Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Internet das coisas e seus riscos: uma análise da exploração de servidores CoAP como refletores de ataques de negação de serviço amplificados

Pedro Henrique Morais Pereira

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Orientador
Prof. Dr. João José Costa Gondim

Brasília
2019

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Internet das coisas e seus riscos: uma análise da exploração de servidores CoAP como refletores de ataques de negação de serviço amplificados

Pedro Henrique Morais Pereira

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Prof. Dr. João José Costa Gondim (Orientador)
CIC/UnB

Prof. Dr. Eduardo Adilio Pelinson Alchieri
CIC/UnB

Me. Luis Alberto Belem Pacheco
CIC/UnB

Prof. Dr. José Edil Guimarães de Medeiros
Coordenador do Curso de Engenharia da Computação

Brasília, 12 de Julho de 2019

Dedicatória

Dedico esse trabalho a minha família, tanto a parte que me criou quanto a que estou construindo, essa conquista também é de vocês.

Agradecimentos

Agradeço aos meus pais, Célio e Neile, que sempre me deram suporte, ajuda e carinho que foram fundamentais para essa jornada. Ao meu irmão, Luiz, que deixou essa jornada mais leve.

Agradeço ao meus amigos, Eduardo, Jeremias, Andressa e Roberto por todo companheirismo e auxilio durante os momentos mais difíceis, além de proporcionar os melhores momentos.

Agradeço ao Professor Dr. Gondim por toda disponibilidade, conhecimento, incentivo e paciência. E ao Igor por sempre me ajudar com qualquer dúvida e por ter elaborado uma fundação sólida sobre a qual construí meu projeto.

Agradeço à minha sogra, Manuela, que sempre me ajudou, motivou na construção do trabalho. Ao meu sogro, Ts, que me apoiou e facilitou a realização dos experimentos. E aos dois eu agradeço por sua filha maravilhosa.

Por fim, agradeço a Aurora que sempre foi uma companheira incrível em todas as ocasiões, que em breve será minha esposa. E que possamos continuar essa jornada um ao lado do outro onde quer que estivermos.

Resumo

Ataques de Negação de Serviço (DoS) são amplamente utilizados e apresentam grandes riscos à estabilidade da Internet. São ataques antigos que foram evoluindo junto com tecnologias e infraestrutura presentes na internet. Com advento e popularização da Internet das Coisas (IoT) muitos dispositivos de baixo poder computacional estão aberto para internet com baixa preocupação com segurança. Como esses dispositivos tem capacidade reduzida, são utilizados protocolos mais modestos que, comumente, trocam segurança por simplicidade. Esse trabalho apresenta uma análise sobre como os vários dispositivos da Internet das Coisas podem ser potenciais vetores de ataques, uma visão geral de ataque DoS e um enfoque em DoS com CoAP para reflexão amplificada de pacotes UDP. E, para simulações e testes foi também implementado um módulo CoAP na ferramenta *Lindehof*[1].

Palavras-chave: CoAP, IoT, DoS, DDoS, Reflexão Amplificada, Linderhof

[1] https://github.com/linderhof-unb/Linderhof

Abstract

Denial of Service (DoS) attacks are widely used and present great risks to the stability of the Internet. These are well known attacks that have evolved along with Internet technologies and infrastructure. With the popularization of the Internet of Things (IoT), many low power computing devices are now open to the Internet with little security concerns. As these devices have reduced computational power, lightweight communication protocols are used that commonly switch security for simplicity. In this work we will present an analysis of how the various IoT devices can potentially be attack vectors, an overview of DoS attack focusing on exploiting CoAP for amplified reflection of UDP packets. Also, a new module for the *Lindehof*[2] framework was implemented for simulations and tests.

Keywords: CoAP, IoT, DoS, DDoS, Amplified Reflexion, Linderhof

[2]https://github.com/linderhof-unb/Linderhof

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AR-DDoS Negação de Serviço Distribuída com Reflexão Amplificada.

CoAP Constrained Application Protocol.

CoRE Constrained RESTful Environments.

CPU Central processing unit.

DDoS Negação de Serviço Distribuída.

DoS Negação de Serviço.

DTLS Datagram Transport Layer Security.

IETF Internet Engineering Task Force.

IoT Internet das Coisas.

IP Internet Protocol.

M2M Machine-to-Machine.

MQTT Message Queuing Telemetry Transport.

PC Computador Pessoal.

RFC Request for Comments.

RFID Radio-frequency identification.

SSD Solid-state drive.

TCP Transmission Control Protocol.

TLV Type-Length-Value.

UDP User Datagram Protocol.

Capítulo 1

Introdução

A Internet das Coisas (IoT) tem ganhado adesão de larga escala e amplo alcance, com sua proposta de conectar dispositivos, pessoas e serviços que trocam informações sobre si e sobre o ambiente em prol de possibilitar ações automatizadas inteligentes em várias aplicações cotidianas[2].

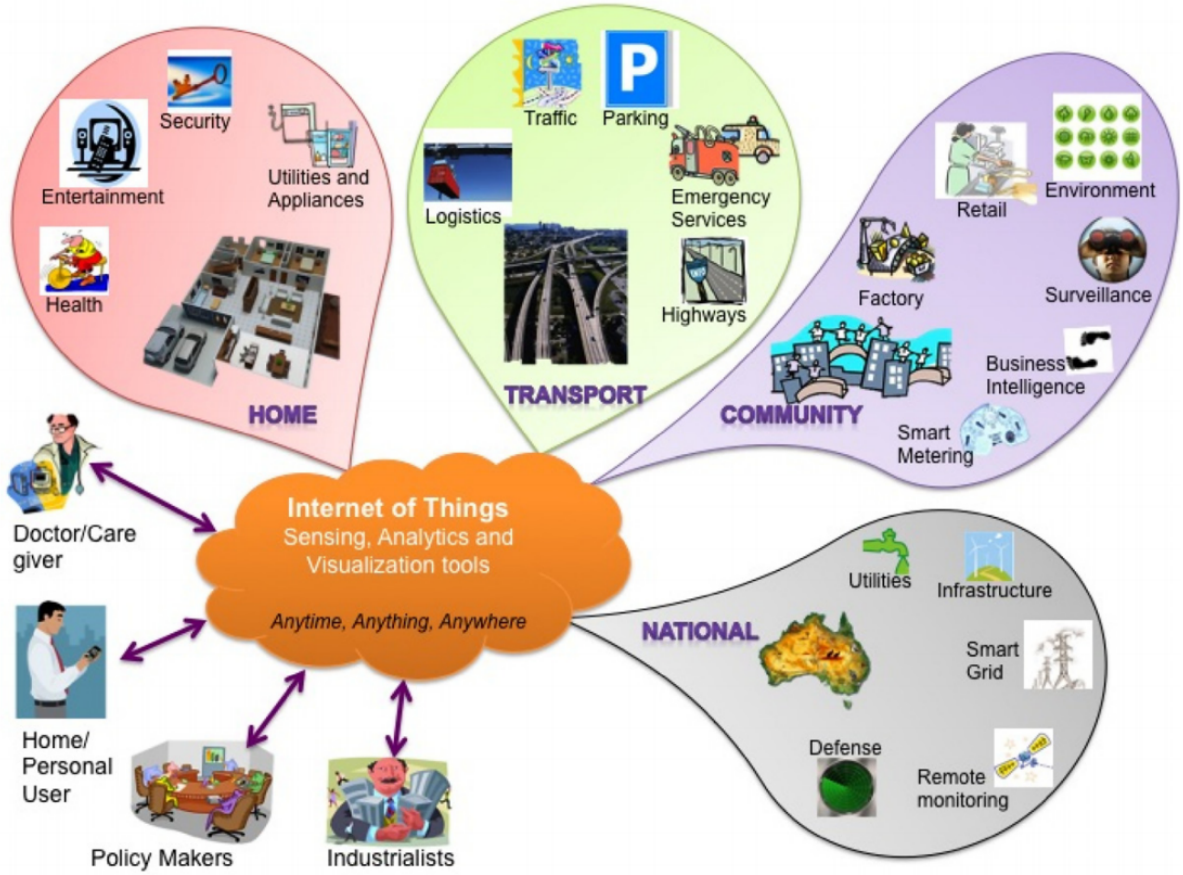

Figura 1.1: Esquemático sobre a Internet das Coisas (IoT), mostrando sob o ponto de vista de usuários finais as áreas de aplicação existentes, com base em dados. (Fonte: [2]).

Tais aplicações são tão diversas quanto os dispositivos IoT: de transporte inteligente à medicina de alta precisão, são milhares os dispositivos necessários para compor toda essa integração[3]. A Figura 1.1 ilustra, com uma abordagem focada em usuário final, as diversas áreas que podem ser beneficiadas pela Internet das Coisas.

Dispositivos IoT no geral possuem tamanho reduzido, baixo poder computacional, baixo consumo de energia e são notoriamente heterogêneos entre si[4]. Logo, faz-se necessária a definição de protocolos de comunicação que sejam capazes de lidar com essa diversidade.

O protocolo Constrained Application Protocol (CoAP) nasceu, então, com o propósito de especificar como dispositivos limitados de baixa potência podem se comunicar na Internet das Coisas[1].

Entretanto, a grande quantidade de dispositivos IoT aliados a um protocolo criado para ser simples e leve é uma combinação com grande potencial de risco: essa estrutura pode ser explorada como vetor para Ataques de Negação de Serviço (DoS) e Negação de Serviço Distribuída (DDoS)[5].

1.1 Motivação

Entidades maliciosas podem usar de técnicas de amplificação e de reflexão para focar uma grande quantidade de tráfego visando a exaustão de banda em um dado alvo[6].

Em outras palavras, a técnica de amplificação visa explorar protocolos que possuem uma resposta com uma quantidade de dados muito superior à quantidade de dados na requisição ao servidor. Assim, um ataque pode aumentar sua força ao escolher bons refletores.

Já a técnica de reflexão faz com que os servidores atuem como refletores, de forma que recebem requisições com um endereço de origem falsificado e acabam por apontar a sua resposta para esse endereço falso. Assim, o tráfego das respostas é redirecionado para a vítima atacada em vez de ser respondido para o atacante.

No geral, há uma preocupação com possíveis ameaças decorrentes da adoção generalizada da Internet das Coisas. O número de dispositivos conectados à Internet tem aumentado drasticamente, potencializando os riscos de segurança[7]. O presente trabalho tem como motivação explorar o potencial de se usar essa crescente quantidade de dispositivos IoT como refletores de ataques Negação de Serviço Distribuída com Reflexão Amplificada.

1.2 Justificativa

O CoAP[1] é amplamente difundido, com mais de 580 000 dispositivos abertos para Internet. Isso é um grande vetor potencial para ataques de Negação de Serviço Distribuída[5]. Tal risco é maior devido à falta de autenticação e ao protocolo funcionar sobre UDP, de forma que o IP é facilmente alterado.[8]

Em específico, o problema é causado devido a cada dispositivo ser um servidor simples que responde a requisições abertas para ler os dados, usando pacotes UDP. Assim, se o endereço de origem IP for falsificado, esses servidores poderiam ser utilizados com refletores. E, dado que a relação entre quantidade de dados respondida é superior à requisitada, ocorre também a amplificação de tráfego durante a reflexão. [6]

1.3 Objetivo

Esse trabalho tem como objetivo analisar os riscos de se expor na rede uma grande quantidade de dispositivos IoT que utilizam o protocolo CoAP, dado que os mesmos tem o potencial de serem utilizados como refletores de ataques de Negação de Serviço Distribuída com Reflexão Amplificada.

1.4 Contribuições

Esse trabalho possui três contribuições principais:

1. Uma análise teórica do potencial de exploração da estrutura da Internet das Coisas (IoT) como vetor para ataques de Negação de Serviço (DoS);
2. Uma extensão da ferramenta *Linderhof*, criando um módulo para ataques explorando o protocolo CoAP;
3. Uma simulação de um ataque Negação de Serviço Distribuída com Reflexão Amplificada, verificando na prática a capacidade de reflexação e amplificação de ataques utilizando o CoAP.

Em resumo, o foco principal é fazer uma análise teórica desse cenário e uma extensão da ferramenta *Linderhof*, criando um novo módulo para possibilitar a simulação e testes de um Ataque de Negação de Serviço (DoS) com reflexão amplificada usando o protocolo CoAP.

1.5 Organização do trabalho

O restante do trabalho é organizado em quatro capítulos principais:

- Capítulo 2: Contém embasamento teórico, desde uma visão geral sobre a Internet das coisas até uma análise da exploração do protocolo CoAP como vetor de ataques de negação de serviço;
- Capítulo 3: Trata da extensão da ferramenta Linderhof para conter um módulo CoAP, bem como a implementação de melhorias;
- Capítulo 4: Dispõe sobre os experimentos de ataques de reflexão amplificada utilizando CoAP, por meio do uso da ferramenta Lindehof (estendida para esse propósito);
- Capítulo 5: Propõe discussão sobre as contribuições desse trabalho, conclusões gerais e possíveis trabalhos futuros.

Capítulo 2

Fundamentação teórica

O capítulo atual aborda os conhecimentos teóricos que motivaram e embasaram este trabalho. Ele explana a respeito da Internet das Coisas em uma seção e em outra seção explica em detalhes um dos seus protocolos (o CoAP), que é especialmente vulnerável a ataques DoS. Tais ataques são, então, o foco de uma terceira seção, seguido de uma quarta seção a respeito da ferramenta Linderhof criada para análise de diversos ataques DoS. Por fim, o capítulo traz uma seção final com considerações gerais pertinentes aos tópicos abordados.

2.1 Internet das Coisas (IoT)

A nova era da computação transcende a estrutura tradicional de computadores pessoais e servidores. Nesse contexto nasceu a Internet das Coisas (IoT), conceito integrador que faz com que muitos dos objetos que nos cercam estejam conectados na rede de uma forma ou outra[2].

O termo *Internet of Things* foi cunhado por Kevin Ashton, como título de uma palestra feita por ele em 1999. A ideia apresentada envolvia o uso de tecnologia RFID para melhorar a capacidade de abastecimento da empresa multinacional Procter & Gamble[9]. Desde então, a ideia de conectar cada vez mais pessoas, dispositivos e ambientes dado o contexto em que se encontram tem se popularizado.

Essa popularização pode ser evidenciada por uma pesquisa feita pela Statista[10] em 2016. Na época, estimou-se 17.68 bilhões de dispositivos IoT conectados na rede e projetou-se que até 2025 esse número cresceria para 75.44 bilhões (como é possível observar na Figura 2.1).

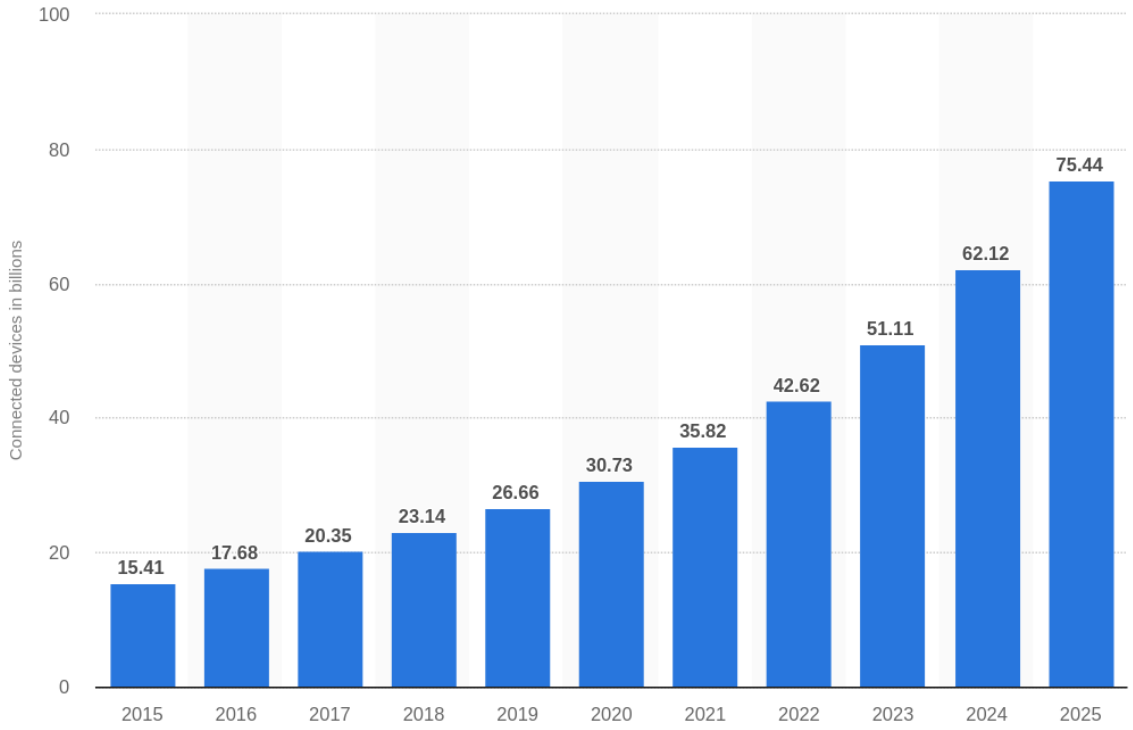

Figura 2.1: A base total de dispositivos conectados à Internet das Coisas está projetada para chegar a 75.44 bilhões em todo o mundo até 2025, seguindo um aumento de cinco vezes em dez anos. (Fonte: [10]).

Conceitualmente, pode-se identificar a Internet das Coisas (IoT) como um paradigma de convergência de três visões propostas por Atzori et al.[3] (que são definidas com mais exemplos na Figura 2.2):

1. **Orientação a Internet:** Foco em elementos arquiteturais da Internet que tornam possível objetos do mundo real serem abstraídos na Internet;
2. **Orientação a coisas:** Foco em elementos tecnológicos simples que se interligam, como o padrão RFID e o padrão NFC;
3. **Orientação a semântica:** Foco em como modelar e representar os nós interligados na Internet das Coisas e a grande quantidade de dados coletada por eles.

Dessas visões, pode-se notar pontos de interesse em relação à estrutura da Internet das Coisas:

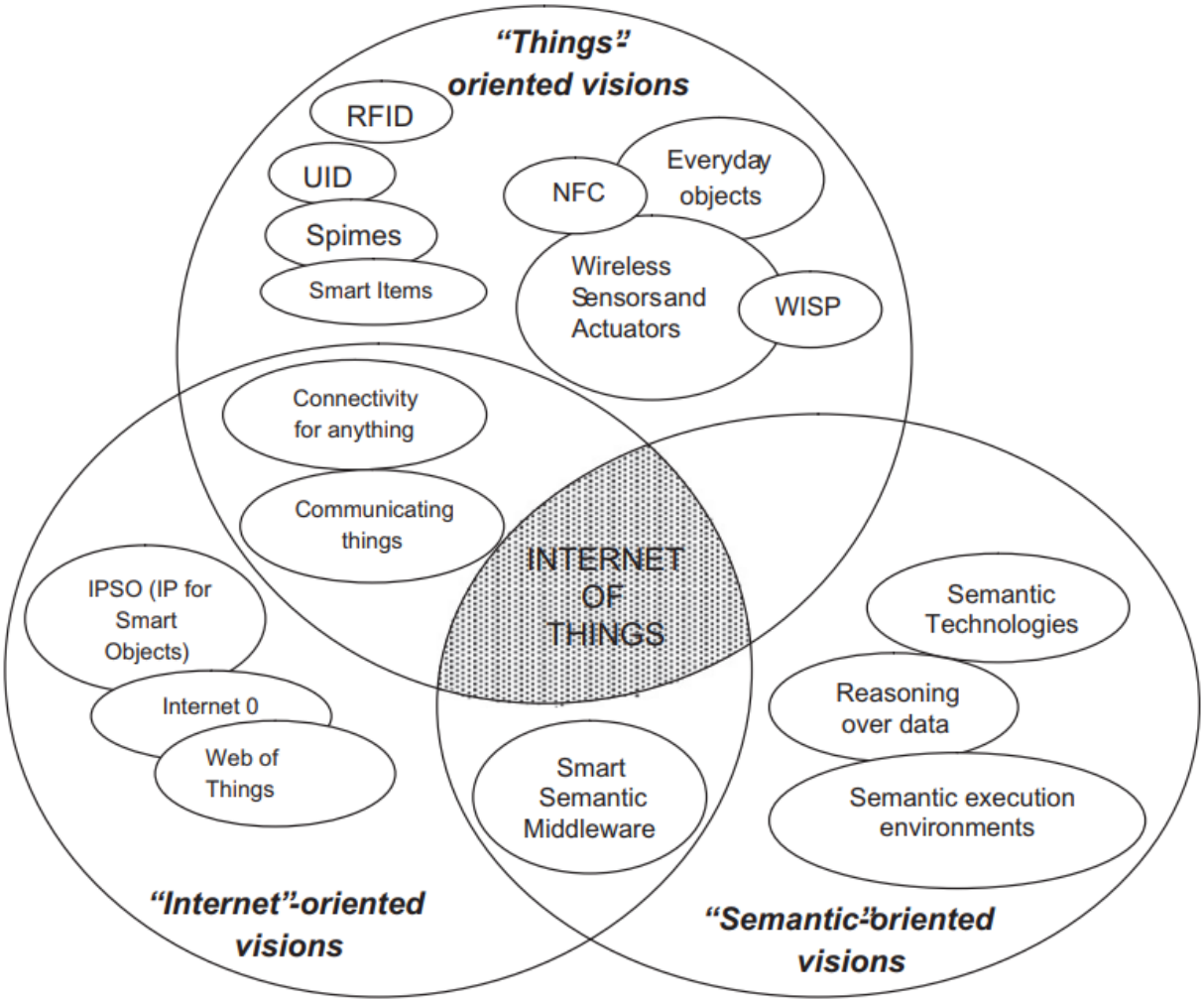

Figura 2.2: As três visões que compõem a Internet das Coisas de acordo com Atzori et al. (Fonte: [3]).

- A visão de orientação a Internet chama a atenção sobre a importância de padrões de comunicação próprios para esse contexto;
- A visão de orientação a coisas mostra que componentes IoT são heterogêneos e, no geral, devem ser simples e leves. Logo, as tecnologias usadas por eles devem se adequar a isso;
- A visão de orientação a semântica ressalta que existe uma grande quantidade de dispositivos interconectados capazes de gerar muito tráfego e quantidade de dados. Necessitando de uma agregação e entendimento.

Nesse contexto, foram criados protocolos de comunicação pensados para suprir as necessidades específicas dessa nova estrutura, como o Message Queuing Telemetry Transport (MQTT) e o Constrained Application Protocol (CoAP).

Ambos os protocolos têm em comum o foco em uma comunicação entre dispositivos com arquiteturas diferentes entre si, bem como com capacidade computacional e potência energética limitadas[5].

Porém, a estrutura da comunicação desses protocolos são diferentes. Enquanto o MQTT possui um modelo de interação centralizado, nos quais os dispositivos dependem de um *broker* para se comunicar, o CoAP é descentralizado e a comunicação funciona de equipamento a equipamento (cada um podendo agir tanto como cliente quanto como servidor). Uma melhor visualização dessas diferenças pode ser vista na Figura 2.3.

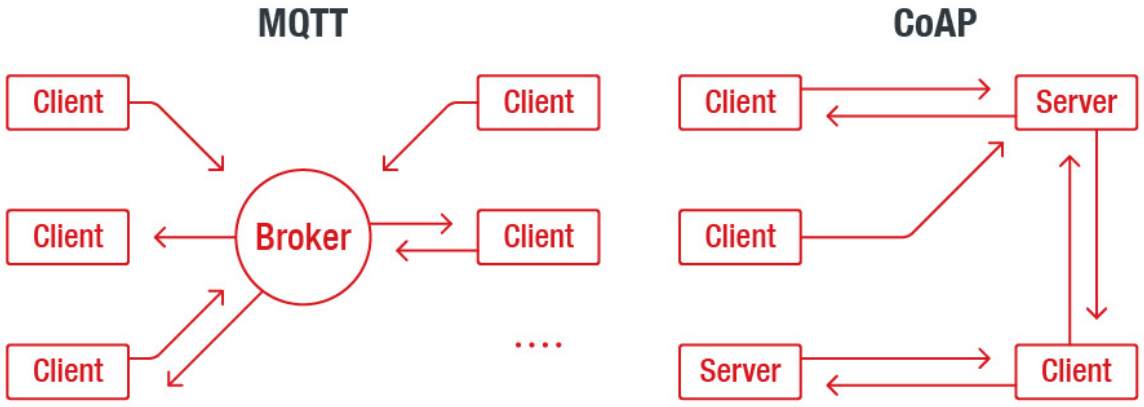

Figura 2.3: Uma visão de alto nível dos modelos de interação de MQTT (esquerda) e CoAP (direita). (Fonte: [5]).

A seção a seguir irá focar no protocolo CoAP, por se tratar de um dos objetos de estudo principais do presente trabalho.

2.2 Constrained Application Protocol (CoAP)

O Constrained Application Protocol (CoAP) foi definido em 2014 por meio do Request for Comments (RFC) 7252[1]. Um RFC é um documento técnico produzido como especificações de padrões a serem utilizados por toda a Internet. Tais documentos são mantidos pela Internet Engineering Task Force (IETF).

Em específico, o CoAP foi projetado por um grupo especializado em Constrained RESTful Environments (CoRE), com foco em possibilitar comunicação genérica entre dispositivos com capacidades restritas e dispositivos da Internet no geral. Ele possibilita, então, uma comunicação eficiente e simples entre nós heterogêneos na rede, otimizada especificamente para comunicação Machine-to-Machine (M2M).

O protocolo é pensado para que seja facilmente traduzido para HTTP (de forma a simplificar a integração com o resto da Internet), bem como prover funcionalidades mais avançadas (tais como suporte a *multicast*). Essa possibilidade de uso similar ao HTTP provê uma familiaridade ao desenvolvedores que já estão acostumados com a popular arquitetura *RESTful*.

Porém, ao contrário do HTTP, o CoAP lida com intercâmbio de mensagens de forma assíncrona, logo acima do protocolo de comunicação por trocas de datagramas na camada de transporte: o User Datagram Protocol (UDP).

2.2.1 Estrutura do protocolo

Seguindo a estrutura de comunicação máquina a máquina, definiu-se os seguintes quatro tipos de mensagens possíveis:

- *Confirmable* (confirmável);
- *Non-confirmable* (não confirmável);
- *Acknowledgement* (confirmada);
- *Reset* ("resetada").

	CON	NON	ACK	RST
Request	X	X	-	-
Response	X	X	X	-
Empty	*	-	X	X

Figura 2.4: Tabela correlacionando tipo de mensagem com a mensagem ser requisição, resposta ou vazia. CON se refere a mensagem do tipo *Confirmable*, NON se refere a mensagem do tipo *Non-confirmable*, ACK se refere a mensagem do tipo *Acknowledgement* e RST se refere a mensagem do tipo *RESET*. (Fonte: [1]).

Requisições podem se feitas por mensagens do tipo *Confirmable* (para comunicações confiáveis) ou do tipo *Non-confirmable* (para comunicações não confiáveis). A resposta de uma requisição pode ser do tipo *Confirmable*, *Non-confirmable* ou simplesmente *Acknowledgement* (para confirmação simples de recebimento). E, quando um recipiente não

consegue de nenhuma forma processar a requisição, ele responde com uma mensagem do tipo *Reset*.

Quanto à sua estrutura, mensagens CoAP são codificadas em um formato binário simples (vide Figura 2.5). Elas começam com um cabeçalho de 4 bytes de tamanho fixo, com:

- **Ver:** versão do CoAP sendo seguida, para permitir identificação de versões futuras;
- **T:** tipo da mensagem (algum dos quatro supramencionados: *Confirmable*, *Non-confirmable*, *Acknowledgement* e *Reset*);
- **TKL:** qual é o tamanho do *token* da mensagem (maiores explicações sobre o *token* poderão ser encontradas abaixo);
- **Code:** para indicar o código e a classe da mensagem, passando informações específicas, tais como se é uma mensagem de sucesso ou de erro e, se for de erro, qual a categoria do erro. Similar a HTTP;
- **Message ID:** identificador da mensagem, usado para identificar casos de duplicatas e para casar mensagens *Acknowledgement*/*Reset* com mensagens *Confirmable*/*Non-confirmable*.

Essas informações são seguidas por um valor de *token* de tamanho variável (entre 0 e 8 bytes de comprimento), sendo que tal tamanho foi passado no cabeçalho. Esse valor tem como propósito correlacionar mensagens de requisição com mensagens de resposta.

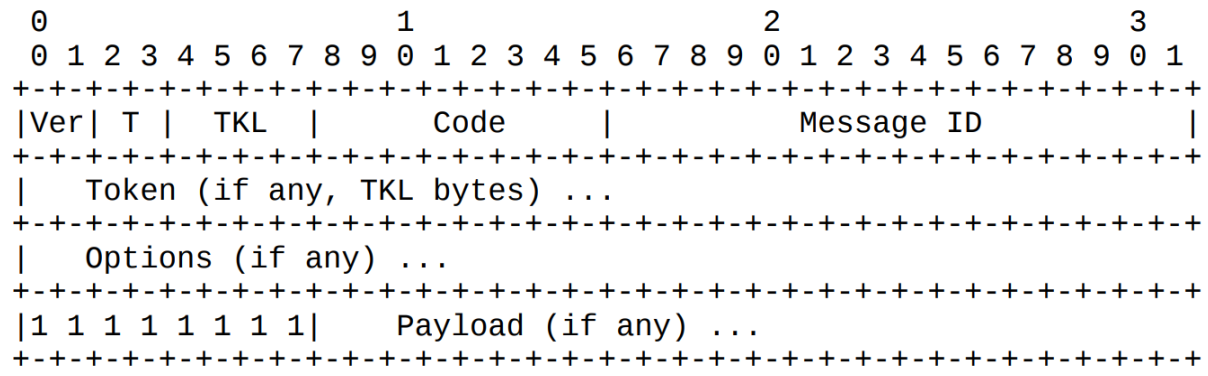

Figura 2.5: Ilustração da estrutura de uma mensagem CoAP especificada pela RFC 7252. (Fonte: [1]).

Depois disso, é possível definir zero ou mais opções usando o Type-Length-Value (TLV). Cada instância de opção tem um número e segue um formato delta definido pelo padrão CoAP. Nesse padrão, há também o tamanho do valor da opção e o valor da opção propriamente dito. Um exemplo de tipo de opção adicionada recentemente é a *Block2* (estabelecida na RFC 7959[11]), que torna possível definir o tamanho da resposta da requisição. A Tabela 2.1 mostra as opções originais definidas pela primeira RFC.

No.	C	U	N	R	Name	Format	Length	Default
1	x			x	If-Match	opaque	0-8	(none)
3	x	x	-		Uri-Host	string	1-255	
4				x	ETag	opaque	1-8	(none)
5	x				If-None-Match	empty	0	(none)
7	x	x	-		Uri-Port	uint	0-2	
8				x	Location-Path	string	0-255	(none)
11	x	x	-	x	Uri-Path	string	0-255	(none)
12					Content-Format	uint	0-2	(none)
14		x	-		Max-Age	uint	0-4	60
15	x	x	-	x	Uri-Query	string	0-255	(none)
17	x				Accept	uint	0-2	(none)
20				x	Location-Query	string	0-255	(none)
35	x	x	-		Proxy-Uri	string	1-1034	(none)
39	x	x	-		Proxy-Scheme	string	1-255	(none)
60			x		Size1	uint	0-4	(none)

Tabela 2.1: Opções delta estabelecidas originalmente pela RFC 7252. (Fonte: [1]).

Por fim, o final do datagrama é ocupado pelo marcador de *payload* e o *payload* em si (carga da dados). Quando não existir marcador de *payload* isso significa que não deverá existir um *payload*.

O CoAP também implementa outras funcionalidades interessantes, como mecanismo de *cache* simples e mecanismo de descoberta de recursos. O mecanismo de *cache* simples tem foco em otimizar a performance, guardando algumas respostas de requisições mais comuns. Já o mecanismo de descoberta de recursos é importante para interações Machine-to-Machine (M2M) pois ele lista os recursos disponíveis no servidor para serem requisitados (geralmente isso pode ser obtido fazendo uma requisição para o caminho: `/.well-known/core` [12], como visto na Figura 2.6).

Em resumo, a comunicação do protocolo é feita na camada de serviços, diretamente entre um dispositivo e outro (M2M). Em adição, ele se usa do User Datagram Protocol (UDP) e opcionalmente do Datagram Transport Layer Security (DTLS) em casos que deseja manter uma comunicação mais segura. Entretanto, existe um *trade-off* entre segurança e eficiência da comunicação, o que faz com que vários dispositivos IoT que utilizam o protocolo escolham não utilizar opções extras de segurança.

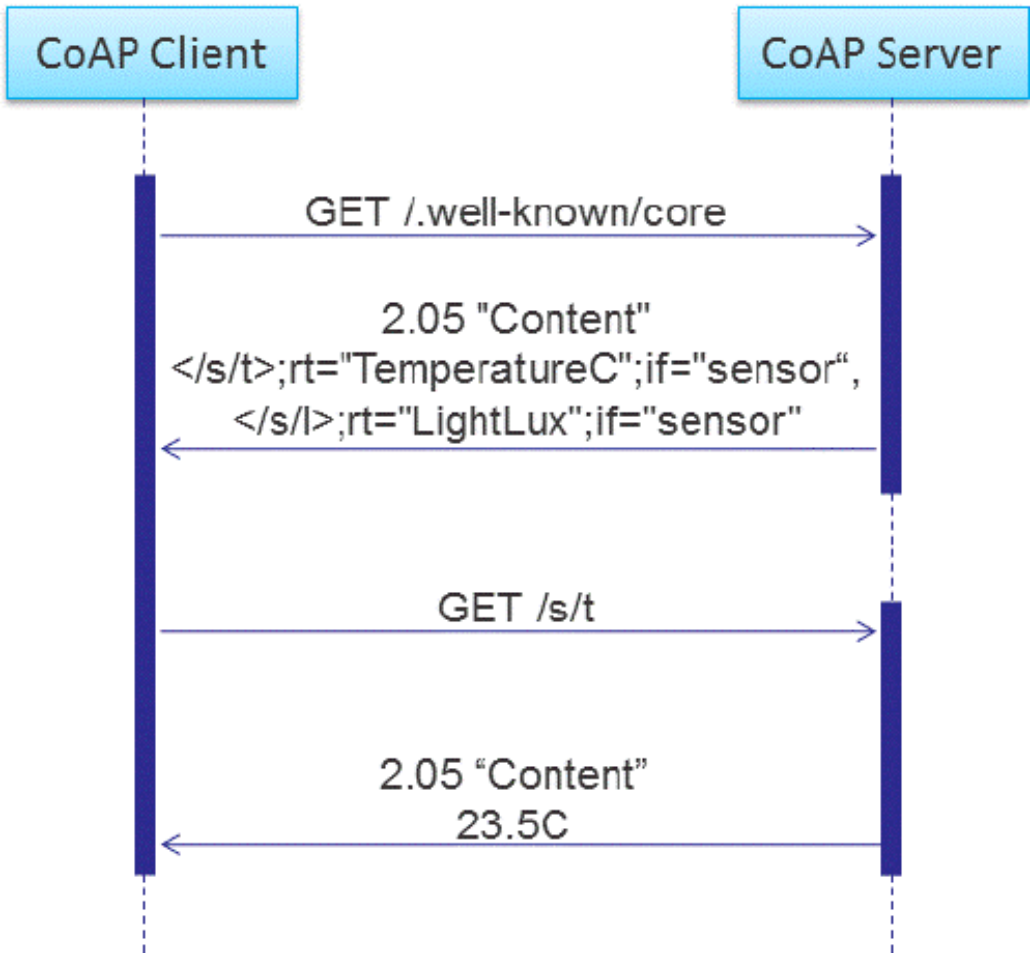

Figura 2.6: Um exemplo de descoberta de recursos do Constrained RESTful Environments (CoRE) através de requisição no `/.well-known/core`, que é um endpoint conhecido e definido especificamente para o protocolo CoAP. (Fonte: [13]).

Vale ressaltar que a natureza do CoAP, a não confiabilidade do UDP e a não priorização de mecanismos de segurança tornam os dispositivos que usam o CoAP vulneráveis. Essa vulnerabilidade pode ser explorada para diversos fins, incluindo a exploração indevida de dispositivos que utilizam CoAP como meio para promover ataques de Negação de Serviço (DoS), que são o tema da próxima seção.

2.3 Ataques de Negação de Serviço (DoS)

Ataques de Negação de Serviço (DoS) são aqueles cujo objetivo é impedir que usuários legítimos acessem algum tipo de recurso ou serviço. Esse é um tipo genérico de ataque que pode ocorrer em diversos contextos e situações, de ataques a sistemas operacionais [14] a serviços em rede [15].

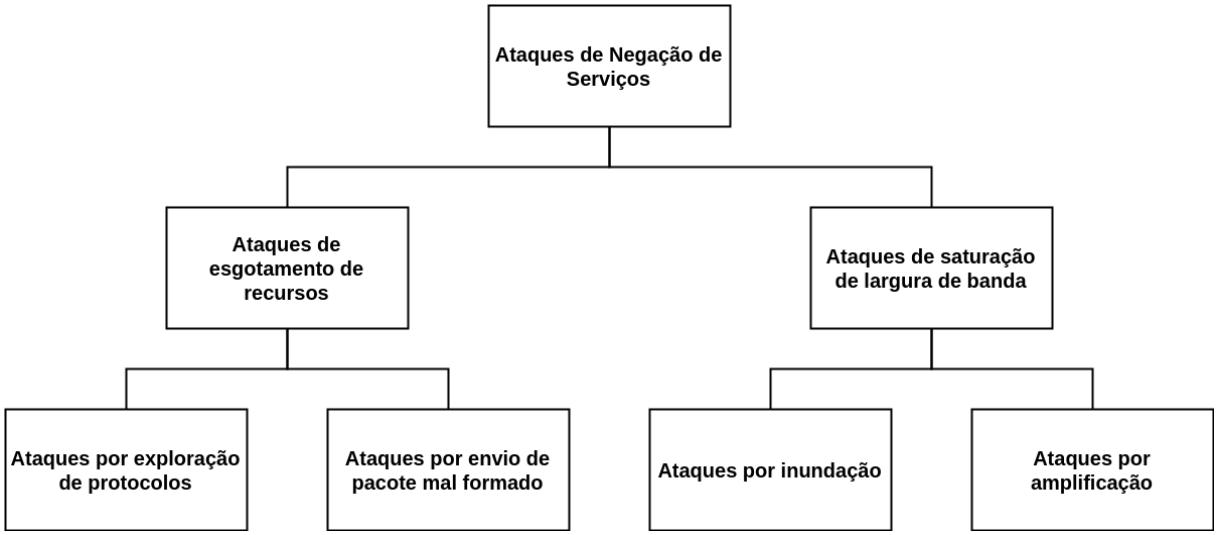

Figura 2.7: Visualização da taxonomia proposta por Specht (Fonte: [16]).

Specht et al. [16], propôs uma taxonomia para ataques de negação de serviço, de forma a separar esses ataques em dois grandes grupos. Esses grupos, por sua vez, podem também ser separados em dois subgrupos cada:

1. **Ataques de esgotamento de recursos:** Tem por finalidade o esgotamento dos recursos disponíveis por um dado serviço, impossibilitando que ele atenda a requisições de usuários legítimos.

    1.1 **Ataques por exploração de protocolos:** Caracterizam-se por explorar alguma falha específica ou de implementação de algum protocolo vulnerável, utilizando isso para comprometer algum recurso da vítima. Um ataque conhecido desse grupo é o ataque TCP SYN, o qual se aproveita do *three-way handshak* (inicialização em três etapas) do protocolo Transmission Control Protocol (TCP). Em resumo, para iniciar uma comunicação TCP um cliente envia para aplicação um pacote do tipo TCP SYN, então o serviço aloca recursos para aquela requisição e responde normalmente com um SYN+ACK. Depois, o cliente envia para aplicação um pacote ACK e começa a fazer as requisições desejadas. Desta forma, o ataque consiste em enviar diversos pacotes do tipo TCP SYN para que a aplicação aloque vários recursos para continuar com essa comunicação. Porém, como o atacante nunca envia o ACK de resposta, cada vez mais recursos diversos são alocados nessa comunicação ilegítima, até o ponto que o serviço não tenha mais recursos suficientes para responder uma requisição de um usuário legítimo;

1.2 **Ataques por envio de pacote mal formado:** Aproveitam-se de sistemas que não preveem o recebimento de pacotes incompletos ou mal formados para enviar tais pacotes com intenção maliciosa, causando falhas e/ou esperas por tempo indeterminado. Por exemplo, um tipo de ataque consiste em enviar um pacote com o mesmo IP de origem e destino, de forma a causar erros no tratamento feito pelo sistema da vítima e travá-lo. Outro exemplo é enviar um pacote TCP/IP com o campo de tamanho total muito grande, de forma que o sistema da vitima ficará esperando receber todo um conteúdo que nunca irá chegar, de forma que o objetivo também é fazer com que a vítima aloque recursos desnecessariamente até esgota-los;

2. **Ataques de saturação de largura de banda:** Como o nome diz, a finalidade é esgotar a largura de banda, de forma que o recurso na rede fique inacessível para usuários legítimos. O sistema atacado poderá ficar lento, sofrer erros ou ficar isolado do resto da rede, uma vez que sua largura de banda esteja esgotada.

   2.1 **Ataques por inundação (*flood*):** Caracterizados pelo envio de grandes volumes de tráfego ao sistema. User Datagram Protocol (UDP) *flood* é exemplo de ataque dessa categoria. Nesse ataque, um grande número de pacotes UDP são enviados para o sistema da vítima para portas randômicas ou específicas. Isso causará um processamento dessas mensagens que tenta destinar as mensagens para alguma aplicação (caso exista alguma aplicação respondendo a porta de destino). E, essa grande quantidade de pacote pode saturar a largura de banda da vítima, de forma a afetar não só a rede da vítima em si mas toda rede ao redor dessa vítima, causando efeitos colaterais em outras aplicações;

   2.2 **Ataques por amplificação:** Caracterizados por explorar um serviço intermediário (refletor) capaz de prover uma resposta a uma requisição com uma quantidade de dados maior do que a requisição em si, de forma a amplificar o volume de dados direcionados a uma vítima. Esse ataque se torna uma eficiente forma de gerar grande volume de dados a partir de um volume reduzido de requisições. Nota-se que esse tipo de ataque também tem efeitos colaterais na rede em torno da vitima, e ainda tem um efeito colateral no intermediário (refletor) que será utilizado para realizar a amplificação.

Gondim et al. [17] explica que existem vários tipos de ataques Negação de Serviço que podem ser organizados em duas classes principais (que também podem ser subdivididas em duas classes cada):

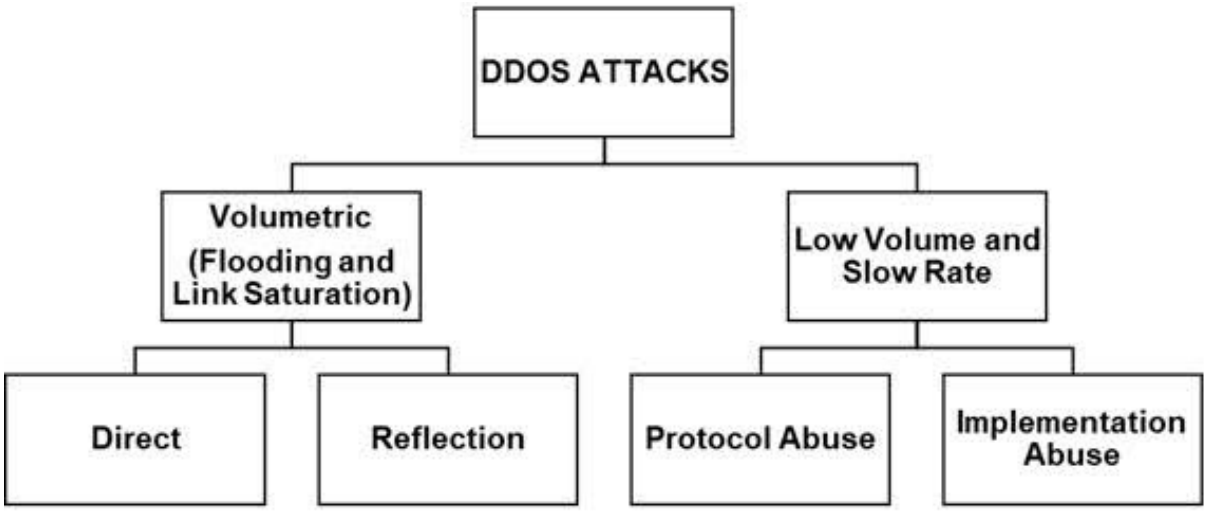

Figura 2.8: Divisão entre os tipos de ataques de DoS proposta por Gondim (Fonte: [17]).

1. **Ataques volumétricos (*Volumetric: Flooding and Link Sauration*):** Incluem ataques nos quais grandes volumes de tráfego dados inundam a vítima, excedendo sua capacidade de processamento ou sua largura de banda, fazendo com que requisições legítimas não possam ser tratadas.

   1.1 **Ataques diretos:** Nos quais nós comprometidos mandam grandes volumes de tráfego diretamente para a vítima;

   1.2 **Ataques de reflexão:** Nos quais entidades maliciosas se aproveitam de nós refletores (intermediários) para inundar a vítima;

2. **Ataques de baixo volume e taxa lenta (*Low Volume and Slow Rate*):** Ataques que promovem exploração de protocolos, visando implementações, características ou funcionalidades específicas para causar exaustão dos recursos da vítima e, consequentemente, impedir requisições legítimas de serem respondidas propriamente.

   2.1 **Abuso de protocolo:** Se aproveitam para explorar falhas no *design* do protocolo em si;

   2.2 **Abuso de implementação:** Se aproveitam para explorar falhas de alguma implementação específica de um protocolo.

O presente trabalho tem foco no uso potencial do grande volume de dispositivos IoT em ataques de DoS, de forma que as subseções a seguir irão abordar conceitos de ataques de Negação de Serviço Distribuída (DDoS) usando amplificação reflexiva.

2.3.1 Ataques de Negação de Serviço Distribuída (DDoS)

Ataques de Negação de Serviço Distribuída (DDoS) são um subtipo de ataques Negação de Serviço (DoS) nos quais o tráfego, inundando uma vítima, irá se originar de diferentes fontes atacantes. São ataques coordenados e indiretos que se aproveitam de sistemas comprometidos[16].

Os mesmos grupos e classes de ataques definidos pelas taxonomias explicadas acima nessa seção podem ser aplicáveis em ataques distribuídos também. A diferença principal é que enquanto em um ataque não distribuído a entidade atacante é fonte do tráfego do ataque, em ataques distribuídos a entidade atacante age como gerenciadora de um grupo de sistemas comprometidos que serão a fonte do tráfego do ataque[18].

Conceitualmente, também é possível definir como **vítima primária** o sistema que é o foco do ataque, no qual o tráfego de dados está direcionado. Entretanto, os sistemas comprometidos que são usados para gerar esse tráfego também podem ser prejudicados, de forma que eles também podem ser considerados **vítimas secundárias**. A(s) entidade(s) maliciosa(s) coordenando o ataque pode(m) ser simplesmente chamada(s) de **atacante(s)**[16].

No geral, esses são ataques mais difíceis de serem prevenidos e mitigados. Isso se dá, por uma série de motivos, sendo um deles o fato de que não é possível parar o ataque simplesmente bloqueando uma única fonte atacante[19]. Além disso, esses ataques tendem a ter uma proporção muito maior e é muito mais difícil encontrar quem é a entidade maliciosa que está coordenando tudo por trás.

O presente trabalho tem principal interesse no funcionamento de ataques de Negação de Serviço Distribuída que inundam uma vítima primária se utilizando de um grande volume de vítimas secundárias capazes de prover respostas para requisições maiores do que as requisições em si, causando assim uma saturação de banda no alvo do ataque. Esses chamados ataques de Negação de Serviço Distribuída com Reflexão Amplificada (AR-DDoS) são o foco da próxima subseção.

2.3.2 Ataques de Negação de Serviço Distribuída com Reflexão Amplificada (AR-DDoS)

Ataques de Negação de Serviço Distribuída com Reflexão Amplificada (AR-DDoS) são um subtipo de ataques Negação de Serviço Distribuída que se utilizam de refletores capazes de amplificar tráfego. Essa combinação de ataque distribuído com amplificação de tráfego faz com que esse tipo de ataque esteja por trás dos incidentes mais significativos de esgotamento de largura de banda desde o ano de 2013[20].

Segundo os conceitos explanados na Seção 2.3, pode-se categorizar esse tipo de ataque como sendo um ataque de saturação de largura de banda por amplificação (de acordo com Specht et al. [16]) e também como um ataque volumétrico de inundação por reflexão (de acordo com Gondim et al. [17]).

Conceitualmente, um refletor é qualquer nó que envia um datagrama IP como resposta para um previamente recebido. Para ataques de Negação de Serviço Distribuída com Reflexão Amplificada, os refletores de interesse são aqueles que conseguem causar amplificação, ou seja, cujas respostas produzem mais pacotes, *bytes* ou ambos do que a requisição de datagrama original[17]. Esse comportamento é caracterizado por um **fator de amplificação**, que indica quanto é o tráfego que um refletor pode gerar. A fórmula desse fator é definida a seguir:

$$\text{Fator de amplificação} = \frac{tamanho(\text{resposta})}{tamanho(\text{requisição})} \tag{2.1}$$

Outro conceito importante que possibilita a reflexão é a falsificação de IP (*IP spoofing*). No geral, essa técnica consiste em se criar endereço(s) de origem arbitrário(s) em datagramas IP, de forma a ser possível ofuscar o atacante. Especificamente para ataques Negação de Serviço Distribuída com Reflexão Amplificada, o IP de origem falsificado é o próprio IP da vítima primária, de forma a direcionar o tráfego dos refletores para ela[17].

A Figura 2.9 ilustra uma arquitetura típica para um ataque AR-DDoS. Essa arquitetura ela é idêntica a uma arquitetura padrão de ataques DDoS, com refletores atuando antes da vítima primária para amplificar o tráfego[20]. Essa visualização mostra duas camadas entre o atacante e a vítima primária:

1. A camada de refletores (*Reflectors*);
2. A camada de escravos (*Slaves or Bots*).

A camada de refletores trata dos nós que efetivamente enviam tráfego amplificado diretamente para a vítima. Já a camada de escravos (ou *bots*) é uma camada de *hosts* comprometidos controlados pelo atacante (ou *mestre*) para enviar requisições com IP falsificado para os refletores. Nota-se que tal camada intermediária não é necessária, é possível que o atacante seja o próprio remetente das requisições. Entretanto, quando essa camada é utilizada, destacam-se as seguintes vantagens:

1. Contribuir para aumentar a escala de ataque;
2. Fornecer ofuscação da fonte de ataque, de modo a proteger o atacante.

Por fim, um detalhamento e uma exemplificação de ataques de Negação de Serviço Distribuída com Reflexão Amplificada na vida real utilizando um protocolo popular (e de especial interesse para esse trabalho) pode ser vista na subseção a seguir.

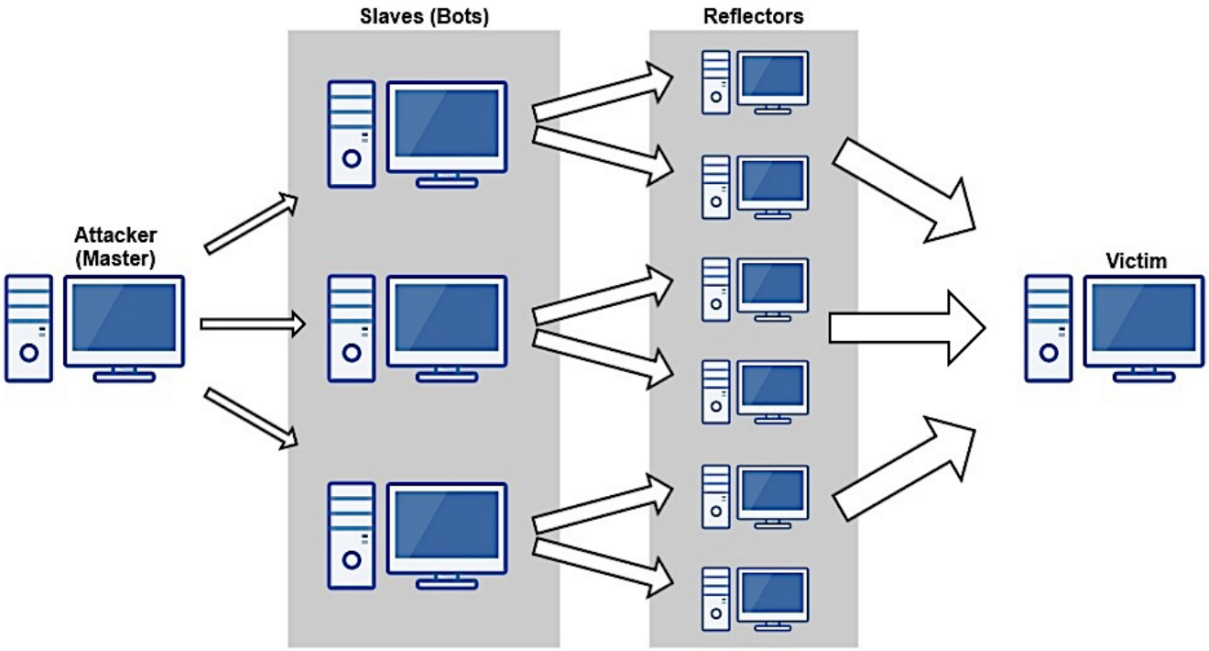

Figura 2.9: Visualização de uma arquitetura típica de um ataque de Negação de Serviço Distribuída com Reflexão Amplificada (AR-DDoS) (Fonte: [20]).

2.3.3 Explorando o CoAP

Na vida real, sistemas que usam protocolo UDP ou protocolos baseados no UDP estão especialmente vulneráveis a serem usados como refletores, devido à suscetibilidade do UDP à falsificação de IP (IP *spoofing*) e à amplificação de pacotes[21]. Como explicado na Seção 2.2, o CoAP é um desses protocolos baseados no UDP.

Nesse contexto, um atacante pode enviar um pequeno pacote UDP para um cliente CoAP (um dispositivo IoT) e o cliente responderia com um pacote muito maior direcionado à vítima. Estima-se que o fator de amplificação para servidores CoAP disponíveis atualmente na rede varie de 10 a 50, mantendo-se numa média de 34[22].

A Figura 2.10 mostra uma visualização genérica de um ataque AR-DDoS utilizando servidores UDP. Para fins dessa explicação, consideraremos que tais servidores são também CoAP, de forma que um possível ataque poderia ocorrer da seguinte forma:

- Atacante descobre uma lista de IPs e portas de servidores CoAP que podem ser utilizados como refletores;
- Atacante gera múltiplos pacotes CoAP em um dado período de tempo, enviando como requisição para os refletores. Esses pacotes tem o IP falsificado como se fosse originário da vítima. Uma requisição comum de ser feita é um GET para o `/.well-known/core` dos servidores;

- Refletores respondem às requisições, enviando-as para a vítima, como o IP de origem foi forjado. No caso de um GET para o `/.well-known/core`, a resposta irá listar todos os recursos do servidor, o que no geral significa um fator de amplificação considerável;
- A depender de condições como quantidade de requisições, quantidade de refletores, condições da rede e etc, pode ser que a vítima tenha sua largura de banda saturada.

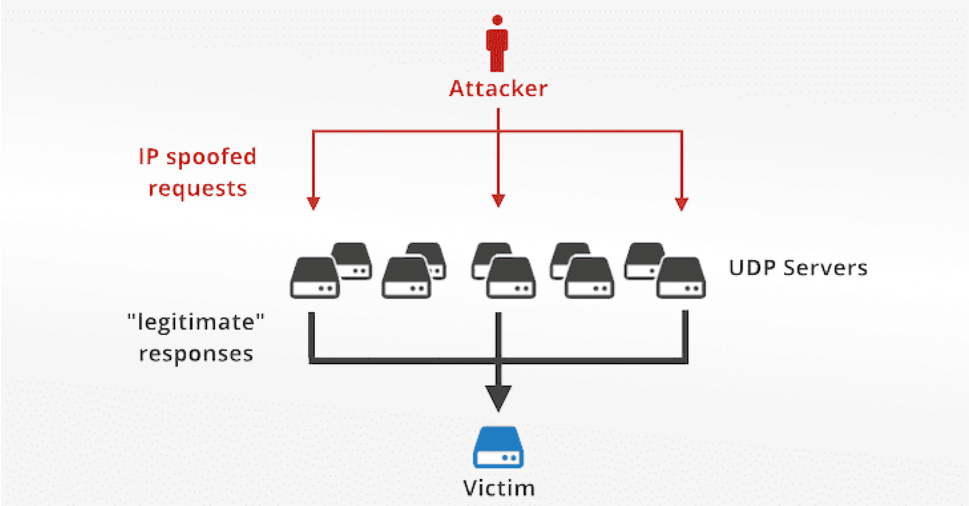

Figura 2.10: Visualização de uma estrutura de ataque de Negação de Serviço Distribuída com Reflexão Amplificada (AR-DDoS) utilizando servidores UDP ou servidores baseados em UDP (como o CoAP). (Fonte: [23]).

Para fins de estudos, existem algumas ferramentas que são capazes de simular ataques de reflexão amplificada. Uma delas está relacionada a esse trabalho e será descrita na subseção a seguir.

2.4 Linderhof

A ferramenta utilizada nesse trabalho para implementação dos ataques foi o *Linderhof*, desenvolvida por Miranda[24] e estendida por Vieira[25] em 2019. Com essa ferramenta é possível desenvolver novos módulos (também chamados de espelhos ou *mirrors*) para explorar outros protocolos.

2.4.1 Visão geral da arquitetura

A arquitetura do Linderhof é separada em 3 módulos principais:

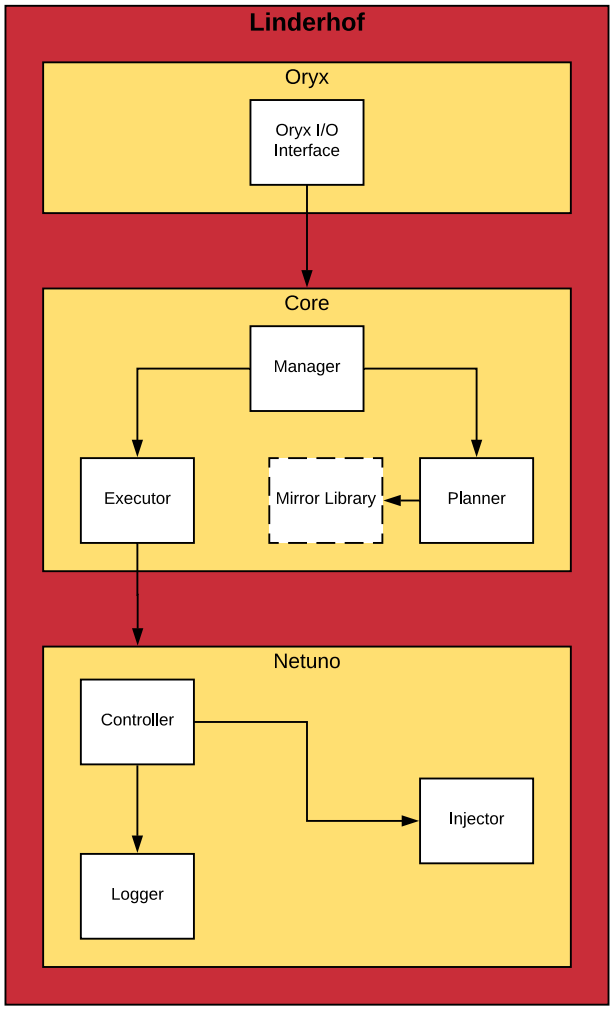

Figura 2.11: Arquitetura do Linderhof, mostrando os módulo principais e os sub-módulos internos (Fonte: [24]).

- **Oryx:** É a interface com o usuário do Linderhof, que atua como ponte de contato que interpreta e encaminha as informações para o Commander;
- **Commander:** É o núcleo da ferramenta, responsável por gerar o plano de ação do ataque, além de usar um *mirror* para gerar o pacote a ser enviado.
- **Netuno:** É a *engine* de injeção de pacotes do Linderhof. Com o plano de ação gerado pelo Commander, o Netuno é responsável por instanciar os injetores que enviam os pacotes. Além disso, também controla o *throughput* de cada injetor.

2.4.2 Oryx

Primeiro módulo a ser executado, Oryx é a interface com o usuário. Basicamente, ele converte os parâmetros passados em uma estrutura *draft*, que pode ser visualizada na

Tabela 2.2. Uma instância dessa estrutura preenchida através de dados passados pelo usuário ou valores padrão é então encaminhada para o Commander.

Para facilitar o uso da ferramenta Linderhof, o módulo do Oryx também implementa um manual resumido com os parâmetros existentes, explicando como podem ser utilizados. Esse manual pode ser acessado via comando, chamando a ferramenta com o parâmetro de ajuda (`-h`).

Opção Longa	**Opção Curta**	**Obrigatório**	**Valor Padrão**	**Descrição**
--mirror	-m	Sim	(Nenhum)	Espelho que será utilizado
--target	-t	Sim	(Nenhum)	IP da vítima
--amplifier	-a	Sim	(Nenhum)	IP do amplificador
--targport	-g	Não	80	Porta da vítima
--amport	-p	Não	Padrão do *Mirror*	Pota do amplificador
--level	-l	Não	1	Nível de ataque
--timer	-c	Não	Infinito	Duração do ataque(segundos)
--log	-f	Não	(stdout)	Nome do arquivo de *Log*
--inc	-i	Não	Infinito	Duração de cada nível(segundos)

Tabela 2.2: Parâmetros interpretados pelo Oryx, via linha de comando, para gerar o *draft* que será passado ao Commander para uso da ferramenta Linderhof

2.4.3 Commander

O núcleo da aplicação Linderhof é o módulo Commander, que é responsável por montar o plano de ataque com base no *draft* recebido pelo Oryx. Esse módulo é divido em 4 submódulos (como pode ser visto na Figura 2.11):

- **Manager:** é o primeiro submódulo a ser executado, ele é responsável por validar o *draft* recebido e funciona como uma ponte entre o Planner e Executor;
- **Planner:** A partir do *draft* o Planner é responsável por gerar um plano de ataque para o *mirror* selecionado. Esse plano contém a função do *mirror* que gerar os dados e executar o ataque, além de todos os dados que o ataque vai necessitar que estavam presentes no *draft*;
- **Hall of Mirrors:** é uma biblioteca dos ataques disponível no Linderhof. Onde são implementados os *mirrors* e os pacotes para o ataque são gerados;
- **Executor:** é submódulo mais simples, depois do plano de ataque estar completo ele apenas o executa passando todos os dados que o ataque requisita.

2.4.4 Netuno

Netuno é o módulo final. Ele atua como a *engine* de injeção. Sendo assim, é responsável por de fato enviar os pacotes do ataque e realizar o controle do fluxo de pacotes. Esse módulo é divido em 3 submódulos:

- **Controller:** é o responsável por criar e gerenciar as *threads* do injetores (*Injectors*). Além disso controla a taxa de injeção de pacotes dependendo do nível de ataque atual. Esse controle é feito indicando a cada *thread* quantos pacotes devem ser enviados em cada intervalo de tempo;
- **Logger:** sua função é registrar em arquivo ou no terminal o estado do Controller. Cada segundo ele imprimi o tempo atual, quantos pacotes foram enviados desde a ultima divisão de tempo e a meta de pacotes para o nível atual;
- **Injector:** é o módulo responsável por enviar os pacotes. Cada injetor é uma *thread* independente com um contador da quantidade de pacotes a ser enviado. Cada vez que um pacote é enviado o contador é decrementado, ao chegar a zero ele para de enviar pacote. Até o controlador reiniciar a variável informando a quantidade de pacotes a ser enviado no intervalo de tempo atual.

Nível

Um conceito muito importante para o Linderhof é o de nível de ataque. O nível define a quantidade de pacotes que será enviada a cada segundo. No nível 1, apenas um pacote por segundo é enviado, no nível 2 serão enviados 10, e assim em diante, seguindo a fórmula:

$$\text{Pacotes enviados} = 10^{N-1} \tag{2.2}$$

Nivel	**Pacotes por segundo**
1	1
2	10
3	100
4	1 000
5	10 000
6	100 000
7	1 000 000

Tabela 2.3: Pacotes por segundo por nível de ataque segundo Equação 2.2

2.5 Considerações finais

Apesar do CoAP prever medidas de segurança, muitas vezes essas medidas de segurança não são utilizadas. Seja por falta de conhecimento mais profundo sobre o protocolo, seja por fabricantes conscientemente não utilizarem devido a restrições de processamento de dispositivos IoT, o fato é que existem muitos potenciais refletores vulneráveis[21, 26].

Por se tratar de um protocolo relativamente recente, o conhecimento dos invasores sobre o protocolo é muito básico, mas tem o potencial de se tornar mais sofisticado a medida em que eles experimentam ataques. Assim ferramentas de *scan* da Internet tem notado cada vez mais ataques de AR-DDoS utilizando o CoAP[22].

Ademais, atacantes felizmente ainda encontram alguns desafios para utilizar servidores CoAP rodando em dispositivos IoT como refletores, dos quais é possível citar:

- **Baixo poder computacional:** Para que ataques de Negação de Serviço Distribuída com Reflexão Amplificada usando dispositivos IoT sejam considerados viáveis, é exigida uma quantidade considerável de refletores e uma coordenação de esforços, dado que as limitações nas capacidades computacionais dos dispositivos fazem com que os mesmos tendam a saturar[7];
- **Transitoriedade:** Estima-se que mais de 80% mudam de endereço a cada duas semanas. Isso pode diminuir a capacidade de abuso dos dispositivos expostos, já que os invasores precisam continuamente fazer *scan* na rede para estabelecer endereços IP a serem usados nos ataques[22].

Mesmo assim, o risco potencial continua significativo. Um relatório recente gerado através de um *scan* da internet pela ferramenta Shodan mostrou que existem aproximadamente 500 mil servidores CoAP respondendo na porta padrão(5683)[27]. A preocupação com tal risco mostra-se então justificada e necessária, dada a alta quantidade de dispositivos IoT vulneráveis e a tendência desse número continuar em crescente ascensão.

Para mitigar ataques de Negação de Serviço Distribuída com Reflexão Amplificada que explorem servidores CoAP como refletores é possível aplicar algumas medidas de segurança. Dentre elas:

- Deixar o servidor em outra porta, para aumentar o custo do atacante ao escanear a rede;
- Limitar o tamanho máximo da resposta usando *Block-Wise Transter*, de forma a atingir uma amplificação menor e ser menos interessante ao atacante como exemplificado na seção 7.4 da RFC7959[11];
- Limitar a taxa de resposta do servidor, de forma a deixar compatível com as necessidades da aplicação em questão, como discutido no capítulo 11 da RFC7252[1];

- Se o hardware permitir, não usar o modo NoSec do CoAP, pois o mesmo não provê nenhum tipo de segurança. E, quando possível, utilizar DTLS.[1] Recomenda-se não deixar hardware, especialmente NoSec, aberto para internet.

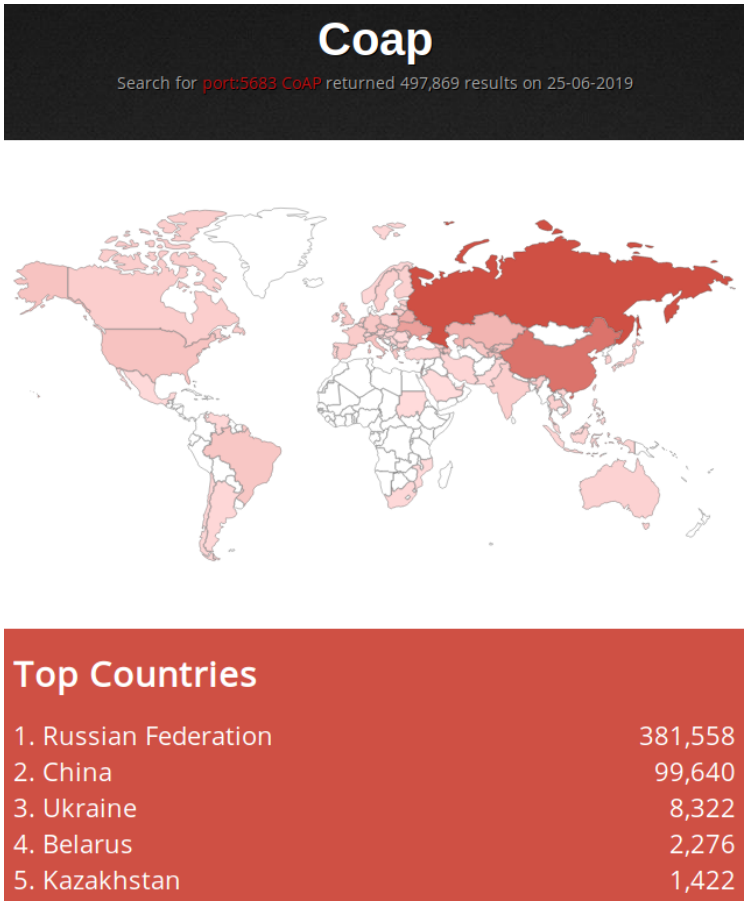

Figura 2.12: Scan feito pela ferramenta Shodan (Fonte: [27]).

Capítulo 3

Extensão do Linderhof

O capítulo atual aborda as contribuições práticas do presente trabalho. Tais contribuições foram agregadas ao *framework* Linderhof, implementado por Miranda[24] em trabalhos anteriores (vide Seção 2.4). No geral, o foco foi evoluir essa ferramenta para que fosse possível simular e validar informações teóricas a respeito da exploração de dispositivos IoT vulneráveis, que se comunicam via CoAP, como refletores de ataques Negação de Serviço Distribuída com Reflexão Amplificada (AR-DDoS).

3.1 *Mirror* CoAP

Uma das contribuições práticas desse trabalho foi a implementação de um novo *mirror* no *Linderhof*, feito para a geração pacotes CoAP. Como explicado na Seção 2.4, *Mirrors* são responsáveis pela geração dos pacotes que serão usado pelos injetores.

Quanto ao pacote enviado pelo *mirror*, seguiu-se a estrutura explicada na Subseção 2.2.1 para gerar o pacote da seguinte forma:

- **Ver:** 1. Foi utilizada a versão 1, que por enquanto é a única versão disponível;
- **T:** CON. Foi utilizado o *Confirmable*, mas esse valor poderia ser o *Non-confirmable* também, o importante nesse caso é ser um tipo de mensagem que requeira a obtenção de uma resposta;
- **TKL:** 0. Como *token* é opcional, TKL recebe o valor 0, de forma a indicar que nenhum *token* será informado;
- **Code:** GET. Indicando que é uma requisição do tipo GET;
- **Message ID:** Aleatório. Valor não é importante como não se tem interesse em manter uma troca de mensagem. E o valor também não precisa ser único;

- **Option delta:** 11(*URI-Path*). Para indicar qual recurso do servidor será requisitado pelo GET;
- **Option length:** Quantidade total de caracteres contém no *URI-Path* informado a seguir;
- **URI-Path:** Caminho para o recurso. Caminho comum pode ser: "*.well-known/core*", que lista todos recursos do servidor[12]. O importante é ter pelo menos 1024 bytes, ou o maior possível. Como o *payload* da resposta será limitada a 1 KiB.
- **2nd Option delta:** 12(BLOCK2). Essa opção é muito importante, ela define o tamanho do *payload* da resposta[11];
- **2nd Option length:** 1. Opção BLOCK2 usa apenas 1 byte para indicar o tamanho;
- **BLOCK2:** 6 (1024). O tamanho do pacote é dado por pela fórmula $2^{(SZX+4)}$, onde SZX é o o valor da opção. Assim, 1024 bytes é representado pelo valor 6 e este é o valor máximo que pode ser usado.[11]

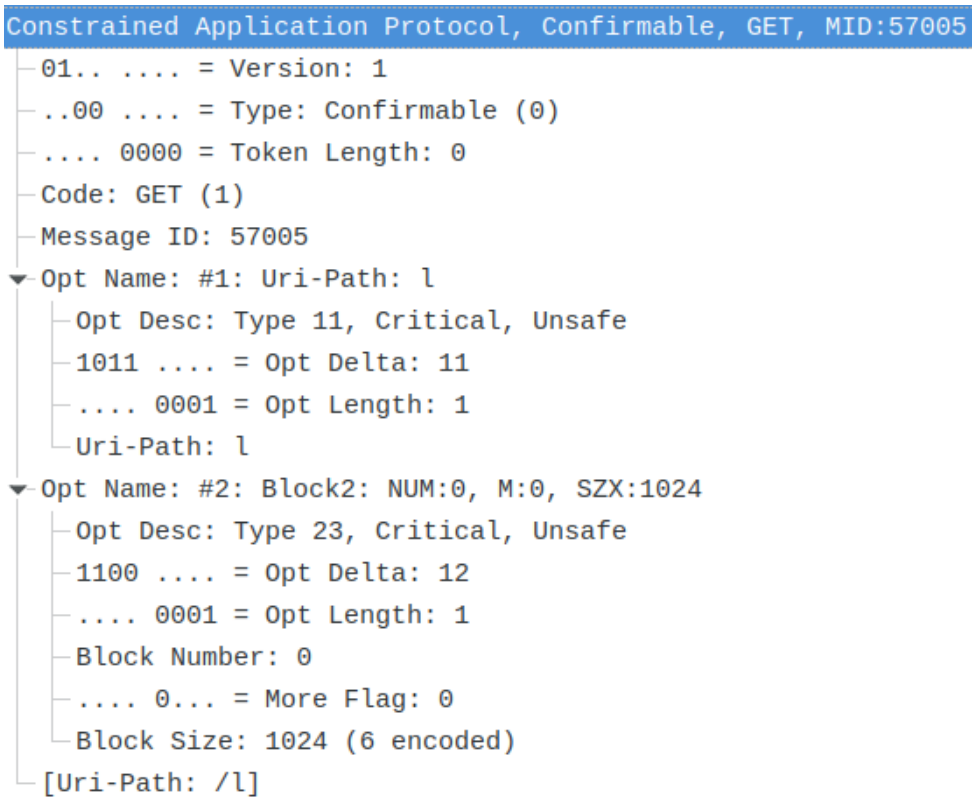

Figura 3.1: Pacote gerado pelo *mirror* CoAP do Linderhof visando aumentar a amplificação, usando a opção *Block2* igual a 6, para o servidor retornar um pacote com 1024 bytes de conteúdo.

A captura de um pacote gerado por esse novo *mirror CoAP* pode ser visualizada na Figura 3.1, que mostra informações detalhadas coletadas pela ferramenta de redes *Wireshark*[28]. A mesma ferramenta foi utilizada para monitorar a interação entre o Linderhof e um servidor CoAP, como é possível visualizar na Figura 3.2

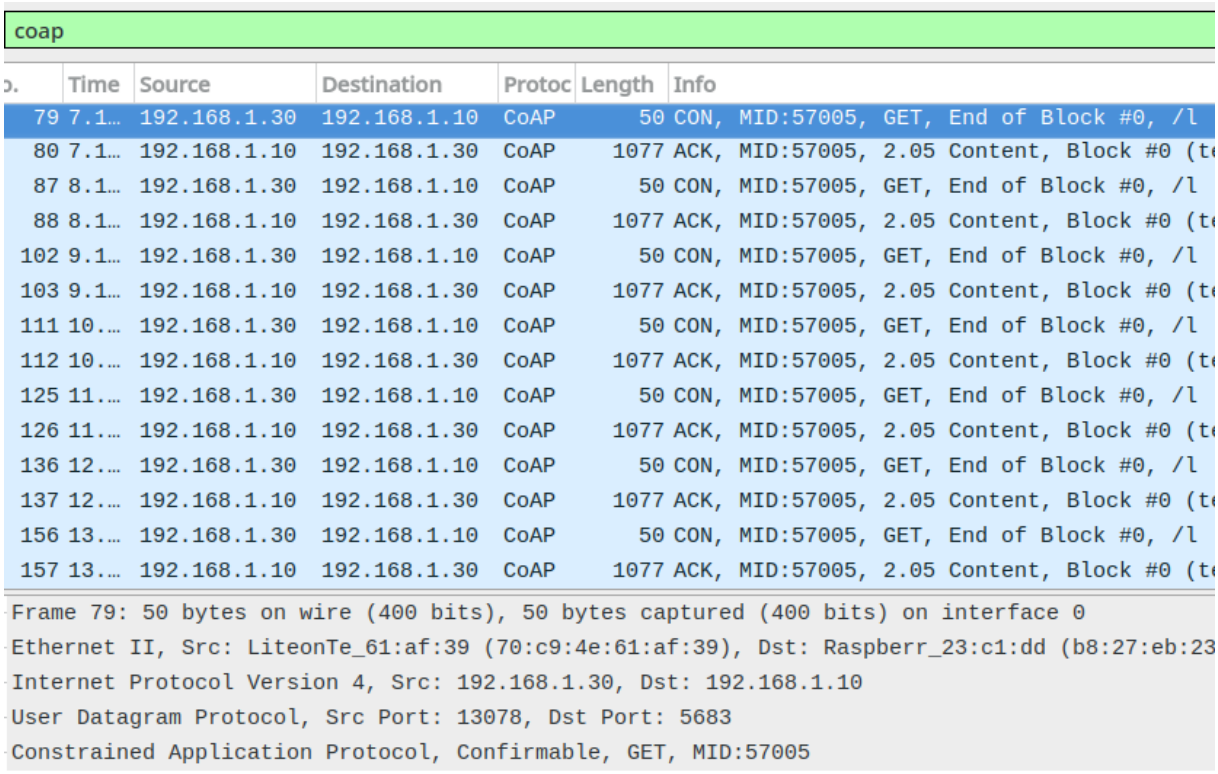

Figura 3.2: Interação entre o Linderhof e um servidor CoAP em nível 1. No caso, o endereço de IP atacante foi selecionado como IP da vítima para poder verificar as repostas do servidor.

3.2 Melhorias adicionais

A proposta inicial de contribuição prática desse trabalho era focada na construção do *mirror* de CoAP para o Linderhof. Mas, dado que o objetivo desse trabalho é focado em explorar o potencial de se utilizar dispositivos IoT em ataques AR-DDoS, decidiu-se que também seria interessante propor incrementos adicionais, como a implementação de ataques de Negação de Serviço Distribuída (DDoS).

Ademais, enquanto essas adições à ferramenta eram implementadas, notou-se a oportunidade de promover melhora no funcionamento de partes já feitas.

Deste modo, as subseções abaixo focam em detalhar demais contribuições práticas feitas por esse trabalho, além da criação de um novo *mirror*.

3.2.1 Implementação ataques de Negação de Serviço Distribuída (DDoS)

O Netuno, responsável por gerar os injetores, foi alterado de forma a suportar injetores que irão enviar pacotes com conteúdos diferentes. Antes de cada uma das *threads* de injeção de pacotes ser instanciadas, são preparados novos pacotes variados dependendo de quem será seu refletor. Cada injetor envia para apenas um servidor.

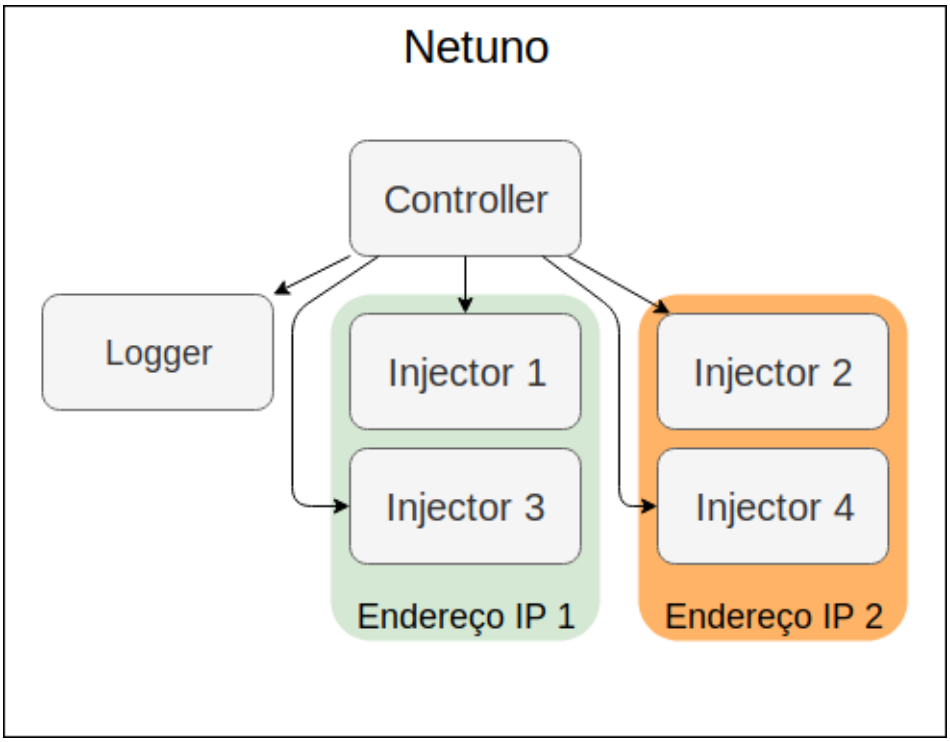

Figura 3.3: Para realizar ataques de Negação de Serviço Distribuída (DDoS) os injetores são divididos em grupos e cada grupo fica responsável por um único IP.

A lista de IPs que será usada no Negação de Serviço Distribuída com o Linderhof é carregada de um arquivo de texto. Então todos os endereços são lidos e são alocados para os injetores através de uma rotação. Por exemplo, usando uma lista com 2 IPs, todos injetores ímpares ficam com o primeiro IP e todos os injetores pares ficam com o segundo IP. Isso é ilustrado na Figura 3.3 que mostra a nova arquitetura do Netuno.

Além disto, para a realização de ataques DDoS o Netuno ajusta os pacotes para atingir os refletores indicados. Assim os pacotes serão distribuídos entre os alvos com uma arquitetura similar a *Fan-out*, como pode ser visto na Figura 3.4.

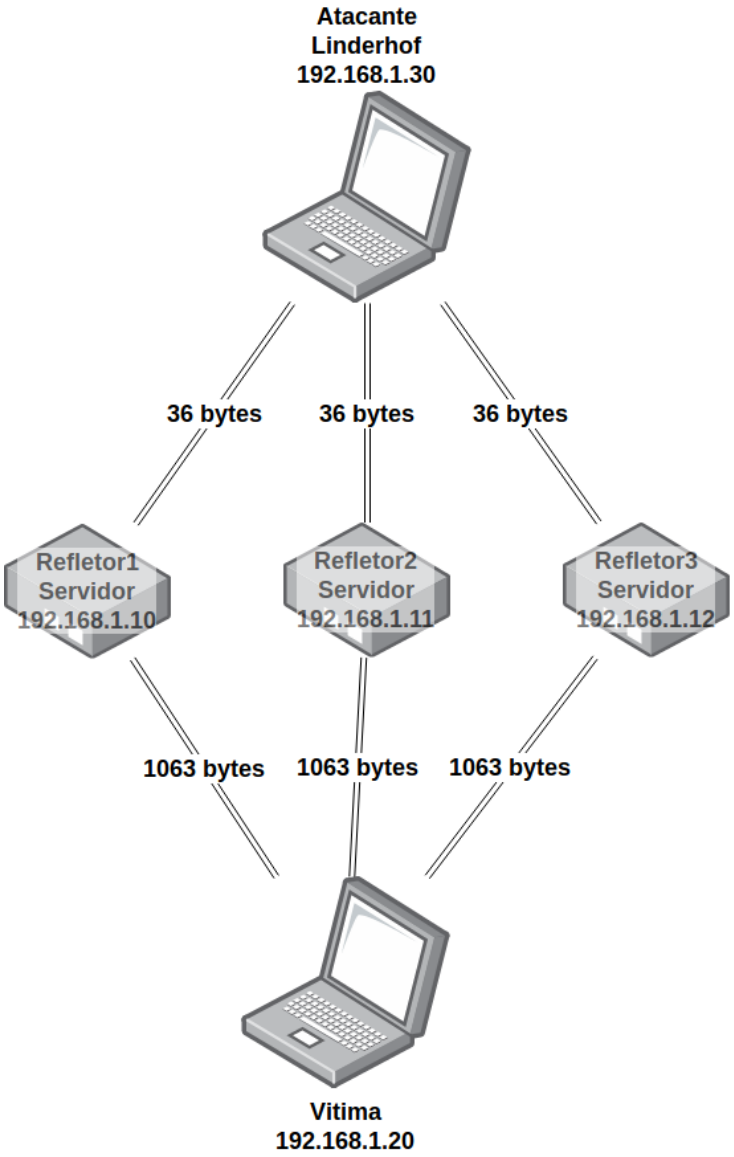

Figura 3.4: Diagrama mostrando um ataque feito utilizando a nova estrutura do Linderhof estendida para a realização de ataques DDoS..

3.2.2 Remoção de espera ocupada (*busy wait*)

Trabalhos anteriores[24] implementaram os injetores do Linderhof como *threads* que funcionam de forma independente entre si e com uma única variável compartilhada entre cada Injetor e Controlador do Netuno. Essa variável é a quantidade de pacotes a ser enviada e a interação funciona da seguinte forma:

- Cada injetor possuía uma variável *bucket*, que continha o valor da quantidade de pacotes que ainda tinham que ser enviados;
- Os injetores esvaziavam esse contador, decrementando cada vez que um pacote era enviado;
- Então, era feita uma checagem se o valor do contador era positivo, para saber se podia enviar mais. Isso era repetido indefinidamente;
- O controlador do Netuno a cada segundo reiniciava todas as variáveis *bucket* com o valor adequado para cada nível de ataque.

Nessa antiga interação, o *bucket* era checado sem nenhum tipo de bloqueio, o que acabava por causar uma espera ocupada (*busy wait*). Para corrigir isso causando o mínimo de detrimento em performance, foi escolhido utilizar semáforos.

Semáforos são variáveis especiais protegidas, que têm como função o controle de acesso a recursos compartilhados em um ambiente *multi-thread*.

Nesse caso, os semáforos adicionados são incrementados (usando a função `sem_post`) sempre que o controlador deseja que os injetores enviem mais um *bucket* de pacotes, fazendo com que essa sincronização seja feita apenas uma vez por segundo. Após essa mudança, o mecanismo passou a funcionar da seguinte forma:

- Cada injetor ainda possui a variável *bucket* indicando a quantidade de pacotes a ser enviados por *batch*;
- Injetores esvaziam o *bucket* enviando todos os pacotes sem checar nenhum tipo semáforo ou *mutex*;
- Ao finalizar esperam no semáforo (chamando a função `sem_wait`);
- No controlador, após ter passado um segundo, é atualizada a variável *maxBucket* caso o nível dos injetores tenha sido alterado. E, então, libera cada *thread*(chamando a função `sem_post`);
- Cada *thread* agora reinicia a própria variável *bucket* com o valor *maxBucket* e repete os envios, até chegar no nível final e as iterações acabarem.

3.3 Considerações finais

A ferramenta Linderhof é bastante útil para a simulação de diversos tipos de ataques de Negação de Serviço, e mostrou-se ser facilmente extensível devido a sua estrutura modular. O presente trabalho estendeu principalmente os módulos Commander e Netuno, com foco principal em adicionar o módulo CoAP à biblioteca de espelhos(*Hall of Mirrors*).

Os incrementos implementados por esse trabalho, principalmente o de possibilitar ataques de Negação de Serviço Distribuída, abrem caminho para mais incrementos futuros interessantes, incluindo a simulação de outros tipos de ataques distribuídos além do ataque de Negação de Serviço Distribuída com Reflexão Amplificada.

De imediato, a extensão para conter um *mirror* de CoAP tornou possível uma análise prática dos conceitos teóricos abordados por esse trabalho, que pode ser vista no capítulo a seguir.

Capítulo 4

Resultados e Discussão

O capítulo atual aborda experimentos realizados utilizando as extensões da ferramenta Linderhof implementadas nesse trabalho para simular um ataque de Negação de Serviço Distribuída com Reflexão Amplificada utilizando servidores CoAP como refletores, bem como os resultados desses experimentos e uma discussão a respeito deles.

Por isso, ele é dividido em cinco seções: a primeira descreve o laboratório criado para os experimentos, a segunda resume a metodologia de testes, a terceira descreve detalhadamente os experimentos, a quarta faz análises dos resultados e a última contém considerações finais.

4.1 Ambiente experimental

Para os testes foram utilizados cinco computadores. Todos esses computadores estavam conectados ao mesmo roteador dedicado ao experimento (que foi utilizado somente por suas capacidades de *switch*/ponto de acesso e não por suas capacidades de roteamento), conforme a rede descrita na Figura 4.1. Em adição, essa rede era isolada, de modo que nenhum outro computador tinha acesso a ela e ela não tinha acesso a Internet. Todas as outras interfaces de rede não utilizadas nos computadores foram desativadas durante o experimento.

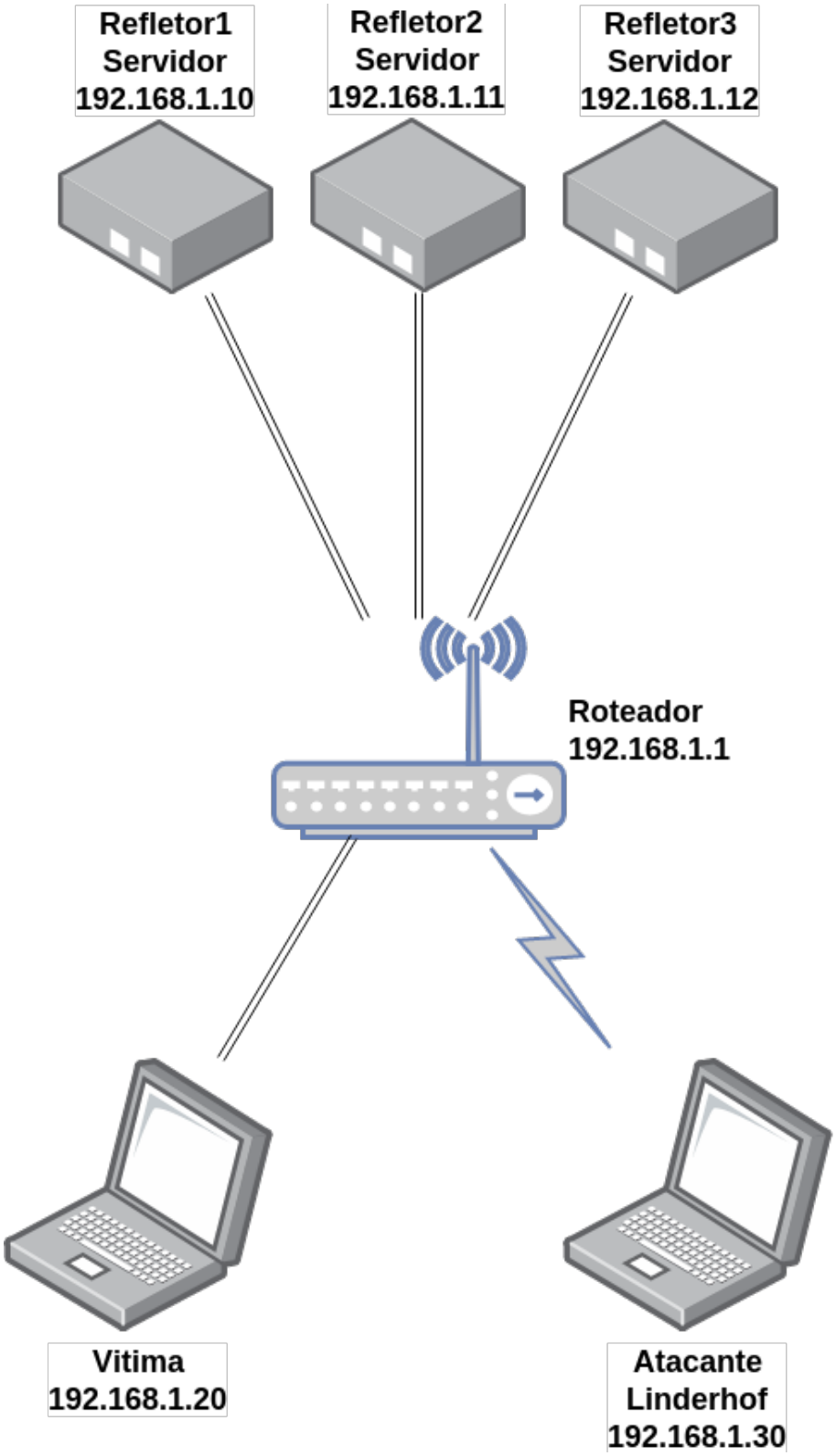

Figura 4.1: Diagrama da rede do ambiente experimental.

4.1.1 Configurações

A seguir, estão detalhadas as informações e configurações técnicas dos cinco computadores utilizados durante os testes, bem como do roteador usado para conectá-los e rotear a comunicação.

Atacante

Para atacante, foi utilizado um Computador Pessoal (PC) com as seguintes configurações:

- **Função:** Executar Linderhof e enviar pacotes gerados pelo *mirror* CoAP para os 3 servidores CoAP que estão atuando como refletores;
- **Processador:** Intel i5-8250U;
- **Placa de rede:** 5Ghz 300Mbps 802.11n;
- **IP:** 192.168.1.30;
- **Memória RAM:** 8GiB DDR4;
- **Sistema Operacional:** Fedora 30;
- **Armazenamento:** SSD.

Refletores

Para refletores, foram utilizados três Raspberry Pi 3 do mesmo modelo. Essa escolha foi feita para que o laboratório de testes simulasse parcialmente a capacidade limitada de dispositivos IoT atuando como refletores. Mais detalhes sobre essas configurações limitadas podem ser vistos a seguir:

- **Função:** Atuar como servidor CoAP para responder a requisição gerada pelo atacante.
- **Processador:** Broadcom BCM2837 Quad Core 1.2GHz 64bit;
- **Placa de rede:** 100 Mbps;
- **IP:** 192.168.1.[10,11,12];
- **Memória RAM:** 1GiB LPDDR2;
- **Sistema Operacional:** Raspbian Buster Lite
- **Armazenamento:** *SD-Card Class 10.*

Vítima

Para vítima, também foi utilizado um Computador Pessoal (PC). Esse computador possui configurações mais robustas do que o computador do atacante, pois em situações reais a vítima costuma possuir configurações superiores ao atacante. Tais configurações são detalhadas a seguir:

- **Função:** Apenas Monitorar o fluxo de rede;
- **Processador:** Intel i7-4710HQ;
- **Placa de rede:** 1 Gbps;
- **IP:** 192.168.1.20;
- **Memória RAM:** 16GiB DDR4;
- **Sistema Operacional:** Fedora 30;
- **Armazenamento:** SSD.

Switch

Para criação da rede, foi utilizado um roteador com interface de rede *ethernet* Gigabit, de forma que ele não fosse o componente a causar gargalo durante o ataque. As configurações desse roteador podem ser vistas a seguir:

- **Função:** Atuar como *switch*/ponto de acesso interconectando os nós da rede e não utilizando nenhuma funcionalidade de roteamento;
- **Modelo:** TP-Link n600 TL-WDR3600;
- **Placa de rede:** 4x1Gbps;
- **Placa de WiFi:** 5Ghz 300Mbps 802.11n;
- **IP:** 192.168.1.1.

4.2 Metodologia de testes

Com os dispositivos corretamente conectados compondo o laboratório de testes, o experimento principal consistiu em executar no atacante o Linderhof, responsável por enviar os pacotes para os 3 refletores. Esses 3 refletores, por sua vez, estavam executando servidores CoAP que respondiam as requisições de mensagem recebidas para o endereço falsificado da vítima, como pode ser visto na Figura 3.4.

Tal execução de experimento simulando um Negação de Serviço Distribuída com Reflexão Amplificada (AR-DDoS) usando o Linderhof demandou passar para ele os seguintes parâmetros:

- **-m (*mirror*):** coap. Com esse parâmetro é definido que o *Mirror* CoAP será utilizado para gerar o pacote de que será utilizado durante o ataque pelo Injetores;
- **-t (*target*):** 192.168.1.20. informa o IP da vítima do ataque que será usado no gerador de pacotes *blacksmith* como IP de origem forjado;
- **-l (*level*):** 1. O ataque será iniciado no nível 1, isto é, enviando 1 pacote por segundo;
- **-i (*increment*):** 10. Assim a cada 10 segundos o nível do ataque é incrementado para o nível seguinte;
- **-c (*counter*):** 70. Tempo máximo de ataque será de 70 segundos, o que com a configuração supracitada levará o ataque do nível 1 até o nível 7, que é o nível máximo que o atacante conseguiu atingir.

Vale lembrar que o nível dita a quantidade de pacotes sendo enviada por segundo. Um nível 1 significa, então, o envio de 1 pacote por segundo, enquanto um nível 7 significa o envio de 1 milhão de pacotes por segundo. Maiores explicações sobre níveis podem ser revistas na Seção 2.4.4.

Durante o experimento todos os computadores capturaram os pacotes usando *dumpcap*, que é o módulo de captura de pacotes do *Wireshark*. Para ter um menor impacto no uso de CPU durante o experimento foi apenas realizada a captura de pacotes, sem nenhum tipo de análise. E para minimizar o impacto do uso da unidade de armazenamento apenas os primeiros 50 *bytes* de cada pacote foram salvos, o que é dado suficiente para analisar o os cabeçalhos IP, UDP e CoAP. Para tal, foi usado o seguinte comando:

```
$ dumpcap -w captura.pcapng -B 200 -i eth0 -s 50
```

Depois da captura de três rodadas de experimentos todas a capturas foram filtradas para deixar apenas os pacotes que fossem CoAP, de forma a deixar os resultados mais claros e também as capturas no refletores foram separadas entre entrada (vindo do atacante) e saída (com destino à vítima). A filtragem em questão foi realizada com o filtros do *tshark*, que é o módulo para terminal do *Wireshark*:

```
1 $ tshark -r captura.pcapng -Y 'coap' -w coap.pcapng
2 $ tshark -r coap.pcapng -Y 'ip.src==192.168.1.20' -w coap_i.pcapng
3 $ tshark -r coap.pcapng -Y 'ip.dst==192.168.1.20' -w coap_o.pcapng
```

Com todas as capturas já filtradas, o *tshark* foi utilizado novamente para gerar estatísticas de entrada e saída da quantidade total de pacotes e dados em *bytes* por segundo. O resultado de todas essas análises foi utilizado para gerar um único arquivo csv com todos os dados.

```
$ tshark -r coap.pcapng -z io,stat,1 > stat.txt
```

Por fim, o arquivo csv gerado foi importado no *Excel* para análise de dados. Durante essa análise, os dados precisaram ser sincronizados para ter uma correlação correta entre segundos gerados pelo *tshark* e o segundos dos *Linderhof* que indicam o nível de ataque atual.

Com os dados sincronizados foi feita a média da quantidade de pacotes dos dez valores que foram capturados em cada nível. Para essa média o primeiro e o último valor não são incluídos para evitar valores transitórios. E, então, foi calculado o volume de dados a partir da quantidade de pacotes, no qual foram multiplicados pelos valores dos cabeçalhos IP, UDP e conteúdo CoAP. Com esses valores finais foi possível calcular os fatores de amplificação em cada nível.

4.3 Experimentos

A seguir, serão expostos dados coletados durante os experimentos feitos de acordo com a metodologia de testes explicada na seção acima. Junto a esses dados constam também informações do porquê esses dados foram coletados e o que eles significam, bem como uma justificativa a respeito de alguns dados experimentais divergirem do previsto na teoria.

4.3.1 Resultados gerais

Pacote e dados por nível de ataque

A Tabela 4.1 mostra, para cada nível, quantidade média de pacotes/bytes enviados por segundo pelo atacante, de acordo com coleta de dados experimental. Nota-se que até o nível 6 os valores coletados são condizentes com o previsto teoricamente (considerando que cada pacote tem tamanho fixo de 36 bytes). Já o nível 7 tem quantidade menor do que esperado, o que pode ser explicado por uma possível saturação nas capacidades de processamento da máquina atacante.

Atacate		
Nível de ataque	**Pacotes enviados**	**Bytes enviados**
1	1	36
2	10	360
3	100	3600
4	1000	36000
5	10000	360000
6	100000	3600000
7	553576	19928735

Tabela 4.1: Dados coletados experimentalmente a respeito da quantidade média de pacotes e bytes enviados pelo atacante por segundo, de acordo com cada nível de ataque.

De modo semelhante, a Tabela 4.2 mostra, para cada nível, quantidade média da soma de pacotes/bytes enviados por segundo por todos os refletores, de acordo com coleta de dados experimental. Nota-se que até o nível 3 os valores coletados são condizentes com o previsto teoricamente (considerando que cada pacote recebido tem tamanho fixo de 36 bytes e cada pacote enviado tem tamanho fixo de 1063 bytes). A partir do nível 4 nota-se que os refletores passam a enviar menos pacotes que recebem, o que pode ser explicado por uma possível saturação causada por limitações de banda nos Raspberry Pi 3. A partir do nível 5, nota-se que estão chegando menos pacotes que o previsto, o que pode ser explicado por uma possível saturação causada por limitações na comunicação entre o atacante e os refletores, seja pelo adaptador de rede com menor capacidade do refletor ou pelo atacante estar conectado via *wi-fi*.

Refletores				
Nível de ataque	**Pacotes recebidos**	**Bytes recebidos**	**Pacotes enviados**	**Bytes enviados**
1	1	36	1	1063
2	10	360	10	10551
3	100	3600	100	106300
4	1000	36000	842	895356
5	9957	358464	6633	7051034
6	18103	651693	12362	13140319
7	47824	1721646	33899	36034327

Tabela 4.2: Dados coletados experimentalmente a respeito da quantidade média da soma dos pacotes e bytes recebidos e enviados por todos os refletores por segundo, de acordo com cada nível de ataque.

Ademais, a Tabela 4.3 mostra, para cada nível, a quantidade média de bytes enviados por segundo por cada refletor, de acordo com coleta de dados experimental. Nota-se que a divisão de pacotes foi feita de acordo com a quantidade de pacotes por segundo, de acordo

com uma distribuição Round Robin. Por exemplo, caso fosse exigido 1 pacote por segundo, apenas o primeiro refletor trabalharia. Já caso fosse exigido 2 pacotes por segundo, os refletores 1 e 2 trabalhariam, enviando 1 pacote por segundo cada. Já caso fossem 4 pacotes por segundo, o refletor 1 trabalharia enviando 2 pacotes por segundo enquanto os refletores 2 e 3 enviariam 1 por segundo. E a lógica de distribuir até a quantidade desejada se aplica para N pacotes. Quanto a comparação de projeções teóricas em relação aos resultados coletados, especula-se que as mesmas saturações que podem justificar os resultados no parágrafo acima sobre todos os refletores se aplicam para a análise individual de cada refletor.

	Refletor 1		**Refletor 2**		**Refletor 3**	
Nível	**Entrada**	**Saida**	**Entrada**	**Saida**	**Entrada**	**Saida**
1	36	1063	0	0	0	0
2	144	4252	108	3150	108	3150
3	1440	42520	1080	31890	1080	31890
4	14400	330947	10800	245642	10800	318767
5	142536	2584382	107964	2258225	107964	2208427
6	218845	4381095	214632	4391105	218216	4368118
7	573924	12011147	574053	12024213	573669	11998967

Tabela 4.3: Quantidade de dados recebidas e enviadas em bytes por segundo, separada por refletor em cada nível de ataque.

Por fim, a Tabela 4.4 mostra, para cada nível, quantidade média de pacotes/bytes recebidos a cada segundo pela vítima, de acordo com coleta de dados experimental. Observa-se que até o nível 3 os valores coletados são condizentes com o previsto teoricamente (considerando que os refletores distribuem entre si o envio dos pacotes e que a mensagem vinda dos refletores tem tamanho fixo de 1063 bytes). A partir do nível 4, especula-se saturações diversas da rede e dos componentes físicos envolvidos, que podem explicar os valores de pacotes recebidos serem um pouco menores do que os esperados teoricamente.

Vítima		
Nível de ataque	**Pacotes recebidos**	**Bytes recebidos**
1	1	1063
2	10	10551
3	100	106300
4	806	857258
5	6631	7048546
6	12314	13090203
7	33855	35987688

Tabela 4.4: Dados coletados experimentalmente a respeito da quantidade média de pacotes e bytes recebidos de fato pela vítima por segundo, de acordo com cada nível de ataque.

4.3.2 Fatores de amplificação práticos

Esses fatores de amplificação foram calculados com a razão entre quantidade de dados ou pacotes enviados e recebidos. E amplificação interna é apenas dentro do refletor usando apenas pacotes que realmente chegaram e saíram do refletor. Já a Amplificação Final é calculada com base nas pontas, a quantidade que saiu do atacante e a quantidade de dados que realmente chegou à vítima.

A Tabela 4.5 mostra a amplificação dada a quantidade de pacotes. O valor máximo é 1, dada a característica do CoAP de responder só um pacote por cada requisição. Pode-se notar que esse valor foi diminuindo conforme o nível de ataque aumentava, o que pode ser explicado por uma possível saturação causada tanto pela limitação dos refletores quanto pela limitação na comunicação.

Amplificação por quantidade de pacotes		
Nível de ataque	**Amplificação interna**	**Amplificação Final**
1	1.00	1.00
2	0.99	0.99
3	1.00	1.00
4	0.84	0.81
5	0.67	0.66
6	0.68	0.12
7	0.71	0.06

Tabela 4.5: Fator de amplificação por quantidade de pacotes, variando de acordo com os níveis, coletado experimentalmente.

Já a Tabela 4.6 mostra a amplificação dado o tamanho dos pacotes. Uma análise mais específica será provida na Seção 4.4. Mas, em resumo, a amplificação final máxima

ficou compatível com os valores esperados teoricamente e esse valor caiu conforme o nível aumentou. Especula-se que isso foi causado por saturações na rede.

Fator de amplificação por tamanho dos pacotes		
Nível de ataque	**Amplificação interna**	**Amplificação Final**
1	29.53	29.53
2	29.31	29.28
3	29.53	29.53
4	24.87	23.81
5	19.67	19.58
6	20.16	3.64
7	20.93	1.81

Tabela 4.6: Fator de amplificação por tamanho dos pacotes, variando de acordo com os níveis, coletado experimentalmente.

4.3.3 Melhorias no uso de CPU do *Lindehof*

Uso de CPU		
Nível de ataque	**Antes**	**Depois**
1	100.00%	0.00%
2	100.00%	0.01%
3	100.00%	0.05%
4	100.00%	0.33%
5	100.00%	2.65%
6	100.00%	21.58%
7	100.00%	100.00%

Tabela 4.7: Tabela mostrando os porcentuais de uso da CPU no computador atacante durante o ataque, comparando o antes e o depois da melhoria de remoção de *busy wait* feita por esse trabalho.

Com a remoção da espera ocupada no Linderhof (que antes fazia 100% dos recursos computacionais fossem usados), é possível perceber que agora o uso da CPU no computador atacante é proporcional ao nível do ataque (ou seja, proporcional a quantidade de pacotes sendo enviada). O impacto dessa mudança pode ser evidenciado na Tabela 4.7 e na Figura 4.2.

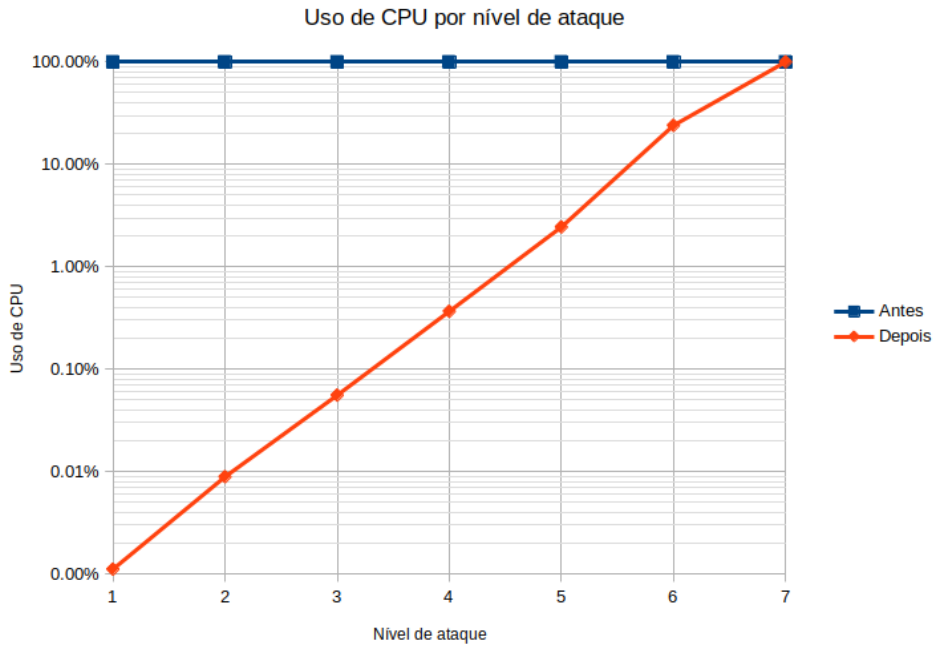

Figura 4.2: Gráfico logaritmo mostrando os porcentuais de uso da CPU no computador atacante pelo nível do ataque em si, comparando o antes e o depois da melhoria de remoção de *busy wait* feita por esse trabalho.

Com o uso de CPU no nível 1, enviando apenas um pacote por segundo é possível verificar que não há mais nenhuma espera ocupada no caso executado. A única que havia era no módulo Netuno para a sincronização entre o Controlador e os Injetores, para o controle de fluxo.

4.4 Análises

A seguir, serão feitas análises gerais de pontos principais a serem discutidos a respeitos dos experimentos e dos resultados. Tais pontos vem com o objetivo de complementar o que já foi exposto de informação na seção acima.

4.4.1 Volume de pacotes

Como já mencionado anteriormente, cada pacote de requisição retorna apenas um pacote de resposta. Isso pode ser visto na Figura 4.3 e na captura de pacote feita pelo *Wireshark* (vide Figura 3.2). Logo, é possível visualizar um possível princípio de saturação no nível 4 e uma possível saturação significativa no nível 6.

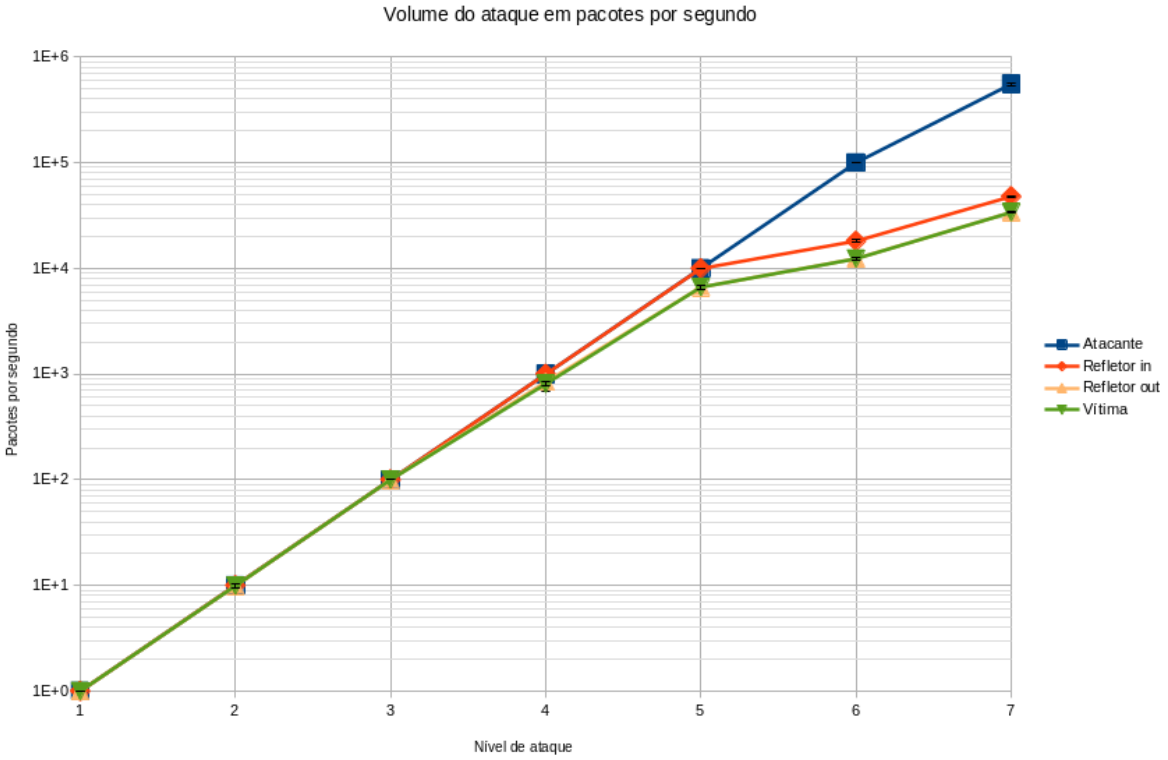

Figura 4.3: Gráfico feito com dados experimentais coletados a respeito do volume de pacotes enviados por segundo variando com os níveis de ataque em si.

Com a Figura 4.5 é possível visualizar que a vítima chegou próximo a 300Mbps de volume de dados recebidos, que é o máximo que os refletores são capazes de enviar, como são 3 e cada um possui controlador de rede de 100 Mbps. E com a Figura 4.4 é possível verificar como o fluxo foi separado entre os 3 refletores, e que no primeiro nível apenas o refletor 1 foi utilizado.

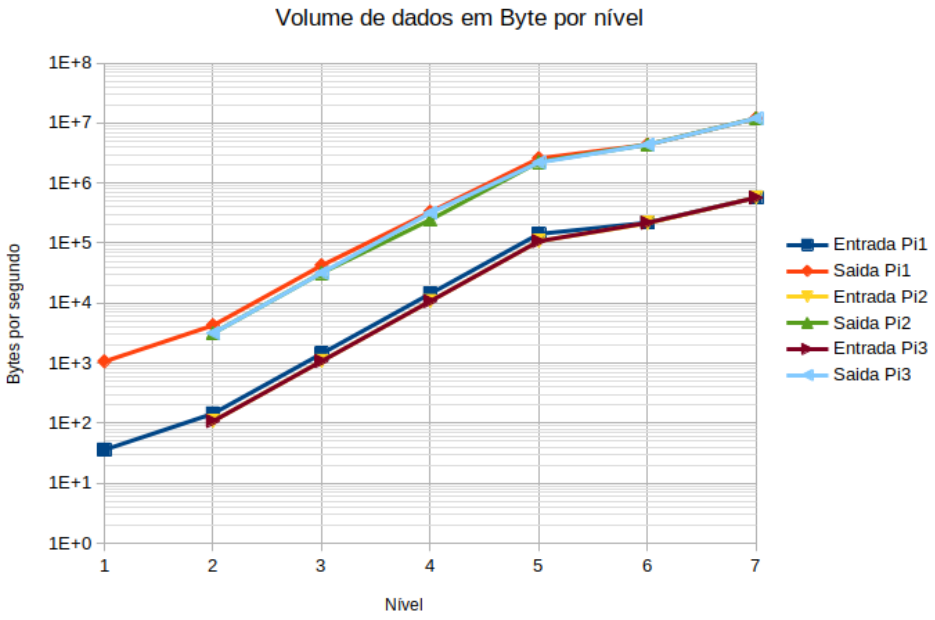

Figura 4.4: Volume do ataque em bytes separado entre cada refletor, onde é possível verificar como foi separado o fluxo entre os refletores.

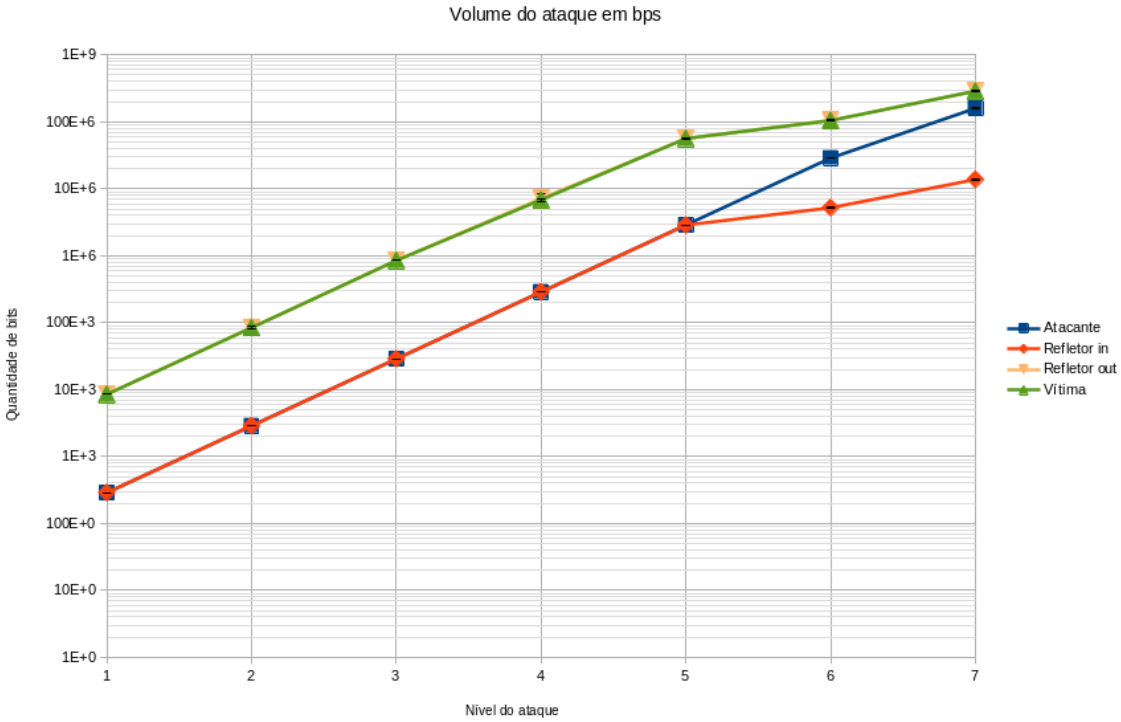

Figura 4.5: Gráfico feito com dados experimentais coletados a respeito do volume do ataque em bits enviados por segundo variando com os níveis de ataque em si.

4.4.2 Amplificação

De acordo com a Equação 2.1, o fator de amplificação esperado foi calculado com os seguintes valores e da seguinte forma:

$$\begin{aligned} \text{Fator de amplificação} &= \frac{\text{tamanho(resposta)}}{\text{tamanho(requisição)}} \\ &= \frac{\text{tamanho}(IP + UDP + CoAP_{resposta})}{\text{tamanho}(IP + UDP + CoAP_{requisição})} \\ &= \frac{20 + 8 + 1035}{20 + 8 + 8} = \frac{1063}{36} \\ &\approx 29.5 \end{aligned} \tag{4.1}$$

Esse comportamento pode ser observado na prática pela Figura 4.6 enquanto não há nenhum outro limitante. A partir do nível 4, o refletor deixa de ter a saturação teórica e a partir do nível 6 há uma queda maior chegando a um fator de amplificação mínimo de valor 1,8.

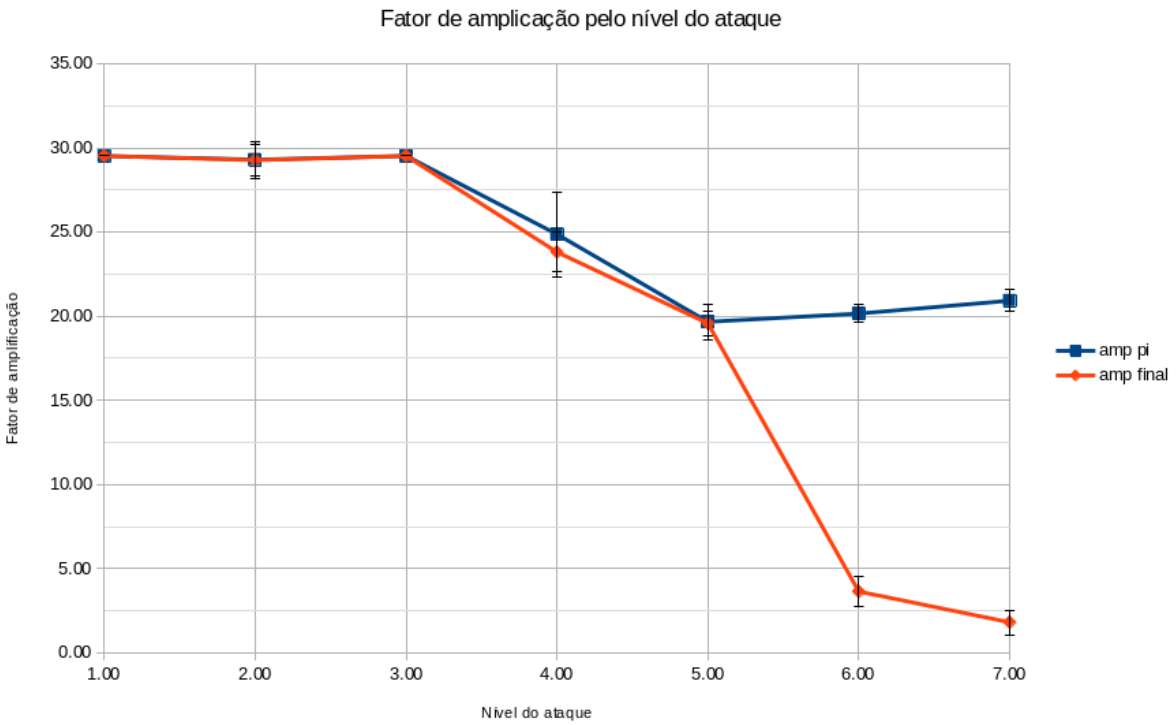

Figura 4.6: Gráfico feito com dados experimentais coletados e com os cálculos supracitados de fatores de amplificação. Ele mostra só fatores de amplificação pelo nível de ataque, tanto para as variações de amplificações internas (em azul) quanto para as variações amplificações finais (em vermelho).

Observa-se que no final ainda há uma amplificação residual, mantendo uma amplificação interna de 20 após uma especulada saturação de rede, devido ao fato que os refletores possuem placa de rede com menor capacidade (como visto na Subseção 4.1.1).

Por fim, nota-se também que no nível de ataque final houve uma saturação do próprio atacante, como pode ser visto pelo uso de CPU na Figura 4.2. E, por isso, foi esse o nível final analisado, já que partir daí a quantidade de pacotes enviados se mantém constante em aproximadamente 554 mil pacotes por segundo.

E, mesmo atingindo amplificações menores ou taxas de envios menores no refletor o atacante poderia apenas usar mais dispositivos para compensar essa diferença. [7, 21, 27]

4.5 Considerações finais

O experimentos demonstraram com sucesso um ataque de Negação de Serviço Distribuída com Reflexão Amplificada utilizando servidores CoAP configurados em dispositivos restritos (Raspberry Pi 3) atuando como refletores.

Além disso, foi interessante fazer uma comparação das projeções teóricas com uma simulação real. Em ambientes reais as limitações impostas por restrições na rede e/ou nos dispositivos envolvidos ficam muito mais evidentes.

No geral, pode-se considerar que os resultados experimentais obtidos foram satisfatórios tanto com o que era esperado dadas as análises teóricas quanto condizentes com a contribuição prática a qual esse trabalho se propôs a fazer.

Capítulo 5

Conclusão e Trabalhos Futuros

5.1 Conclusão

O presente trabalho de graduação fez uma análise sobre o cenário atual e uma projeção de um cenário futuro a respeito da Internet das Coisas (IoT), sob o ponto de vista de vulnerabilidades e riscos de segurança. Nota-se que ele foi bem sucedido no seu objetivo principal, pois mostrou de forma teórica e de forma prática os riscos de se expor na rede uma grande quantidade de dispositivos IoT que utilizam o CoAP.

Ressalta-se, então, tanto uma contribuição teórica (que aborda riscos e propõe *insights* sobre um ramo que está em alta), quanto contribuições práticas que tanto ajudaram a simular as informações teóricas no contexto desse trabalho quanto irão ser usadas em outros contextos e outros trabalhos (dada que o *framework* Linderhof é uma ferramenta com código *open source*).

A respeito do contexto teórico, projeta-se que vantagens e facilidades promovidas pela Internet das Coisas nas mais diversas áreas e contextos farão com que sua adesão seja cada vez maior. E, dada a grande quantidade de dispositivos IoT necessários para coletar informações e manter tantos objetos e pessoas conectados, destaca-se a existência de limitações físicas de tamanho e limitações econômicas de custos na fabricação e criação desses dispositivos.

Nesse contexto, observou-se que o foco de fabricantes do ramo tem sido em leveza e simplicidade. Infelizmente, como em vários outros ramos, a preocupação com segurança tem sido negligenciada. É por esse motivo que foi foco principal desse trabalho abordar os riscos de entidades maliciosas se aproveitarem para fazerem ataques de Negação de Serviço Distribuída cada vez mais massivos e impactantes. Tais ataques mesmo sendo ataque de natureza mais antiga, com surgimento quase junto ao da internet, continuam extremamente presentes.

Ademais, esse trabalho demonstrou na prática que é possível se utilizar de refletores com CoAP e de capacidades computacionais limitadas para promover ataques Negação de Serviço Distribuída com Reflexão Amplificada efetivos. Tal simulação foi possível devido a incrementos feitos na ferramenta Linderhof, que incluem a adição de um *mirror* CoAP e um mecanismo para a execução de tais ataques de Negação de Serviço Distribuída (DDoS).

Felizmente, o protocolo CoAP em específico já prevê mecanismos de segurança. Além do mais, fazer um grande ataque de AR-DDoS demanda grande coordenação e preparação. Isso aliado com a transitoriedade de IPs em servidores CoAP por enquanto têm prevenido esse tipo de ataque de ser mais amplamente explorado.

Em resumo, o risco da exploração de servidores CoAP como refletores de ataques de Negação de Serviço amplificados é real. Todavia, ressalta-se que existem maneiras de lidar com isso, que irão requerer maiores esforços dos fabricantes de dispositivos IoT, maiores esforços no gerenciamento dos dispositivos e aprimoramentos de *software* IoT no geral. Talvez um passo interessante que possa ser tomado rumo a esse cenário é aumentar a conscientização a respeito do porquê devemos ter segurança sempre como uma preocupação.

5.2 Trabalhos Futuros

Propõe-se trabalhos futuros tanto relacionados ao risco da exploração de servidores CoAP como refletores de ataques de Negação de Serviço amplificados quanto relacionados com a ferramenta Linderhof:

- **Mecanismo de *scan* de servidores CoAP:** como mencionado na Seção 2.5, servidores CoAP tem natureza transitória, o que significa que é comum trocarem de IP ao longo do tempo. Um trabalho futuro interessante seria fazer um mecanismo capaz de escanear parte da rede a procura dos IPs desses servidores, dado características peculiares à implementação do protocolo (como a existência da URL `/.well-known/core`), e verificar com qual recurso seria possível obter a maior amplificação;
- **Implementações de outros ataques de Negação de Serviço Distribuída:** agora que existe o mecanismo que é capaz de fazer ataque distribuído integrado ao *framework* Linderhof, é possível usá-lo como base para a implementação de outros tipos de ataques distribuídos (inclusive os mencionados e categorizados nas definições da Seção 2.3);

- **Melhora de balanceamento entre injetores para ataques de Negação de Serviço Distribuída:** Como mostrado na Figura 3.4, cada injetor só recebe um único IP de refletor, o que causa desbalanceamento. Seria pertinente, então, fazer um mecanismo mais flexível que levasse em conta as capacidades do ambiente e os refletores disponíveis para balancear essa injeção de pacotes entre os refletores;
- **Reestruturação dos ataques de Negação de Serviço Distribuída para comportarem organização de *Botnet*:** Como mostrado na figura Figura 2.9 e explanado na Subseção 2.3.2, a existência de uma organização *Botnet* na qual há uma camada de escravos (*slaves* ou *bots*) comandada por um mestre atacante é vantajosa tanto para o aumento da escala do ataque quanto para fornecer maior ofuscação do atacante propriamente dito.

Referências

[1] Shelby, Z., K. Hartke e C. Bormann: *The constrained application protocol (coap).* Rfc 7252, RFC Editor, June 2014. `http://www.rfc-editor.org/rfc/rfc7252.txt`, `http://www.rfc-editor.org/rfc/rfc7252.txt`. xi, 2, 3, 8, 9, 10, 11, 23, 24

[2] Gubbi, Jayavardhana, Rajkumar Buyya, Slaven Marusic e Marimuthu Palaniswami: *Internet of things (iot): A vision, architectural elements, and future directions.* Future generation computer systems, 29(7):1645–1660, 2013. 1, 5

[3] Atzori, Luigi, Antonio Iera e Giacomo Morabito: *The internet of things: A survey.* Computer networks, 54(15):2787–2805, 2010. 2, 6, 7

[4] Sehgal, Anuj, Vladislav Perelman, Siarhei Kuryla e Jurgen Schonwalder: *Management of resource constrained devices in the internet of things.* IEEE Communications Magazine, 50(12):144–149, 2012. 2

[5] *Mqtt and coap: Security and privacy issues in iot and iiot communication protocols.* `https://www.trendmicro.com/vinfo/hk-en/security/news/internet-of-things/mqtt-and-coap-security-and-privacy-issues-in-iot-and-iiot-communication-protocols` 2, 3, 8

[6] Paxson, Vern: *An analysis of using reflectors for distributed denial-of-service attacks.* ACM SIGCOMM Computer Communication Review, 31(3):38–47, 2001. 2, 3

[7] Pacheco, Luis Alberto B, Joao JC Gondim, Priscila A Solis Barreto e Eduardo Alchieri: *Evaluation of distributed denial of service threat in the internet of things.* Em *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)*, páginas 89–92. IEEE, 2016. 2, 23, 46

[8] Lipson, Howard F: *Tracking and tracing cyber-attacks: Technical challenges and global policy issues.* Relatório Técnico, CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 2002. 3

[9] Ashton, Kevin *et al.*: *That 'internet of things' thing.* RFID journal, 22(7):97–114, 2009. 5

[10] *Iot: number of connected devices worldwide 2012-2025.* `https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/`. 5, 6

[11] Bormann, C. e Z. Shelby: *Block-wise transfers in the constrained application protocol (coap).* Rfc 7959, RFC Editor, August 2016. 11, 23, 26

[12] Shelby, Z.: *Constrained restful environments (core) link format.* Rfc 6690, RFC Editor, August 2012. `http://www.rfc-editor.org/rfc/rfc6690.txt`, `http://www.rfc-editor.org/rfc/rfc6690.txt`. 11, 26

[13] Ishaq, Isam, David Carels, Girum Teklemariam, Jeroen Hoebeke, Floris Abeele, Eli Poorter, Ingrid Moerman e Piet Demeester: *Ietf standardization in the field of the internet of things (iot): a survey.* Journal of Sensor and Actuator Networks, 2(2):235–287, 2013. 12

[14] Gligor, V. D.: *A note on denial-of-service in operating systems.* IEEE Transactions on Software Engineering, SE-10(3):320–324, May 1984, ISSN 0098-5589. 12

[15] Needham, Roger M: *Denial of service: an example.* Communications of the ACM, 37(11):42–46, 1994. 12

[16] Specht, Stephen e Ruby Lee: *Distributed denial of service: Taxonomies of attacks, tools, and countermeasures.* páginas 543–550, janeiro 2004. 13, 16, 17

[17] Costa Gondim, João e de Oliveira Albuquerque, Robson: *Mirror saturation in amplified reflection ddos.* JNIC, 2019. 14, 15, 17

[18] Bawany, Narmeen Zakaria, Jawwad A Shamsi e Khaled Salah: *Ddos attack detection and mitigation using sdn: methods, practices, and solutions.* Arabian Journal for Science and Engineering, 42(2):425–441, 2017. 16

[19] Lau, Felix, Stuart H Rubin, Michael H Smith e Ljiljana Trajkovic: *Distributed denial of service attacks.* Em *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics.'cybernetics evolving to systems, humans, organizations, and their complex interactions'(cat. no. 0*, volume 3, páginas 2275–2280. IEEE, 2000. 16

[20] Costa Gondim, João, Robson de Oliveira Albuquerque, Anderson Clayton Alves Nascimento, Luis García Villalba e Tai Hoon Kim: *A methodological approach for assessing amplified reflection distributed denial of service on the internet of things.* Sensors, 16(11):1855, 2016. 16, 17, 18

[21] Cimpanu, Catalin: *The coap protocol is the next big thing for ddos attacks*, Jan 2019. `https://www.zdnet.com/article/the-coap-protocol-is-the-next-big-thing-for-ddos-attacks/`. 18, 23, 46

[22] *Coap attacks in the wild.* `https://www.netscout.com/blog/asert/coap-attacks-wild`. 18, 23

[23] *Memcached servers abused for massive amplification ddos attacks*, Feb 2018. `https://thehackernews.com/2018/02/memcached-amplification-ddos.html`. 19

[24] Miranda, Igor F.: *Ataque de negação de serviço por reflexão amplificada explorando memcached*, 2019. 19, 20, 25, 29

[25] Souza Vieira, Alexander André de: *Ataque distribuído de negação de serviço por reflexão amplificada usando network time protocol*, 2019. 19

[26] /@FrankSEC42: *What is coap and is it the next ddos for iot - nsc42*, Dec 2018. https://medium.com/nsc42/what-is-coap-and-is-it-the-next-ddos-for-iot-de8ee97e57e6. 23

[27] *Coap search report*, 2019. https://www.shodan.io/report/WSmfpmde. 23, 24, 46

[28] Lamping, Ulf e Ed Warnicke: *Wireshark user's guide*. Interface, 4(6), 2004. 27