtual seja executada em uma mesma máquina física. O segundo *software* trata-se do sistema operacional Ubuntu versão 22.04 LTS, o qual é *open source,* possui compatibilidade com a maior parte dos *hardwares* domésticos e utilização gratuita. Na Figura 13 abaixo pode-se observar a arquitetura proposta.

Figura 13 – Arquitetura do Block Suricata Anti-DDoS

Fonte: elaborado pelo autor.

A Figura 13 acima ilustra a arquitetura em diagrama de blocos do sistema proposto. Na parte superior da figura, o microcomputador Beelink recebe o fluxo de pacotes da rede residencial. Tal *hardware* contém os *softwares* Suricata, ELK, Wazuh e o código em Python. Inicialmente, os pacotes brutos são decodificados e processados pelo Suricata, para então passarem pela etapa de detecção de ataques e posterior geração de alertas e *logs* em *json*.

A partir dessa etapa, existem dois destinos diferentes para os *logs*: a pilha ELK e o código em Python. Na pilha ELK, os logs passam por tratamento e organização, sendo direcionados ao Wazuh, o qual exibe uma interface visual ao usuário com os alertas de detecção. A partir dessa informação, o usuário pode desligar o equipamento que está passando pelo ataque DDoS.

O outro destino dos *logs* consiste em interação com código Python, passando pelo módulo Telepot, com configuração de uma camada de segurança com *token* e *chat id*. Os *logs* são percorridos e o último registro é enviado para o Telegram do usuário da rede doméstica, para que a cada novo ataque os registros antigos não sejam enviados novamente, apenas o ataque mais recente. Munido dessa informação em tempo real e de forma amigável a um usuário doméstico, o mesmo pode desligar o equipamento e mitigar a propagação do ataque DDoS na rede doméstica.

4.3 IMPLEMENTAÇÃO

Esta seção descreve a criação e a implementação da solução de segurança chamada de Block Suricata Anti-DDoS. Os detalhes sobre os procedimentos de instalação dos *softwares* necessários, as telas de configurações e os códigos das principais funcionalidades implementadas nesta dissertação estão localizados no Apêndice.

4.3.1 Instalações

Inicialmente, procedeu-se à instalação do ambiente de desenvolvimento, sendo instalado no VMware vSphere Hypervisor 7.0 o sistema operacional Ubuntu versão 22.04 LTS, também conhecida como Jammy Jellyfish. A instalação do Ubuntu é compatível com o *software* Suricata e com os demais *softwares* que são utilizados neste projeto. Além disso, o Ubuntu possui ampla compatibilidade com inúmeros

hardwares, o que possibilita sua ampla utilização. Outra vantagem é o fato de ele ser *open source* e de possuir atualizações de segurança mais frequentes, com suporte registrado até o ano de 2027 (UBUNTU, 2024).

Após essa primeira fase, já é possível a instalação do IDPS Suricata. Primeiramente, é adicionado o repositório do Suricata ao Ubuntu.

Com o repositório adicionado, prossegue-se com a instalação do Suricata. Há a execução de comando de instalação do *software*, e posteriormente a adição de pacotes e árvore de dependências. É necessário ainda habilitar o *suricata.service*, o que significa que o *software* está em funcionamento.

Outro ponto importante é a configuração do Suricata. O *software* vem por padrão com o modo Sistema de Detecção de Intrusão (IDS) ativado, ou seja, caso seja necessário efetivamente bloquear o tráfego, o modo terá de ser alterado para Sistema de Prevenção de Intrusão (IPS). Dessa forma, as configurações do Suricata podem ser alteradas de acordo com o objetivo desejado e com o cenário de trabalho disponível.

4.3.2 Configuração das Interfaces de Rede

O servidor utilizado neste experimento possui duas interfaces de rede, ens33 e ens37. A primeira interface recebe *Internet* e é a interface de rede principal do servidor, enquanto a outra interface distribui *internet* para o restante da residência e dos dispositivos IoTs. Dessa forma, a interface ens33 foi configurada como padrão no Suricata, ou seja, o tráfego a ser inspecionado é proveniente dessa interface.

Para alterar a interface de rede, é necessário acessar o arquivo de configurações do Suricata, *suricata.yaml*. Dessa forma, a interface padrão eth0 foi substituída pela interface ens33, a qual recebe a *internet* de agora em diante.

A partir desse ponto, inicia-se a configuração do Sistema de Detecção de Intrusão (IDS) dentro do Suricata. Nessa configuração, existe uma variável chamada HOME_NET, a qual é preenchida com o endereço da sub-rede interna. Assim, a proteção ocorre na HOME_NET. A outra variável a ser configurada é a EXTERNAL_NET, que se refere à rede externa e desprotegida.

Dessa forma, a variável EXTERNAL_NET foi configurada como !$HOME_NET, onde a exclamação significa diferente. Assim, o Suricata considera

que qualquer valor que seja diferente da minha sub-rede HOME_NET é considerado como rede externa, ou seja, EXTERNAL_NET.

4.3.3 Implementação da Regra de Alerta

O *af-packet* no arquivo *suricata.yaml* guarda, além das configurações de rede, outras informações relevantes. Por exemplo, nesse arquivo, é indicado onde estão os arquivos de regras, no caso, *suricata.rules* e *detectando-ddos.rules* no caminho */var/lib/suricata/rules*. Todas essas indicações são necessárias para que o Suricata busque os arquivos de regras no caminho informado pelo usuário.

Nesse arquivo, é adicionada a regra de alerta, que está direcionada a ataques DDoS. A regra contém configurações do modo Sistema de Detecção e Intrusão (IDS), relacionadas à função de alertar e mostrar ao SYSADMIN do Suricata quando um determinado evento está ocorrendo. Dessa forma, a regra utilizada nesta dissertação é ilustrada na Figura 14 a seguir, na qual o texto em vermelho significa a ação da regra, o texto em verde significa o cabeçalho da regra e o texto em azul indica as opções da regra.

Figura 14 – Regra implementada no Suricata

```
alert tcp any any -> $HOME_NET 80 (msg: "Possivel attack
DDoS"; flags: S; flow: from_client; threshold: type both, track
by_dst, count 200, seconds 1; sid:99126611; rev:1;)
```

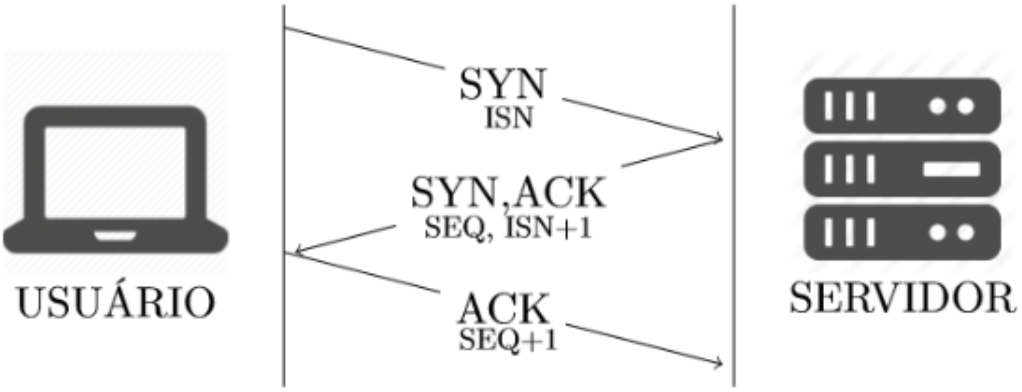
Fonte: elaborado pelo autor.

Nesse sentido, a ação da regra é alertar sobre a ocorrência de eventos, por isso a ação foi definida como *alert*, em vermelho. Já o cabeçalho da regra, em verde, indica qual o protocolo a ser inspecionado, nesse caso, o protocolo TCP. Após o protocolo, é determinada a fonte do tráfego e a porta de origem, as quais foram definidas como *any* e *any*, respectivamente, ou seja, a regra analisa o tráfego de qualquer IP fonte, proveniente de qualquer porta. A seta direcional indica a direção do tráfego, com origem em qualquer fonte, e com destino à HOME_NET, porta 80.

Além desses parâmetros, a regra também contém uma mensagem, um pequeno texto que será exibido nos *logs* quando o alerta ocorrer, sendo definida neste

trabalho como *possível ataque DDoS*. O uso da opção *flags S* significa que o fluxo analisado será TCP SYN, fazendo referência à primeira etapa do *Three-way Handshake*, em que o cliente envia um pacote com a flag SYN ativa, significando uma solicitação de sincronização com o servidor.

Figura 15 – Flag SYN no Three-way Handshake



Fonte: ALBUQUERQUE, et al. (2018).

De forma mais detalhada, esse processo de comunicação entre o servidor e o cliente nos moldes do protocolo TCP é ilustrado na Figura 15. A primeira etapa consiste no pedido de sincronização, o que ocorre quando o usuário envia um pacote com a *flag* SYM e um *Initial Sequence Number* (ISN), que é um número aleatório. Em sequência, inicia-se a segunda etapa, na qual o servidor envia ao usuário uma resposta contendo um pacote SYN/ACK, outro número aleatório, chamado de SEQ, que consiste no número de sequência gerado pelo servidor. Além disso, o servidor também envia ao usuário o ISN+1, o qual consiste em um reconhecimento do servidor em resposta ao pedido de sincronização recebido inicialmente com número de sequência ISN.

Na terceira etapa do *Three-way Handshake*, o usuário confirma ao servidor que recebeu o pacote, enviando um pacote ACK, um reconhecimento, SEQ+1, que é o valor enviado anteriormente pelo servidor mais um, e mais o número ISN+1 (ALBUQUERQUE et al., 2018).

Neste trabalho, foi optado por rastrear a primeira etapa da comunicação TCP, ou seja, a etapa de sincronização SYN, já que alertar conexões ilegítimas no início da comunicação TCP evita toda a cascata de resposta do servidor, evitando que a sua banda seja esgotada. Portanto, alertar o usuário na primeira etapa do *Three-*

way Handshake previne que os dispositivos IoT sofram e perpetuem os malefícios do ataque DDoS.

A próxima opção da regra se refere ao fluxo, ou *flow*. Assim, a opção definida como *flow from_client* indica que o fluxo a ser analisado possui direção se originando do cliente para o servidor, já que é dessa forma que se originam os ataques DDoS, do cliente para o servidor. Já a opção limiar (*threshold)* responsável por definir um limite de quantas vezes a regra será acionada em um determinado tempo, além de também definir um limiar mínimo de acionamento para que a regra seja alertada. Na regra do Suricata, por exemplo, foi optado por utilizar essas duas funcionalidades, por isso *threshold* foi definido como *both*. Dessa forma, os parâmetros *count* 200 e *seconds* 1 significam que a regra do Suricata somente dará origem a um alerta se durante o intervalo de 1 segundo forem identificados pelo menos 200 acionamentos da regra pelo Suricata.

A próxima opção de regra possível de ser implementada é a palavra-chave *sid*. Ela funciona como um identificador de cada regra, de tal forma que cada regra possui um *id* único. Para este trabalho, foi escolhido o *sid* 99126611, o qual estará presente nos *logs* quando a regra for acionada. Já a opção *rev* indica a versão da regra, ou seja, caso a regra seja alterada, é uma boa prática de desenvolvimento alterar o número de revisão para que haja registro nos *logs* indicando que a regra alvo de alguma modificação. Este trabalho contém *rev* 1, porque se trata da primeira versão da regra.

4.3.4 Configurações de Firewall

Após definir a regra para o Suricata, é importante alterar configurações de *firewall*, posto que o *firewall* padrão do sistema Ubuntu nessa versão é o Uncomplicated Firewall (UFW). Nessa configuração padrão, todo o tráfego que entra e sai da rede deve ser redirecionado ao firewall UFW.

Já na regra modificada, todo tráfego do *firewall* é enviado para o NFQUEUE do Suricata, o qual consiste um *iptables target* utilizado em *softwares* IDPS fazendo com que todo o tráfego que chegar e sair dessa rede seja aceito e redirecionado para o Suricata. Essa configuração possibilita que o tráfego de rede trafegue pelo *software* Suricata, para então ser analisado e gerar alertas.

4.3.5 Instalação e Configuração do Wazuh

Após alterar as configurações de *firewall*, outro ponto importante é a instalação e a configuração do *software* Wazuh no Ubuntu. Antes de realizar a instalação do Wazuh, deve-se adicionar a *GNU Privacy Guard* (GPG), que se trata de uma chave pública de criptografia do Wazuh no Ubuntu, para que seja possível efetuar as atividades de encriptar e desencriptar pacotes, necessárias às operações do Wazuh.

Somente após a chave GPG ser adicionada, passa-se a adicionar o repositório do Wazuh no Ubuntu. Ressalte-se que é necessário estar com o privilégio de usuário *root* para executar todos os comandos de instalação. Durante a instalação, executada por um assistente, são adicionadas as dependências, as configurações e o indexador, este último sendo responsável por pesquisar e analisar os alertas gerados pelo Wazuh.

Figura 16 – Integração do *software* Wazuh com os *logs* do Suricata

Fonte: elaborada pelo autor

Com o *software* instalado, segue-se à configuração do agente para permitir que o Wazuh possa realizar a leitura do arquivo de *log* proveniente do Suricata. Existem duas maneiras de realizar essa etapa, sendo a primeira acessando o arquivo no seguinte diretório */var/ossec/etc/ossec.conf*, ou a segunda maneira acessando as configurações do Wazuh por meio da própria interface gráfica, e então adicionar na penúltima linha do arquivo as configurações presentes da linha 405 a 408 da Figura 16 acima.

A partir de então, todos os *logs* que são gerados pelo Suricata serão analisados pelo Wazuh. Entretanto, para que o administrador da rede doméstica possa acessar de forma fácil os alertas recebidos, este trabalho implementou um *bot* no Telegram utilizando a linguagem de programação Python versão 3 para que esses alertas sejam enviados de forma automática via Telegram, etapa que será tratada no tópico a seguir.

4.3.6 Integração com o Telegram

Para implementar tal funcionalidade, foi necessário instalar o Python versão 3. Após a instalação, adicionar a biblioteca *pip*, responsável por efetuar instalações dentro do Python, para em seguida adicionar o *telepot.* Esse *framework* é capaz de integrar uma *Application Programming Interface* (API) criada pelo usuário ao *bot* do Telegram.

Com o *telepot* instalado e operante, é preciso criar um *bot* no Telegram. Tal atividade é realizada ao iniciar uma conversa com o BotFather, um *bot* nativo do Telegram que permite ao usuário criar e gerenciar seus próprios *bots*. Durante a interação com o BotFather, são disponibilizadas algumas opções ao usuário, desde criar um novo *bot,* deletar um *bot* já existente, gerar um novo *token* de autorização, revogar um token de autorização recém-implementado, configurar um nome para o *bot* criado, alterar imagem do *bot*, definir se o *bot* pode participar de grupos, alterar a descrição do *bot*, listar os bots criados por este usuário, bem como configurar a privacidade do *bot* em grupos. A opção selecionada foi */newbot* para criar um novo *bot.*

Em seguida, o BotFather permite ao usuário escolher um nome para o *bot*, e o nome escolhido para este trabalho foi Segurança da Rede. Esse nome aparece

como um nome de contato quando o usuário receber a notificação de alerta de ataques.

Outras informações repassadas posteriormente pelo BotFather são referentes à segurança, tais como um *token* e o *chat_id.* Essas informações são confidenciais e não podem ser repassadas a terceiros, já que são a garantia de que quem criou o *bot* é quem realmente está sendo o responsável por enviar as mensagens de agora em diante.

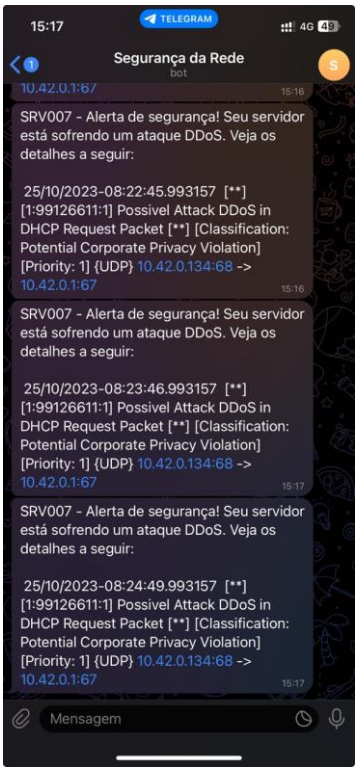
Em seguida, passe-se a criar um arquivo em Python e adicionar código nele. É necessário estabelecer uma comunicação entre a API recém-criada e o Telegram. Para este trabalho, foi criado o arquivo chamado de *trabalhador24h.py*. Dentro de tal arquivo, primeiramente deve-se importar o módulo *telepot*, para então repassar o *token* e o *chat_id* fornecidos pelo BotFather para a *API,* pois essas informações de segurança são necessárias para conectar ao Telegram.

Após a configuração de segurança, passa-se a tratar a variável que será recebida como alerta, posto que enviar todos os alertas antigos novamente a cada mensagem não teria utilidade para o usuário, além de dificultar a visualização de novos alertas. Pensando nisso, foi implementado um código em Python 3 com o intuito de percorrer os *logs* do Suricata e retornar apenas o último *log*, para que ele seja enviado via Telegram. Dessa forma, a cada novo alerta proveniente do Suricata, apenas esse último será enviado via Telegram, já que os alertas antigos já foram enviados, um por vez.

Nesse código, disponibilizado na íntegra no Apêndice A ao final desta dissertação, existe uma condicional, determinando que se o *id* da regra do Suricata for encontrado, será enviada uma mensagem ao Telegram através do *bot* definido anteriormente. A mensagem definida no código deste trabalho foi ‘SRV007 – Alerta de segurança! Seu servidor está participando de um ataque DDoS. Veja os detalhes a seguir:’, sendo complementada pelo texto do alerta proveniente do *log* do Suricata, o qual contém a data e hora do alerta, mensagem de possível ataque DDoS e os IPs envolvidos.

A Figura 17 a seguir demonstra a API em funcionamento, com o recebimento de mensagens via Telegram informando quando ocorrem ataques DDoS. É possível visualizar que as mensagens foram enviadas do *bot* chamado Segurança da Rede, que foi o *bot* criado e configurado anteriormente neste trabalho.

Figura 17 – Alertas de ataques DDoS recebidos via Telegram



Fonte: elaborada pelo autor

A mensagem enviada contém a data e hora do alerta, bem como a mensagem que foi definida nas regras do Suricata anteriormente, informando sobre possível ataque DDoS, e outras informações pertinentes sobre os alertas de segurança. Assim, o *log* do Suricata é acessado pelo Python, filtrado pelo comando *tail* do Linux, integrado ao Telegram com o *framework telepot*, culminando no envio de alertas de ataques por meio do *bot* do Telegram.

4.3.7 Considerações sobre a Implementação

Ressalta-se que o procedimento para todas as instalações e implementações supracitadas foi permeado por erros e por tentativas repetidas até se alcançar o resultado almejado. Por esse motivo, foi levantada a ideia de criar uma solução de segurança com esses procedimentos, ficando como legado deste trabalho.

Nesse sentido, por ser bastante complicado implementar todas essas funcionalidades em um sistema operacional, e com objetivos educacionais, científicos e também para ajudar as pessoas, foi criada uma solução que permitisse ao usuário doméstico instalar em um *hardware* de baixo custo essa ferramenta para alertar sobre

incidentes envolvendo a segurança doméstica dos seus equipamentos. É importante ressaltar que para chegar a esse resultado foram instaladas 137 máquinas virtuais, das quais 136 apresentaram algum comportamento de erro e tiveram que ser descartadas. Para usá-lo, é necessário o usuário alterar o *token* e *chat_id* do Telegram. Essa implementação foi nomeada de Block Suricata Anti-DDoS.

Dessa forma, o Block Suricata Anti-DDoS está pronto para o uso por um usuário residencial, sem conhecimentos aprofundados de programação ou de redes de computadores. Além disso, a solução de segurança apresentada neste trabalho supera a simples instalação das ferramentas apresentadas, a saber, sistema operacional Ubuntu, Suricata, Wazuh, posto que inclui configurações de interfaces de rede e de *firewall* para o funcionamento correto ao proposto, inclui programação para integração entre Wazuh e *logs* do Suricata, inclui programação de código para leitura dos *logs* do Suricata e envio do resultado desses *logs* por meio de notificações ao Telegram, além de regra de alerta modificada para melhor desempenho na detecção de ataques DDoS pelo sistema Suricata.

5 AVALIAÇÃO DE DESEMPENHO

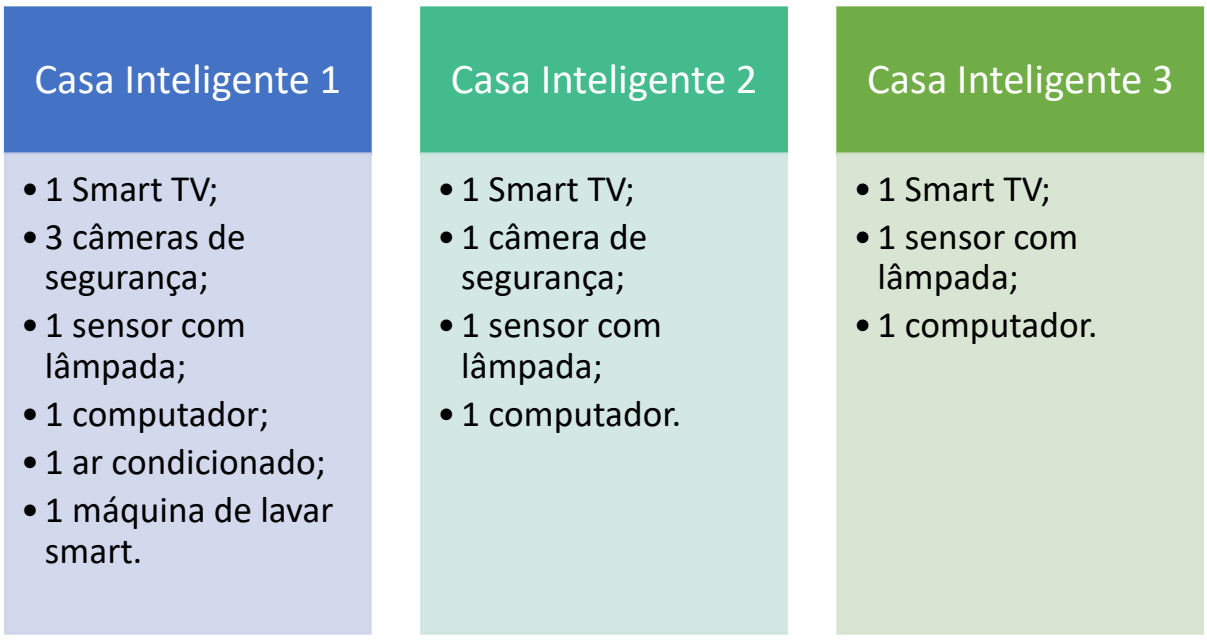
Nesta etapa do trabalho, foi realizada a avaliação de desempenho da solução proposta Block Suricata Anti-DDoS, com o intuito de verificar quais fatores influenciam na acurácia da detecção de ataques e no consumo de memória e de CPU.

Devido ao caráter experimental deste trabalho, a avaliação de desempenho foi realizada utilizando a técnica de medição. Foram realizados testes em um ambiente controlado, apenas dentro da rede interna das casas inteligentes.

5.1 CENÁRIO

A avaliação de desempenho foi composta pelo cenário de três casas inteligentes, contendo ao todo três *smart tv*, quatro câmeras de segurança, três sensores com lâmpadas, três computadores, um ar condicionado e uma máquina de lavar *smart*. Dessa forma, a avaliação inclui residências com dispositivos IoT e não IoT conectados. A Figura 18 a seguir ilustra com detalhes quais são os dispositivos presentes em cada casa inteligente.

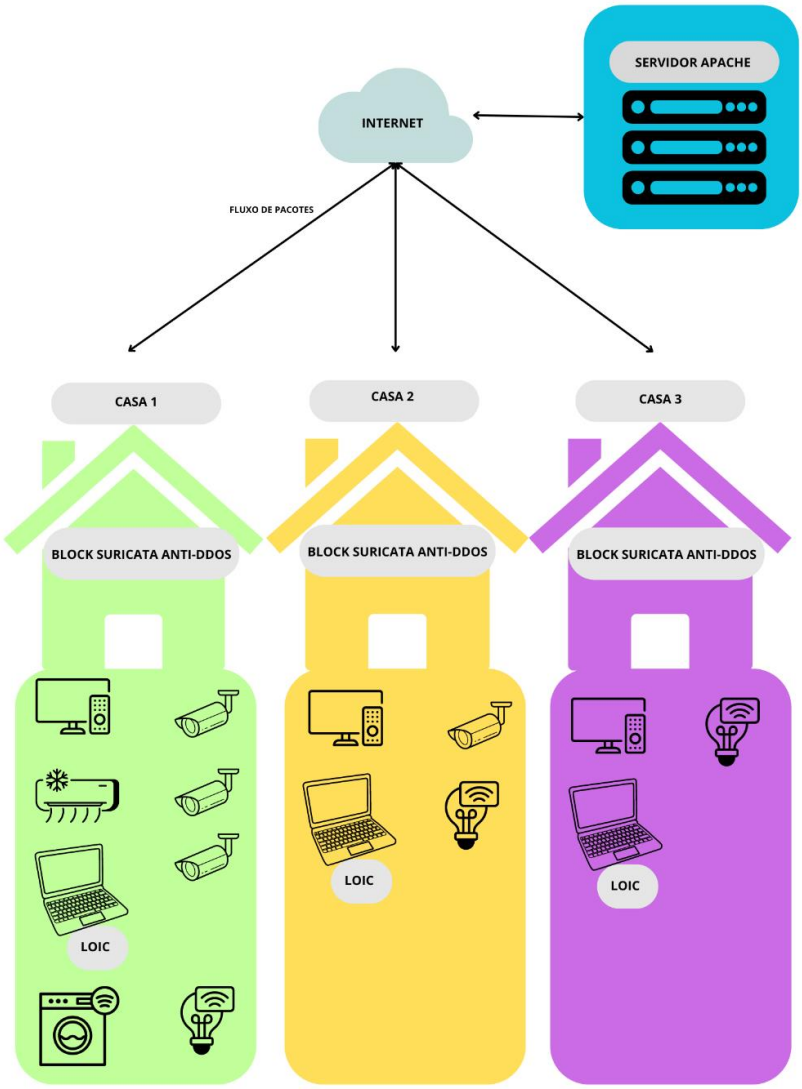
Figura 18 – Dispositivos presentes em cada casa inteligente



Fonte: elaborada pelo autor

A Figura 19, por sua vez, ilustra o cenário contendo as três casas inteligentes, bem como os dispositivos presentes em cada residência. Nota-se que na parte superior da imagem está representado o servidor *web* que recebe os ataques, estando conectado à *Internet*, por onde ocorre o fluxo de pacotes. Já na parte inferior da imagem, é possível observar as três casas de teste conectadas à *Internet.* Cada casa possui a solução de segurança Block Suricata Anti-DDoS instalada em um microcomputador, além dos dispositivos que compõem a rede. Ressalta-se a presença do *software* Low Orbit Ion Cannon (LOIC) instalado em um *notebook* da rede, ou seja, um dispositivo que está dentro da rede ataca o servidor *web*, que está fora da rede. Então, as tentativas de ataque DDoS são detectadas pela solução de segurança.

Figura 19 – Cenário da Avaliação de Desempenho



Fonte: elaborada pelo autor

5.2 SISTEMA, SERVIÇO E MÉTRICAS

No presente trabalho, o sistema consiste em um computador de baixo custo com a solução Block Suricata Anti-DDoS em funcionamento. Ressalte-se que o microcomputador concentra as duas tarefas, ou seja, processamento e análise dos dados de monitoramento, permitindo que os dispositivos IoT da rede residencial realizem as suas atividades habituais, sem lhes causar prejuízo da capacidade de processamento.

Quanto ao serviço, consiste em detectar as tentativas de ataques DDoS tipo inundação HTTP. Dessa forma, as medições do sistema proposto são concentradas na detecção de ataques recebidos pelos clientes.

Para além da definição do sistema e do serviço que são alvo da presente avaliação de desempenho, este trabalho se utiliza das seguintes métricas: acurácia de detecção de ataques e métricas relacionadas à utilização dos recursos. Quanto às métricas de utilização de recursos, foram usadas o consumo de memória e o consumo de CPU.

A escolha de tais métricas se baseou no fato de elas estarem presentes em muitos trabalhos relacionados, e que são frequentemente utilizadas em trabalhos da área. Além disso, outro fator preponderante é o fato de tais métricas impactarem no desempenho do sistema durante o período de testes.

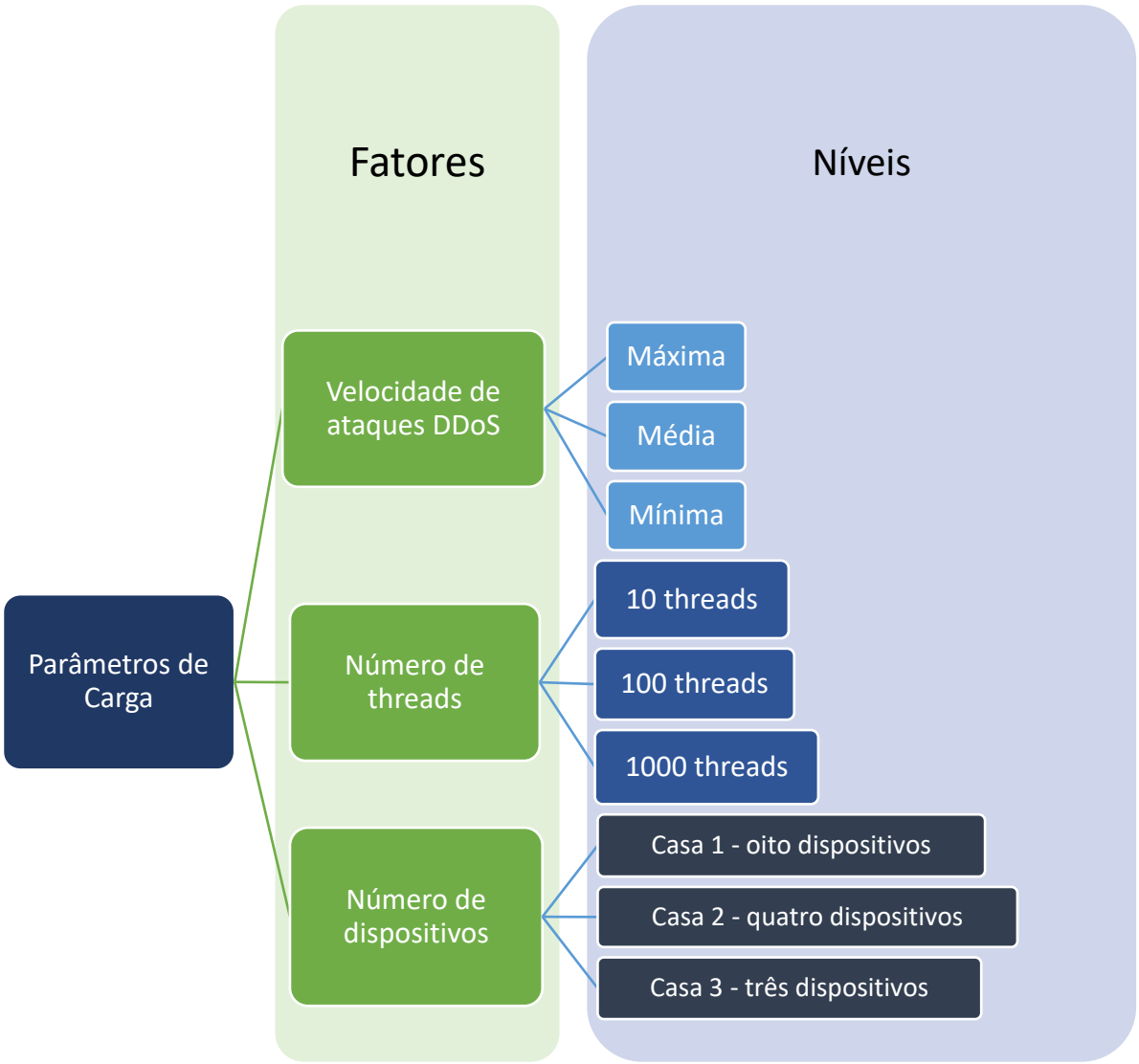
5.3 PARÂMETROS DE CARGA

Neste trabalho, foram considerados três fatores de desempenho: a velocidade de ataques gerada pelo *software Low Orbit Ion Cannon* (LOIC), o número de dispositivos IoT e o número de *threads*, que simula a quantidade de usuários fazendo requisições no site. Tais fatores possuem variações chamadas de níveis, as quais impactarão no desempenho do sistema.

Quanto à velocidade de ataques, existem três diferentes níveis. O primeiro nível foi configurado com a velocidade máxima de ataques DDoS gerados pelo *software* LOIC, o segundo nível com a velocidade média, e o terceiro nível com a velocidade mínima de ataques DDoS. Quanto ao número de *threads* utilizados pelo *software* LOIC, este trabalho utilizou as opções de 10, 100 e 1000 *threads*. Por último, o número de dispositivos IoT em cada casa também se configura como um fator de

desempenho, sendo um fator relacionado ao sistema, variando de oito dispositivos na Casa 1, quatro dispositivos na Casa 2 e três dispositivos na Casa 3. Tais configurações estão organizadas na Figura 20 abaixo, contendo a divisão de fatores e níveis que foram adotados como parâmetros de carga dos experimentos nesta dissertação.

Figura 20 – Parâmetros de Carga: Fatores e Níveis



Fonte: elaborada pelo autor

Essa definição de diferentes parâmetros de carga tem o objetivo de observar se de acordo com a variação de ataques, de *threads* e de número de dispositivos, tanto o *software* quanto o *hardware* da solução proposta obtêm alterações em seu desempenho, influenciando na acurácia de detecção.

5.4 DESCRIÇÃO DOS EXPERIMENTOS

O experimento consistiu em gerar ataques DDoS do tipo inundação HTTP utilizando o software LOIC, para então analisar a eficiência do painel do Block Suricata Anti-DDoS em detectar tais ataques e notificá-los via aplicativo Telegram.

No experimento, são conectados os IoT nas redes das três casas inteligentes, a saber, três *smart tv*, quatro câmeras de segurança, três sensores com lâmpadas, três computadores, um ar-condicionado e uma máquina de lavar *smart*, conforme apresentado na Figura 18 anteriormente.

O ataque é simulado através de um dispositivo que está dentro da rede, por meio do software *Low Orbit Ion Cannon* (LOIC), a ferramenta mais indicada para realizar o ataque DDoS tipo inundação HTTP, podendo realizar ataques DDoS de inundação HTTP, *Transmission Control Protocol* (TCP) e UDP.

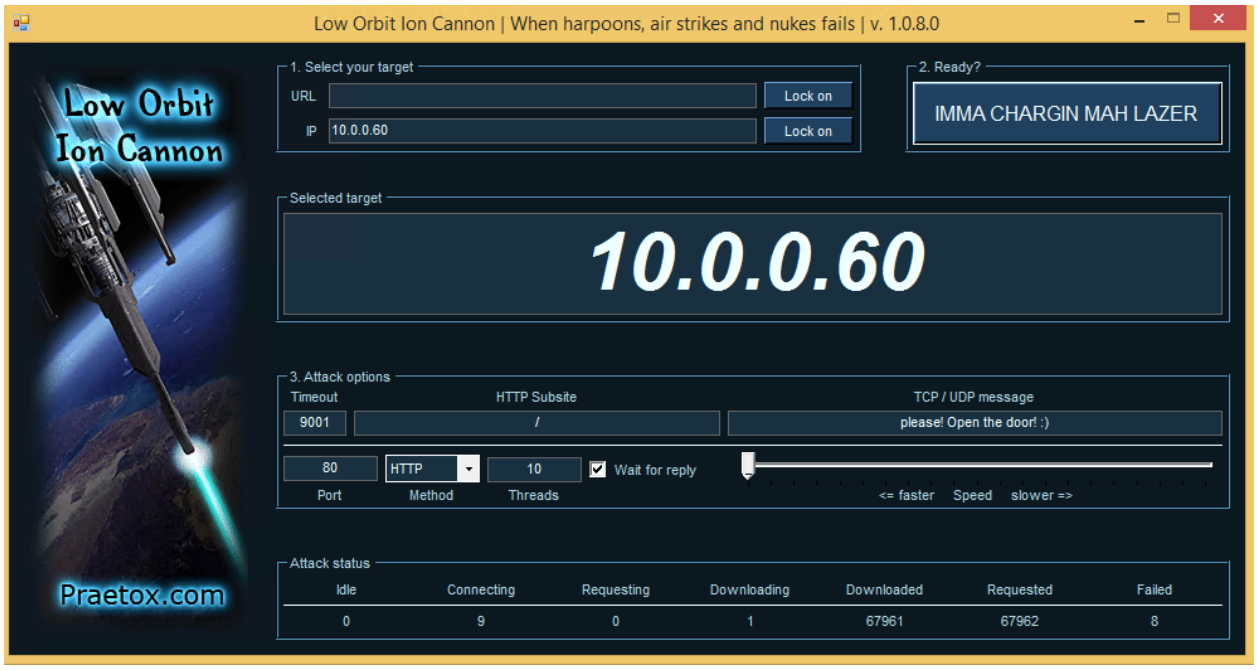
Nesse *software*, é possível inserir o endereço da vítima nos modos *Uniform Resource Locator* (URL) ou Internet Protocol (IP), para então gerar um grande tráfego UDP, TCP ou HTTP, conforme escolha (ARAÚJO, 2017).

Dessa forma, um dispositivo que está dentro da rede ataca um servidor *web*, que está fora da rede. Então, as tentativas de ataque DDoS são detectadas pelo *software* Suricata, com envio de alerta via aplicativo de mensagem instantânea Telegram para o responsável pela residência.

Diante dessas considerações, o *software Low Orbit Ion Cannon* (LOIC) é responsável por gerar um volume grande de tráfego HTTP, ou seja, neste trabalho o LOIC tem o objetivo de realizar ataques DDoS tipo inundação HTTP.

Na Figura 21 a seguir, verifica-se que existem algumas informações a serem adicionadas ao LOIC para que ele possa entrar em funcionamento. Primeiramente, é necessário selecionar o alvo na seção ‘*Select your target*’ o qual pode ser uma URL ou um IP. Após definir o alvo, é necessário clicar no botão ‘*lock on’* para bloquear o IP no qual se deseja realizar ataques, e então o IP aparecerá no campo ‘*selected target’*. Na seção 3, são disponibilizadas algumas opções para o ataque, incluindo desde a configuração do *timeout*, que significa o tempo máximo de resposta; a opção de selecionar um sub-site, em que foi selecionada a raiz do site; a opção de exibir uma mensagem TCP; a seleção da porta 80 e do método HTTP, já que se trata de ataque de inundação HTTP.

Figura 21 – LOIC realizando ataques DDoS tipo inundação HTTP.



Fonte: elaborada pelo autor

Já na opção *threads*, há a possibilidade de simular a quantidade de usuários fazendo requisições no site. A opção '*wait for reply*' permite que o *software* não desconecte quando o servidor começar a responder, mantendo a conexão. É possível ainda escolher a velocidade do ataque, desde mais rápida até uma velocidade mais lenta.

Após configurar o alvo e as opções do ataque, pode-se clicar no botão '*IMMA CHARGIN MAH LAZER*', disponível na área superior direita da tela. Tal botão é responsável por iniciar o ataque. Após o início do ataque, aparecem informações no campo de *status* do ataque, o qual fica localizado na parte inferior da tela. O campo IDLE mostra quantos *threads* não estão trabalhando no momento, e esse número deve ser zero ou próximo de zero para maior eficiência do ataque. A seguir, o campo *connecting* demonstra quantos *threads* estão tentando conectar, enquanto o campo *requesting* mostra quantos *threads* estão efetivamente requisitando conexão com o servidor.

O campo *downloading* significa quantas *threads* estão baixando dados do servidor no momento, enquanto *downloaded* se refere a quantos *downloads* ocorreram no total. O campo *requested* mostra quantas requisições foram feitas no total, enquanto o campo *failed* mostra quantas vezes no total o servidor não

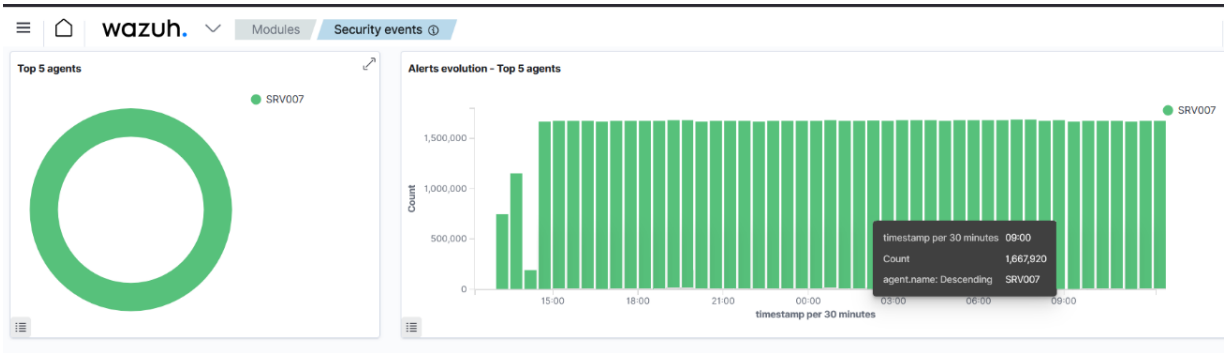
respondeu, o que pode significar um servidor desligado se houver um número alto de falhas, ou a própria limitação de *hardware* ao receber requisições, se houver um número menor de falhas.

Diante do exposto, o presente trabalho considerou o número de ataques DDoS disparados pelo LOIC e o número de alertas gerados pela solução Block Suricata Anti-DDoS. Após a etapa de coleta de dados, foram calculados o intervalo de confiança, o desvio padrão e a média aritmética das medições obtidas.

5.5 AVALIAÇÃO DOS RESULTADOS

Após os ataques pelo LOIC iniciarem, houve uma primeira mudança no painel, indicando que uma atividade anormal estava ocorrendo no sistema. Assim, a contagem de alertas gerados pelo Suricata foi visualizada no painel do *software* Wazuh, conforme a Figura 22 a seguir. Este painel é chamado de *security events*, e exibe os principais eventos de segurança. Do lado esquerdo da referida figura, é possível observar que os alertas de seguranças foram todos provenientes do agente SRV007, o qual é a máquina onde está instalado o Wazuh.

Figura 22- Painel Wazuh demonstrando eventos de segurança

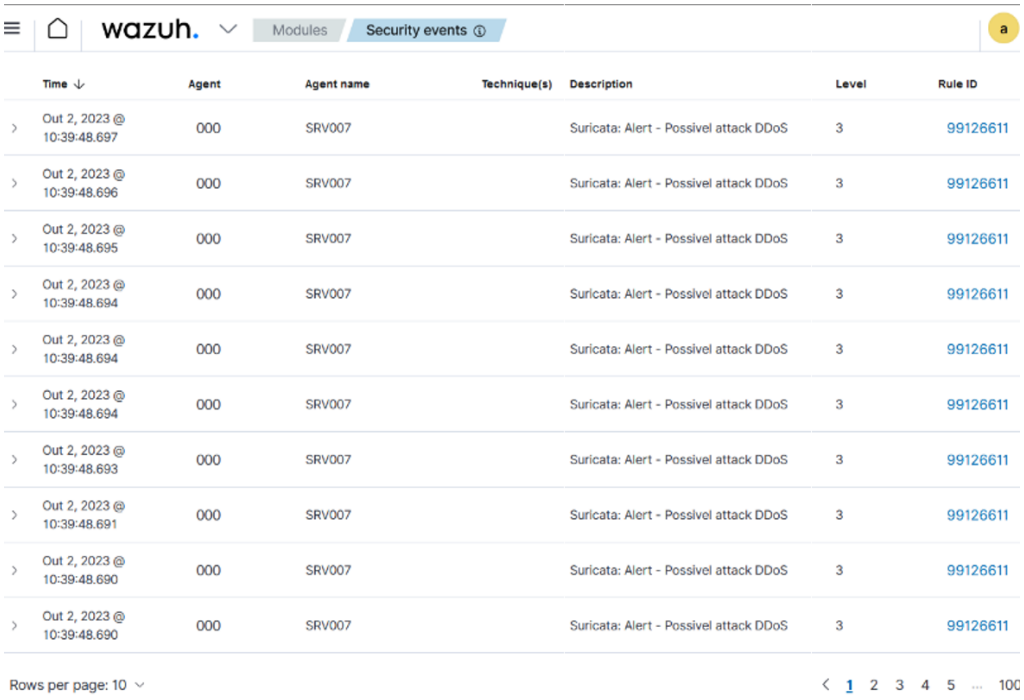


Fonte: elaborada pelo autor

Já do lado direito, em um gráfico de barras, observa-se a evolução de alertas. O Wazuh identificou um alto número de alertas, indicando um alto tráfego de ataques, chegando-se ao patamar de um milhão e meio de alertas identificados a cada meia hora. Cada barra verde representa um período de meia hora no eixo horizontal, chamado de *timestamp*, e a quantidade de alertas no eixo vertical.

Ainda no painel *security events* do Wazuh, existe a possibilidade de detalhar os alertas identificados. Na Figura 23 a seguir, é possível visualizar alguns dos primeiros alertas recebidos, com a seguinte estrutura: na primeira coluna, é detalhada a data e a hora do alerta, seguida pelo nome do agente SRV007 e a descrição do alerta definida na regra do suricata, ou seja, 'Suricata: Alerta – Possível ataque DDoS', a qual foi definida dentro da regra do Suricata, no campo *msg*, conforme implementado na seção 4.2. Além dessas informações, é mostrado ainda o *rule id*, que foi configurado na regra do Suricata como 99126611. Esse número é importante para identificar a regra que gerou o alerta, já que se trata de um número único.

Figura 23- Painel Wazuh detalhando os alertas de segurança



Fonte: elaborada pelo autor

Quanto aos resultados, foram executados os experimentos acima descritos, no período de 02 de abril de 2024 a 20 de maio de 2024, sendo realizadas 4 aferições por dia, nos horários de 0 horas, 6 horas, 12 horas e 18 horas, totalizando 66 aferições em cada casa inteligente. Foram realizadas médias aritméticas das aferições para obter a média de consumo de memória, consumo de CPU e acurácia de detecção.

Foram considerados os valores obtidos nos testes nas três casas inteligentes, separadamente. Foi considerado um nível de confiança de 95% para calcular o intervalo de confiança. Observa-se nas Tabelas 4, 5 e 6 abaixo os

resultados dos experimentos em cada casa inteligente, de acordo com a velocidade de ataques DDoS gerados pelo software Low Orbit Ion Cannon (LOIC) - mínima, média e máxima, e com as opções de 10, 100 e 1000 threads.

Tabela 4 - Resultados dos experimentos na Casa Inteligente 1

CASA INTELIGENTE 1			
VELOCIDADE DE ATAQUES DDOS: MÍNIMA			
QUANTIDADE DE THREADS	**10**	**100**	**1000**
CONSUMO DE MEMÓRIA (%)	35,1%	78%	78%
DESVIO PADRÃO (%)	0,327	0,394	0,386
NÍVEL DE CONFIANÇA	0,5	0,5	0,5
MARGEM DE ERRO	0,07889	0,09505	0,09312
INTERVALO DE CONFIANÇA (%)	35,021	77,905	77,907
	35,179	78,095	78,093
CONSUMO DE CPU (%)	41,3%	67%	87%
DESVIO PADRÃO (%)	0,285	0,341	0,276
NÍVEL DE CONFIANÇA	0,5	0,5	0,5
MARGEM DE ERRO	0,06805	0,08226	0,06685
INTERVALO DE CONFIANÇA (%)	41,231	66,918	86,933
	41,369	67,082	87,067
ACURÁCIA DE DETECÇÃO (%)	99,9%	99,9%	99,8%
DESVIO PADRÃO (%)	0,405	0,293	0,301
NÍVEL DE CONFIANÇA	0,5	0,5	0,5
MARGEM DE ERRO	0,097708	0,07688	0,072618
INTERVALO DE CONFIANÇA (%)	99,802	99,83	99,727
	99,998	99,97	99,873

VELOCIDADE DE ATAQUES DDOS: MÉDIA			
QUANTIDADE DE THREADS	**10**	**100**	**1000**
CONSUMO DE MEMÓRIA (%)	57,1%	77%	79%
DESVIO PADRÃO (%)	0,314	0,389	0,397
NÍVEL DE CONFIANÇA	0,5	0,5	0,5
MARGEM DE ERRO	0,075754	0,093848	0,095778

INTERVALO DE CONFIANÇA (%)	57,024	76,906	78,904
	57,176	77,094	79,096
CONSUMO DE CPU (%)	59,7%	83%	94%
DESVIO PADRÃO (%)	0,218	0,368	0,311
NÍVEL DE CONFIANÇA	0,5	0,5	0,5
MARGEM DE ERRO	0,052594	0,088782	0,07503
INTERVALO DE CONFIANÇA (%)	59,647	82,911	93,925
	59,753	83,089	94,075
ACURÁCIA DE DETECÇÃO (%)	99,9%	99,8%	99,8%
DESVIO PADRÃO (%)	0,535	0,325	0,409
NÍVEL DE CONFIANÇA	0,5	0,5	0,5
MARGEM DE ERRO	0,129071	0,078408	0,098673
INTERVALO DE CONFIANÇA (%)	99,77	99,722	99,701
	100	99,878	99,899
VELOCIDADE DE ATAQUES DDOS: MÁXIMA			
QUANTIDADE DE THREADS	**10**	**100**	**1000**
CONSUMO DE MEMÓRIA (%)	59,5%	76%	87%
DESVIO PADRÃO (%)	0,325	0,401	0,346
NÍVEL DE CONFIANÇA	0,5	0,5	0,5
MARGEM DE ERRO	0,078408	0,096743	0,083474
INTERVALO DE CONFIANÇA (%)	59,422	75,903	86,917
	59,578	76,097	87,083
CONSUMO DE CPU (%)	63,7%	85%	96%
DESVIO PADRÃO (%)	0,204	0,289	0,234
NÍVEL DE CONFIANÇA	0,5	0,5	0,5
MARGEM DE ERRO	0,049216	0,069723	0,056454
INTERVALO DE CONFIANÇA (%)	63,651	84,93	95,944
	63,749	85,07	96,056
ACURÁCIA DE DETECÇÃO (%)	95,10%	59,54%	5,58%
DESVIO PADRÃO (%)	0,628	0,457	0,482
NÍVEL DE CONFIANÇA	0,5	0,5	0,5
MARGEM DE ERRO	0,151518	0,110253	0,116285
	94,948	54,43	5,4637

INTERVALO DE CONFIANÇA (%)	95,252	54,65	5,6963

Fonte: elaborada pelo autor

Tabela 5 - Resultados dos experimentos na Casa Inteligente 2

CASA INTELIGENTE 2			
VELOCIDADE DE ATAQUES DDOS: MÍNIMA			
QUANTIDADE DE THREADS	**10**	**100**	**1000**
CONSUMO DE MEMÓRIA (%)	33,6%	56%	56%
DESVIO PADRÃO (%)	0,351	0,248	0,299
NÍVEL DE CONFIANÇA	0,5	0,5	0,5
MARGEM DE ERRO	0,08468	0,059831	0,072135
INTERVALO DE CONFIANÇA (%)	33,515	55,94	55,928
	33,685	56,06	56,072
CONSUMO DE CPU (%)	40,7%	57%	79%
DESVIO PADRÃO (%)	0,233	0,246	0,212
NÍVEL DE CONFIANÇA	0,5	0,5	0,5
MARGEM DE ERRO	0,056212	0,059349	0,051146
INTERVALO DE CONFIANÇA (%)	40,644	56,941	78,949
	40,756	57,059	79,051
ACURÁCIA DE DETECÇÃO (%)	99,8%	99,8%	99,8%
DESVIO PADRÃO (%)	0,369	0,377	0,403
NÍVEL DE CONFIANÇA	0,5	0,5	0,5
MARGEM DE ERRO	0,089023	0,090953	0,97226
INTERVALO DE CONFIANÇA (%)	99,71	99,71	99,703
	99,89	99,89	99,897
VELOCIDADE DE ATAQUES DDOS: MÉDIA			
QUANTIDADE DE THREADS	**10**	**100**	**1000**
CONSUMO DE MEMÓRIA (%)	43,2%	56%	56%

DESVIO PADRÃO (%)	0,385	0,391	0,354
NÍVEL DE CONFIANÇA	0,5	0,5	0,5
MARGEM DE ERRO	0,092883	0,09433	0,085404
INTERVALO DE CONFIANÇA (%)	43,107	55,906	55,915
	43,293	56,094	56,085
CONSUMO DE CPU (%)	91,3%	66%	83%
DESVIO PADRÃO (%)	0,402	0,387	0,321
NÍVEL DE CONFIANÇA	0,5	0,5	0,5
MARGEM DE ERRO	0,096984	0,093366	0,077443
INTERVALO DE CONFIANÇA (%)	91,203	65,907	82,923
	91,397	66,093	83,077
ACURÁCIA DE DETECÇÃO (%)	99,8%	99,8%	99,8%
DESVIO PADRÃO (%)	0,428	0,459	0,481
NÍVEL DE CONFIANÇA	0,5	0,5	0,5
MARGEM DE ERRO	0,103257	0,11736	0,116044
INTERVALO DE CONFIANÇA (%)	98.997	99,69	99,684
	99.203	99,91	99,916
VELOCIDADE DE ATAQUES DDOS: MÁXIMA			
QUANTIDADE DE THREADS	**10**	**100**	**1000**
CONSUMO DE MEMÓRIA (%)	74,8%	56%	56%
DESVIO PADRÃO (%)	0,358	0,297	0,346
NÍVEL DE CONFIANÇA	0,5	0,5	0,5
MARGEM DE ERRO	0,08637	0,071653	0,083474
INTERVALO DE CONFIANÇA (%)	74,714	55,928	55,917
	74,886	56,072	56,083
CONSUMO DE CPU (%)	99,6%	77%	94%
DESVIO PADRÃO (%)	0,212	0,249	0,285
NÍVEL DE CONFIANÇA	0,5	0,5	0,5
MARGEM DE ERRO	0,051146	0,060072	0,068758

INTERVALO DE CONFIANÇA (%)	99,549	76,94	93,931
	99,651	77,06	94,069
ACURÁCIA DE DETECÇÃO (%)	96,9%	54,92%	29,31%
DESVIO PADRÃO (%)	0,534	0,547	0,698
NÍVEL DE CONFIANÇA	0,5	0,5	0,5
MARGEM DE ERRO	0,12883	0,131966	0,291416
INTERVALO DE CONFIANÇA (%)	96,771	54,788	0,294784
	97,029	55,052	0,168396

Fonte: elaborada pelo autor

Tabela 6 - Resultados dos experimentos na Casa Inteligente 3

CASA INTELIGENTE 3			
VELOCIDADE DE ATAQUES DDOS: MÍNIMA			
QUANTIDADE DE THREADS	**10**	**100**	**1000**
CONSUMO DE MEMÓRIA (%)	26,4%	59%	57%
DESVIO PADRÃO (%)	0,158	0,172	0,203
NÍVEL DE CONFIANÇA	0,5	0,5	0,5
MARGEM DE ERRO	0,038118	0,041496	0,048975
INTERVALO DE CONFIANÇA (%)	26,362	58,959	56,951
	26,438	59,041	57,049
CONSUMO DE CPU (%)	34,7%	61%	66%
DESVIO PADRÃO (%)	0,217	0,189	0,267
NÍVEL DE CONFIANÇA	0,5	0,5	0,5
MARGEM DE ERRO	0,052352	0,045597	0,064415
INTERVALO DE CONFIANÇA (%)	34,6476	60,954	65,936
	34,7524	61,046	66,064
ACURÁCIA DE DETECÇÃO (%)	99,9%	99,9%	99,8%
DESVIO PADRÃO (%)	0,102	0,207	0,299
NÍVEL DE CONFIANÇA	0,5	0,5	0,5
MARGEM DE ERRO	0,024608	0,04994	0,072135
INTERVALO DE CONFIANÇA (%)	99,875	99,85	99,728
	99,925	99,95	99,872

VELOCIDADE DE ATAQUES DDOS: MÉDIA			
QUANTIDADE DE THREADS	**10**	**100**	**1000**
CONSUMO DE MEMÓRIA (%)	37,1%	59%	50%
DESVIO PADRÃO (%)	0,254	0,269	0,314
NÍVEL DE CONFIANÇA	0,5	0,5	0,5
MARGEM DE ERRO	0,064415	0,064415	0,064415
INTERVALO DE CONFIANÇA (%)	37,0356	58,936	49,936
	37,1644	59,064	50,064
CONSUMO DE CPU (%)	59%	60%	73%
DESVIO PADRÃO (%)	0,352	0,246	0,321
NÍVEL DE CONFIANÇA	0,5	0,5	0,5
MARGEM DE ERRO	0,064415	0,064414	0,064415
INTERVALO DE CONFIANÇA (%)	58,936	59,935	72,936
	59,136	60,064	73,064
ACURÁCIA DE DETECÇÃO (%)	99,9%	99,8%	99,8%
DESVIO PADRÃO (%)	0,298	0,314	0,387
NÍVEL DE CONFIANÇA	0,5	0,5	0,5
MARGEM DE ERRO	0,103257	0,075754	0,093366
INTERVALO DE CONFIANÇA (%)	99,828	99,724	99,707
	99,972	99,876	99,893
VELOCIDADE DE ATAQUES DDOS: MÁXIMA			
QUANTIDADE DE THREADS	**10**	**100**	**1000**
CONSUMO DE MEMÓRIA (%)	59%	59%	51%
DESVIO PADRÃO (%)	0,376	0,341	0,302
NÍVEL DE CONFIANÇA	0,5	0,5	0,5
MARGEM DE ERRO	0,064415	0,082268	0,072859
INTERVALO DE CONFIANÇA (%)	58,936	58,918	50,927
	59,064	59,082	51,073
CONSUMO DE CPU (%)	63%	66%	84%
DESVIO PADRÃO (%)	0,245	0,213	0,287
NÍVEL DE CONFIANÇA	0,5	0,5	0,5
MARGEM DE ERRO	0,059107	0,051387	0,06924
	62,941	65,949	63,931

INTERVALO DE CONFIANÇA (%)	63,059	66,051	64,069
ACURÁCIA DE DETECÇÃO (%)	99,7%	57,58%	35,21%
DESVIO PADRÃO (%)	0,409	0,435	0,496
NÍVEL DE CONFIANÇA	0,5	0,5	0,5
MARGEM DE ERRO	0,098673	0,104946	0,119662
INTERVALO DE CONFIANÇA (%)	99,601	57,475	35,09
	99,799	57,685	35,33

Fonte: elaborada pelo autor

De acordo com as Tabelas 4, 5 e 6 dispostas acima, o Block Suricata Anti-DDoS alcançou um desempenho mais satisfatório com a carga de 10 *threads*, velocidade de ataques mínima e o cenário da Casa Inteligente 3. Nessas configurações, calculando-se a média aritmética dos cenários, a solução proposta obteve 26,4% de consumo de memória, 34,7% de consumo de CPU e 99,9% de acurácia de detecção, apresentando o menor consumo de *hardware* e a maior acurácia de detecção de todos os testes realizados.

Esse melhor resultado provavelmente ocorreu pelo número reduzido de *threads* e da velocidade de ataques mínima, diminuindo as requisições de *hardware*. Além disso, o cenário da Casa Inteligente 3 possui o menor número de dispositivos conectados, influenciando positivamente no desempenho do sistema de detecção, porque apesar de o sistema estar hospedado em um *hardware* separado dos dispositivos residenciais, o tráfego a ser analisado aumenta conforme a quantidade de dispositivos presentes na rede residencial. Assim, quanto menor o número de dispositivos, há menos tráfego de rede na casa, menos tráfego no IDS e uma melhor acurácia de detecção.

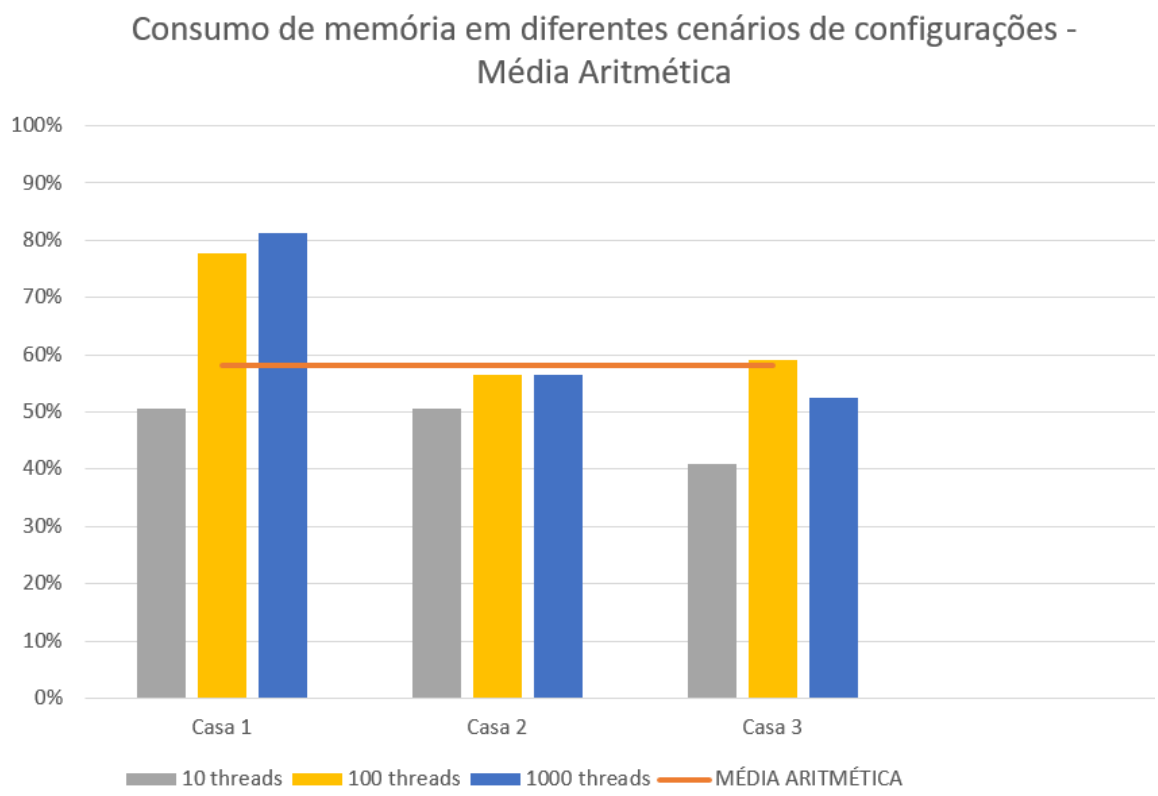
Nesse sentido, os resultados apresentam uma piora quando são analisadas as Casas Inteligentes 1 e 2, provavelmente por causa do aumento do número de dispositivos presentes nessas residências, sendo a Casa Inteligente 1 a que possui mais dispositivos e um pior resultado nos testes efetuados, ou seja, aumento do consumo de CPU e memória, e diminuição da acurácia de detecção. Dessa forma, o pior cenário avaliado consistiu na Casa Inteligente 1 com a carga de 1000 *threads* e a velocidade máxima de ataques, resultando em um consumo de memória médio de 87%, 96% de consumo de CPU e 5,58% de acurácia de detecção, apresentando o

maior consumo de *hardware* e a menor acurácia de detecção de todos os experimentos.

Quanto ao desvio padrão, os valores obtidos se mantêm baixos durante os testes efetuados, indicando pouca variabilidade nos resultados e confiabilidade da solução Block Suricata Anti-DDoS.

Em se tratando do consumo de memória, houve na primeira residência uma média aritmética de 69,63% das aferições, na segunda residência uma média aritmética de 54,17% e na terceira residência uma média aritmética de 50,83%. Ao se considerar uma média aritmética dos três cenários, a solução proposta alcança uma média de 58,21% de consumo de memória. Tais resultados estão dispostos no Gráfico 2 a seguir.

Gráfico 2 – Consumo de memória em diferentes cenários de configurações



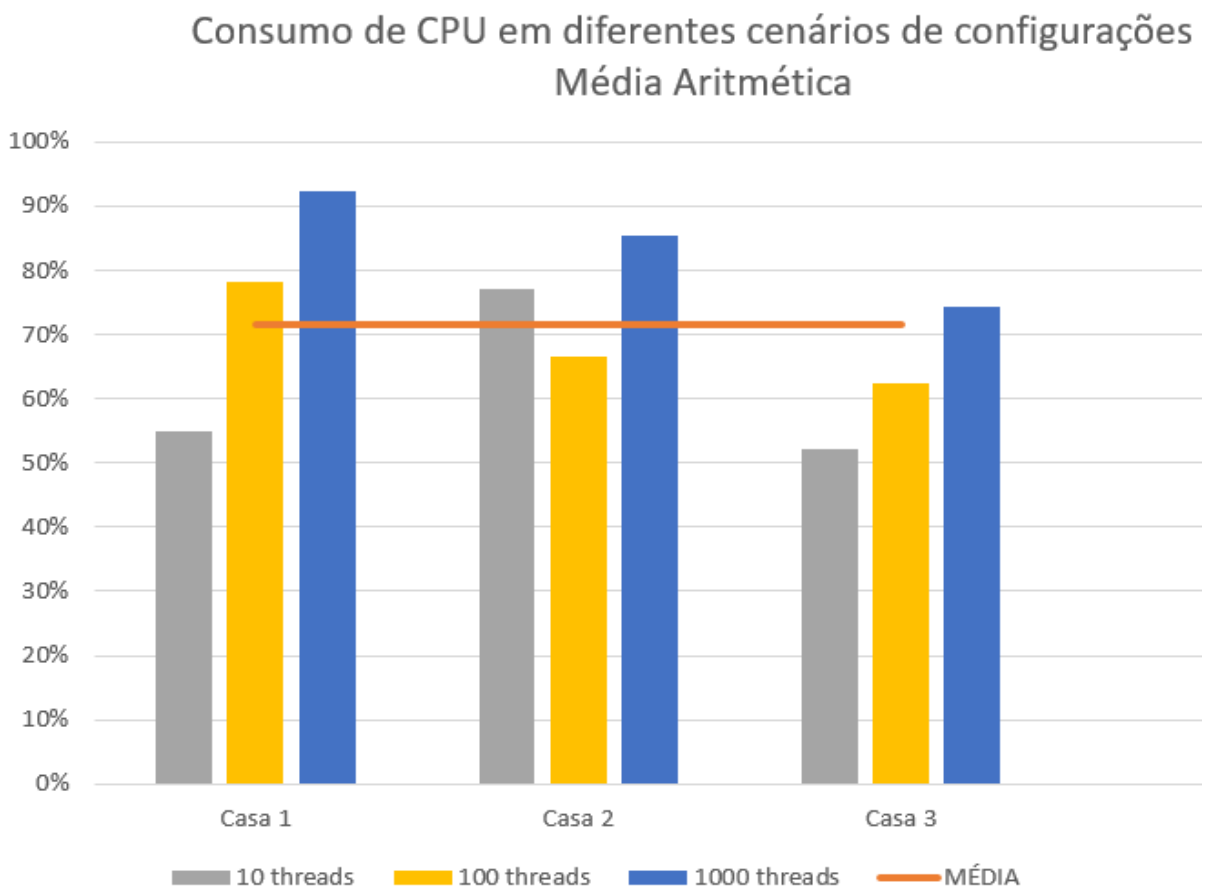
Fonte: elaborada pelo autor

Observa-se que houve uma discreta diminuição do consumo de memória no cenário de 1000 *threads*, em comparação com o cenário de 100 *threads*. Esse fato se deve ao fenômeno de as milhares de *threads* criadas possuírem baixa complexidade, reduzindo o consumo de memória em valores muito altos de *threads*, como observado

também no experimento realizado por Aversari, Kulesza e Moreira (2017). Além disso, é necessário destacar que existe uma limitação tanto de *hardware* quanto de rede, limitando o número de conexões simultâneas que podem ser gerenciadas. Consequentemente, o servidor diminui o desempenho ou até bloqueia algumas das conexões. Assim, esse fenômeno é observado quando mesmo aumentando o número de threads, o valor do consumo de memória e de CPU apresenta variações mínimas.

No quesito consumo de CPU, os testes na primeira residência demonstraram um consumo de CPU de 75,18%, enquanto na segunda demonstrou 76,4%, e na terceira residência resultou em um consumo de CPU de 62,96%. A média aritmética dos três cenários resultou em um consumo médio de CPU de 71,51%. No Gráfico 3 abaixo é possível verificar tais dados.

Gráfico 3 – Consumo de CPU em diferentes cenários de configurações

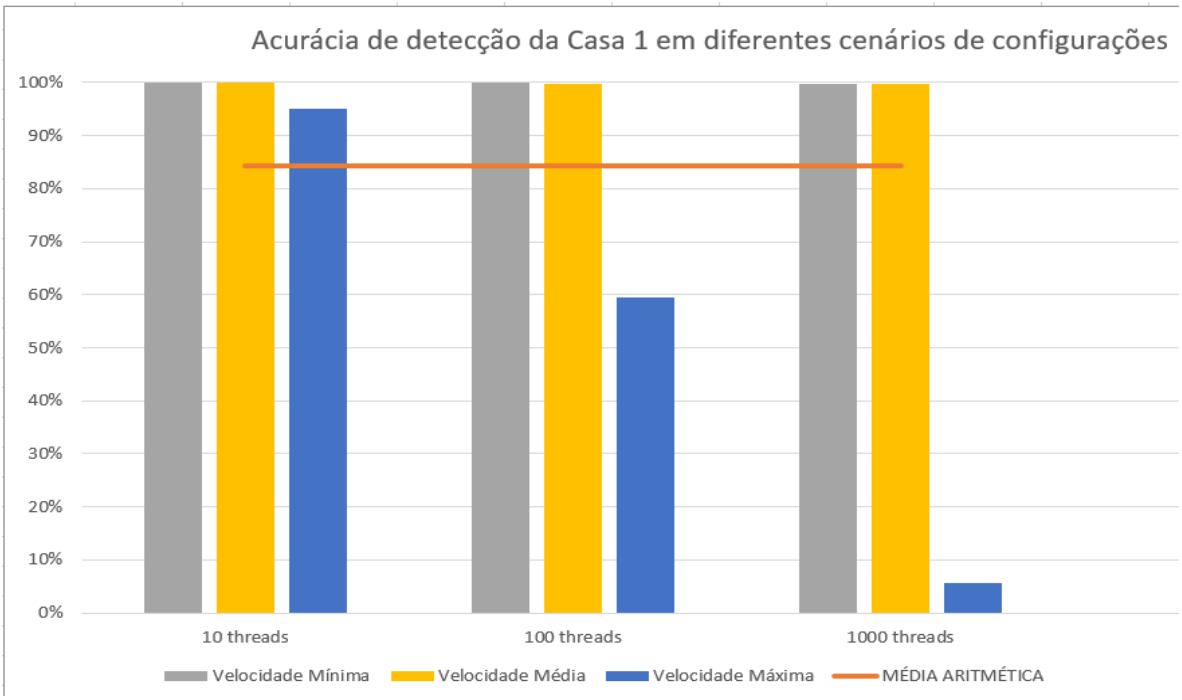


Fonte: elaborada pelo autor

Quanto à acurácia de detecção, sendo a porcentagem de ataques detectados em relação aos ataques disparados, a solução proposta alcançou uma média aritmética geral de 86,32% considerando todos os cenários analisados.

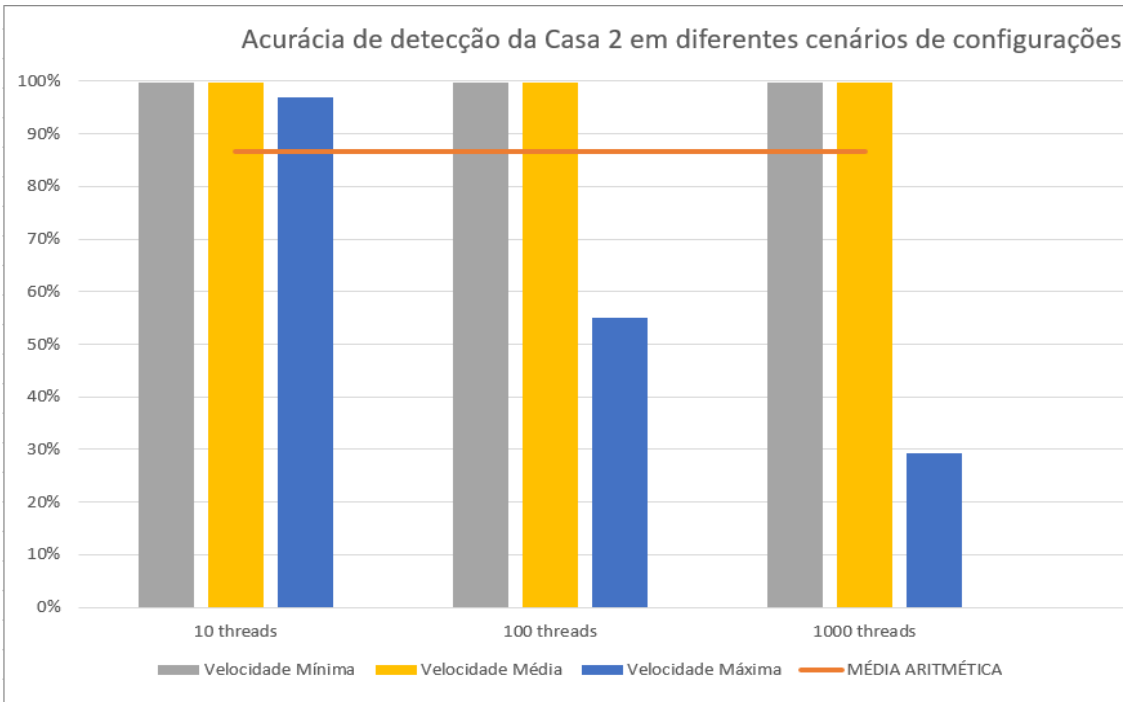
Conforme os Gráficos 4, 5 e 6, a primeira residência obteve uma média aritmética de acurácia de detecção de 84,36%, enquanto a segunda residência obteve 86,65% e a terceira uma média aritmética de 87,95%.

Gráfico 4 – Acurácia de detecção da Casa 1 em diferentes cenários de configurações



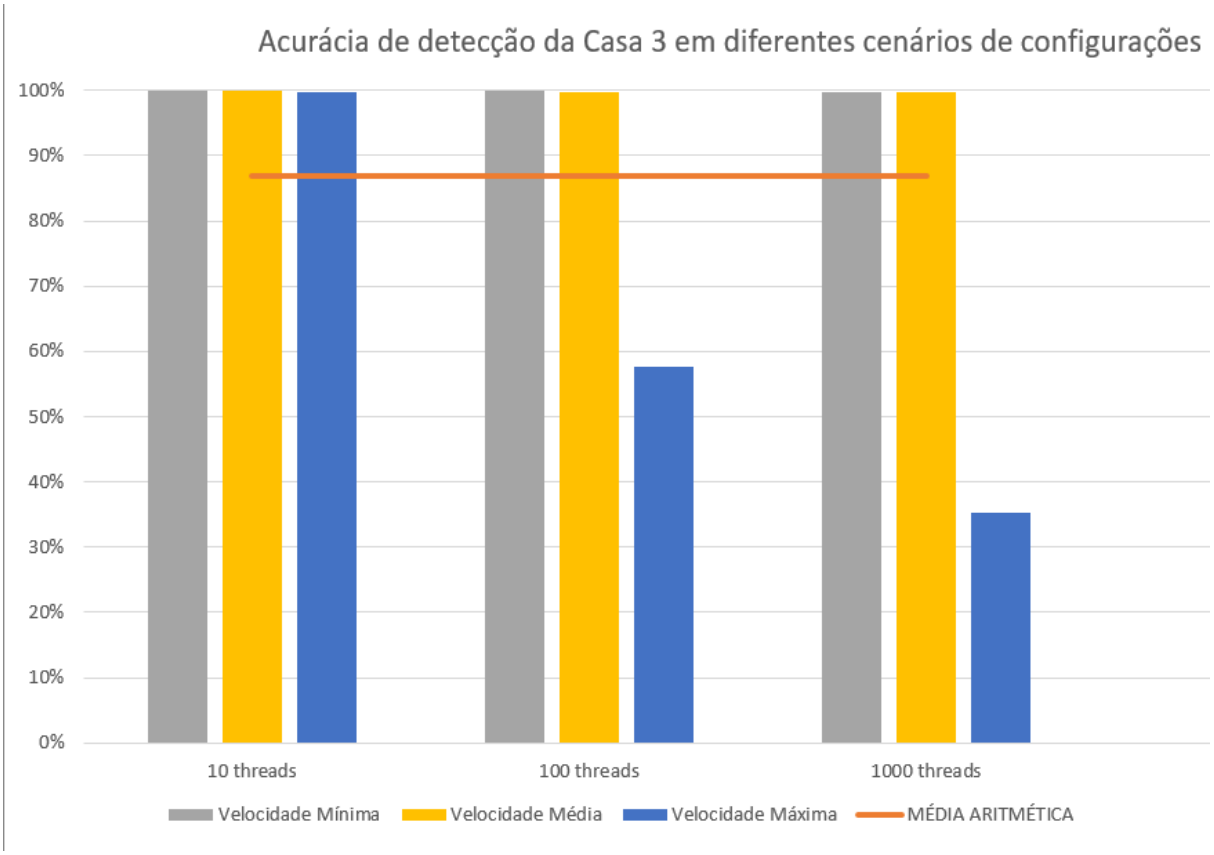
Fonte: elaborada pelo autor

Gráfico 5 – Acurácia de detecção da Casa 2 em diferentes cenários de configurações



Fonte: elaborada pelo autor

Gráfico 6 – Acurácia de detecção da Casa 3 em diferentes cenários de configurações



Fonte: elaborada pelo autor

Dessa forma, na primeira residência, obteve-se um consumo de memória de 69,63% e consumo de CPU de 75,18%, impactando na diminuição da acurácia de detecção para 84,36%. Na segunda residência, obteve-se um consumo de memória de 54,17% e consumo de CPU de 76,4%, impactando na acurácia da detecção em 86,65%. Na terceira residência, obteve-se um consumo de memória de 50,83% e consumo de CPU de 71,51%, impactando na maior acurácia da detecção, 87,95%.

Em uma análise quantitativa, foi utilizada a Análise de Variância (ANOVA), a qual consiste em um método estatístico para definir se a diferença entre grupos distintos possui ou não significância estatística. Dessa forma, foi analisado o impacto que os fatores exercem sobre a acurácia de detecção, sobre o consumo de memória e sobre o consumo de CPU.

Com tal método, o impacto é considerado significativo quando variável valor-*p* é menor do que 0,05. Assim, a Tabela 7 abaixo apresenta os resultados para o valor-*p* de acordo com os fatores testados.

Tabela 7 – Impacto dos fatores na acurácia de detecção e consumo de memória e CPU

	VARIÁVEL *p* DE ACORDO COM FATORES		
	NÚMERO DE DISPOSITIVOS	**NÚMERO DE THREADS**	**VELOCIDADE DO ATAQUE**
ACURÁCIA DE DETECÇÃO	0,962314	0,146742	0,001873
CONSUMO DE MEMÓRIA	0,01324	0,020193	0,310624
CONSUMO DE CPU	0,210522	0,012696	0,026397

Fonte: elaborada pelo autor

Nos resultados acima, observa-se que a acurácia de detecção recebeu impacto significante com a mudança da velocidade do ataque (p=0,001873), e impacto não significante com o aumento do número de dispositivos na casa (p=0,962314) e número de *threads* (p=0,146742). Já a análise do consumo de memória revelou que variar o número de dispositivos e de *threads* impacta significativamente no consumo de memória, enquanto a mudança da velocidade do ataque não possui significância estatística. Por fim, o consumo de CPU foi afetado pelo número de *threads* e pela velocidade de ataque, revelando que variar o número de dispositivos não tem impacto relevante sobre tal métrica.

Diante desses experimentos, verifica-se que a acurácia de detecção não é afetada significativamente pela quantidade de dispositivos presentes na casa, e nem pelo número de *threads* do ataque, mas sim pela mudança na velocidade do ataque DDoS. Apesar da presença de tais variações, a média aritmética de acurácia resultou em 86,32% de detecção nos cenários propostos, em 198 aferições, sendo 66 aferições em cada casa inteligente, confirmando a eficácia do Block Suricata Anti-DDoS em enviar alertas de acordo com os ataques disparados.

6 CONCLUSÕES E TRABALHOS FUTUROS

Neste Capítulo são apresentadas as conclusões e limitações deste trabalho, assim como sugestões para trabalhos futuros.

6.1 CONCLUSÕES

Os benefícios e facilidades trazidos pela Internet das Coisas constituem um avanço inegável nos campos da tecnologia, saúde, educação e indústria. Por outro lado, houve aumento do número de ataques DDoS mundialmente nos últimos anos, principalmente envolvendo dispositivos IoT, ilustrando a importância da discussão sobre a segurança, com foco nos dispositivos residenciais. Nesse contexto, as últimas pesquisas demonstram uma crescente vulnerabilidade nos dispositivos IoT residenciais, sendo necessária uma maior atenção da comunidade científica nesse aspecto.

Nesse sentido, esta dissertação apresentou um sistema operacional personalizado para detecção de ataques DDoS de inundação HTTP, na camada de aplicação da fonte, em redes IoT residenciais, chamado de Block Suricata Anti-DDoS. Essa solução de segurança foi desenvolvida para ser compatível com um *hardware* de baixo custo e para não diminuir os recursos computacionais escassos dos dispositivos IoT da rede doméstica sendo, portanto, escalável e aplicável no cenário residencial.

A avaliação de desempenho utilizando tráfego real em três casas inteligentes demonstraram a média aritmética de 86,32% de acurácia para detectar ataques DDoS de inundação HTTP, comprovando o benefício da solução de segurança proposta.

6.2 LIMITAÇÕES

Diante do contexto de segurança em redes residenciais, esta dissertação apresentou avanços, mas também limitações. Por exemplo, foram feitos testes em casas inteligentes que continham no máximo oito dispositivos IoT. Portanto, o desempenho do sistema pode sofrer alterações em residências onde haja um número maior de dispositivos.

Outra limitação consiste na ação do sistema em alertar o usuário, cabendo a ele a tarefa de desligar o equipamento envolvido no ataque, ou seja, a função de IPS de bloquear o tráfego malicioso automaticamente não está disponível nesta versão do sistema proposto, restando essa opção para os trabalhos futuros.

Além das limitações apresentadas acima, destaca-se uma limitação típica de ambientes residenciais. A casa não está totalmente protegida porque uma pessoa, desavisadamente, pode conectar um novo roteador em uma porta LAN do roteador que está sendo usado atualmente, passando a distribuir *internet* a partir do novo roteador, o que acarreta no tráfego não passar pela solução Block Suricata Anti-DDoS, colocando a casa em vulnerabilidade. Ressalte-se que esta é uma limitação proveniente do ambiente residencial, causada por uma falha humana, não estando relacionada intrinsicamente à solução apresentada neste trabalho, uma vez que geralmente dispositivos como roteadores são de fácil acesso, diferentemente de um ambiente corporativo, onde o servidor e outros equipamentos de rede frequentemente estão localizados em uma sala cofre, com acesso limitado de pessoas. Portanto, lembra-se que a segurança é multifatorial, estando envolvidos fatores humanos, além dos fatores tecnológicos.

Nesse sentido, é importante ressaltar que o trabalho possui limitações técnicas, e também humanas decorrentes do ambiente residencial, que podem ser abordadas com mais detalhes em novos trabalhos.

6.3 TRABALHOS FUTUROS

O presente trabalho possuiu como alvo alertar os usuários na ocorrência de ataques DDoS. Portanto, configura-se como um possível trabalho futuro a implementação de soluções capazes de não só alertar sobre ataques, mas efetivamente bloqueá-los e impedir a sua concretização em redes IoT residenciais, bem como envolver o bloqueio de ataques de outras categorias, além de DDoS.

Além disso, como forma de fortalecer a segurança, poderia ser criado um site repositório colaborativo, em que as pessoas que usassem a ferramenta Block Suricata Anti-DDoS ajudassem na coleta de IPs suspeitos. Dessa forma, as pessoas poderiam consultar quais são os principais ataques e de quais IPs eles são originados. Tais IPs poderiam ser listados de forma pública, de tal forma o próprio sistema Block Suricata Anti-DDoS envie automaticamente para o repositório como um IP suspeito, e

enviar para os outros dispositivos que participam do repositório, de tal forma que esse IP fique em um ambiente de revisão, e caso sejam recebidos mais alertas por segundo de outros equipamentos, esse IP entre na lista de IP bloqueado.

Diante do exposto, a Internet das Coisas traz evolução em diversos campos do conhecimento e também desafios a serem pesquisados e solucionados pela ciência.

REFERÊNCIAS

2001- UMA ODISSÉIA NO ESPAÇO; Direção: Stanley Kubrick. Produção: Stanley Kubrick Productions. Estados Unidos: Metro-Goldwyn-Mayer, 1968.

ABDUL-QAWY, Antar; TADISETTY, Srinivasulu. 2015. The Internet of Things (IoT): An Overview. **Journal of Engineering Research and Applications**, [s. l.], v. 5, n. 12, p.71-82, 2015. Disponível em: <https://www.researchgate.net/publication/323834996_The_Internet_of_Things_IoT_An_Overview>. Acesso em: 15 abr. 2023.

ALANI, M. An explainable efficient flow-based Industrial IoT intrusion detection system. **Computers and Electrical Engineering**, [s. l.], v. 108, [s.n.], 2023. Disponível em: < https://www.sciencedirect.com/science/article/abs/pii/S0045790623001568>. Acesso em: 05 abr. 2024.

ALBUQUERQUE, D. et. al. Proposta de um Modelo para Reduzir o Impacto de Ataques Maliciosos Ddos Syn-Flood. In: CONFERÊNCIA SUL EM MODELAGEM COMPUTACIONAL (MCSul), VIII., 2018, Rio Grande, Brasil. **Conferência** [...] Rio Grande: 2018, p. 1-13. Disponível em: < https://www.researchgate.net/profile/Marcos-Santos-85/publication/328886106_Proposta_de_um_modelo_para_reduzir_o_impacto_de_ataques_maliciosos_DDoS_SYN-FLOOD/links/5be97405a6fdcc3a8dd04341/Proposta-de-um-modelo-para-reduzir-o-impacto-de-ataques-maliciosos-DDoS-SYN-FLOOD.pdf >. Acesso em: 25 nov. 2023.

ALCANTARA, G. et al. Syrius: Synthesis of Rules for Intrusion Detectors. **IEEE Transactions on Reliability**, v. 71, n. 1, pp. 370-381, 2022. Disponível em: <https://ieeexplore-ieee-org.ez16.periodicos.capes.gov.br/document/9380789>. Acesso em: 25 jul. 2023.

ALGHAMDI, R.; BELLAICHE, M. A cascaded federated deep learning based framework for detecting wormhole attacks in IoT networks. **Computers & Security**, [s. l.], v. 125, [s.n.], 2023. Disponível em: < https://www.sciencedirect.com/science/article/abs/pii/S0167404822004060>. Acesso em: 05 abr. 2024.

ALTAN, G. SecureDeepNet-IoT: A deep learning application for invasion detection in industrial Internet of Things sensing systems. **Transactions on Emerging Telecommunications Technologies**, [s. l.], v. 32, n. 4, p. 2161-3915, 2021. Disponível em: <https://onlinelibrary.wiley.com/doi/epdf/10.1002/ett.4228>. Acesso em: 05 abr. 2024.

ARAÚJO, T. **Avaliando o desempenho dos sistemas de detecção de intrusão Snort e Suricata em ataques de negação de serviço**. Orientador: Dr. Fernando Menezes Matos. 2017. 73 f. Dissertação (Mestrado) - Centro de Informática, Universidade Federal da Paraíba, João Pessoa, 2017. Disponível em: <https://repositorio.ufpb.br/jspui/bitstream/123456789/12933/1/Arquivototal.pdf>. Acesso em: 25 jul. 2023.

AVERSARI, L.; KULESZA, R.; MOREIRA, J. Um Estudo Prático sobre o Potencial do Ataque Slowloris a partir de Dispositivos Móveis. In: SIMPÓSIO BRASILEIRO DE SEGURANÇA DA INFORMAÇÃO E DE SISTEMAS COMPUTACIONAIS, 17., 2017, Brasília. **Anais** [...]. Porto Alegre: Sociedade Brasileira de Computação, 2017. Disponível em: <https://sol.sbc.org.br/index.php/sbseg/article/view/19523>. Acesso em: 10 jun. 2024.

BANERJEE, U.; PATHAK, A.; CHANDRAKASAN, A. 2.3 An Energy-Efficient Configurable Lattice Cryptography Processor for the Quantum-Secure Internet of Things. **IEEE International Solid- State Circuits Conference**, San Francisco, USA, p. 46-48, 2019. Disponível em: <https://ieeexplore.ieee.org/document/8662528/>. Acesso em: 05 jul. 2023.

BIBI, Nighat *et al*. IoMT-Based Automated Detection and Classification of Leukemia Using Deep Learning. **Journal of Healthcare Engineering**, [s. l.], v. 2020, n. 10, p. 1-12, 2020. Disponível em: <https://pubmed.ncbi.nlm.nih.gov/33343851/>. Acesso em: 08 jun. 2023.

CHEN, Ling et. al. Design of Visual Seismic Monitoring and Warning System Based on Zabbix. **Journal of Physics: Conference Series**, Wuhan, China, vol. 2519, [s. n.], p. 1-12, 2023. Disponível em: <https://iopscience.iop.org/article/10.1088/1742-6596/2519/1/012008/meta>. Acesso em: 15 jul. 2023.

DE MELO, P.; MIANI, R.; ROSA, P. FamilyGuard: A Security Architecture for Anomaly Detection in Home Networks. **Sensors**, [s. l.], v. 22, n. 8, p. 2895-2919, 2022. Disponível em: <https://www.mdpi.com/1424-8220/22/8/2895>. Acesso em: 22 jun. 2023.

DJEHAICHE, Rania et al. Adaptive Control of IoT/M2M Devices in Smart Buildings Using Heterogeneous Wireless Networks. **IEEE Sensors Journal**, [s. l.], v. 23, n. 7, p. 7836-7849, 2023. Disponível em: <https://ieeexplore.ieee.org/document/10054551>. Acesso em: 05 jul. 2023.

FLEURY, M.; WERLANG, S. Pesquisa aplicada: reflexões sobre conceitos e abordagens metodológicas. **FGV SB**, [s. l.], p. 10-15, 2017. Disponível em: <https://bibliotecadigital.fgv.br/dspace/bitstream/handle/10438/18700/A_pesquisa_aplicada_conceito_e_abordagens_metodol%C3%B3gicas.pdf>. Acesso em: 01 abr. 2023.

FRUSTACI, M. et al. Evaluating Critical Security Issues of the IoT World: Present and Future Challenges. **IEEE Internet of Things Journal**, v. 5, n. 4, p. 2483-2495, 2018. Disponível em: <https://ieeexplore.ieee.org/document/8086136>. Acesso em: 05 jul. 2023.

GONDIM, J.; ALBUQUERQUE, R.; OROZCO, A. Mirror saturation in amplified reflection Distributed Denial of Service: A case of study using SNMP, SSDP, NTP and DNS protocols. **Future Generation Computer Systems**, [s. l.], v. 108, [s.n.], p. 68-81, 2020. Disponível em:

<https://www.sciencedirect.com/science/article/pii/S0167739X19322745>. Acesso em: 25 jul. 2023.

GUPTA, Neha; JINDAL, Vinita; BEDI, Punam. A Survey on Intrusion Detection and Prevention Systems. **SN Computer Science**, [s. l.], v. 4, n. 439, p. 1-28, 2023. Disponível em: <https://link.springer.com/article/10.1007/s42979-023-01926-7>. Acesso em: 07 jul. 2023.

HASSAN, S. et. al. Implementation of a Low-Cost IoT Enabled Surveillance Security System. In: INTERNATIONAL CONFERENCE ON APPLIED SYSTEM INNOVATION (ICASI), VII., 2021, Chiayi, Taiwan. **Conferência** [...] Chiayi: IEEE, 2021, p. 101-104. Disponível em: < https://ieeexplore.ieee.org/document/9568426>. Acesso em: 14 abr. 2023.

JURCUT, Anca; RANAWEERA, Pasika; XU, Lina. Introduction to IoT Security. **S. of Computer Science**, Dublin, Ireland, v. 2, [s. n.], p. 1-37, 2019. Disponível em: < https://www.researchgate.net/publication/336406296_Introduction_to_IoT_Security >. Acesso em: 14 abr. 2023.

KHOIROM, S. et. al. Comparative Analysis of Python and Java for Beginners. **International Research Journal of Engineering and Technology** (IRJET), [s. l.], v. 07, n. 08, p. 4384-4407, 2020. Disponível em: <https://https://d1wqtxts1xzle7.cloudfront.net/64732739/IRJET_V7I8755-libre.pdf?1603282304=&response-content-disposition=inline%3B+filename%3DIRJET_Comparative_Analysis_of_Python_and.pdf&Expires=1713556386&Signature=VVoQ7taRlh~g8Ar6Qal5HL7oApDDUitvbLt9dNNb9P~prCT4G2kP~J5uxuoNaT4MbwBRYlppoAo6eb2rxOtYRFeA6VBuHGFrLdXaKkCxX~U8WNSYpK24WcVq~ebXcDQslSwmlN6VcQvVuzgf2GwGEYnagYy~U2dat4piKy0T5mKtpAYqoBSoGsLNkXx5dAM12EpYmvpTEmQ-8C-Bc8lA0w0hLT4R88lbmIrSuLSyN9Bltz3eT1PP8GRb5bR2ZAYv3OVJF12-ei~senUB9kprtJSZ2cmKMdB5G2RCAwwrl65gQzzwiNUxGnK8i0qR1PHfrS9AE15w3b0Z5HbxlAfioA__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA>. Acesso em: 15 abr. 2024.

KORONIOTIS, N. ; MOUSTAFA, N.; SITNIKOVA, E. Forensics and Deep Learning Mechanisms for Botnets in Internet of Things: A Survey of Challenges and Solutions. **IEEE Access**, [s. l.], v. 7, n. 3, p. 61764-61785, 2019. Disponível em: <https://ieeexplore.ieee.org/document/8713986>. Acesso em: 05 jul. 2023.

LING, Zhen *et al*. Security Vulnerabilities of Internet of Things: A Case Study of the Smart Plug System. **IEEE Internet of Things Journal**, [s. l], p. 1-11, 2017. Disponível em: <https://www.researchgate.net/publication/317146761_Security_Vulnerabilities_of_Internet_of_Things_A_Case_Study_of_the_Smart_Plug_System>. Acesso em: 07 mar. 2023.

MANAVI, Taghizadeh. Defense mechanisms against Distributed Denial of Service attacks: A survey. **Computers & Electrical Engineering**, [s. l.], v. 72, [s. n.], pp. 26-38, 2018. Disponível em:

<https://www.sciencedirect.com/science/article/abs/pii/S0045790616307029>. Acesso em: 17 jul. 2023.

MANTOVANI, A. Pesquisa qualitativa e quantitativa: definições e conceitos. **USPSI ,** [s. l.], v. 3, p. 1-20, 2017. Disponível em: <https://edisciplinas.usp.br/pluginfile.php/2945625/mod_resource/content/1/pesquisa%20quanti_quali.pdf>. Acesso em: 01 abr. 2023.

MAZHAR, T. et. al. Analysis of IoT Security Challenges and Its Solutions Using Artificial Intelligence. **Brain Sciences,** [s. l.], v. 13, n. 4, p. 683-713, 2023. Disponível em: <https://www.mdpi.com/2076-3425/13/4/683>. Acesso em: 05 jul. 2023.

MEYER, Bruno; GEMMER, Davi; ANDRADE, Allex; MELLO, Emerson; NOGUEIRA, Michele; WANGHAM, Michelle. Criação de Redes Virtuais no MENTORED Testbed: Uma Análise Experimental. **Workshop de Testbeds, Computer Science**, [s. l.], p. 24-35, 2022. Disponível em: <https://www.researchgate.net/publication/362380020_Criacao_de_Redes_Virtuais_no_MENTORED_Testbed_Uma_Analise_Experimental/>. Acesso em: 7 mar. 2023.

NASIR, M.; ARSHAD, J; KHAN, M. Collaborative device-level botnet detection for internet of things. **Computers & Security**, [s. l.], v. 129, p. 1-20, 2023. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167404823000822>. Acesso em: 16 jun. 2023.

NATH, Renya; NATH, Hiran. Critical analysis of the layered and systematic approaches for understanding IoT security threats and challenges. **Computers and Electrical Engineering**, [s. l.], v. 100, p. 12-25, 2022. Disponível em: < https://www.sciencedirect.com/science/article/abs/pii/S0045790622002658>. Acesso em: 14 abr. 2023.

NAWROCKI, Sok et al. A Data-driven View on Methods to Detect Reflective Amplification DDoS Attacks Using Honeypots. In: PROCEEDINGS OF EURO S&P, XXIII., 2023, Piscataway, USA. **Conferência** [...] Piscataway: IEEE, 2023, p. 1-16. Disponível em: <https://arxiv.org/abs/2302.04614>. Acesso em: 17 jul. 2023.

NAZARI, M.; DAHMARDEH, Z.; ALIABADY, S. A Novel Approach of Botnets Detection Based on Analyzing Dynamical Network Traffic Behavior. **SN Computer Science**, [s. l.], v. 2, n. 247, pp. 1-11, 2021. Disponível em: <https://link.springer.com/article/10.1007/s42979-021-00634-4> . Acesso em: 15 jul. 2023.

NEVES, G. CheckBERT: Um Sistema de Identificação de Informações Falsas Através da
Comparação Semântica de Sentenças com Notícias de Portais Jornalísticos de Língua Inglesa Utilizando Google BERT. Trabalho de Conclusão de Curso em Sistemas de Informação –Universidade Federal de Santa Catarina. Florianópolis, p. 141, 2023. Disponível em: <https://repositorio.ufsc.br/bitstream/handle/123456789/248713/TCC-Gabriel-Medeiros-das-Neves-final.pdf?sequence=1&isAllowed=y>. Acesso em: 02 dez. 2023.

NGUYEN, Ngoc; LE, Hoang; DO, Dinh. Study to analyse, compare and evaluate the performance of next general firewalls: Case of Palo Alto and Fortigate Firewall. DOAJ Directory of Open Access Journals, **Vinh University Journal of Science**, v.51, n. 2A, 2022. Disponível em: <https://vujs.vn/api/view.aspx?oldId=NT08-2022>. Acesso em: 15 jul. 2023.

NOZOMI NETWORKS. 2022. Relatório de segurança de OT/IoT: análise aprofundada do cenário de ameaças em ICS. Disponível em: <https://www.nozominetworks.com/downloads/PT/Nozomi-Networks-OT-IoT-Security-Report-ES-2022-2H-PT.pdf>. Acesso em: 31 mar. 2023.

NOZOMI NETWORKS. 2023. OT/IOT Security Report Executive Summary:Unpacking the Threat Landscape
With Unique Telemetry Insight. Disponível em: < https://assets-global.website-files.com/645a4534705010e2cb244f50/64c953f5fbcde39471610fa3_EXECUTIVE-SUMMARY-Nozomi-Networks-OT-IoT-Security-Report-ES-August-2023.pdf>. Acesso em: 29 mar. 2024.

PARK, W.; AHN, S. Performance Comparison and Detection Analysis in Snort and Suricata Environment. **Wireless Pers Commun**, [s. l.], v. 94, p. 241–252, 2017. Disponível em: <https://link.springer.com/article/10.1007/s11277-016-3209-9>. Acesso em: 08 jul. 2023.

PATACA, C. C. A Internet das Coisas: Tipologias, Protocolos e Aplicações. **The Law, State and Telecommunications Review**, Brasília, v. 13, n. 2, p. 198-220, 2021. Disponível em: <https://periodicos.unb.br/index.php/RDET/issue/view/2333>. Acesso em: 14 abr. 2023.

RAFIQUE, Wajid et al. Complementing IoT Services Through Software Defined Networking and Edge Computing: A Comprehensive Survey. **IEEE Communications Surveys & Tutorials**, [s. l.], p. 1-46, 2020. Disponível em: <https://www.researchgate.net/publication/341657809_Complementing_IoT_Services_Through_Software_Defined_Networking_and_Edge_Computing_A_Comprehensive_Survey>. Acesso em: 05 jul. 2023.

SAARIKKO, Ted; WESTERGREN, Ulrika; BLOMQUIST, Tomas. The Internet of Things: Are you ready for what's coming? **Business Horizons**, Indiana, v. 60, n. 5, p. 667-676, 2017. Disponível em: < https://www.sciencedirect.com/science/article/pii/S000768131730068X/>. Acesso em: 25 mar. 2023.

SAFEATLAST. 25 Insightful IoT Statistics to Make Your Life Easier. SAFEATLAST, 2022. Disponível em: < https://safeatlast.co/blog/iot-statistics/>. Acesso em: 14 abr. 2023.

SANTOS *et al*. A flow-based intrusion detection framework for internet of things networks. **Cluster Comput**, [s. l.], v. 26, p. 37–57, 2023. Disponível em: <https://link.springer.com/article/10.1007/s10586-021-03238-y>. Acesso em: 17 jun. 2023.

SIDDHANT *et al.* 2018. Wireless Technology: Bluetooth. **International Journal for Scientific Research & Development**, [s. l.], v. 6, n. 3, p. 1-4, 2018. Disponível em: <https://www.researchgate.net/publication/362491660_Wireless_Technology_Bluetooth>. Acesso em: 30 abr. 2023.

SILVA, R. Análise comparativa de soluções de Open Source para cibersegurança por monitorização de tráfico de rede. **IP Beja**, Beja, Portugal, v. 2020, [s. n.], 2020. Disponível em: <https://repositorio.ipbeja.pt/handle/20.500.12207/5539>. Acesso em: 15 jul. 2023.

SILVA, W.; MEDEIROS, R.; MARTINS, R. Análise e Gerenciamento de Redes usando uma Metodologia Proativa com Zabbix. **HOLOS**, [s. l.], v. 8, [s. n.], p. 277-289, 2015. Disponível em: <https://www2.ifrn.edu.br/ojs/index.php/HOLOS/article/view/2441/1328>. Acesso em: 08 jul. 2023.

SIVASANKARI, N; KAMALAKKANNAN, S. Detection and prevention of man-in-the-middle attack in iot network using regression modeling. **Advances in Engineering Software**, [s. l.], v. 169, p. 103-126, 2022. Disponível em: <https://www.sciencedirect.com/science/article/abs/pii/S0965997822000370/>. Acesso em: 30 abr. 2023.

SOBH, Tarek. Wired and wireless intrusion detection system: Classifications, good characteristics and state-of-the-art. **Computer Standards & Interfaces**, [s. l.], v. 28, n. 6, p. 670-694, 2016. Disponível em: <https://www.sciencedirect.com/science/article/abs/pii/S092054890500098X>. Acesso em: 08 jul. 2023.

SOUZA, Allan et. al. Elasticsearch, Kibana e Logstash (Pilha ELK): transformando dados brutos em informações poderosas de forma rápida e eficiente. UNICAMP: Cinfotec, Campinas, Brasil, ed. 8, [s. n.], p. 1-2, 2019. Disponível em: <https://www.cinfotec.unicamp.br/wp-content/uploads/sites/25/2023/08/Transformando_dados.pdf>. Acesso em: 02 dez. 2023.

STADDON, E.; & LOSCRI, V; MITTON, N. Attack Categorisation for IoT Applications in Critical Infrastructures, a Survey. **Applied Sciences**, [s. l.], v. 11, n. 16, p. 1-39, 2021. Disponível em: < https://www.researchgate.net/publication/353723022_Attack_Categorisation_for_IoT_Applications_in_Critical_Infrastructures_a_Survey>. Acesso em: 30 abr. 2023.

SWANDI, C. et. al. Middleware Development to Connect Telegram Messenger and Instant Messenger for the Elderly. **IOP Conference Series: Materials Science and Engineering**, [s. l.], v. 1077, p. 1-8, 2021. Disponível em: <https://iopscience.iop.org/article/10.1088/1757-899X/1077/1/012007/pdf>. Acesso em: 15 abr. 2024.

TANWEER, Alam. A Reliable Communication Framework and Its Use in Internet of Things (IoT). **International Journal of Scientific Research in Computer Science,**

Engineering and Information Technology, [s. l.], v. 3, n. 5, p. 449-456, 2018. Disponível em: < https://www.researchgate.net/publication/325645304_A_Reliable_Communication_Framework_and_Its_Use_in_Internet_of_Things_IoT >. Acesso em: 05 jul. 2023.

TREDINNICK, Luke; LAYBATS, Claire. Being smart with smart technology. **Business Information Review**, [s. l.], v. 35, n. 2, p. 54-55, 2018. Disponível em: <https://www.researchgate.net/publication/325560392_Being_smart_with_smart_technology>. Acesso em: 05 jul. 2023.

Ubuntu lifecycle and release cadence. **Ubuntu**, 2024. Disponível em: <https://ubuntu.com/about/release-cycle>. Acesso em: 19 fev. 2024.

ULLAH, I.; MAHMOUD, Q. 2020. A Two-Level Flow-Based Anomalous Activity Detection System for IoT Networks. **Electronics**, [s. l.], v. 9, n. 3, 2020. Disponível em: <https://www.researchgate.net/pu3blication/340114433_A_Two-Level_Flow-Based_Anomalous_Activity_Detection_System_for_IoT_Networks>. Acesso em: 22 jun. 2023.

VALADARES, Dalton et al. (2022). Segurança em Cenários de Internet das Coisas em Redes 5G: Desafios e Recomendações. In: SIMPÓSIO BRASILEIRO DE TELECOMUNICAÇÕES E PROCESSAMENTO DE SINAIS (SBRT 2022), XL., 2022, Santa Rita do Sapucaí, MG, Brasil. **Conferência** [...] Santa Rita do Sapucaí: ResearchGate, 2022, p. 1- 6. Disponível em: <https://www.researchgate.net/publication/362559101_Seguranca_em_Cenarios_de_Internet_das_Coisas_em_Redes_5G_Desafios_e_Recomendacoes>. Acesso em: 30 mar. 2024.

WALEED, Abdul; JAMALI, Fareed; MASOOD, Ammar. Which open-source IDS? Snort, Suricata or Zeek. **Computer Networks**, [s. l.], v. 213, p. 1-11, 2022. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1389128622002420>. Acesso em: 08 jul. 2023.

WOEI-KAE, Chen *et al*. 2018. ICAT: An IoT Device Compatibility Testing Tool. In: ASIA-PACIFIC SOFTWARE ENGINEERING CONFERENCE (APSEC), XXV., 2018, Nara, Japão. **Conferência** [...] Nara: IEEE, 2018, p. 668-672. Disponível em: <https://ieeexplore.ieee.org/document/8719553>. Acesso em: 8 mar. 2023.

YADAV, Er Pooja; YADAV, Hemant. IoT: Challenges and Issues in Indian Perspective. In: IEEE INTERNATIONAL CONFERENCE ON INTERNET OF THINGS: SMART INNOVATION AND USAGES (IoT-SIU 2018), III., 2018, Bhimtal, India. **Conferência** [...] Bhimtal: ResearchGate, 2018, p. 1- 6. Disponível em: <https://www.researchgate.net/publication/326929378_IoT_Challenges_and_Issues_in_Indian_Perspective>. Acesso em: 07 mar. 2023.

ZAHID, H.; HINA, S; HAYAT, M.; SHAH, G. Agentless Approach for Security Information and Event Management in Industrial IoT. **Electronics**, [s. l.], v. 12, p. 1831, 2023. Disponível em: < https://www.mdpi.com/2079-9292/12/8/1831>. Acesso em: 09 jul. 2023.

ZIKMUND, William. **Business Research Methods**. [S. l.]: South-Western College Pub, 8th Edition, 2009. Disponível em: <https://www.academia.edu/33978482/Business_Research_Method_Zikmund_8th_edition_pdf>. Acesso em: 31 mar. 2023.

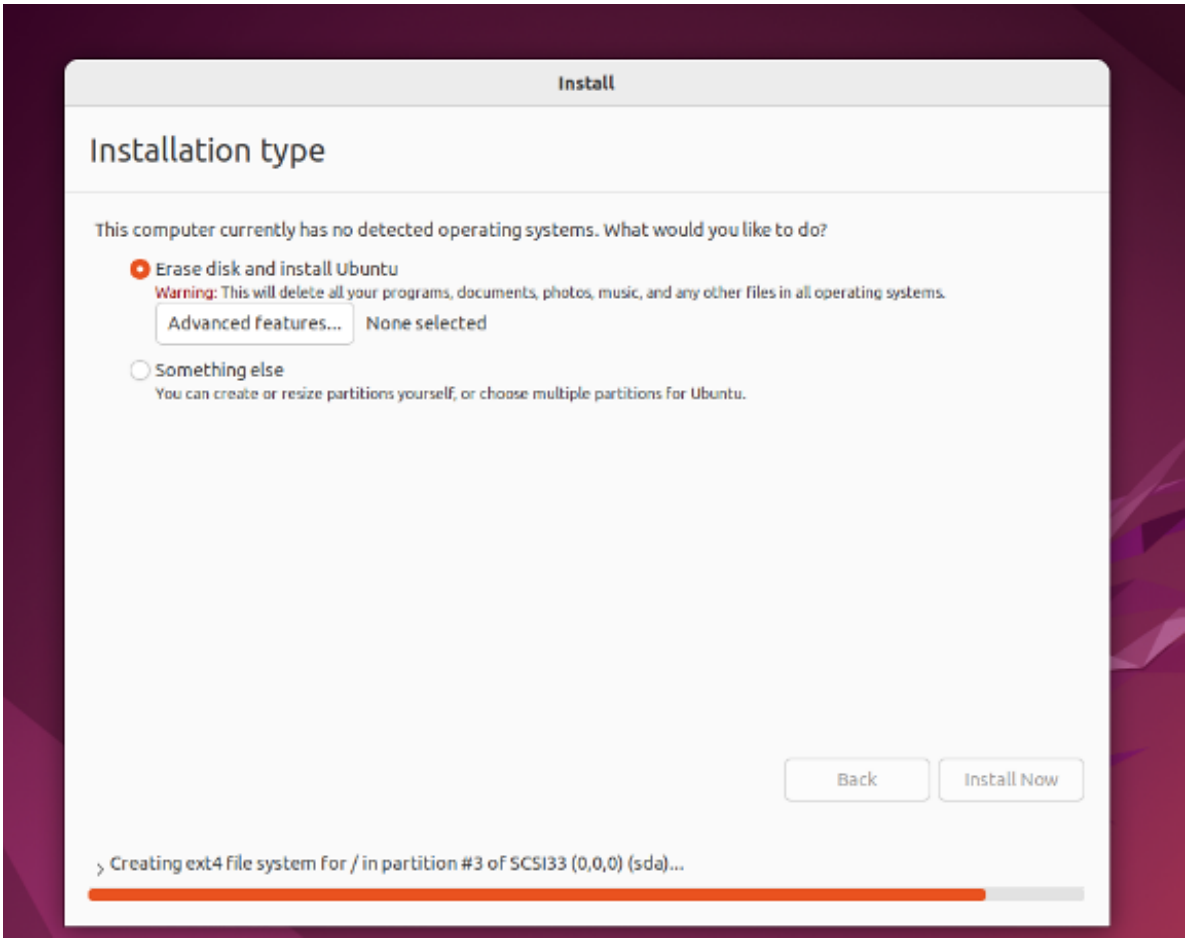
ZHANG, P. et. al. A Blockchain-Based Authentication Scheme and Secure Architecture for IoT-Enabled Maritime Transportation Systems. **IEEE Transactions on Intelligent Transportation Systems**, [s. l.], v. 24, no. 2, p. 2322-2331, 2023. Disponível em: <https://ieeexplore.ieee.org/document/9745459>. Acesso em: 05 abr. 2024.

APÊNDICE A – CÓDIGOS DA IMPLEMENTAÇÃO

Este Apêndice tem o objetivo de detalhar os procedimentos de instalação de *softwares*, bem como demonstrar os códigos implementação utilizados no presente trabalho.

A Figura 24 ilustrada abaixo demonstra a instalação do ambiente de desenvolvimento, sendo instalado no VMware vSphere Hypervisor 7.0 o sistema operacional Ubuntu versão 22.04 LTS.

Figura 24 – Instalação do sistema operacional Ubuntu



Fonte: elaborada pelo autor

Para a instalação do IDPS Suricata, é necessário antes adicionar o repositório do Suricata ao Ubuntu, etapa que se encontra representada na Figura 25 abaixo.

Figura 25 – Repositório do sistema Suricata

```
root@machine-suricata:/home/casa# sudo add-apt-repository ppa: oisf/suricata-stable
Repository: 'deb http://ppa.launchpadcontent.net/oisf/suricata-stable/ubuntu/jammy main'
Description:
Suricata IDS/IPS/NSM stable packages
http://suricata.io/
https://oisf.net/

Suricata IDS/IPS/NSM - Suricata is a high performance intrusion Detection and Prevention System and
Network Security Monitoring engine.

Open source and owned by a community run non-profit foundation, the Open Information Security
Foundation (OISF). Suricata is developed by the OISF, its supporting vendors and the community.

This Engine supports:

-Multi-Threading - provides for extremely fast and flexible operation on multicore systems.
-Multi tenancy - Per vlan/Per interface
-Uses Rust for most protocol detection/parsing
-TLS/SSL certificate matching/logging
-JA3 TLS client fingerprinting
-JA3S TLS server fingerprinting
-IEEE 802.1ad (QinQ) and IEEE 802.1Q (VLAN) support
-VXLAN support
-All JSON output/logging capability
-IDS runmode
-IPS runmode
-IDPS runmode
-NSM runmode
-eBPPF/XDP
-Automatic Protocol Detection and logging - IPv4/6, TCP, UDP, ICMP, HTTP, SMTP, TLS, SSH, FTP, SMB,
DNS, NFS, TFTP, KRB5, DHCP, IKEv2, SNMP, SIP, RDP
-SCADA automatic protocol detection - ENIP/DNP3/MODBUS
-File Extraction HTTP/SMTP/FTP/NFS/SMB - over 4000 file types recognized and extracted from live
traffic.
-File MD5/SHA1/SHA256 matching/logging
-Gzip Descompression
-Fast IP Matching
-Datasets Matching-Rustlang enabled protocol detection
-Lua scripting

and more great features -
https://suricata.io/features/all-features/
More info: https://launchpad.net/~oisf/+archive/ubuntu/suricata-stable
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
```

Fonte: elaborada pelo autor

Após a adição do repositório, pode-se prosseguir à instalação do Suricata. A Figura 26 a seguir ilustra o comando de instalação na primeira linha. Após a execução desse comando, haverá a adição de pacotes e árvore de dependências do Suricata.

Figura 26 – Instalação do sistema Suricata

```
root@machine-suricata:/home/casa# sudo apt install suricata
```

Fonte: elaborada pelo autor

Quanto à configuração das interfaces de rede, são utilizadas duas interfaces de rede, ens33 e ens37. A interface ens33 é a interface responsável por receber a *Internet*, sendo considerada, portanto, a interface de rede principal do servidor doméstico.

Já a interface ens37 é responsável por realizar a distribuição de *Internet* para o restante da residência e dos dispositivos IoTs. Abaixo segue a Figura 27 especificando a interface ens33 como padrão no Suricata, ou seja, definindo que o tráfego a ser inspecionado será proveniente dessa interface.

Figura 27 – Configuração da Interface de Rede

```
root@machine-suricata:/home/casa# ip -p -j route show default

[  {
        "dst": "default",
        "gateway": "10.0.0.1",
        "dev": "ens33",
        "protocol": "dhcp",
        "metric": 100,
        "flags": [ ]
    } ]
```

Fonte: elaborada pelo autor

A próxima configuração se refere à variável HOME_NET, que aponta para o endereço da sub-rede interna, a qual estará protegida. Além dessa configuração, existe também a variável EXTERNAL_NET, a qual aponta para a rede externa, que está desprotegida.

A Figura 28 a seguir ilustra a configuração das sub-redes no Suricata, em que a variável EXTERNAL_NET foi configurada como !$HOME_NET, significando que o Suricata considera que qualquer valor que seja diferente da sub-rede HOME_NET é considerado como rede externa, ou seja, EXTERNAL_NET, já que o sinal de exclamação significa ‘diferente’.

Figura 28 – Configuração das Sub-redes no Suricata

```
/etc/suricata/suricata.yaml

vars:
    # more specific is better for alert accuracy and performance
    address-groups:
    HOME_NET: "[10.42.0.1/24]"
    #HOME_NET: "[192.168.0.0/16]"
    #HOME_NET: "[10.0.0.0/8]"
    #HOME_NET: "[172.16.0.0/12]"
    #HOME_NET: "any"

    EXTERNAL_NET: "!$HOME_NET"
    #EXTERNAL_NET: "any"
```

Fonte: elaborada pelo autor

A próxima configuração de rede importante é a interface de rede ens37. Está localizada no *af-packet*, dentro do arquivo *suricata.yaml*. Já a interface de rede ens33 se refere à rede externa, ou seja, a qual deve ser monitorada neste trabalho. A Figura 29 abaixo demonstra a configuração dessa interface no *suricata.yaml*.

Figura 29 – Configuração da Interface de Rede no *suricata.yaml*

```
/etc/suricata/suricata.yaml

# Linux high speed capture support
af-packet:
  - interface: ens37
    # Number of receive threads. "auto" uses the number of cores
    #threads: auto
    # Default clusterid. AF_PACKET will load balance packets based on flow.
```

Fonte: elaborada pelo autor

Além disso, no arquivo *suricata.yaml* existem outras alterações relevantes. Por exemplo, é indicado onde estão os arquivos de regras, no caso, *suricata.rules* e *detectando-ddos.rules* no caminho */var/lib/suricata/rules*. Todas essas indicações são necessárias para que o Suricata busque os arquivos de regras no caminho informado pelo usuário, conforme Figura 30 abaixo.

Figura 30 – Arquivo de regras no Suricata

```
/etc/suricata/suricata.yaml

##
## Configure Suricata to load Suricata-Update managed rules.
##

default-rule-path: /var/lib/suricata/rules

rule-files:
   - suricata.rules
   # - Custom Test rules
   - detectando-ddos.rules
```

Fonte: elaborada pelo autor

Após informar ao Suricata o caminho do arquivo de regras, é necessário escrever tal arquivo. No diretório */var/lib/suricata/rules/* no arquivo *detectando-ddos.rules* foi adicionada a regra de alerta visualizada na Figura 31 abaixo, direcionada a ataques DDoS. Cada posição na regra tem um significado, por exemplo, primeira informação “alert” corresponde à ação que se deve tomar, no caso do IDS, alertar e mostrar ao SYSADMIN do Suricata quando um determinado evento está ocorrendo.

A segunda posição se refere ao protocolo a ser inspecionado, nesse caso, protocolo TCP. Após o protocolo, é determinada a fonte do tráfego e a porta de origem, as quais foram definidas como *any* e *any*, respectivamente, ou seja, a regra analisará o tráfego de qualquer IP fonte, proveniente de qualquer porta. A seta direcional indica a direção do tráfego, com origem em qualquer fonte, e com destino à HOME_NET, porta 80.

Figura 31 – Adicionando Regras no Suricata

```
detectando-ddos.rules

alert tcp any any -> $HOME_NET 80 (msg: "Possivel attack  DDoS"; flags: S; flow: from_client;
threshold: type both, track by_dst, count 200, seconds 1; sid:99126611; rev:1;)
```

Fonte: elaborada pelo autor

Seguindo-se a explicação da regra, a opção *msg* se refere a uma informação textual que será exibida nos *logs* quando o alerta ocorrer, sendo definida neste

trabalho como *possível ataque DDoS*. Já a opção *flags S* significa que o fluxo analisado será TCP SYN, referindo-se à primeira etapa do *Three-way Handshake*, em que o cliente envia um pacote com a flag SYN ativa, significando uma solicitação de sincronização com o servidor.

Já a alteração das configurações de *firewall* pode ser executada no arquivo *sudo nano /etc/ufw/before.rules*, representado abaixo na Figura 32. Pode-se observar nas duas últimas linhas as regras indicando que todo o tráfego que entra e sai deve ser redirecionado ao firewall UFW.

Figura 32 – Arquivo *before.rules* com *firewall* UFW ativo

```
1  # rules.before
2  #
3  # Rules that should be run before the ufw command line added rules. Custom
4  # rules should be added to one of these chains:
5  #   ufw-before-input
6  #   ufw-before-output
7  #   ufw-before-forward
8  #
9
10 # Don't delete these required lines, otherwise there will be errors
11 *filter
12 :ufw-before-input - [0:0]
13 :ufw-before-output - [0:0]
14 :ufw-before-forward - [0:0]
15 :ufw-not-local - [0:0]
16 # End required lines
17
18 # allow all on loopback
19 -A ufw-before-input -i lo -j ACCEPT
20 -A ufw-before-output -o lo -j ACCEPT
```

Fonte: elaborada pelo autor

Tais regras são modificadas a fim de que todo tráfego do *firewall* seja enviado para o NFQUEUE do Suricata, o qual consiste um *iptables target* utilizado em *softwares* IDPS fazendo com que todo o tráfego que chegar e sair dessa rede seja aceito e redirecionado para o Suricata.

A Figura 33 ilustra tal mudança de regra, sendo a penúltima linha responsável por redirecionar todo o tráfego de entrada para o NFQUEUE do Suricata, e a última linha responsável por redirecionar todo o tráfego de saída para o NFQUEUE

do Suricata. Portanto, na prática, tudo que entra e sai do servidor é encaminhado para o Suricata analisar.

Figura 33 – Arquivo *before.rules* com NFQUEUE ativo

```
# rules.before
#
# Rules that should be run before the ufw command line added rules. Custom
# rules should be added to one of these chains:
#   ufw-before-input
#   ufw-before-output
#   ufw-before-forward
#

# Don't delete these required lines, otherwise there will be errors
*filter
:ufw-before-input - [0:0]
:ufw-before-output - [0:0]
:ufw-before-forward - [0:0]
:ufw-not-local - [0:0]
# End required lines

###Suricata NFQUEUE
- I INPUT -j NFQUEUE
- I OUTPUT -j NFQUEUE
```

Fonte: elaborada pelo autor

Quanto à configuração do *software* Wazuh no Ubuntu deve ser adicionada a *GNU Privacy Guard* (GPG). A GPG key é uma chave pública de criptografia, sendo necessária às atividades do Wazuh de encriptar e desencriptar pacotes. Na Figura 34 abaixo encontra-se o *download* e importação da chave GPG, bem como aplicação da permissão para acessar e modificar o arquivo.

Figura 34 – GPG *key* no Ubuntu

```
curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | gpg --no-default-keyring
--keyring gnupg-ring:/usr/share/keyrings/wazuh.gpg --import && chmod 644
/usr/share/keyrings/wazuh.gpg
```

Fonte: elaborada pelo autor

Com a adição da chave GPG, prossegue-se à etapa de adição do repositório do Wazuh no Ubuntu, etapa que pode ser observada na Figura 35 a seguir.

Figura 35 – Repositório do *software* Wazuh

```
root@SRV007:/home/casa# echo "deb [signed-by=/usr/share/keyrings/wazuh.gpg] https://packages.wazuh.com/4.x/apt/ stable main" | tee -a /etc/apt/sources.list.d/wazuh.list

deb [signed-by=/usr/share/keyrings/wazuh.gpg] https://packages.wazuh.com/4.x/apt/ stable main

root@SRV007:/home/casa#
```

Fonte: elaborada pelo autor

Neste momento, pode-se prosseguir à instalação propriamente dita, representada na Figura 36 abaixo.

Figura 36 – Instalação do *software* Wazuh

```
apt-get -y install wazuh-manager
```

Fonte: elaborada pelo autor

A próxima configuração se refere à permissão para que o *software* Wazuh possa efetuar a leitura do arquivo de *log* proveniente do Suricata. Na própria interface gráfica do Wazuh, pode-se adicionar na penúltima linha do arquivo as configurações presentes da linha 405 a 408 da Figura 37 abaixo.

Figura 37 – Integração do *software* Wazuh com os *logs* do Suricata

```
396     <log_format>syslog</log_format>
397     <location>/var/log/dpkg.log</location>
398   </localfile>
399
400   <localfile>
401     <log_format>syslog</log_format>
402     <location>/var/log/kern.log</location>
403   </localfile>
404
405   <localfile>
406     <log_format>json</log_format>
407     <location>/var/log/suricata/eve.json</location>
408   </localfile>
409
410 </ossec_config>
411
```

Fonte: elaborada pelo autor

Quanto à integração entre o Python e o Telegram, a Figura 38 abaixo ilustra a instalação do *telepot,* um *framework* capaz de integrar uma *Application Programming Interface* (API) criada pelo usuário ao *bot* do Telegram. Essa etapa é necessária para que se possa enviar alertas via aplicativo ao usuário.

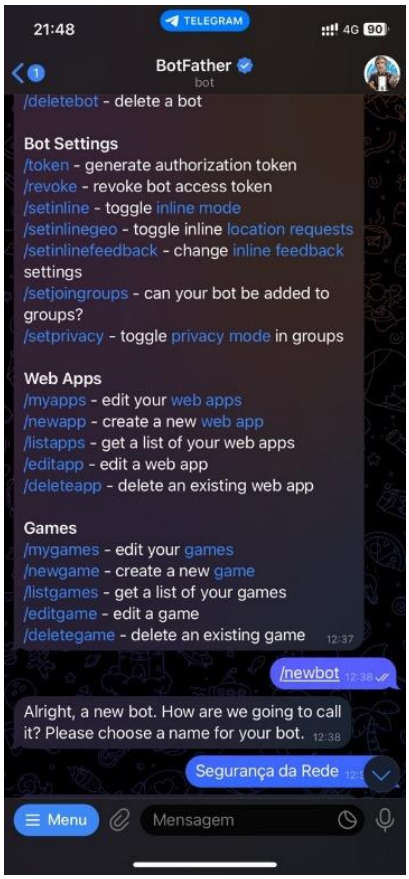
Figura 38 – Instalação do Telepot no Python

```
1 root@SRV007:/home/casa# pip3 install telepot
```

Fonte: elaborada pelo autor

Na etapa de criação do *bot* para o Telegram, é necessário interagir com o BotFather, situação representada na Figura 39 a seguir, em que é possível escolher a criação de um novo *bot,* bem como o nome do *bot,* definido como Segurança da Rede.

Figura 39 – Interação com o BotFather



Fonte: elaborada pelo autor

A Figura 40 abaixo ilustra o procedimento de entrega dos dados de segurança para a API do Python, permitindo que a conexão com o Telegram seja estabelecida. Ressalte-se que os valores do *token* e o *chat_id* foram mascarados para preservar a segurança da aplicação.

Figura 40– Fornecimento de *Token* e *Chat_id* para API do Telegram

```
1  import telepot
2
3  TOKEN="xxxxxxxxxxxxxxxxxxxxxxxxxxxx"
4  chat_id="yyyyyyyyyyyyyyyyyyyyyyyyyy"
5  bot=telepot.Bot(TOKEN)
```

Fonte: elaborada pelo autor

A Figura 41 a seguir ilustra o código em Python responsável por percorrer os *logs* do Suricata e retornar apenas o último *log*, para que ele seja enviado via Telegram. Na parte superior do código, é ilustrada a importação dos módulos necessários: *telepot* e *subprocess*. O módulo *telepot* é necessário para criar a API para o Telegram, conforme explicado anteriormente. Já o módulo *subprocess* é responsável por permitir ao Python receber comandos do Linux, e processar sua saída no próprio código Python. A seguir, é ilustrada a etapa de segurança com *token* e *chat-id*, já explicada anteriormente.

Figura 41 – Código em Python percorrendo os *logs* do Suricata e enviando alerta via Telegram

```
1  import telepot
2  import subprocess
3
4  TOKEN="xxxxxxxxxxxxxxxxxxxxxxxxxxxx"
5  chat_id="yyyyyyyyyyyyyyyyyyyyyyyyyy"
6  bot=telepot.Bot(TOKEN)
7
8
9
10 alertaDeSeguranca = subprocess.run(["sudo tail -n 1 /var/log/suricata/fast.log"], shell
   =True, capture_output=True, text=True)
11
12 if '99126611' in alertaDeSeguranca.stdout:
13   mensagem = "SRV007 - Alerta de segurança! Seu servidor está sofrendo um ataque DDoS.
     Veja os detalhes a seguir:\n\n" + alertaDeSeguranca.stdout
14
15 #enviando mensagem criada
16
17 bot.sendMessage(chat_id,mensagem)
```

Fonte: elaborada pelo autor

Logo abaixo dessa etapa, é criada uma variável chamada de alertaDeSeguranca, a qual tem o intuito de trazer para o Python o resultado de um comando Linux executado, nesse caso, o módulo *subprocess* foi utilizado com o intuito de executar o comando *tail -n*, capaz de exibir as últimas linhas de um arquivo de texto, o arquivo *fast.log* do Suricata. No campo *n* do comando *tail*, foi passado como parâmetro o número 1, que significa que será exibida somente a última linha do arquivo de *logs*.

Os últimos três parâmetros são valores booleanos, ou seja, aceitando os valores verdadeiro ou falso, sendo o parâmetro *shell*, responsável por executar o comando no *shell* do próprio Linux, quando for definido com *true*. Quanto ao parâmetro *capture_output*, quando *true*, é capaz de capturar a saída do *shell* e exibir na função. O último parâmetro usado foi o *text*, responsável por retornar a saída no formato de *string,* quando configurado como *true*. Por meio desse código, o Python tem acesso apenas ao alerta mais recente proveniente dos *logs* do Suricata.

Ainda em referência ao código da Figura 40 acima, a próxima etapa do código é uma condicional, determinando que se o *id* da regra do Suricata 99126611 for encontrado na variável alertaDeSeguranca, será enviada uma mensagem ao Telegram através do *bot* definido anteriormente. A mensagem definida no código deste trabalho foi ‘SRV007 – Alerta de segurança! Seu servidor está participando de um ataque DDoS. Veja os detalhes a seguir:’, sendo complementada pelo texto do alerta proveniente do *log* do Suricata, o qual contém a data e hora do alerta, mensagem de possível ataque DDoS e os IPs envolvidos.