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

# Dambuster: uma ferramenta para avaliação de soluções de mitigação de ataques de flooding

Eduardo Sousa da Silva

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

Orientador
Prof. Dr. João J. C. Gondim

Brasília
2022

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

# Dambuster: uma ferramenta para avaliação de soluções de mitigação de ataques de flooding

Eduardo Sousa da Silva

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

Prof. Dr. João J. C. Gondim (Orientador)
CIC/UnB

Prof. Dr. Robson de O. Albuquerque
UnB

Prof.a Dr.a Aletéia P. F. Araújo
UnB

Prof. Dr. João J. C. Gondim
Coordenador do Curso de Engenharia da Computação

Brasília, 3 de maio de 2022

# Dedicatória

Dedico este trabalho primeiramente a Deus, que me deu a capacidade de concluí-lo, aos meus pais que sempre me apoiaram para que eu pudesse alcançar o meu melhor, ao meu avô Pedro Barbosa (in memoriam) — meu grande exemplo para a vida — e à toda minha família. Dedico à minha namorada, Érica de Oliveira Lopes, que acreditou em mim e me acompanhou até nos momentos mais difíceis.

# Agradecimentos

Agradeço ao meu orientador e coordenador, Dr. Gondim, cujo auxílio, dedicação e paciência foram essenciais para a produção deste trabalho.

Agradeço às grandes amizades que fiz durante o curso e que tornaram essa jornada mais tranquila.

Agradeço à Universidade de Brasília por me acolher e proporcionar essa experiência única, cheia de altos e baixos que finalmente se conclui, e à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) por investir no meu futuro por meio do programa Ciência sem Fronteiras (CSF).

# Resumo

Ataques de negação de serviço estão cada vez mais frequentes e poderosos. Tornando, então, sistemas de proteção e mitigação uma necessidade. Para estudar a perspectiva do atacante, este trabalho desenvolveu o Dambuster, uma ferramenta de injeção de pacotes que pode ser utilizada na avaliação de sistemas de proteção a ataques de flooding. A ferramenta pode ser executada em ambientes na nuvem e utiliza uma arquitetura modular, bem estruturada e bem documentada. O desempenho da aplicação foi mensurado comparando os resultados dos vários ataques implementados, da execução em diferentes configurações de hardware e em relação a uma ferramenta já existente, o T50. Os resultados demonstraram que o Dambuster é uma ferramenta de geração de tráfego de pacotes consistente, escalável e eficaz. Suas taxas de envio e produção de pacotes foram superiores e com maior consistência que os valores da ferramenta de referência, produzindo, em média, 32% mais pacotes durante o ataque. Trabalhos futuros podem expandir o Dambuster, melhorando sua performance e implementando novos ataques e opções de configuração e customização.

**Palavras-chave:** negação de serviço, injeção de pacotes, ataques de flooding, t50

# Abstract

Denial-of-service (DoS) attacks have been more frequent and impactful than ever before. Thus making mitigation and protection systems a must. In order to research the attacker perspective, this work developed Dambuster, a packet injection tool that can be used to evaluate DoS flooding attacks protection systems. This tool is cloud-ready and implements a well structured, well documented modular architecture. The performance of the application is analyzed by comparing the results of the available attacks, the execution on different hardware settings, and against a commonly used tool (T50). The results show that Dambuster is a viable, scalable and efficient tool for packet injection and traffic generation. Its packet production and sending rates are shown to be higher and more consistent than T50's. On average, Dambuster's production outperformed T50's by sending 32% more packets during the attacks. Future work can improve Dambuster by increasing its performance and implementing new attacks and customization options.

**Keywords:** denial of service, packet injection, flooding attacks, t50

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Lista de Abreviaturas e Siglas

**CIDR** *Class Inter-Domain Routing.*

**CLI** *Command Line Interface.*

**DDoS** *Distributed Denial-of-Service.*

**DoS** *Denial-of-Service.*

**GUI** *Graphical User Interface.*

**ICMP** *Internet Control Message Protocol.*

**IP** *Internet Protocol.*

**TCP** *Transmission Control Protocol.*

**UDP** *User Datagram Protocol.*

**XML** *Extensible Markup Language.*

# Capítulo 1

# Introdução

## 1.1 Justificativa e Motivação

Juntamente à constante expansão da Internet e da quantidade de dispositivos conectados, o número e a capacidade de ameaças à segurança de computadores tem aumentado bastante nos últimos anos. Dentre estas ameaças, Ataques de Negação de Serviço (do inglês, *Denial-of-Service* (DoS)) têm destaque. Sua variante distribuída, ataques DDoS (do inglês, *Distributed Denial-of-Service*), é uma forma simples [1], barata e eficaz de comprometer o acesso a serviços e recursos de rede [2]. Técnicas de proteção contra ataques DDoS foram desenvolvidas, porém os atacantes continuam a evoluir suas técnincas, burlando as defesas implantadas [3]. A maioria destes ataques afligem as vítimas enviando um volume massivo de dados e pacotes, ou seja, são ataques volumétricos. De acordo com Cao [4], cerca de 65% de todos os ataques DDoS são volumétricos.

O crescimento exponencial da Internet permitiu que a escala e magnitude dos ataques DDoS alcançassem níveis jamais vistos, como apresenta a Figura 1.1. Em setembro de 2017, a infraestrutura do Google Cloud recebeu um ataque de 2,5 Tbps, cerca de quatro vezes maior do que o ataque do Mirai, de 2016 [5]. De acordo com o recente relatório de segurança da Microsoft [2], o segundo semestre de 2021 apresentou ataques com frequência e complexidades sem precedentes. O sistema de mitigação implantado impediu 40% mais ataques do que no semestre anterior, alcançando 1.955 ataques por dia. Em novembro de 2021, foi identificado e mitigado um ataque de 3,47 Tbps e 340 milhões de pacotes por segundo, valores inéditos. De todos os ataques identificados, 55% foram do tipo UDP Flood [6] e 19% visaram o protocolo TCP.

Neste cenário, soluções de mitigação tornam-se essenciais para o funcionamento estável do ecossistema da Internet. O investimento em segurança cibernética por provedores de serviços em nuvem é massivo. Algumas plataformas investem mais de um bilhão de dólares anualmente em soluções de segurança [2]. Estas soluções precisam ser implementadas

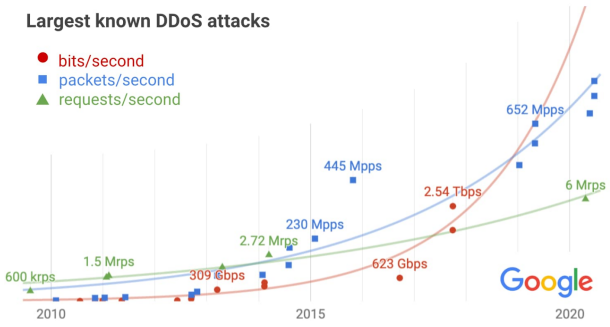


Figura 1.1: Evolução dos maiores Ataques DDoS na última década (Fonte: [5]).

prevenindo as diversas variantes de ataques, defendendo múltiplas camadas e dispositivos de rede, além de se preparar para ataques desconhecidos e inesperados [5]. As soluções de mitigação atuais implantam sistemas baseados em nuvem, com suporte às diversas camadas, adaptáveis e respaldados por técnicas de inteligência artificial e modelos estatísticos ([3]; [7]).

## 1.2 Objetivos

Este trabalho apresenta o Dambuster, uma ferramenta projetada com o propósito de auxiliar o desenvolvimento e avaliação de sistemas de mitigação, além de estudos relacionados a ataques DoS volumétricos. O software desenvolvido se baseou na ferramenta Linderhof ([8]; [9]), que é uma aplicação para o estudo de ataques DoS por reflexão, bem como avaliar soluções para sua mitigação.

O objetivo geral deste trabalho é desenvolver uma ferramenta de simulação de ataques de negação de serviço para entender o funcionamento destes ataques, sistemas de mitigação, além da perspectiva do atacante.

Os objetivos específicos são:

- Avaliar o desempenho da ferramenta desenvolvida;
- Verificar a viabilidade modular inspirada no Linderhof;
- Comparar o funcionamento de diversos ataques DoS;
- Compreender a perspectiva do atacante no desenvolvimento e implantação de técnicas de ataque.

## 1.3 Organização do trabalho

Para isto, este trabalho está organizado em capítulos. O Capítulo 2 explica o funcionamento de ataques de negação de serviço e apresenta técnicas e variantes destes ataques. O Capítulo 3 se aprofunda na arquitetura, desenvolvimento e execução da ferramenta. O Capítulo 4 esclarece a metodologia utilizada na avaliação da viabilidade e desempenho da ferramenta. Os resultados desta avaliação são, então, apresentados e discutidos no Capítulo 5. Ao final, no Capítulo 6, as implicações deste trabalho são expostas e algumas direções para trabalhos futuros são propostas.

# Capítulo 2

# Referencial Teórico

Neste capítulo são apresentados os principais conceitos referentes aos ataques DoS e sua variante, DDoS, bem como os ataques específicos implementados na ferramenta Dambuster.

## 2.1 Ataques DoS e DDoS

Ataques de negação de serviço (do inglês, *Denial-of-Service* (DoS)) são utilizados por usuários mal-intencionados para impedir que usuários legítimos tenham acesso a recursos de redes e computadores [10].

Quando ataques DoS utilizam múltiplos dispositivos para atacar um único alvo, são nomeados Ataques de Negação de Serviço Distribuídos, ou, *Distributed Denial-of-Service* (DDoS) [11]. A primeira etapa de um ataque distribuído é infectar sistemas vulneráveis conectados à Internet, instalando o software nocivo. Estes sistemas comprometidos são denominados *zombies*. A segunda etapa consiste em enviar um sinal na rede para que os *zombies* iniciem o ataque à vítima [12], conforme apresentado na Figura 2.1. Desta forma, ataques distribuídos amplificam o poder de um ataque DoS, além de esconder mais efetivamente os rastros do invasor [13].

## 2.2 Taxonomia dos Ataques DoS e DDoS

Ao analisar a quantidade de pacotes enviados durante um ataque, pode-se dividir ataques DDoS em dois tipos [14]: **baixo volume** e **volumétricos**. A Figura 2.2 apresenta a taxonomia proposta por [14]. Neste trabalho, esta classificação será utilizada também para ataques DoS, pois não é influenciada pela característica distribuída de ataques DDoS.

Ataques de baixo volume exploram vulnerabilidades em aplicações ou protocolos para limitar o funcionamento correto do serviço [15]. Normalmente o objetivo é alcançado

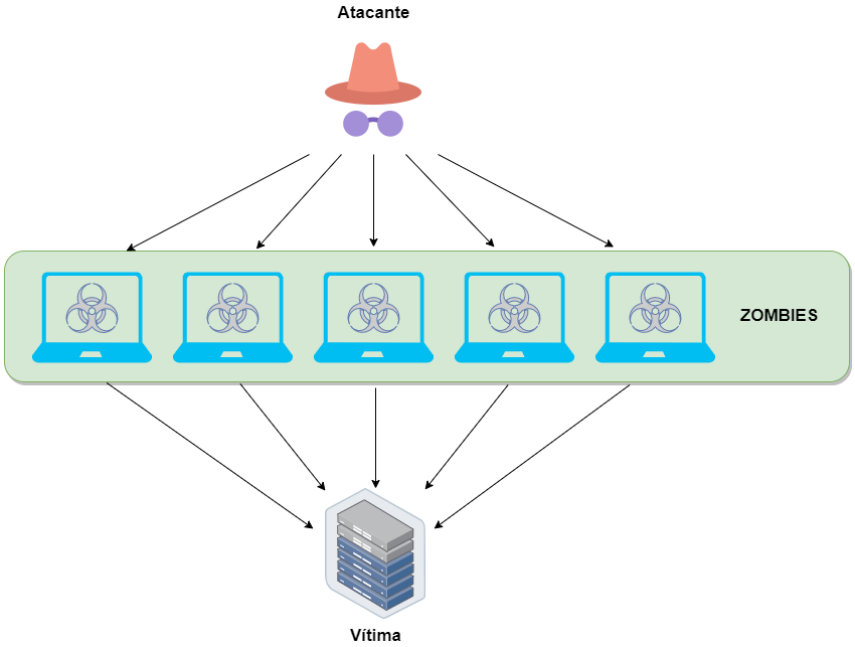


Figura 2.1: Representação de Ataques DDoS.

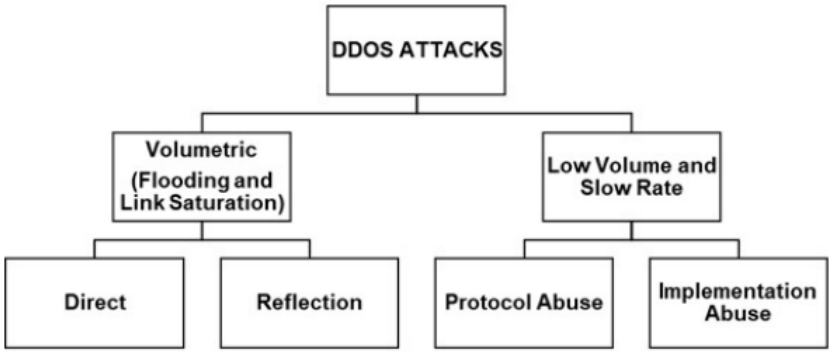


Figura 2.2: Taxonomia de ataques DDoS (Fonte: [14]).

construindo e enviando um ou mais pacotes nocivos para a vítima. Estes pacotes são construídos cuidadosamente para atacarem vulnerabilidades de software, desabilitando serviços e sistemas nos dispositivos desejados [12].

Ataques volumétricos são caracterizados por um grande e constante fluxo de pacotes de rede, com o objetivo de esgotar recursos de rede, CPU e memória da vítima. Eles são chamados também de ataques por inundação, ou *flooding* [16]. A força destes ataques é o tráfego volumoso, e não o conteúdo dos pacotes, o que dificulta a distinção entre tráfego legítimo e tráfego malicioso [12].

Douligeris e Mitrokotsa [17] categorizam ataques DoS em ataques por reflexão e ataques diretos. Ataques por reflexão alteram o endereço de origem dos pacotes, substituindo pelo endereço do alvo, técnica conhecida como *IP Spoofing*. Esses pacotes são então enviados para dispositivos de rede denominados refletores [18]. Ao responder à comunicação recebida, os refletores enviam pacotes para o endereço adulterado, efetivamente redirecionando o tráfego para a vítima. Assim, os refletores são dispositivos inocentes que são utilizados como intermediários pelo atacante [12]. Ataques por reflexão dificultam o rastreamento do ataque, além da capacidade de amplificar o tráfego, permitindo que atacantes com uma menor vazão de pacotes possam exaurir serviços superdimensionados [4]. Ataques diretos não utilizam intermediários e o tráfego acontece apenas entre o atacante e o alvo. *IP Spoofing* também é utilizado por ataques diretos, dificultando a identificação da origem dos pacotes [18].

No decorrer deste trabalho, o foco será nos ataques volumétricos diretos que estão implementados no Dambuster.

## 2.3 Técnicas de Ataque DoS

Para que os ataques sejam mais efetivos em desabilitar serviços e burlar soluções de proteção, os atacantes utilizam diversas técnicas na execução de ataques de negação de serviço. Algumas técnicas comumente utilizadas são apresentadas nas seções seguintes.

### 2.3.1 *IP Spoofing*

A técnica de mascaramento de endereço IP, ou *IP Spoofing*, consiste em falsificar o endereço de origem de um pacote dificultando o rastreamento do endereço original [19]. Esta técnica é utilizada na maioria dos ataques DoS e essencial em ataques por reflexão [17]. Ehrenkranz e Li [20] demonstram que embora muitas técnicas de prevenção e identificação de pacotes adulterados tenham sido propostas, nenhuma foi capaz de erradicar o problema. Cao [4] afirma que presumir que a presença de *IP Spoofing* tenha sido eliminada da Internet é fora da realidade. Mirkovic e Reiher [21] classificam ataques de acordo com a autenticidade do IP de origem. As classificações propostas são:

- Endereço adulterado: ataques que utilizam a técnica de *IP Spoofing* para dificultar a identificação do atacante ou executar o ataque:

    1. Acessibilidade:

        1.1 Endereço acessível: endereços utilizados por algum dispositivo e que são acessíveis por roteadores;

1.2 Endereço inacessível: endereços que não são utilizados por nenhum dispositivo e que não podem ser acessados.

2. Técnica de mascaramento:

   2.1 Aleatório: os endereços selecionados são escolhidos de forma aleatória;

   2.2 Subrede: os endereços selecionados estão na mesma subrede que o atacante;

   2.2 *En route*: os endereços selecionados estão na rota entre o atacante e a vítima;

   2.3 Fixo: os endereços selecionados são fixados pelo atacante.

- Endereço não adulterado: Ataques que não mascaram o IP, como o NAPHTA.

### 2.3.2 *Carpet Bombing*

Uma nova variante dos ataques volumétricos chamada de *Carpet Bombing* consiste em atacar simultaneamente vários dispositivos em uma subrede ou em um bloco de endereços *Class Inter-Domain Routing* (CIDR) [22]. Essa técnica dificulta a detecção de um ataque em andamento de forma precisa e ágil, além de sobrecarregar sistemas de mitigação [23].

### 2.3.3 Alteração de taxa de ataque

Ataques DoS podem utilizar várias técnicas para a taxa de envio dos pacotes durante a execução. Essas técnicas podem ser agrupadas da seguinte maneira [21]:

- Taxa constante: Os pacotes nocivos são enviados em uma taxa fixa, a maioria dos ataques DoS utilizam esta forma. O ataque súbito é capaz de desabilitar um serviço rapidamente, porém é mais facilmente detectado por sistemas de mitigação;
- Taxa incremental: A taxa de envio cresce durante o ataque, sobrecarregando a vítima de forma lenta por um longo período de tempo. Essa técnica reduz a agilidade com que o ataque é descoberto por sistemas de segurança;
- Taxa flutuante: Durante o ataque a taxa de envio é alterada de acordo com o comportamento da vítima ou de forma predeterminada. Um exemplo são ataques em pulsos, que desestabilizam a vítima periodicamente. Ataques em pulsos podem ser coordenados entre diversos atacantes para manter a vítima continuamente indisponível sem que sistemas de mitigação identifiquem anormalidades prolongadas.

## 2.4 Ataques ao protocolo TCP

O *Transmission Control Protocol* (TCP) é definido na RFC 793 [24] como um protocolo ponto a ponto, orientado a conexão e confiável. Seu funcionamento depende que uma conexão entre um cliente e um servidor seja estabelecida. Esta conexão é efetuada por meio do *three-way handshake*, realizado em três etapas. Diversos ataques DoS abusam das vulnerabilidades presentes no comportamento esperado do TCP [11].

### 2.4.1 SYN Flood

O SYN Flood é o mais conhecido [18] e um dos mais poderosos [12] ataques DoS. O ataque explora o *three-way handshake* para esgotar os recursos do servidor e torná-lo inoperante [10]. O atacante envia vários segmentos SYN, requisitando que novas conexões sejam estabelecidas com a vítima. Estes pacotes têm seus endereços de origem mascarados, às vezes com endereços inexistente ou inoperantes [12]. Ao receber a solicitação, a vítima imediatamente aloca um espaço considerável para aquela conexão [21], responde com um segmento SYN/ACK e aguarda o recebimento de um ACK, mantendo a conexão aberta. Devido ao volume de requisições, a CPU precisa analisar os pacotes recebidos [18] e a memória é sobrecarregada por conexões em aberto [12]. Assim, o SYN Flood é capaz de debilitar ou desativar o servidor [6]. O ataque está apresentado na Figura 2.3.

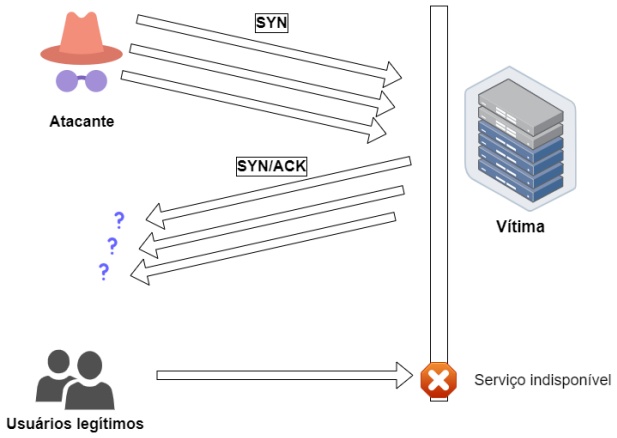


Figura 2.3: Ataque SYN Flood.

### 2.4.2 SYN/ACK Flood

Na operação normal do TCP, um servidor responde a um segmento SYN enviando um segmento SYN/ACK, indicando que recebeu o pedido de conexão. O ataque de SYN/ACK Flood (Figura 2.4) é caracterizado por uma enorme quantia de pacotes SYN/ACK enviada

pelo atacante. A vítima não reconhece aquela tentativa de conexão e, na tentativa de computar essa quebra de protocolo, esgota seus recursos computacionais, tornando-se inoperante [6].

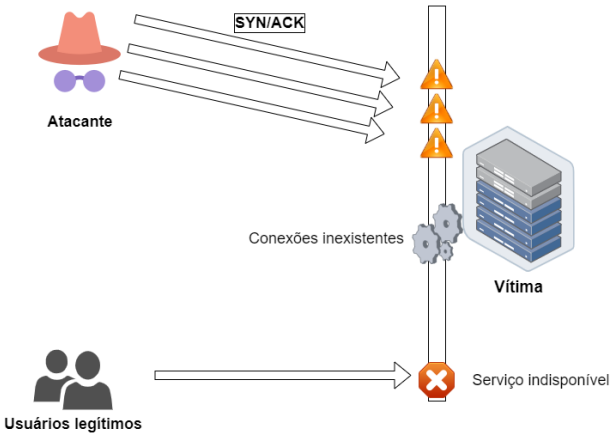


Figura 2.4: Ataque SYN/ACK Flood.

### 2.4.3 RST Flood

Quando um segmento TCP recebido por um servidor não pertence à nenhuma de suas conexões abertas, o servidor retorna um segmento RST para o cliente, indicando que aquela conexão não existe e que pode ser encerrada [24]. O ataque de RST Flood (Figura 2.5) consiste no envio intenso e contínuo de segmentos RST. Ao receber os pacotes, a vítima exaure seus recursos tentando, inutilmente, localizar em sua memória as conexões inexistentes [6].

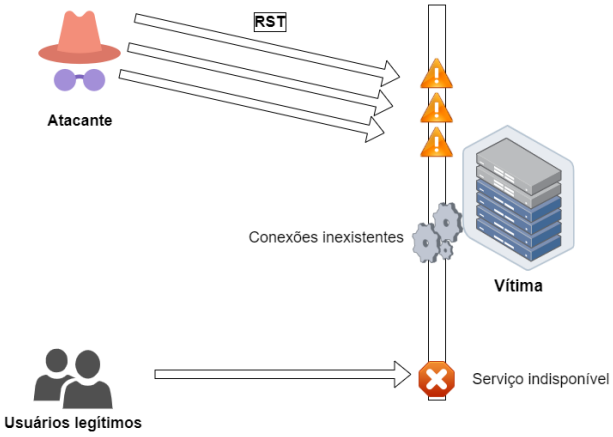


Figura 2.5: Ataque RST Flood.

## 2.5 UDP Flood

O *User Datagram Protocol* (UDP) é um protocolo orientado a transações e que não garante ordenamento na entrega dos pacotes. Ele foi projetado para que aplicações possam se comunicar com o mínimo de burocracia. Assim sendo, a comunicação entre cliente e servidor é feita de maneira direta, sem a necessidade de estabelecer uma conexão [25].

Por não precisar de uma conexão estabelecida para a transferência de dados, o UDP se torna especialmente vulnerável a ataques extremamente efetivos e difíceis de serem detectados. O principal destes é o UDP Flood (Figura 2.6), que inunda e sobrecarrega a vítima com uma imensa quantidade de pacotes em alta frequência. Este ataque consome recursos de rede e largura de banda até que a vítima não seja mais acessível [6].

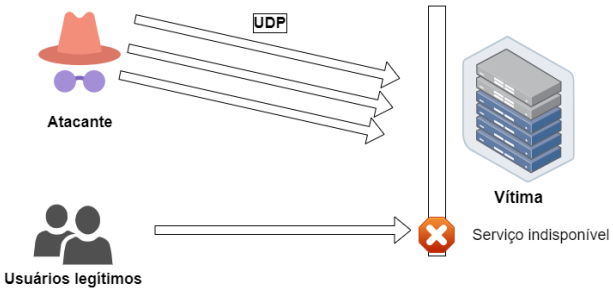


Figura 2.6: Ataque UDP Flood.

## 2.6 ICMP Flood

O *Internet Control Message Protocol* (ICMP) é uma parte integral do *Internet Protocol* (IP) e tem o propósito de permitir que *hosts* reportem problemas na comunicação com outros *hosts* [26]. As mensagens trocadas pelo ICMP são utilizadas para comunicar sobre as condições da rede, erros no processamento de datagramas e ajudar no diagnóstico de problemas de conexão ([12]; [26]; [27]). Estas mensagens são essenciais para o funcionamento correto de protocolos como TCP e IP [27].

Como o ICMP é essencial para o funcionamento de comunicações entre *hosts*, a maioria dos dispositivos em uma rede IP permitem a chegada de mensagens ICMP de consulta de rede, denominadas de *Echo Request*, o que torna estes dispositivos vulneráveis a ataques como o ICMP Flood [10].

O ICMP Flood é similar ao UDP Flood e consiste em enviar uma grande quantidade de mensagens ICMP para a vítima, esgotando os recursos de rede e a largura de banda, e impedindo tráfego de usuários legítimos [6]. As mensagens ICMP enviadas no ataque podem ser do tipo *Echo Request* e *Echo Reply*. Na primeira variante (Figura 2.7), a

vítima, além de sobrecarregada com o fluxo intenso de pacotes, tenta responder a todas as requisições do atacante, congestionando o tráfego de saída [28].

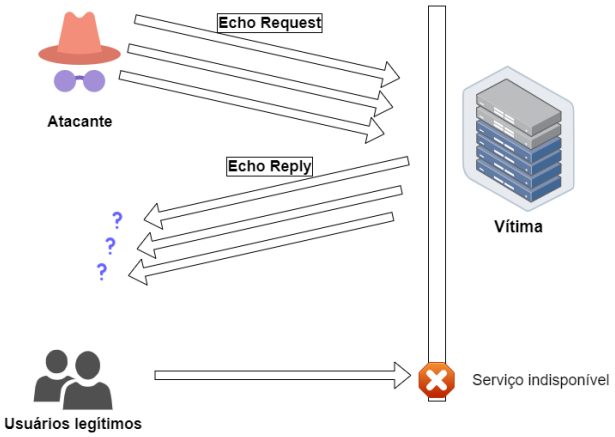


Figura 2.7: Ataque ICMP *Echo Request* Flood.

## 2.7 Considerações finais

Neste capítulo foram apresentados os principais conceitos referentes aos ataques DoS e DDoS. Também foram abordados os ataques específicos implementados na ferramenta. Assim, o Dambuster é apresentado, em detalhes, no capítulo que segue.

# Capítulo 3

# Dambuster

O Dambuster foi desenvolvido como uma ferramenta para estudar, entender e aferir o funcionamento de ataques de negação de serviço volumétricos diretos, com a finalidade de avaliar soluções de mitigação para tais ataques. O usuário pode configurar e executar diversos ataques DoS com alta performance, configurabilidade e usabilidade.

O nome faz alusão à esquadra de aviação inglesa, os *Dambusters*. Essa divisão da força aérea inglesa atuou durante a Segunda Guerra Mundial e executaram a Operação *Chastise*, encarregada de bombardear represas alemãs, causando inundações e desestruturando econômica, política e moralmente a Alemanha Nazista [29]. Assim, a partir dessa associação, os diferentes ataques suportados pelo Dambuster são chamados de bombas. Na ferramenta proposta, estão implementadas as bombas: **SYN Flood, SYN/ACK Flood, RST Flood, UDP Flood,** e duas variações do **ICMP Flood: *Echo Request*** e ***Echo Reply***.

A ferramenta foi desenvolvida em linguagem C, para sistemas Linux e é baseada na versão 2.0.0 do Linderhof [30], uma ferramenta modular de ataques DoS por reflexão. Embora o projeto seja inspirado no Linderhof, várias melhorias e modificações foram implementadas:

- Refatoração e melhor documentação do código;
- Aperfeiçoamento da interface gráfica e de linha de comando;
- Criação de imagem Docker para ambientes de nuvem e melhor distribuição;
- Geração e disponibilização automática de documentação;
- Geração de arquivos de saída com relatórios de execução;
- Alterações na injeção de pacotes para ser utilizada por ataques volumétricos diretos;
- Refatoração completa da lógica de criação de cabeçalhos e implementação de protocolos;

- Refatoração da comunicação com o usuário por meio da criação do módulo `logger`;
- Implementação de três novos protocolos e seis novos ataques.

Nas seções seguintes a arquitetura, a execução e as funcionalidades do Dambuster são apresentadas.

## 3.1 Arquitetura

A arquitetura do Dambuster é baseada na da ferramenta Linderhof ([8]; [9]; [30]) sendo dividida em quatro módulos ou camadas: **Interface, Commander, Arsenal e Injector**. Cada módulo é responsável por uma etapa do ataque. Além dos quatro módulos básicos, algumas estruturas contém funções para auxiliar na execução do Dambuster, essas estruturas fazem parte do módulo auxiliar **Common**.

Um diagrama da arquitetura do projeto é apresentado na Figura 3.1, e cada módulo é descrito em detalhes nas seções seguintes.

### 3.1.1 Interface

O módulo de Interface é responsável por implementar a entrada e a saída de dados para comunicação com o usuário. O Dambuster contém dois tipos de interface, uma por linha de comando (do inglês, *Command Line Interface* (CLI)) e uma gráfica (do inglês, *Graphical User Interface* (GUI)). O módulo de Interface também lida com a validação de entrada e leitura de arquivos de configuração. A saída dos dados de execução é por meio do *logger*.

Após ler a entrada do usuário, a Interface gera um *draft* do ataque, ou seja, o rascunho das informações gerais do ataque que será utilizado pelas camadas inferiores do Dambuster.

### 3.1.2 Commander

O módulo Commander é dividido em dois componentes. O primeiro, *commander*, é responsável por fazer inicializações de variáveis necessárias para a executar os ataques. Além disso, ele gerencia o *manager* na criação e execução dos ataques.

Já o *manager* é responsável por planejar e executar o ataque de acordo com o *draft*. Com base nessa informação, ele requisita ao módulo Arsenal a criação de uma bomba referente ao ataque escolhido.

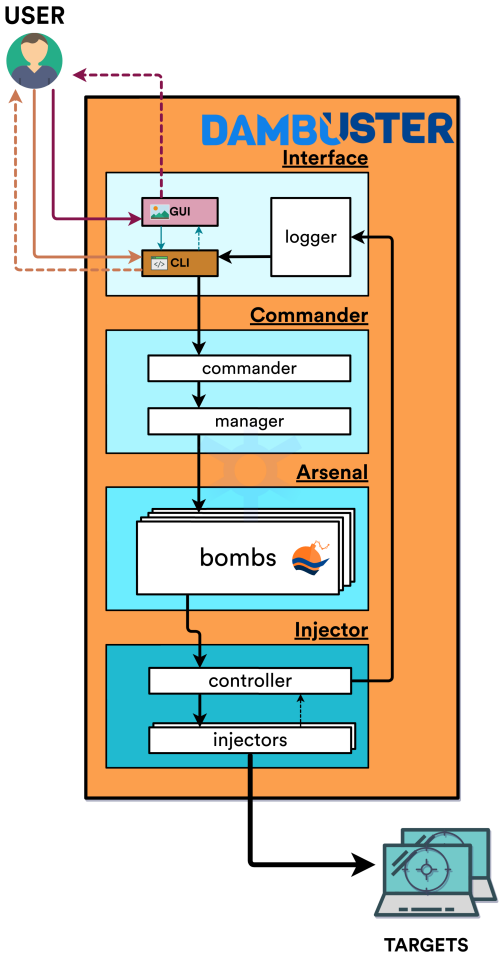


Figura 3.1: Arquitetura do Dambuster.

### 3.1.3 Arsenal

O módulo Arsenal contém todas as bombas disponíveis e é responsável pela criação dos pacotes que serão enviados durante o ataque. Nele, os pacotes são totalmente construídos de acordo com as especificações informadas, para que o módulo Injector possa executar o ataque corretamente.

Novos ataques devem ser implementados no módulo Arsenal. A estrutura dos ataques é extremamente modular, facilitando a adição de novos tipos de bombas em iterações futuras do projeto. O preenchimento dos cabeçalhos dos datagramas é feita utilizando o *blacksmith*, parte do módulo Common. Novos protocolos de rede ou transporte podem ser incorporados neste módulo.

### 3.1.4 Injector

Este módulo é dividido em dois componentes: *controller* e *injector*. O *controller* é responsável por gerenciar os *injectors* e orquestrar o ataque. Enquanto cada *injector* é responsável por enviar os pacotes para um alvo específico, ou seja, a quantidade de alvos determina a quantidade de *injectors*. O *injector* lida com a criação de *sockets* e envio dos pacotes pela rede, e é a camada mais inferior da arquitetura. Cada *injector* ocupa uma *thread* da aplicação. Durante o ataque, o *controller* emite *feedback* do ataque para o módulo Interface, por meio do *logger*.

## 3.2 Execução

A execução dos ataques pode ser dividida em três etapas: configuração, construção de pacotes e injeção de pacotes. A configuração é a etapa em que a entrada do usuário é usada para criar o *draft*. Os parâmetros e opções disponíveis são explorados no decorrer deste capítulo.

### 3.2.1 Construção de Pacotes

Após o *draft* ser criado durante a configuração, estes dados são enviados para o Commander fazer as preparações para o ataque. O Arsenal é chamado então para a construção do pacote. Os pacotes do Dambuster são implementados por meio da estrutura de dados *packet_wrapper*, que encapsula um pacote de rede, além de conter outros dados úteis para a aplicação.

Cada bomba deve implementar sua própria função para a construção destes pacotes. Normalmente, a função de construção vai criar o *payload* do pacote e chamar o *blacksmith* presente no módulo auxiliar Common para encapsular o *payload* colocando os cabeçalhos necessários e gerando um *packet_wrapper*.

No *blacksmith* estão implementadas funções para a criação de cabeçalhos dos protocolos IP, TCP, UDP e ICMP. A Figura 3.2 apresenta como o *draft* se relaciona com o *packet_wrapper*, e com um datagrama.

### 3.2.2 Injeção de Pacotes

O módulo Injector é chamado pelo Arsenal para que o ataque ocorra. Primeiramente são criadas as *threads* injetoras e seus respectivos *buckets*. O *bucket* corresponde à quantidade de pacotes que devem ser enviados. O *controller* é responsável por controlar a taxa de injeção, por meio do *bucket*.

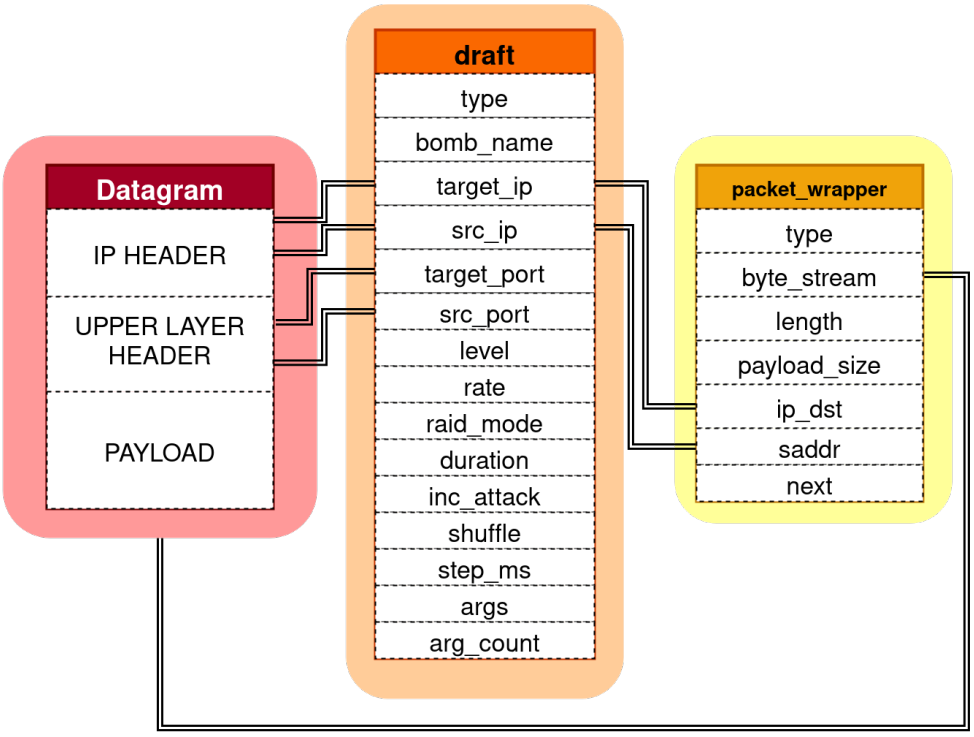


Figura 3.2: Relacionamentos de estruturas com o *draft*.

Cada alvo é designado para uma *thread* injetora. Logo, o número de *threads* é equivalente ao número de alvos. Normalmente o Dambuster é executado com apenas um alvo, o que significa apenas um *injector*. Como a ferramenta não cria vários pacotes, mas envia sempre o mesmo pacote alterando-o em tempo real, a estrutura de *bucket* é apenas uma abstração. O *bucket* é na verdade um contador que decresce quando um pacote é enviado.

A injeção não é feita de forma contínua, mas em ciclos, ou ondas. O intervalo entre essas ondas é definido pelo usuário. Em cada onda, o *injector* reinicia seu *bucket* e tenta esvaziá-lo. Todos os pacotes enviados por um *injector* durante uma onda são idênticos, pois o custo para calcular o *checksum* de cada pacote individualmente é muito alto. Isso é uma das limitações identificadas no projeto, pois facilita a filtragem de pacotes em troca de um aumento no desempenho.

Durante o ataque, o *controller* é responsável por alterações que afetam todos os pacotes. Assim sendo, caso seja necessário fazer alterações apenas para um alvo, o *injector* é responsável por fazê-las. Na implementação atual, as alterações que podem ser feitas durante a injeção são nos campos: endereço de origem, porta de origem e porta de saída.

## 3.3 Geração de tráfego

Os ataques do Dambuster são efetuados por meio da geração de tráfego de rede. Estes ataques são controlados por diferentes modos de ataque, além de parâmetros gerais, compartilhados entre os modos. Os parâmetros e modos são utilizados para determinar como esses pacotes serão enviados no decorrer da execução do programa. Os parâmetros gerais utilizados para configurar as taxas esperadas do ataque efetuado pelo *injector* são:

- *duration*: determina a duração do ataque em milisegundos. Por padrão, o Dambuster irá executar o ataque até que seja interrompido manualmente;
- *basetime*: determina o intervalo entre cada onda de envio de pacotes, em milisegundos. Ou seja, o intervalo entre cada iteração. Por padrão, o Dambuster irá enviar uma onda de pacotes por segundo;
- *level*: determina o nível de envio, ou seja a quantidade de pacotes enviados em cada iteração. A quantidade de pacotes enviados em cada nível é determinado pela Equação 3.1:

$$X = 10^{(L-1)} \tag{3.1}$$

  Por padrão, o *injector* assumirá nível 1 e irá enviar apenas um pacote em cada iteração. É importante notar que a quantidade de pacotes determinada pelo nível é o objetivo do Dambuster, porém não é garantido que a máquina conseguirá produzir e enviar todos os pacotes desejados. Além disso, não é certeza que todos os pacotes consigam chegar ao alvo. A Tabela 3.1 detalha quantos pacotes por iteração serão enviados, de acordo com o nível escolhido.

Tabela 3.1: Níveis de ataque.

| Nível | Pacotes/iteração |
|---|---|
| 1 | 1 |
| 2 | 10 |
| 3 | 100 |
| 4 | 1000 |
| 5 | 10000 |
| 6 | 100000 |
| 7 | 1000000 |
| 8 | 10000000 |
| 9 | 100000000 |
| 10 | 1000000000 |

Estão implementados quatro modos de ataques, que alteram como a taxa de envio de pacotes se comportam: Modo Padrão, Modo Incremental, Modo *Raid* e Modo Personalizado.

No **Modo Padrão** do Dambuster, apenas os parâmetros gerais determinarão a taxa de injeção dos pacotes. Cada *injector* atacará um alvo com a quantidade de pacotes determinadas pelo nível do ataque. A Figura 3.3 apresenta o funcionamento.

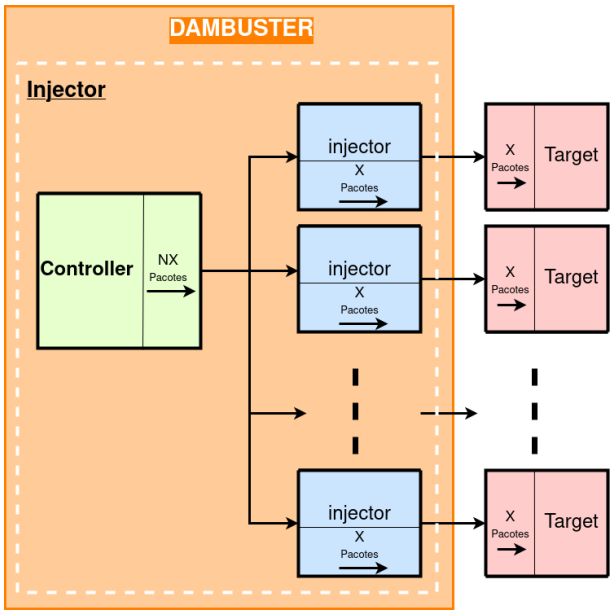


Figura 3.3: Modo Padrão do Dambuster.

No **Modo Incremental**, o nível do ataque não é constante. O valor inicial é passado pelo parâmetro de nível. A cada $i$ iterações, o nível do ataque é incrementado em 1. O valor de $i$ é um número inteiro passado como argumento do Modo Incremental. Quando o nível chegar a 10, ele para de aumentar.

No **Modo *Raid***, o nível do ataque é ignorado e o Dambuster tenta enviar o máximo de pacotes possível em cada iteração. A quantidade de pacotes enviada dependerá da capacidade de hardware e rede do usuário.

No **Modo Personalizado**, o usuário pode especificar um arquivo contendo a quantidade de pacotes que serão enviados por cada *injector* em cada iteração. Ao chegar no final do arquivo, o Dambuster voltará para o início, gerando assim um comportamento cíclico. Com isso, o modo personalizado pode ser utilizado para criar ataques de taxa flutuante.

## 3.4 Técnicas de ataque implementadas

O Dambuster contém múltiplas funcionalidades que aumentam o nível de customização dos ataques, permitindo uma grande flexibilidade para o usuário e incorporando as técnicas apresentadas na Seção 2.3.

A ferramenta faz uso da técnica de ***IP Spoofing***, permitindo que o usuário selecione o endereço de origem. Este endereço pode ser fixo ou aleatório. Além disso, também é possível informar múltiplos endereços, na forma de uma lista de endereços IP ou usando a notação CIDR. Esses endereços serão usados de forma aleatória ou na ordem informada, de acordo com a preferência do usuário. Os endereços informados podem ser dos protocolos IPv4 e IPv6.

Assim como no IP de origem, o usuário pode informar um ou múltiplos endereços de destino. Assim sendo, o usuário pode efetuar um ataque de ***Carpet Bombing*** ao informar uma lista de endereços ou um bloco CIDR.

Os diversos tipos de taxa de ataque apresentados na Seção 2.3 foram implementados no Dambuster por meio dos diversos modos de ataque explicados na Seção 3.3.

## 3.5 Suporte a diversos ataques

Uma das características principais da arquitetura utilizada é a modularização na implementação e inserção de novos ataques. Os ataques no Dambuster são denominados *bombs* e estão contidos no módulo **Arsenal**. As *bombs* são implementadas de forma independente, utilizando apenas as funções do módulo auxiliar **Common**.

A versão desenvolvida neste trabalho tem suporte aos ataques: **SYN Flood, SYN/ACK Flood, RST Flood, UDP Flood,** e duas variações do **ICMP Flood: *Echo Request*** e ***Echo Reply***.

Em iterações futuras do Dambuster, novos ataques podem ser desenvolvidos na forma de *bombs* e facilmente adicionados à ferramenta. O procedimento para a implementação de um ataque DoS como uma *bomb* do Dambuster é descrito na seção seguinte.

### 3.5.1 Inclusão de novos ataques

Para a criação de novas *bombs* devem ser implementadas três funções:

1. `Execute<Bomb>`: Responsável pela execução do ataque desejado. Essa função recebe o *draft* como parâmetro e invoca a função de criação do pacote padrão do ataque. Após a criação do pacote, esta função inicia o módulo Injector.

2. `CreateAttackData`: Responsável pela criação do pacote padrão do ataque. Essa função invoca a função de criação do *payload* de dados e encapsula com os cabeçalhos dos protocolos utilizados, gerando um *packet_wrapper*. O módulo Common possui diversas funções para a criação e encapsulamento de pacotes em seu submódulo *blacksmith*.

3. `Forge<Bomb>`: Responsável pela criação do *payload* de dados do pacote. Esta função é invocada durante a criação dos pacotes e pode utilizar argumentos passados pelo usuário para o preenchimento dos dados.

Após a implementação do ataque, é necessário integrá-lo ao programa para que possa ser executado pelo usuário. Para adicionar a *bomb*, esta deve ser adicionada às opções presentes no *draft*. Em seguida, é necessário alterar o módulo Commander para que a função de execução do ataques seja associada à opção adicionada ao *draft*. Deve-se então adicioná-la às opções aceitas pelo módulo Interface, associando o parâmetro de entrada com a opção inserida no *draft*. Dessa forma, o ataque estará disponível na ferramenta.

Os procedimentos descritos nesta seção são necessários para que a *bomb* seja integrada à arquitetura. Durante a execução do Dambuster, os quatro módulos principais são acionados e estão diretamente relacionados às alterações efetuadas. Inicialmente, o usuário seleciona o ataque desejado no módulo Interface. Este módulo preenche o *draft* com a opção definida e envia-o para o módulo Commander. O Commander invoca a função `Execute<Bomb>` associada a esta opção, invocando assim a função de execução da *bomb* implementada no módulo Arsenal. Esta função invoca a função `CreateAttackData` da *bomb* que, então, invoca a função `Forge<Bomb>` e encapsula o *payload* com os cabeçalhos, efetivamente gerando um *packet_wrapper* a partir de um *draft*. Após a criação do pacote, a função de execução aciona o módulo Injector para a execução do ataque e injeção de pacotes. A Figura 3.4 apresenta o fluxo de dados durante a execução da ferramenta e como os módulos principais se relacionam com as funções das *bombs* implementadas no Arsenal.

## 3.6 Usabilidade

Algumas funcionalidades adicionais foram incluídas para aumentar a usabilidade e portabilidade da aplicação proposta neste trabalho, os quais serão descritas nas próximas seções.

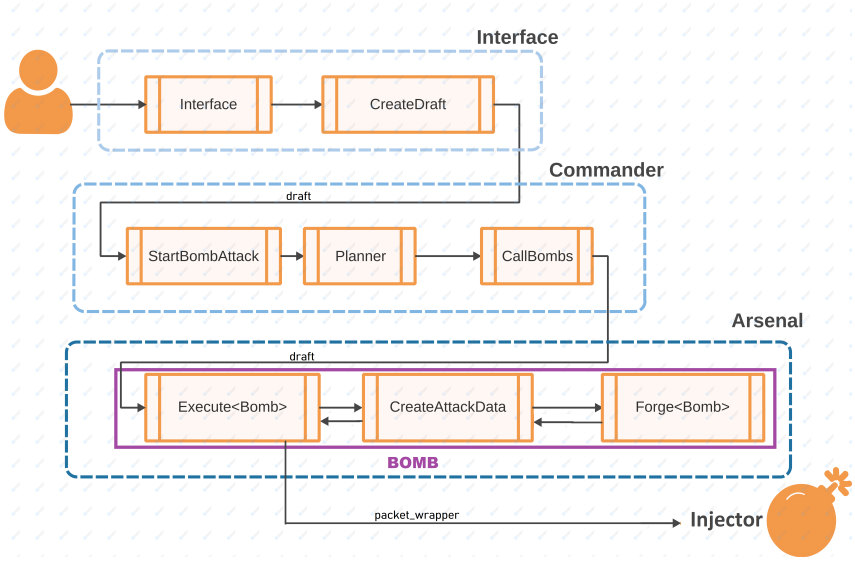


Figura 3.4: Fluxo de dados do Dambuster.

### 3.6.1 Interface

Com o intuito de facilitar o uso da aplicação, foram implementados dois tipos de interface. Uma por linha de comando (CLI) e uma interface gráfica (GUI). As Figuras 3.5 a 3.6 mostram as duas interfaces desenvolvidas.


Figura 3.5: Interface CLI.

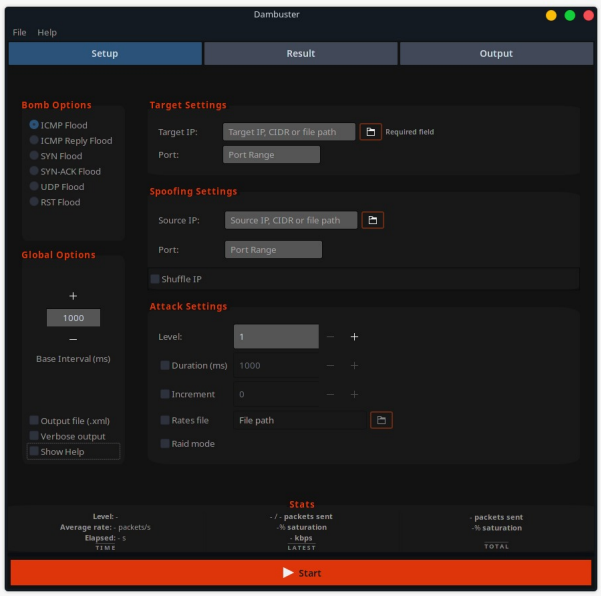


Figura 3.6: Interface GUI.

A GUI utilizou a biblioteca gráfica GTK na versão 3.22 [31], e foi escrita em linguagem C. A GUI permite a visualização do ataque por meio de gráficos (conforme apresentado na Figura 3.7), indicando a taxa de pacotes enviados por segundo, além da forma textual (apresentada na Figura 3.8) que detalha a injeção de pacotes. A forma textual é a mesma utilizada na interface CLI.


Figura 3.7: Visualização gráfica do ataque.

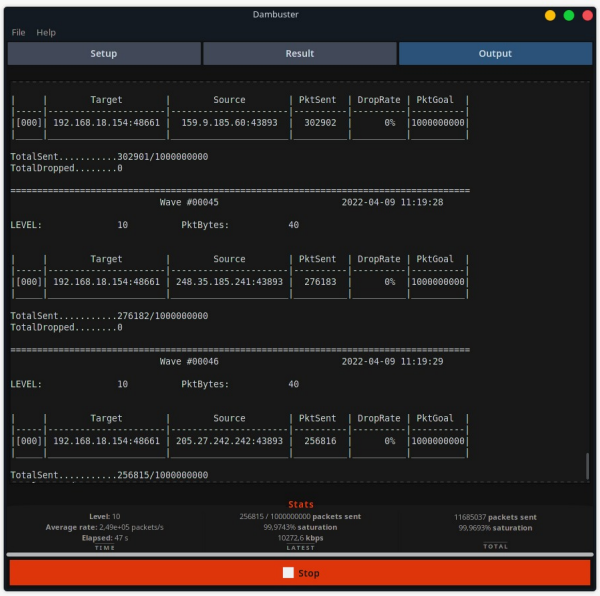


Figura 3.8: Visualização textual do ataque.

### 3.6.2 Entrada e Saída por Arquivos

O Dambuster disponibiliza opções para que a interação com o programa seja feita por meio de arquivos texto. Todos os parâmetros de entrada podem ser informados em um arquivo de configuração. Para dar um *feedback* completo do ataque e agilizar o processo de automação e testes, a ferramenta produz um relatório do ataque no formato *Extensible Markup Language* (XML)[32].

### 3.6.3 Docker

Docker é uma plataforma com código aberto de empacotamento ou "containerização" de software. Ela facilita a distribuição e a portabilidade de uma aplicação por meio da criação de imagens que contém a aplicação e os requisitos necessários para a sua execução [33].

A partir disto, uma imagem Docker do Dambuster foi gerada utilizando uma configuração mínima, removendo a interface gráfica e arquivos redundantes. A imagem foi disponibilizada em um repositório[1] do Docker Hub.

[1] `https://hub.docker.com/repository/docker/eduardoforca/dambuster`

## 3.7 Considerações finais

Este capítulo apresentou os principais aspectos da arquitetura, desenvolvimento, utilização e disponibilização do Dambuster. No capítulo que segue, a metodologia e os procedimentos dos testes utilizando a ferramenta são descritos.

# Capítulo 4

# Metodologia e Procedimentos

Este capítulo apresenta os métodos utilizados para testar o desempenho do Dambuster e dos ataques implementados.

## 4.1 Metodologia

Como explicado anteriormente, ataques DoS volumétricos visam sobrecarregar a vítima enviando pacotes de forma massiva. Logo, é possível avaliar o sucesso de uma ferramenta de ataques volumétricos pela quantidade de pacotes enviados em um período de tempo. Desta forma, os testes propostos visam identificar a taxa máxima de envio que pode ser alcançada pela ferramenta.

Com o intuito de mensurar o desempenho da ferramenta, ataques incrementais foram efetuados. Estes ataques foram feitos de acordo com a Tabela 3.1, iniciando no nível 1 até alcançar o nível 10, identificando assim o ponto de saturação da taxa de envio, ou seja, em que nível a ferramenta se torna incapaz de enviar a quantidade desejada de pacotes.

Três métricas foram escolhidas para descrever a taxa de saturação:

- Taxa máxima de envio;
- Taxa média de saturação;
- Nível médio de saturação.

A taxa máxima de envio é a maior taxa alcançada durante a execução do ataque. A taxa média de saturação é a taxa média alcançada durante o período de saturação. O nível médio de saturação é o nível equivalente da taxa média de saturação.

Assim sendo, o desempenho do Dambuster foi avaliado por três perspectivas:

- Hardware do atacante;

- Comparação com ferramentas similares;
- Ataque DoS utilizado.

## 4.2 Procedimentos

A partir da metodologia proposta, três cenários de testes foram projetados. O primeiro cenário testou os diversos ataques no Dambuster, estabelecendo os valores de referência do desempenho da ferramenta. No segundo cenário, o Dambuster foi executado a partir de outro dispositivo e, desta forma, permitiu comparar o desempenho de acordo com as configurações de hardware do atacante. Já no terceiro cenário, uma aplicação já existente e similar ao Dambuster foi avaliada para motivos de comparação. A ferramenta escolhida foi o T50 [34]. Contudo, como ataques incrementais não são suportados pelo T50, o ataque foi efetuado na taxa máxima alcançada pela aplicação, denominado Modo *Flood*, similar ao Modo *Raid* do Dambuster.

No total, dois dispositivos foram utilizados como atacante e dois foram utilizados como vítima, a mesma vítima foi atacada nos Cenários 1 e 3, enquanto a outra foi atacada no Cenário 2. O roteador utilizado para a conexão dos computadores foi o mesmo em todos os testes, do modelo *TP-LINK Archer C7 AC1750*. A conexão utilizou portas Gigabit Ethernet. As especificações de todos os dispositivos utilizados estão na Tabela 4.1.

Tabela 4.1: Especificações dos dispositivos de teste.

| | **Atacante** | **Vítima** |
|---|---|---|
| *Cenário 1* | | |
| Sistema | Manjaro 21.2.5 64-bit | Windows 11 Pro |
| Processador | Intel Core i5-5200U @ 2,20GHz | Intel Core i5-8400 @ 2,80GHz |
| Memória | 8GB DDR3 @ 1600MHz | 16GB DDR4 @ 2666Mhz |
| Placa de Rede | TP-Link UE300 USB Gigabit | Intel Gigabit Ethernet |
| *Cenário 2* | | |
| Sistema | ClearLinux OS 36010 | Manjaro 21.2.5 64-bit |
| Processador | Intel Core i5-8400 @ 2,80GHz | Intel Core i5-5200U @ 2,20GHz |
| Memória | 16GB DDR4 @ 2666Mhz | 8GB DDR3 @ 1600MHz |
| Placa de Rede | Intel Gigabit Ethernet | TP-Link UE300 USB Gigabit |
| *Cenário 3* | | |
| Sistema | Manjaro 21.2.5 64-bit | Windows 11 Pro |
| Processador | Intel Core i5-5200U @ 2,20GHz | Intel Core i5-8400 @ 2,80GHz |
| Memória | 8GB DDR3 @ 1600MHz | 16GB DDR4 @ 2666Mhz |
| Placa de Rede | TP-Link UE300 USB Gigabit | Intel Gigabit Ethernet |

Em cada cenário foram testados os ataques: **ICMP *Echo Request* Flood, ICMP *Echo Reply* Flood, SYN Flood, SYN/ACK Flood, RST Flood** e **UDP Flood**. A Tabela 4.2 sintetiza os cenários de teste descritos abaixo:

### Cenário 1

Neste cenário, a ferramenta Dambuster foi testada utilizando os parâmetros da Tabela 4.3. As especificações dos dispositivos estão na Tabela 4.1. O ataque durou 100 segundos, aumentando gradualmente a taxa de envio. Os dados foram coletados usando o software de captura Wireshark. A partir destes dados pretende-se identificar em qual momento ocorreu a saturação no envio dos pacotes e, assim, averiguar o limite da ferramenta neste cenário.

### Cenário 2

Este cenário foi testado da mesma forma que o anterior, porém utilizando uma máquina com melhores especificações, como apresentado na Tabela 4.1. Logo, espera-se que os resultados sejam melhores do que os encontrados no cenário 1.

### Cenário 3

O Cenário 3 avaliou a ferramenta T50, um injetor de pacotes de código aberto utilizado para executar testes de estresse em infraestruturas de rede. Ele foi escolhido porque implementa todos os protocolos necessários, é bem documentado e apresenta uma alta performance, chegando a enviar mais de 1.000.000 de pacotes por segundo durante um ataque do tipo SYN Flood [34]. Dessa forma, mostrou-se uma escolha adequada para a comparação. O T50 tem algumas diferenças na implementação que podem causar discrepância nos resultados. Os pacotes enviados pela ferramenta carregam dados aleatórios, além de preencher aleatoriamente alguns campos do cabeçalho. Isso resulta em pacotes maiores do que no Dambuster. Além disso, esses pacotes são enviados continuamente, enquanto o Dambuster faz envios em intervalos configurados pelo usuário, podendo apresentar períodos de ociosidade. O T50 também aumenta seu desempenho ao não computar corretamente o *checksum* da camada de transporte do pacote. Assim como explicado anteriormente, o T50 não possui um modo de ataque incremental, então a configuração de taxa máxima foi escolhida. O ataque foi efetuado pelo período de 15 segundos e a total duração do ataque foi considerada saturada. O endereço de origem foi mascarado de forma aleatória. Os dispositivos foram os mesmos utilizados no cenário 1. As suas especificações estão dispostas na Tabela 4.1.

Tabela 4.2: Cenários de testes.

| Cenário | Ferramenta | Ataques | Taxa | Objetivo |
|---|---|---|---|---|
| 1 | **Dambuster** | Todos | Incremental | Estabelecer valores de referência |
| 2 | **Dambuster** | Todos | Incremental | Avaliar melhorias de hardware |
| 3 | T50 | Todos | Máxima | Avaliar ferramentas similares |

Para avaliar os Cenários 1 e 2 de forma similar, os parâmetros de configuração do Dambuster foram os mesmos para todos os ataques e estão dispostos na Tabela 4.3. O teste durou um total de 100 segundos, incrementando o nível a cada 10 segundos. Os envios foram feitos em intervalos de 1 segundo, ou seja, 10 envios para cada nível. O endereço de origem dos pacotes foi mascarado e as portas de origem e destino foram escolhidas aleatoriamente no início do ataque. A avaliação do Cenário 1 em relação ao Cenário 3 não foi tão equivalente, pois o T50 não efetua ataques incrementais, então a comparação foi feita entre o período total do Cenário 3, e o período saturado do Cenário 1.

Tabela 4.3: Parâmetros de ataque do Dambuster.

| **Parâmetro** | | **Valor** |
|---|---|---|
| Duração | Intervalo | 1 s |
| | Nível | 10 s |
| | Total | 100 s |
| Nível | Inicial | 1 |
| | Final | 10 |
| Endereço | Origem | Aleatório |
| Porta | Origem | Aleatória |
| | Destino | Aleatória |

A disposição do ambiente de testes (Figura 4.1) foi consistente entre todos os cenários. A máquina do atacante e da vítima foram conectadas a uma rede local por um roteador. Ambas as máquinas executaram o Wireshark [35] para a captura de pacotes da execução do ataque. Os testes foram realizados por meio de *scripts* escritos em Python e Bash. Os dados coletados foram agregados e analisados em Python com auxílio da plataforma Jupyter Lab [36] e da biblioteca de análise de dados Pandas ([37]; [38]).

Para manter consistência e replicabilidade dos testes, foram desenvolvidos *scripts*. Para os Cenários 1 e 2, um *script* em Python foi executado. Uma seção do código está presente no Apêndice A. Ele administra a execução do Dambuster e do software de captura de pacotes, um ataque de cada vez. De forma similar, a execução do teste do Cenário 3 foi efetuada utilizando os *scripts* em Bash presentes no Apêndice B.

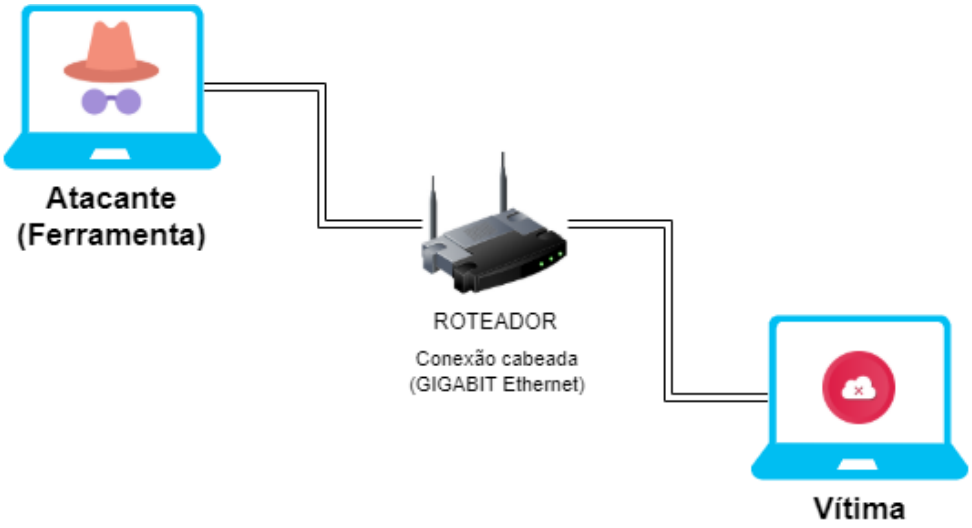


Figura 4.1: Ambiente de Testes.

## 4.3 Considerações finais

Neste capítulo foram apresentadas as configurações e os procedimentos dos testes de desempenho do Dambuster. Na sequência, os resultados dos testes são apresentados e discutidos no próximo capítulo.

# Capítulo 5

# Resultados e Discussão dos Testes

Este capítulo apresenta os gráficos e tabelas produzidos durante os testes propostos. Estes resultados são analisados e discutidos em detalhe, abordando aspectos da arquitetura do Dambuster e do funcionamento de ataques de negação de serviço.

## 5.1 Cenário 1

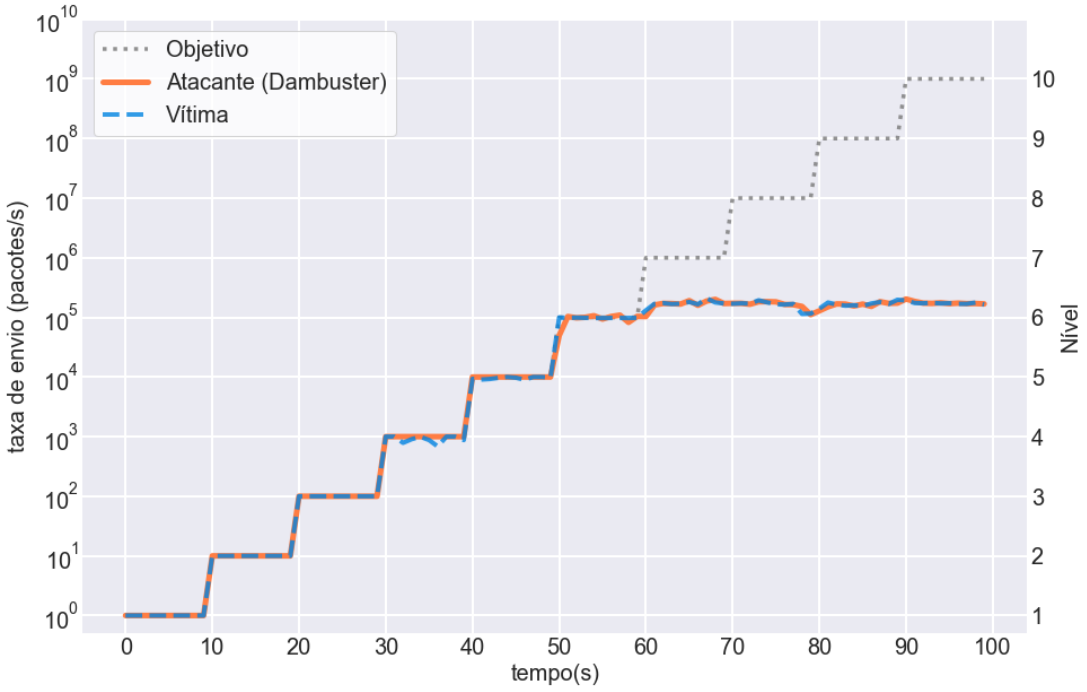


Figura 5.1: Cenário 1 - ICMP *Echo Request* Flood.

É possível visualizar por meio dos gráficos dispostos nas Figuras 5.1 a 5.6 que, em todos os ataques, a saturação total ocorreu a partir dos 60 segundos de teste, logo o período

de saturação definido foi entre 60 e 100 segundos de ataque. Além dos gráficos, os dados numéricos dos ataques foram separados por nível e estão incluídos no Apêndice C, na Tabela C.1. As métricas propostas foram extraídas e estão na Tabela 5.1. As taxas de envio foram apresentadas em pacotes por segundo (PPS) e Megabits por segundo (Mbps). O nível médio de saturação equivale ao primeiro nível de saturação que a taxa média de saturação não é capaz de alcançar.

Tabela 5.1: Cenário 1 - Métricas de envio dos ataques efetuados pelo Dambuster.

| **Ataque** | PPS | | Mbps | | Nível médio de saturação |
|---|---|---|---|---|---|
| | Máx | Média | Máx | Média | |
| **ICMP *Request*** | 202870 | 167861,62 | 5,68 | 4,70 | 7 |
| **ICMP *Reply*** | 203628 | 190204,62 | 5,70 | 5,33 | 7 |
| **RST** | 167271 | 154383,25 | 6,69 | 6,18 | 7 |
| **SYN** | 184843 | 157711,45 | 7,39 | 6,31 | 7 |
| **SYN/ACK** | 163531 | 155901,15 | 6,54 | 6,24 | 7 |
| **UDP** | 192453 | 175935,60 | 5,39 | 4,93 | 7 |

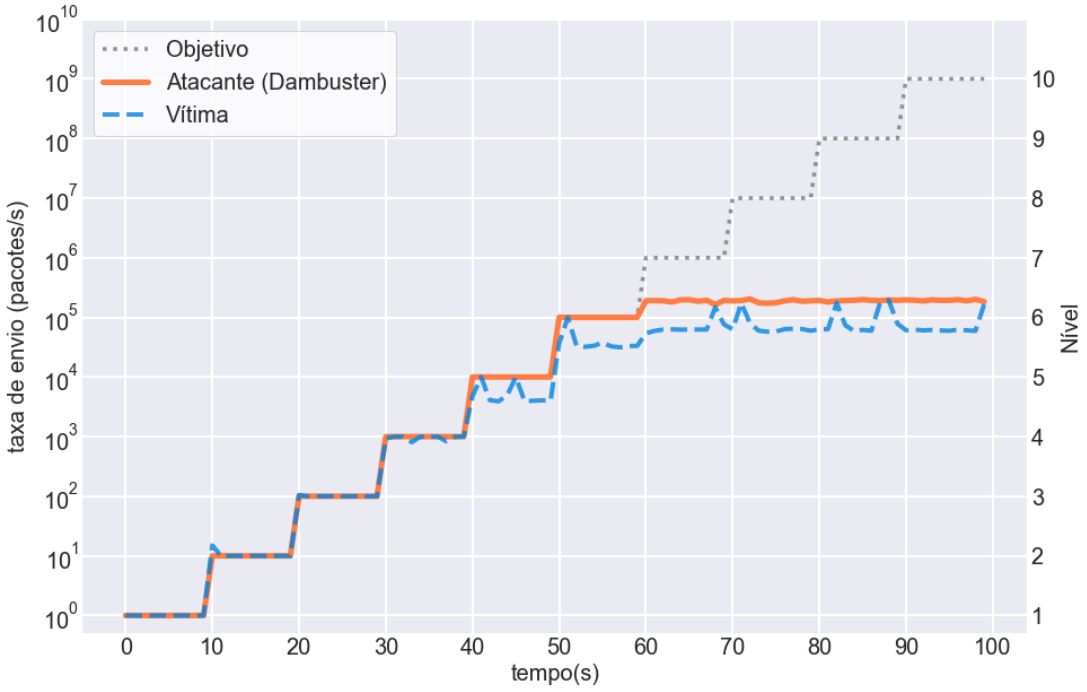


Figura 5.2: Cenário 1 - ICMP *Echo Reply*.

A Tabela C.1 mostra que, em todos os ataques, o nível médio de saturação foi 7, embora tenha ocorrido alguma saturação no nível 6 nos ataques ICMP *Echo Request* Flood, RST Flood e SYN/ACK Flood. Assim sendo, foi possível perceber pela Tabela 5.1 que os ataques ao protocolo TCP tiveram uma taxa menor em PPS. Porém, devido ao tamanho maior dos pacotes, uma taxa maior em Mbps. Isso já era esperado pois na arquitetura do Dambuster o esforço para produzir os pacotes é o mesmo entre os protocolos, diferindo apenas no tamanho, o que permite uma maior produção de pacotes mais leves como UDP e ICMP. Mas, essa diferença de processamento é ínfima, comparada ao aumento da taxa de bits decorrente da diferença de tamanho dos pacotes.

Em termos de Mbps, o SYN Flood obteve a maior taxa observada, assim como a maior taxa média. As vítimas apresentaram uma taxa de entrada similar à saída do atacante, com exceção do ataque ICMP *Echo Reply*, no qual a vítima apresentou bastante inconstância no recebimento de pacotes, com picos seguidos de saturação. Esse efeito ocorreu pois a capacidade de processamento dos pacotes recebidos pela vítima foi afetada pelo ataque.

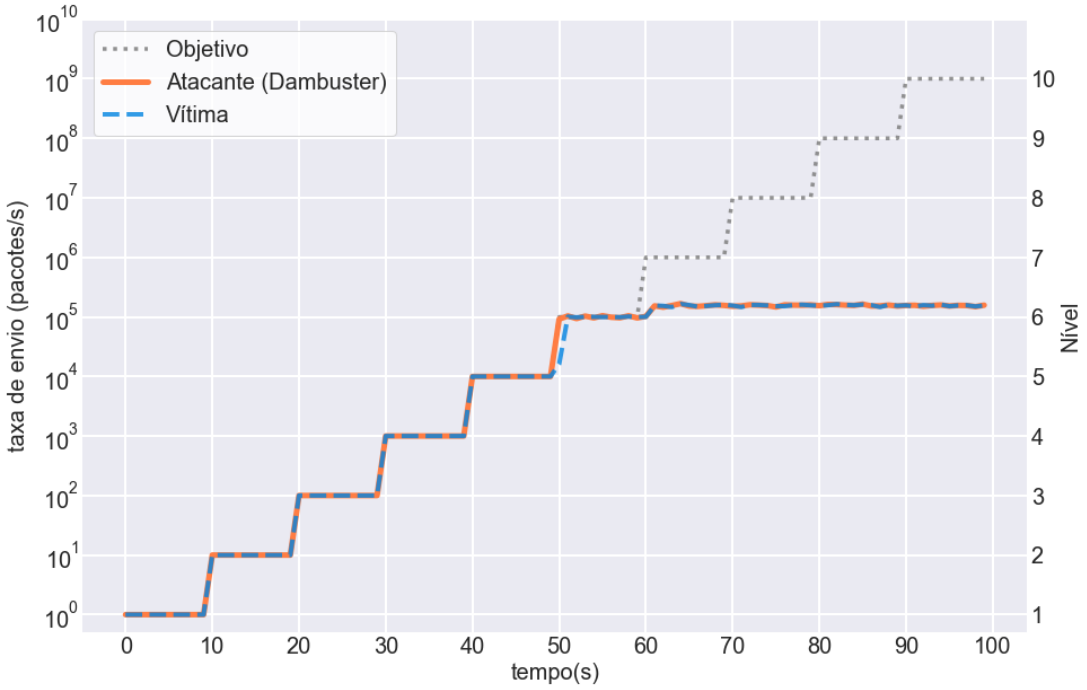


Figura 5.3: Cenário 1 - RST Flood.

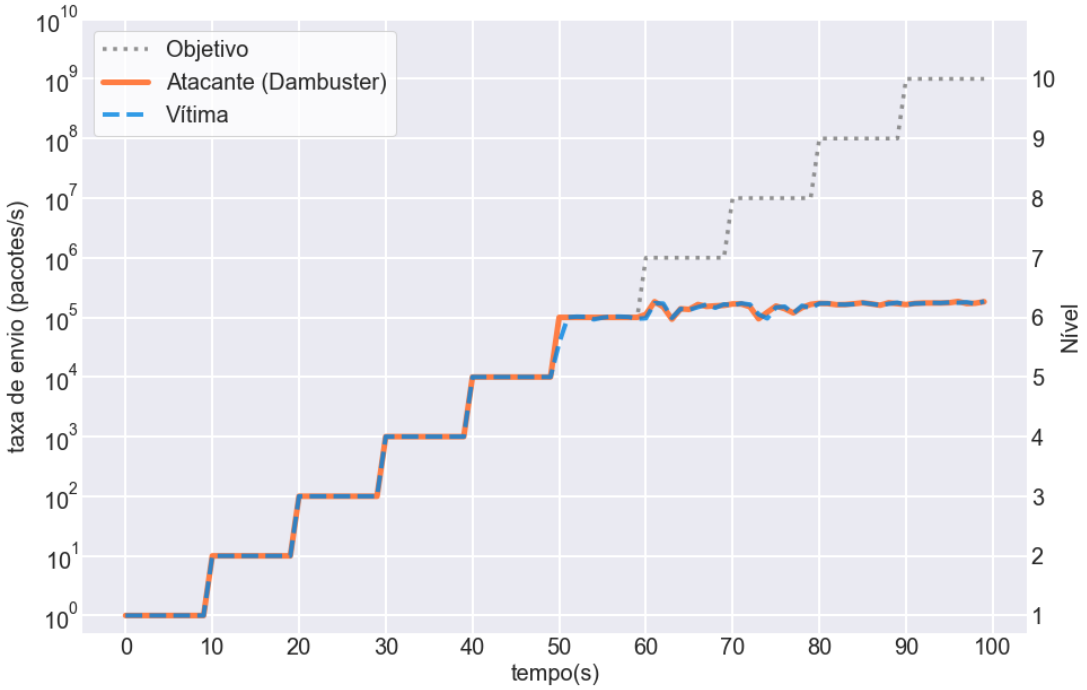


Figura 5.4: Cenário 1 - SYN Flood.

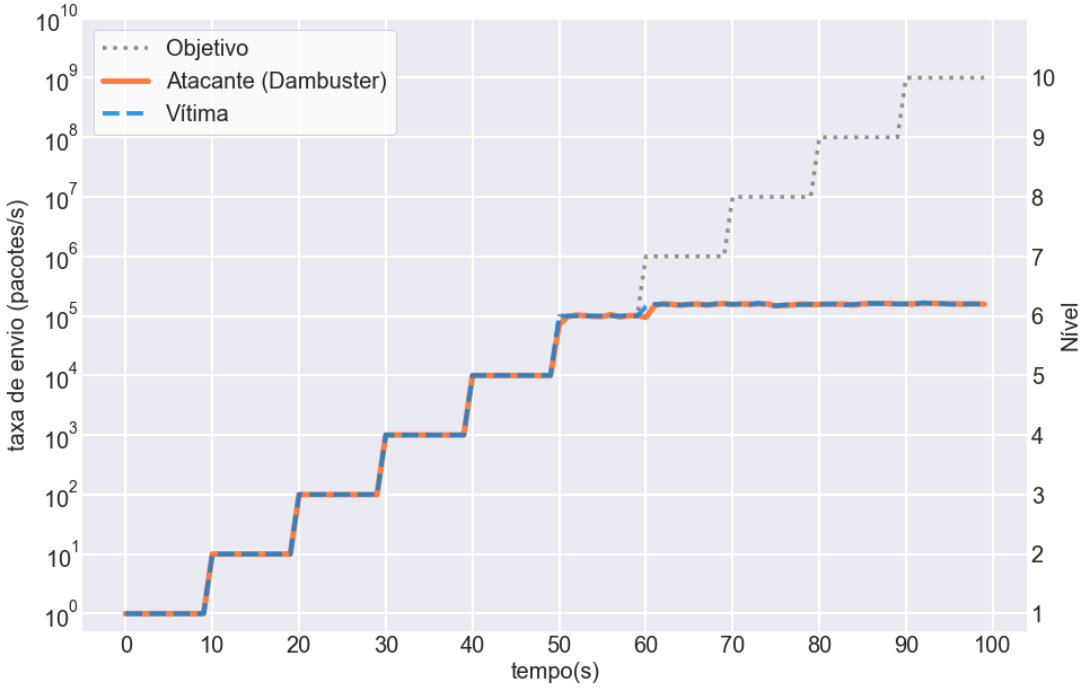


Figura 5.5: Cenário 1 - SYN/ACK Flood.

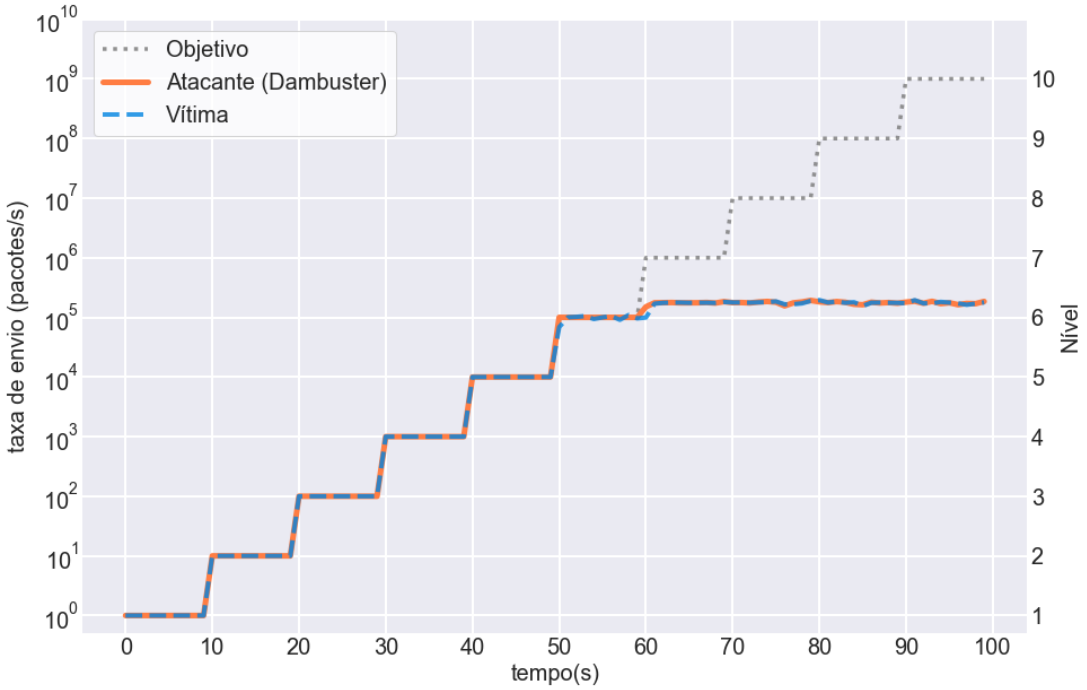


Figura 5.6: Cenário 1 - UDP Flood.

No Cenário 1, todos os ataques saturaram logo após entrar no nível 7, mais especificamente na faixa de 150.000 a 200.000 pacotes por segundo. A ferramenta foi capaz de produzir mais pacotes ICMP e UDP do que pacotes TCP, porém devido ao tamanho maior dos pacotes TCP (40 bytes) comparado ao ICMP e UDP (ambos com 28 bytes), a taxa de bits por segundo foi maior do que nos outros ataques. Este resultado é validado pela implementação do Dambuster. Como grande parte dos esforço da produção de pacotes acontece antes da execução do ataque, o custo de processamento durante o ataque é similar entre os protocolos, com uma pequena vantagem para pacotes mais leves. Entretanto, a diferença entre os tamanhos dos pacotes é considerável, refletindo diretamente na taxa de transferência de bits.

## 5.2 Cenário 2

Os dados coletados durante o teste foram separados por nível e estão na Tabela C.2 no Apêndice C, as métricas propostas foram aferidas e estão dispostas na Tabela 5.2. Pela Tabela 5.2, conclui-se que a produção de pacotes foi similar entre as execuções, com a média em torno de 600.000 PPS. Assim como ocorreu no Cenário 1, em termos de vazão de bits, os ataques ao protocolo TCP apresentaram o melhor rendimento, na faixa de 24 Mbps, o que é relacionado ao tamanho maior dos pacotes em relação aos outros protocolos.

Os ataques aos protocolos ICMP e UDP apresentaram valores na faixa de 17 Mbps. A arquitetura do Dambuster é compatível com o resultado, já que a produção de pacotes TCP tem custo de processamento maior, porém similar à produção de pacotes mais leves. Os gráficos de taxa de fluxo de pacotes no decorrer do tempo estão nas Figuras 5.7 a 5.12.

Tabela 5.2: Cenário 2 - Métricas de envio dos ataques efetuados pelo Dambuster.

| **Ataque** | PPS | | Mbps | | Nível médio de saturação |
|---|---|---|---|---|---|
| | Máx | Média | Máx | Média | |
| **ICMP *Request*** | 614008 | 600310,60 | 17,19 | 16,81 | 7 |
| **ICMP *Reply*** | 623612 | 575295,55 | 17,46 | 16,11 | 7 |
| **RST** | 609797 | 597814,22 | 24,39 | 23,91 | 7 |
| **SYN** | 617539 | 606918,12 | 24,70 | 24,28 | 7 |
| **SYN/ACK** | 618074 | 595311,53 | 24,72 | 23,81 | 7 |
| **UDP** | 615185 | 608314.28 | 17,23 | 17,03 | 7 |

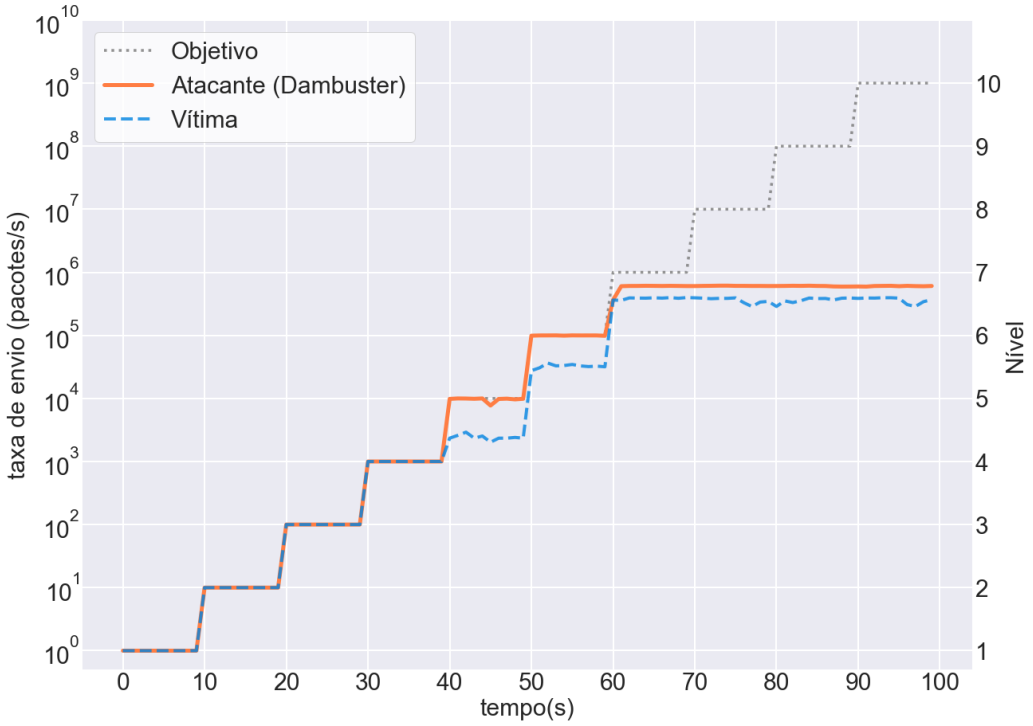


Figura 5.7: Cenário 2 - ICMP *Echo Request* Flood.

Ao analisar os gráficos das Figuras 5.7 a 5.12, percebe-se que todos os ataques saturaram completamente no nível 7, não ultrapassando a marca de $10^6$ PPS. Ou seja, o período de saturação foi a partir dos 60 segundos de execução. As representações visuais dos resultados indicam que os valores chegaram próximos do nível seguinte. É plausível

que aumentando ainda mais as especificações de hardware do atacante e com otimizações na produção de pacotes do Dambuster, a ferramenta alcance a taxa de 1 milhão de pacotes por segundos.

Durante os ataques, a vítima apresentou saturação a partir do nível 5, indicando que já não suportava a taxa recebida de pacotes e que seus recursos de rede e de processamento estavam sobrecarregados. Isto demonstra o impacto dos ataques DoS, mesmo em dimensões pequenas.

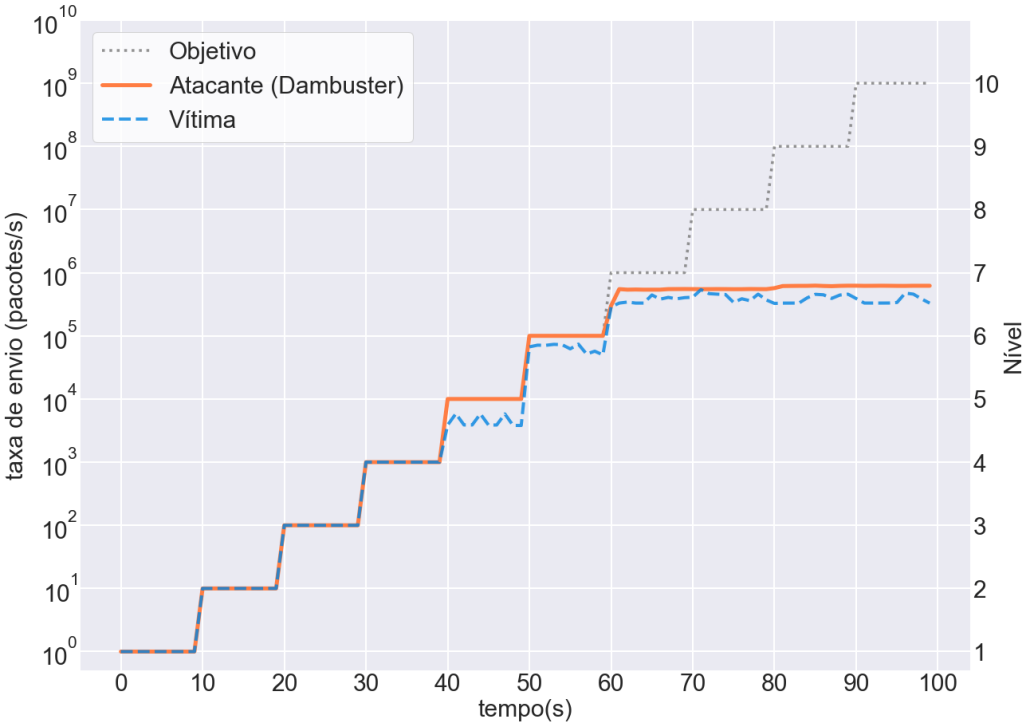


Figura 5.8: Cenário 2 - ICMP *Echo Reply*.

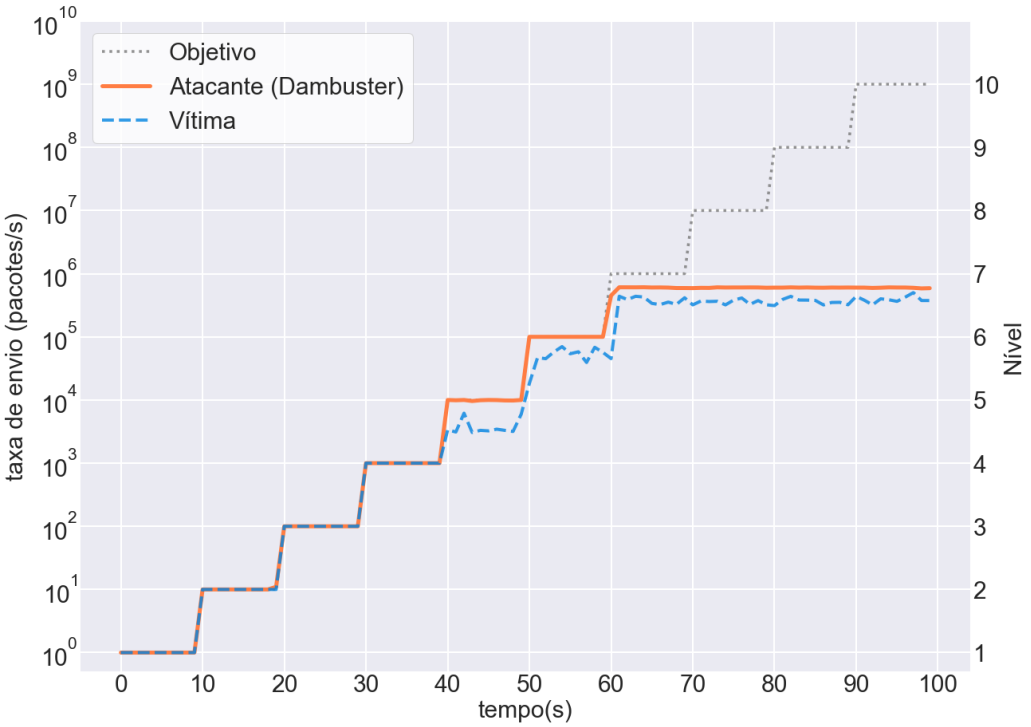


Figura 5.9: Cenário 2 - RST Flood.

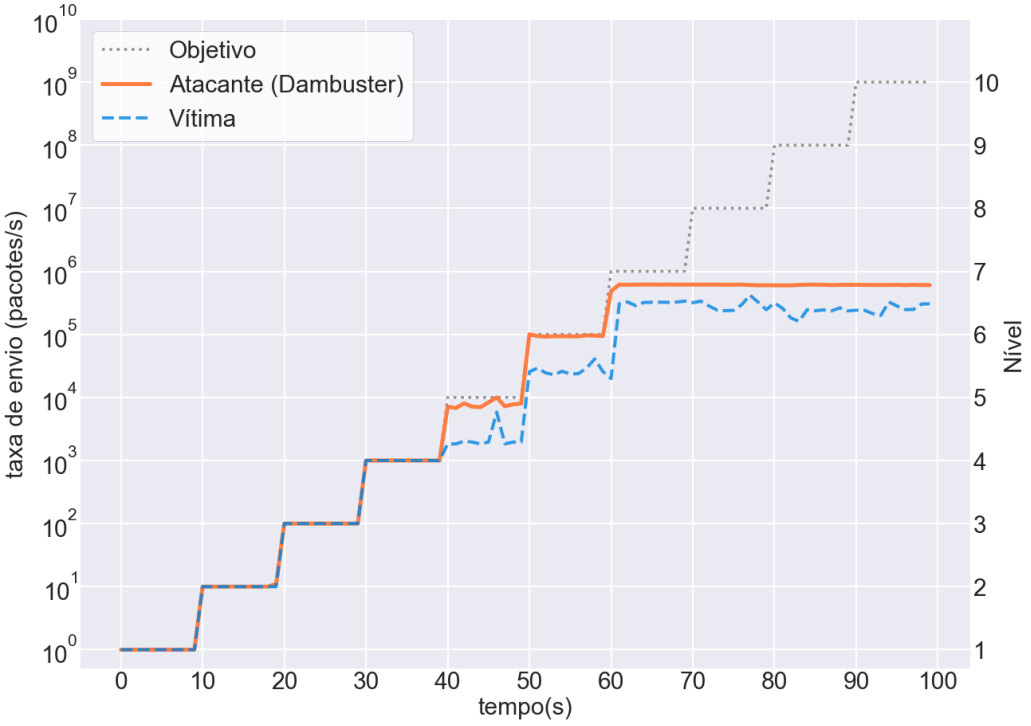


Figura 5.10: Cenário 2 - SYN Flood.

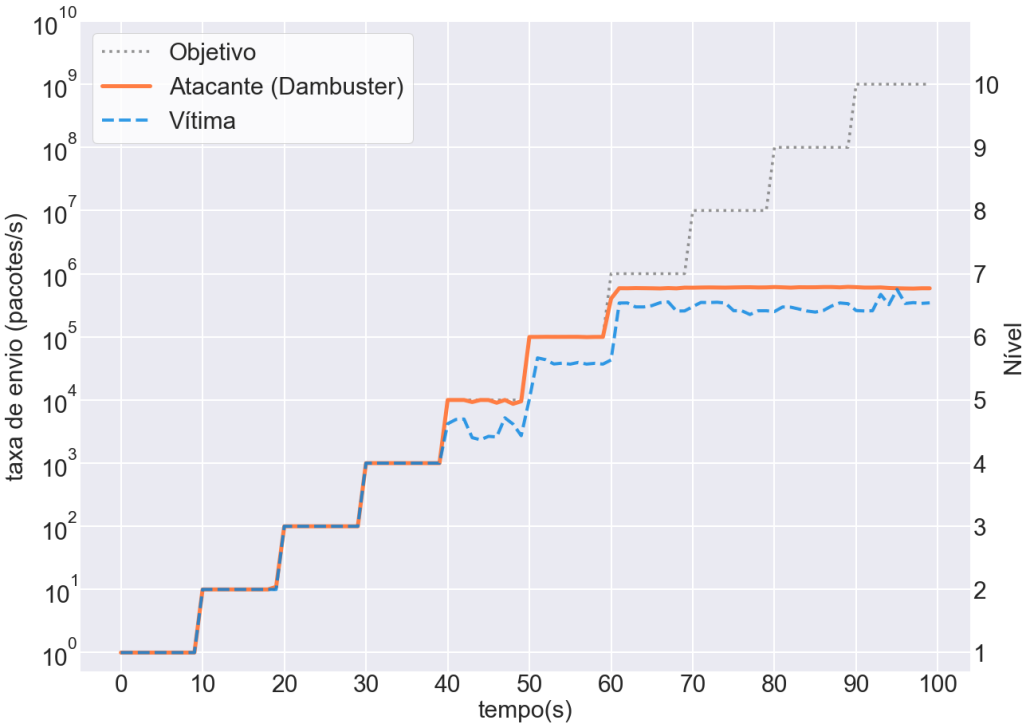


Figura 5.11: Cenário 2 - SYN/ACK Flood.

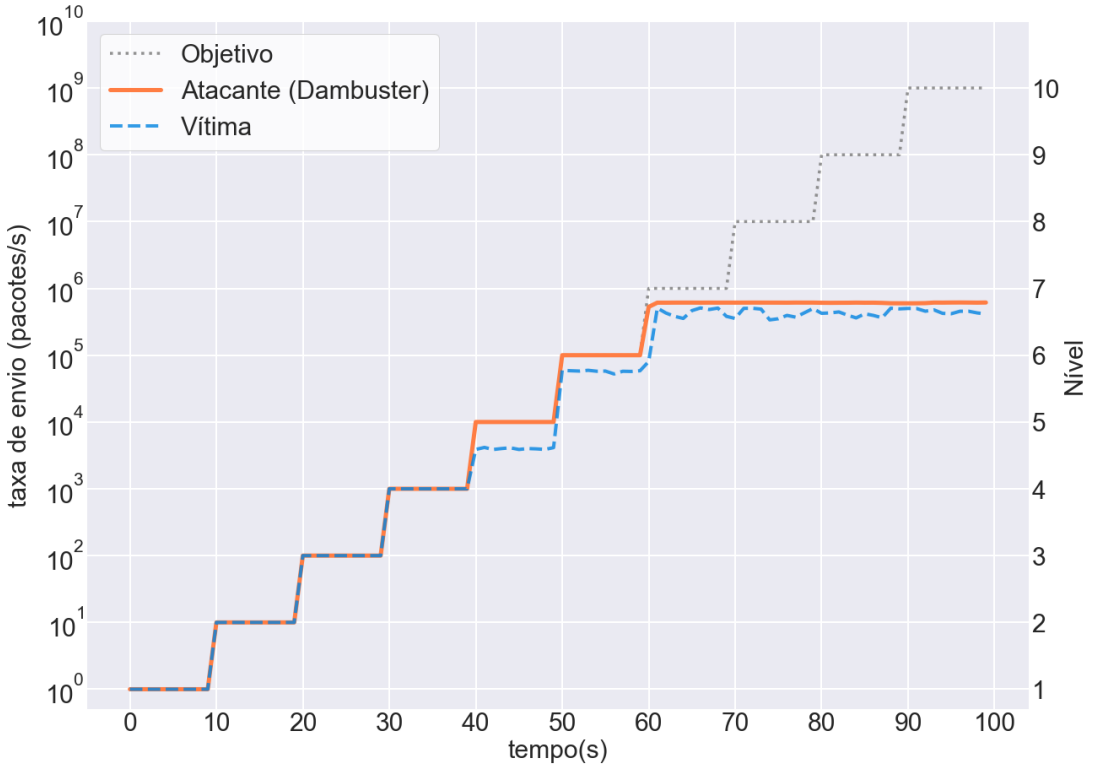


Figura 5.12: Cenário 2 - UDP Flood.

O desempenho da ferramenta no Cenário 2 foi muito superior ao Cenário 1, embora o nível de saturação seja o mesmo do cenário anterior. A quantidade de pacotes produzidos foi cerca de três vezes maior do que na máquina inferior, como demonstra o gráfico na Figura 5.13. No gráfico, os diversos ataques foram sobrepostos e formaram dois blocos distintos, referentes aos cenários, indicando a diferença de proporção entre os cenários e a consistência dos resultados entre os ataques. Assim, podemos concluir que a performance do Dambuster é proporcional ao hardware utilizado, indicando ótima escalabilidade na taxa de injeção de pacotes.

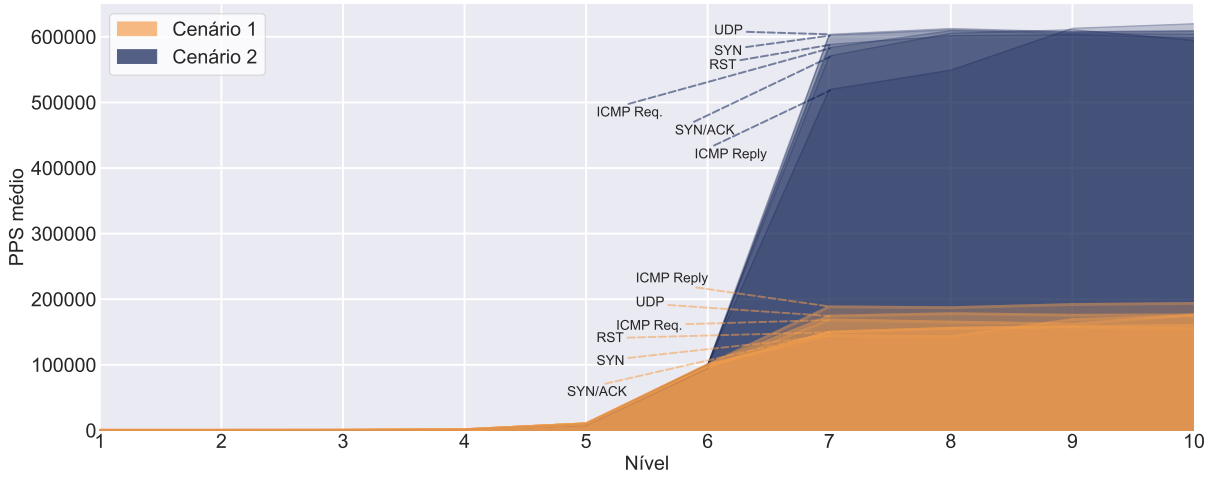


Figura 5.13: Comparação de produção de pacotes do Dambuster em máquinas diferentes.

Em termos de taxa de transferência de dados, o gráfico da Figura 5.14 também apresenta uma grande superioridade do cenário 2, além de evidenciar as diferenças já relatadas em relação aos protocolos. O gráfico mostra a taxa de vazão nos dois cenários. Os ataques foram sobrepostos e as cores indicam o protocolo. Para facilitar a interpretação, os ataques foram individualmente identificados. No cenário 2, a taxa de bits também foi maior nos ataques ao protocolo TCP. Essa diferença é respaldada pela arquitetura do Dambuster, que tem esforço similar para produzir pacotes independentemente do protocolo, mantendo a taxa de PPS próxima entre os ataques e a vazão em Mbps proporcional ao tamanho dos datagramas.

Além disso, os gráficos nas (Figuras 5.7 a 5.12) demonstram como um dispositivo atacante com melhores especificações do que a vítima é capaz de esgotar recursos e limitar a largura de banda de entrada do dispositivo alvo.

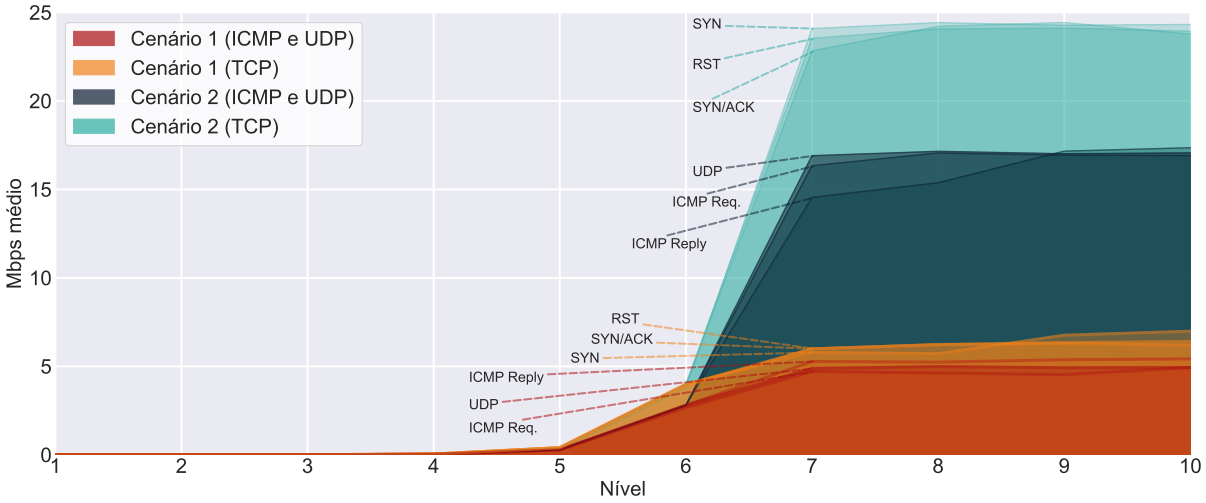


Figura 5.14: Comparação de vazão (Mbps) do Dambuster em máquinas diferentes.

### 5.2.1 Cenário 3

A ferramenta T50 foi executada durante 15 segundos para cada ataque. Este período foi considerado inteiramente como período de saturação. As métricas encontradas estão na Tabela 5.3.

Tabela 5.3: Cenário 3 - Métricas dos ataques efetuados pelo T50.

| **Ataque** | PPS | | Mbps | | Nível médio de saturação |
|---|---|---|---|---|---|
| | Máx | Média | Máx | Média | |
| **ICMP *Request*** | 148904 | 106288,40 | 4,17 | 2,98 | 7 |
| **ICMP *Reply*** | 148995 | 138300,87 | 4,17 | 3,87 | 7 |
| **RST** | 139060 | 138601,40 | 7,23 | 7,21 | 7 |
| **SYN** | 132054 | 103575,73 | 6,87 | 5,39 | 7 |
| **SYN/ACK** | 139130 | 133930,60 | 7,23 | 6,96 | 7 |
| **UDP** | 149030 | 136194,53 | 5,96 | 5,45 | 7 |

Nesse caso, é possível ver que todos os ataques testados saturaram no nível 7. O ataque RST Flood apresentou as melhores taxas em Mbps da ferramenta. Os ataques ao protocolo ICMP apresentaram um rendimento significantemente abaixo dos outros ataques. Entre os ataques ao protocolo TCP, o SYN Flood apresentou um rendimento bem abaixo dos outros. Vale ressaltar que no ataque RST Flood, o nível máximo antigido foi bem similar à média. Os resultados entre os ataques foi muito variado, o que pode ser explicado pelo o uso de técnicas de otimização diferentes para cada ataque ou protocolo.

Os testes realizados no Cenário 3 utilizando a ferramenta T50 apresentaram alguns resultados semelhantes aos encontrados no Cenário 1, que utilizou os mesmo dispositivos.

Todos os ataques saturaram no nível 7 e a ferramenta apresentou mais facilidade em produzir pacotes ICMP e UDP, porém a taxa de envio em Mbps foi maior em ataques do protocolo TCP, com exceção do SYN Flood que apresentou um desempenho menor que os outros, enquanto foi o ataque com a melhor taxa em Mbps do Dambuster. Essa diferença entre os ataques do T50 decorre de diferenças de implementação e de otimizações na produção e envio de pacotes de ataques distintos. Os gráficos das Figuras 5.15 a 5.16 comparam as taxas de envio do T50 com as do Dambuster, tanto em Mbps quanto em PPS.

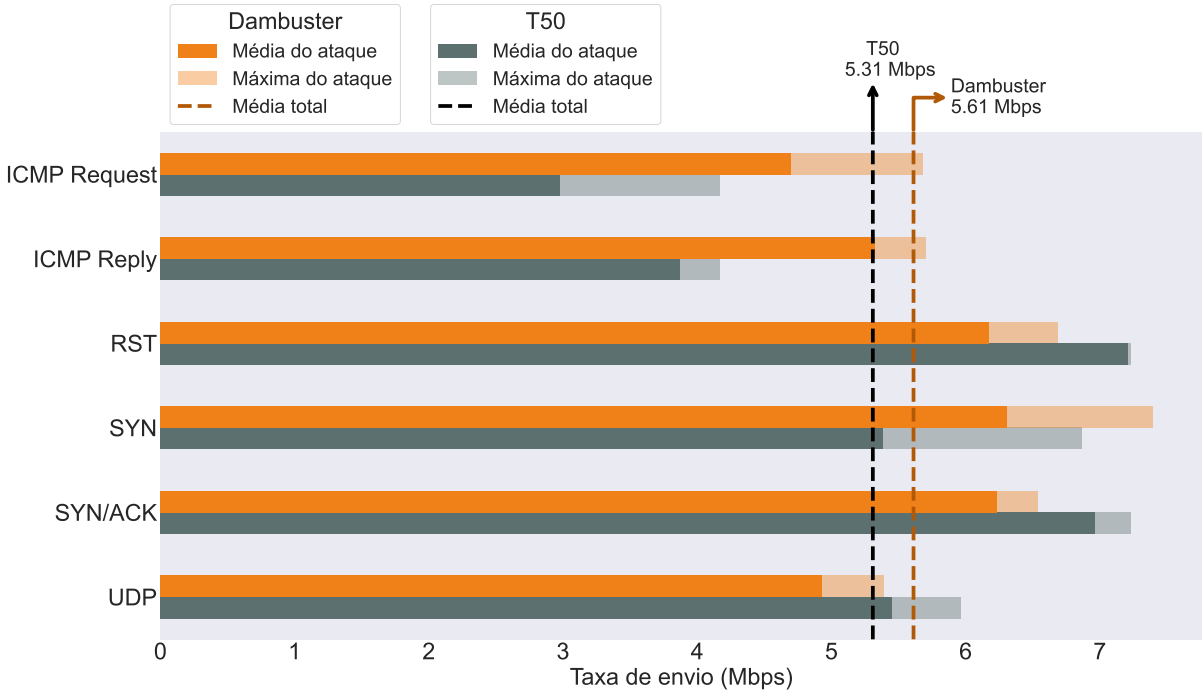


Figura 5.15: Comparação de taxas média e máxima de envio de bits (Mbps) entre Dambuster e T50.

Como apresentado na Figura 5.15, as diferenças nas taxas de transferência de dados foram pequenas entre as ferramentas, com a vantagem média sendo do Dambuster. O T50 apresentou vantagem nos ataques RST Flood, SYN/ACK Flood e UDP Flood. Já o Dambuster apresentou grandes vantagens nos ataques ICMP *Echo Request* Flood e ICMP *Echo Reply* Flood, além de uma pequena vantagem no SYN Flood.

Porém, como o Dambuster envia pacotes mais leves, por não conter informações no campo de dados, este apresentou uma vantagem extrema na taxa de produção de pacotes. A Figura 5.16 mostra que o Dambuster foi capaz de produzir, em média, cerca de 32% mais pacotes que o T50 e apresentou melhor desempenho em todos os ataques, tanto nas taxas máximas como médias.

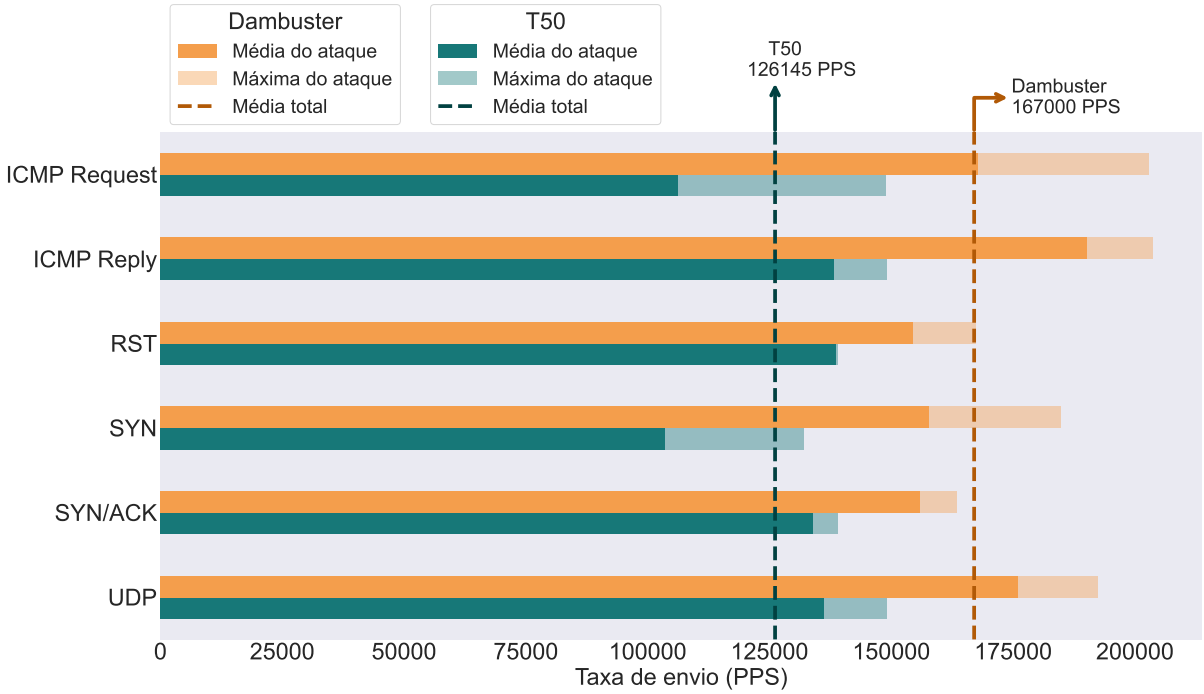


Figura 5.16: Comparação de taxas média e máxima de envio de pacotes (PPS) entre Dambuster e T50.

## 5.3 Considerações finais

Nos resultados apresentados nesta seção, em todos os cenários testados a saturação do dispositivo atacante ocorreu a partir do nível 7, com alguns sinais de saturação já no nível 6. Esse resultado é similar ao encontrado em [30] e [39] na análise de ataques de negação de serviço por reflexão, corroborando os resultados aqui obtidos.

Este capítulo apresentou e analisou os resultados dos testes realizados neste trabalho. Cabe ressaltar o desempenho comparável, e em alguns casos superior à ferramenta de referência escolhida, a ferramenta T50. A seguir, são apresentadas as conclusões e indicados alguns desdobramentos deste trabalho.

# Capítulo 6

# Conclusão

Neste trabalho, foram expostos os aspectos do desenvolvimento e da avaliação do Dambuster, a qual é uma ferramenta proposta neste trabalho para testes de sistemas de mitigação de ataques de negação de serviço volumétricos.

A arquitetura do Dambuster foi baseada no Linderhof ([8]; [9]), uma ferramenta para estudar ataques DoS por reflexão, uma proposta análoga à do Dambuster com ataques volumétricos. A sua estrutura extremamente modular permitiu a adaptação e o reuso de funções e implementações, diminuindo esforços de refatoração e otimização de código. O Dambuster seguiu a filosofia modular no seu desenvolvimento, expandindo e evoluindo o conceito proposto. Conclui-se então que a arquitetura escolhida foi bem projetada e, ao mantê-la e incrementá-la, viabilizará e facilitará as iterações futuras deste trabalho.

Os testes efetuados e seus resultados evidenciaram que o Dambuster é uma ferramenta que pode ser utilizada para simular ataques DoS e avaliar sistemas de mitigação. Todos os ataques testados tiveram resultados semelhantes e consistentes, mostrando a eficácia da estrutura modular da aplicação. A comparação do Dambuster com o T50, um software semelhante, indicou a viabilidade da versão atual. O Dambuster obteve resultados superiores nas métricas propostas, alcançando em torno de 40.000 PPS de diferença, isso é produziu em média 132% da produção do T50.

Além disso, foram identificadas formas de melhorar o desempenho em atualizações futuras, por meio de técnicas presentes no T50. Por exemplo, a taxa de envio de bits pode ser aumentada adicionando conteúdo aleatório ao campo de dados do pacote. Já a taxa de produção de pacotes pode ser melhorada, mesmo ao adicionar maior aleatoriedade aos campos, ignorando o cálculo do *checksum* dos pacotes.

Os testes também ressaltaram o impacto do hardware da máquina atacante nas taxas de injeção de pacotes. Ao alterar o computador utilizado, o Dambuster apresentou resultados aproximadamente 200% melhores, chegando à taxas maiores que 600.000 pacotes por segundo. Isso demonstra a escalabilidade da ferramenta, de maneira que dispositivos

atacantes de alto desempenho podem ser utilizados para testar os extremos de sistemas de segurança.

A implementação atual do Dambuster se mostrou viável para a produção de tráfego de pacotes de rede, simulando ataques de negação de serviço. Dessa forma, a ferramenta pode ser utilizada para testar soluções de mitigação e segurança, produzindo tráfego semelhante a um ataque real. Além de um desempenho satisfatório, o Dambuster permite várias opções de configuração, contendo uma gama de ataques populares e funcionalidades que emulam técnicas reais de ataques DoS, e proporcionando um alto grau de customização para os usuários. A imagem Docker produzida permite, ainda, a execução da ferramenta em ambientes na nuvem, um grande diferencial em relação a outras ferramentas.

Trabalhos futuros podem abordar as limitações do Dambuster aqui evidenciadas. O desempenho pode ser melhorado adaptando técnicas encontradas no T50. Mais funcionalidades e opções para os ataques podem ser incluídas, combatendo diretamente técnicas de mitigação utilizadas por sistemas de segurança. Futuras iterações podem, também, adaptar a aplicação para ataques distribuídos (DDoS), amplificando os ataques ao utilizar múltiplos dispositivos simultaneamente.

# Referências


[1] Sharafaldin, Iman, Arash Habibi Lashkari, Saqib Hakak e Ali Ghorbani: *Developing realistic distributed denial of service (ddos) attack dataset and taxonomy.* páginas 1–8, outubro 2019. 1

[2] Toh, Alethea: *Azure ddos protection-2021 q3 and q4 ddos attack trends.* `https://azure.microsoft.com/en-us/blog/azure-ddos-protection-2021-q3-and-q4-ddos-attack-trends/`, Acesso em 21 abr. 2022. 1

[3] *An on-premises defense is the cornerstone for multilayer ddos protection.* White paper, NETSCOUT, April 2022. `https://www.netscout.com/sites/default/files/2022-04/NETSCOUT-AnOn-PremisesDefenseistheFoundation.pdf`. 1, 2

[4] Cao, Yuan, Yuan Gao, Rongjun Tan, Qingbang Han e Zhuotao Liu: *Understanding internet ddos mitigation from academic and industrial perspectives.* IEEE Access, 6:66641–66648, 2018. 1, 6

[5] Menscher, Damian: *Identifying and protecting against the largest ddos attacks | google cloud blog*, Oct 2020. `https://cloud.google.com/blog/products/identity-security/identifying-and-protecting-against-the-largest-ddos-attacks`, Acesso em 21 abr. 2022. 1, 2

[6] Biolchini, Jorge, P. Gomes Mian, A. Candida Cruz Natali e G. Horta Travassos: *Systematic review in software engineering.* System Engineering and Computer Science Department COPPE/UFRJ, Technical Report ES, 679(05), 2005. 1, 8, 9, 10

[7] Khalaf, Bashar Ahmed, Salama A. Mostafa, Aida Mustapha, Mazin Abed Mohammed e Wafaa Mustafa Abduallah: *Comprehensive review of artificial intelligence and statistical approaches in distributed denial of service attack and defense methods.* IEEE Access, 7:51691–51713, 2019. 2

[8] Dantas, Amanda, Matheus Vieira, Alan Vasques e João Gondim: *Linderhof: uma ferramenta para avaliação de sistemas de mitigação de ataques reflexivos volumétricos (ddos).* Em *Anais Estendidos do XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, páginas 25–32, Porto Alegre, RS, Brasil, 2020. SBC. `https://sol.sbc.org.br/index.php/sbrc_estendido/article/view/12398`. 2, 13, 43

[9] Vieira, Matheus, Amanda Dantas, Alan Vasques e João Gondim: *Linderhof v2.0.0*. Em *Anais Estendidos do XXI Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, páginas 9–17, Porto Alegre, RS, Brasil, 2021. SBC. https://sol.sbc.org.br/index.php/sbseg_estendido/article/view/17334. 2, 13, 43

[10] Lau, Felix, Soklin Va, Stuart Rubin, M.H. Smith e Ljiljana Trajkovic: *Distributed denial of service attacks*. março 2001. 4, 8, 10

[11] Mahajan, Deepika e Monika Sachdeva: *Ddos attack prevention and mitigation techniques-a review*. International Journal of Computer Applications, 67(19), 2013. 4, 8

[12] Peng, Tao, Christopher Leckie e Kotagiri Ramamohanarao: *Survey of network-based defense mechanisms countering the dos and ddos problems*. ACM Comput. Surv., 39(1):3–es, apr 2007, ISSN 0360-0300. https://doi.org/10.1145/1216370.1216373. 4, 5, 6, 8, 10

[13] Stone, Robert: *Centertrack: An ip overlay network for tracking dos floods*. USENIX Security Symposium, 9, janeiro 2000. 4

[14] Gondim, João JC, Robson de Oliveira Albuquerque e Ana Lucila Sandoval Orozco: *Mirror saturation in amplified reflection distributed denial of service: A case of study using snmp, ssdp, ntp and dns protocols*. Future Generation Computer Systems, 108:68–81, 2020. 4, 5

[15] Hussain, Alefiya, John Heidemann e Christos Papadopoulos: *A framework for classifying denial of service attacks*. Em *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, páginas 99–110, 2003. 4

[16] Filho, Francisco, Frederico Silveira, Agostinho Junior, Genoveva vargas solar e Luiz Silveira: *Smart detection: An online approach for dos/ddos attack detection using machine learning*. Security and Communication Networks, 2019:1–15, outubro 2019. 5

[17] Douligeris, Christos e Aikaterini Mitrokotsa: *Ddos attacks and defense mechanisms: a classification*. páginas 190 – 193, janeiro 2004, ISBN 0-7803-8292-7. 6

[18] Moore, David, Geoffrey M. Voelker e Stefan Savage: *Inferring internet Denial-of-Service activity*. Em *10th USENIX Security Symposium (USENIX Security 01)*, Washington, D.C., agosto 2001. USENIX Association. https://www.usenix.org/conference/10th-usenix-security-symposium/inferring-internet-denial-service-activity. 6, 8

[19] Ali, Farha: *Ip spoofing*. The Internet Protocol Journal, 10(4):1–9, 2007. 6

[20] Ehrenkranz, Toby e Jun Li: *On the state of ip spoofing defense*. ACM Trans. Internet Technol., 9(2), may 2009, ISSN 1533-5399. https://doi.org/10.1145/1516539.1516541. 6

[21] Mirkovic, Jelena e Peter Reiher: *A taxonomy of ddos attack and ddos defense mechanisms*. ACM SIGCOMM Computer Communication Review, 34(2):39–53, 2004. 6, 7, 8

[22] Bjarnason, Steinthor: *Withstanding the infinite: Ddos defense in the terabit era*. Presentation at NANOG-74, October, 2018. 7

[23] Bhardwaj, Aanshi, Veenu Mangat, Vig Renu, Subir Halder e Mauro Conti: *Distributed denial of service attacks in cloud: State-of-the-art of scientific and commercial solutions*. Computer Science Review, 39:100332, fevereiro 2021. 7

[24] Postel, Jon: *Transmission control protocol*. Std 7, RFC Editor, September 1981. `http://www.rfc-editor.org/rfc/rfc793.txt`. 8, 9

[25] Postel, J.: *User datagram protocol*. Std 6, RFC Editor, August 1980. `http://www.rfc-editor.org/rfc/rfc768.txt`. 10

[26] Postel, J.: *Internet control message protocol*. Std 5, RFC Editor, September 1981. `http://www.rfc-editor.org/rfc/rfc792.txt`. 10

[27] Chauhan, Varun e Pranav Saini: *ICMP Flood Attacks: A Vulnerability Analysis*, páginas 261–268. janeiro 2018, ISBN 978-981-10-8535-2. 10

[28] Sonar, Krushang e Hardik Upadhyay: *A survey: Ddos attack on internet of things*. International Journal of Engineering Research and Development, 10(11):58–63, 2014. 11

[29] Dildy, Douglas C: *Dambusters: Operation Chastise 1943*. Bloomsbury Publishing, 2012. 12

[30] Vieira, Matheus de Oliveira: *Ataque de negação de serviço por reflexão amplificada sobre CLDAP utilizando a ferramenta Linderhof*. Trabalho de conclusão de curso (bacharelado em engenharia da computação), Universidade de Brasília, Brasília, 2021. `https://bdm.unb.br/handle/10483/29879`. 12, 13, 42

[31] *The gtk project - a free and open-source cross-platform widget toolkit*. `https://www.gtk.org/`, Acesso em 09 abr. 2022. 22

[32] *Extensible markup language (xml)*. `https://www.w3.org/XML/`, Acesso em 09 abr. 2022. 23

[33] *Docker documentation*. `https://docs.docker.com/`, Acesso em 09 abr. 2022. 23

[34] `https://gitlab.com/fredericopissarra/t50`, Acesso em 09 abr. 2022. 26, 27

[35] `https://www.wireshark.org/`, Acesso em 09 abr. 2022. 28

[36] `https://jupyter.org/`, Acesso em 09 abr. 2022. 28

[37] team, The pandas development: *pandas-dev/pandas: Pandas*, fevereiro 2020. `https://doi.org/10.5281/zenodo.3509134`. 28

[38] McKinney Wes: *Data Structures for Statistical Computing in Python*. Em Walt Stéfan van der e Jarrod Millman (editores): *Proceedings of the 9th Python in Science Conference*, páginas 56 – 61, 2010. 28

[39] Costa Gondim, João José, Robson de Oliveira Albuquerque, Anderson Clayton Alves Nascimento, Luis Javier 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. 42

# Apêndice A

# Script de teste e captura do Dambuster

```
### Test configuration
CONFIG_FILE = "dbst.conf"
TARGET = "192.168.1.127"
INTERFACE = "1"
#List of attacks and protocols
BOMBS = [("ICMPFLOOD", "icmp"), ("ICMPREPLYFLOOD", "icmp"),
        ("RSTFLOOD", "tcp"), ("SYNFLOOD", "tcp"),
        ("SYNACKFLOOD", "tcp"), ("UDPFLOOD", "udp")]
#Runs capture and dambuster for each attack
for bomb, protocol in BOMBS:
    # Create dumpcap(Wireshark) filter
    filter_ = create_dc_filter(protocol, TARGET)
    # Create capture CLI command
    dc_cmd = "dumpcap -q -i {} -P -w output/{}_host.pcap -f {}".format(
   INTERFACE, bomb, filter_)

    #Setup dambuster cli command
    dbst_args = [
        ('-c', CONFIG_FILE), ('-t', TARGET),
        ('-b', bomb), ('-o', "output/{}_log.xml".format(bomb))
    ]
    args_ = join_args(dbst_args)
    dbst_cmd = "../bin/dbst {}".format(args_)

    #Starts processes
    dumpcap = subprocess.Popen(dc_cmd, shell=True)
    dambuster = subprocess.Popen(dbst_cmd, shell=True)
    #Waits for attack to end and kills capture process
    dambuster.wait(); dumpcap.terminate()
```

# Apêndice B

# Script de teste e captura do T50

```
### run_t50.sh
dumpcap -q -i 1 -P -w $1 -f "$2" &
t50 192.168.1.127 --flood $3 &
sleep 15 && pkill -P $$

killall dumpcap & killall t50
wait
```

```
### Attacking script
#RST
./run_t50.sh t_50_rst.pcap "(tcp) && (dst host 192.168.1.127)" "--
    protocol TCP --rst"
#SYN
./run_t50.sh t_50_syn.pcap "(tcp) && (dst host 192.168.1.127)" "--
    protocol TCP --syn"
#SYNACK
./run_t50.sh t_50_synack.pcap "(tcp) && (dst host 192.168.1.127)" "--
    protocol TCP --syn --ack"
#UDP
./run_t50.sh t_50_udp.pcap "(udp) && (dst host 192.168.1.127)" "--
    protocol UDP"
#ICMPREQUEST
./run_t50.sh t_50_icmp.pcap "(icmp) && (dst host 192.168.1.127)" "--
    protocol ICMP"
#ICMPREPLY
./run_t50.sh t_50_icmp_reply.pcap "(icmp) && (dst host 192.168.1.127)" "
    --protocol ICMP --icmp-type 0"
```

# Apêndice C

# Dados coletados nos testes

Os dados apresentados nas Tabelas C.1 a C.2 foram obtidos durante os testes do Dambuster.

São apresentados os valores médios por nível das taxas de envio do atacante e das taxas de recebimento da vítima. As taxas estão em pacotes por segundo (PPS) e megabits por segundo (Mbps). As taxas de entrada da vítima podem ser afetadas por atrasos na transmissão de dados, o que causa um aumento na taxa de recebimento do nível seguinte, podendo ser maior que a taxa de envio do atacante.

Além disso, são apresentados os valores esperados por nível em PPS e as eficiências do atacante e da vítima. As eficiências representam a proporção entre a taxa de pacotes aferida nos testes e a taxa de pacotes esperada por nível. As equações utilizadas estão nas Equações C.1 a C.2.

$$\text{Eff}_{out} = \frac{\text{PPS}_{Nível}}{\text{PPS}_{Saída}} \tag{C.1}$$

$$\text{Eff}_{in} = \frac{\text{PPS}_{Nível}}{\text{PPS}_{Entrada}} \tag{C.2}$$

Tabela C.1: Cenário 1 - Valores médios por nível.

| Ataque | Nível | | Saída (atacante) | | | Entrada (vítima) | | |
|---|---|---|---|---|---|---|---|---|
| | $L$ | PPS | PPS | Mbps | $\text{Eff}_{out}(\%)$ | PPS | Mbps | $\text{Eff}_{in}(\%)$ |
| **ICMP** ***Echo Request*** | 1 | 1 | 1 | ≤,01 | 100,00% | 1 | ≤,01 | 100,00% |
| | 2 | 10 | 10 | ≤,01 | 100,00% | 10 | ≤,01 | 100,00% |
| | 3 | $10^2$ | 100 | ≤,01 | 100,00% | 100 | ≤,01 | 100,00% |
| | 4 | $10^3$ | 1000 | 0,03 | 100,00% | 912 | 0,03 | 91,24% |
| | 5 | $10^4$ | 10000 | 0,28 | 100,00% | 9578 | 0,27 | 95,78% |
| | 6 | $10^5$ | 95095 | 2,66 | 95,10% | 98227 | 2,75 | 98,23% |
| | 7 | $10^6$ | 168354 | 4,71 | 16,84% | 171608 | 4,81 | 17,16% |
| | 8 | $10^7$ | 165122 | 4,62 | 1,65% | 161148 | 4,51 | 1,61% |
| | 9 | $10^8$ | 161868 | 4,53 | 0,16% | 167119 | 4,68 | 0,17% |
| | 10 | $10^9$ | 176102 | 4,93 | 0,02% | 173681 | 4,86 | 0,02% |
| **ICMP** ***Echo Reply*** | 1 | 1 | 1 | ≤,01 | 100,00% | 1 | ≤,01 | 100,00% |
| | 2 | 10 | 10 | ≤,01 | 100,00% | 10 | ≤,01 | 100,00% |
| | 3 | $10^2$ | 100 | ≤,01 | 100,00% | 100 | ≤,01 | 100,00% |
| | 4 | $10^3$ | 1000 | 0,03 | 100,00% | 957 | 0,03 | 95,74% |
| | 5 | $10^4$ | 10000 | 0,28 | 100,00% | 5362 | 0,15 | 53,62% |
| | 6 | $10^5$ | 100000 | 2,80 | 100,00% | 40379 | 1,13 | 40,38% |
| | 7 | $10^6$ | 188364 | 5,27 | 18,84% | 71208 | 1,99 | 7,12% |
| | 8 | $10^7$ | 187031 | 5,24 | 1,87% | 74314 | 2,08 | 0,74% |
| | 9 | $10^8$ | 191882 | 5,37 | 0,19% | 100340 | 2,81 | 0,10% |
| | 10 | $10^9$ | 193541 | 5,42 | 0,02% | 71227 | 1,99 | 0,01% |
| **RST** | 1 | 1 | 1 | ≤,01 | 100,00% | 1 | ≤,01 | 100,00% |
| | 2 | 10 | 10 | ≤,01 | 100,00% | 10 | ≤,01 | 100,00% |
| | 3 | $10^2$ | 100 | ≤,01 | 100,00% | 100 | ≤,01 | 100,00% |
| | 4 | $10^3$ | 1000 | 0,04 | 100,00% | 1000 | 0,04 | 100,00% |
| | 5 | $10^4$ | 10000 | 0,40 | 100,00% | 10000 | 0,40 | 100,00% |
| | 6 | $10^5$ | 99036 | 3,96 | 99,04% | 91615 | 3,66 | 91,62% |
| | 7 | $10^6$ | 149678 | 5,99 | 14,97% | 149736 | 5,99 | 14,97% |
| | 8 | $10^7$ | 155763 | 6,23 | 1,56% | 155470 | 6,22 | 1,55% |
| | 9 | $10^8$ | 157184 | 6,29 | 0,16% | 157336 | 6,29 | 0,16% |
| | 10 | $10^9$ | 154908 | 6,20 | 0,02% | 155058 | 6,20 | 0,02% |

(continua)

(conclusão)

| Ataque | Nível | | Saída (atacante) | | | Entrada (vítima) | | |
|---|---|---|---|---|---|---|---|---|
| | $L$ | PPS | PPS | Mbps | $\text{Eff}_{out}(\%)$ | PPS | Mbps | $\text{Eff}_{in}(\%)$ |
| **SYN** | 1 | 1 | 1 | ≤,01 | 100,00% | 1 | ≤,01 | 100,00% |
| | 2 | 10 | 10 | ≤,01 | 100,00% | 10 | ≤,01 | 100,00% |
| | 3 | $10^2$ | 100 | ≤,01 | 100,00% | 100 | ≤,01 | 100,00% |
| | 4 | $10^3$ | 1000 | 0,04 | 100,00% | 1000 | 0,04 | 100,00% |
| | 5 | $10^4$ | 10000 | 0,40 | 100,00% | 10000 | 0,40 | 100,00% |
| | 6 | $10^5$ | 100000 | 4,00 | 100,00% | 93796 | 3,75 | 93,80% |
| | 7 | $10^6$ | 144597 | 5,78 | 14,46% | 143427 | 5,74 | 14,34% |
| | 8 | $10^7$ | 142814 | 5,71 | 1,43% | 142029 | 5,68 | 1,42% |
| | 9 | $10^8$ | 168877 | 6,76 | 0,17% | 169006 | 6,76 | 0,17% |
| | 10 | $10^9$ | 174559 | 6,98 | 0,02% | 174257 | 6,97 | 0,02% |
| **SYN/ACK** | 1 | 1 | 1 | ≤,01 | 100,00% | 1 | ≤,01 | 100,00% |
| | 2 | 10 | 10 | ≤,01 | 100,00% | 10 | ≤,01 | 100,00% |
| | 3 | $10^2$ | 100 | ≤,01 | 100,00% | 100 | ≤,01 | 100,00% |
| | 4 | $10^3$ | 1000 | 0,04 | 100,00% | 1000 | 0,04 | 100,00% |
| | 5 | $10^4$ | 10000 | 0,40 | 100,00% | 10000 | 0,40 | 100,00% |
| | 6 | $10^5$ | 97341 | 3,89 | 97,34% | 100000 | 4,00 | 100,00% |
| | 7 | $10^6$ | 149701 | 5,99 | 14,97% | 155090 | 6,20 | 15,51% |
| | 8 | $10^7$ | 155466 | 6,22 | 1,55% | 155548 | 6,22 | 1,56% |
| | 9 | $10^8$ | 158538 | 6,34 | 0,16% | 158610 | 6,34 | 0,16% |
| | 10 | $10^9$ | 159899 | 6,40 | 0,02% | 159602 | 6,38 | 0,02% |
| **UDP** | 1 | 1 | 1 | ≤,01 | 100,00% | 1 | ≤,01 | 100,00% |
| | 2 | 10 | 10 | ≤,01 | 100,00% | 10 | ≤,01 | 100,00% |
| | 3 | $10^2$ | 100 | ≤,01 | 100,00% | 100 | ≤,01 | 100,00% |
| | 4 | $10^3$ | 1000 | 0,03 | 100,00% | 1000 | 0,03 | 100,00% |
| | 5 | $10^4$ | 10000 | 0,28 | 100,00% | 10000 | 0,28 | 100,00% |
| | 6 | $10^5$ | 100000 | 2,80 | 100,00% | 97223 | 2,72 | 97,22% |
| | 7 | $10^6$ | 174204 | 4,88 | 17,42% | 168266 | 4,71 | 16,83% |
| | 8 | $10^7$ | 177914 | 4,98 | 1,78% | 176894 | 4,95 | 1,77% |
| | 9 | $10^8$ | 175556 | 4,92 | 0,18% | 177052 | 4,96 | 0,18% |
| | 10 | $10^9$ | 176069 | 4,93 | 0,02% | 175265 | 4,91 | 0,02% |

Tabela C.2: Cenário 2 - Valores médios por nível.

| Ataque | Nível | | Saída (atacante) | | | Entrada (vítima) | | |
|---|---|---|---|---|---|---|---|---|
| | $L$ | PPS | PPS | Mbps | $\text{Eff}_{out}(\%)$ | PPS | Mbps | $\text{Eff}_{in}(\%)$ |
| **ICMP** ***Echo Request*** | 1 | 1 | 1 | ≤,01 | 100,00% | 1 | ≤,01 | 100,00% |
| | 2 | 10 | 10 | 0,00 | 100,00% | 10 | 0,00 | 100,00% |
| | 3 | $10^2$ | 100 | 0,00 | 100,00% | 100 | 0,00 | 100,00% |
| | 4 | $10^3$ | 1000 | 0,03 | 100,00% | 1000 | 0,03 | 100,00% |
| | 5 | $10^4$ | 9665 | 0,27 | 96,65% | 2420 | 0,07 | 24,20% |
| | 6 | $10^5$ | 99541 | 2,79 | 99,54% | 32432 | 0,91 | 32,43% |
| | 7 | $10^6$ | 583318 | 16,33 | 58,33% | 385866 | 10,80 | 38,59% |
| | 8 | $10^7$ | 609256 | 17,06 | 6,09% | 362911 | 10,16 | 3,63% |
| | 9 | $10^8$ | 604761 | 16,93 | 0,60% | 362984 | 10,16 | 0,36% |
| | 10 | $10^9$ | 603908 | 16,91 | 0,06% | 365112 | 10,22 | 0,04% |
| **ICMP** ***Echo Reply*** | 1 | 1 | 1 | ≤,01 | 100,00% | 1 | ≤,01 | 100,00% |
| | 2 | 10 | 10 | 0,00 | 100,00% | 10 | 0,00 | 100,00% |
| | 3 | $10^2$ | 100 | 0,00 | 100,00% | 100 | 0,00 | 100,00% |
| | 4 | $10^3$ | 1000 | 0,03 | 100,00% | 1000 | 0,03 | 100,00% |
| | 5 | $10^4$ | 10000 | 0,28 | 100,00% | 4445 | 0,12 | 44,45% |
| | 6 | $10^5$ | 100000 | 2,80 | 100,00% | 64906 | 1,82 | 64,91% |
| | 7 | $10^6$ | 519422 | 14,54 | 51,94% | 363866 | 10,19 | 36,39% |
| | 8 | $10^7$ | 549094 | 15,37 | 5,49% | 423277 | 11,85 | 4,23% |
| | 9 | $10^8$ | 612850 | 17,16 | 0,61% | 390623 | 10,94 | 0,39% |
| | 10 | $10^9$ | 619816 | 17,35 | 0,06% | 369364 | 10,34 | 0,04% |
| **RST** | 1 | 1 | 1 | ≤,01 | 100,00% | 1 | ≤,01 | 100,00% |
| | 2 | 10 | 10 | 0,00 | 101,00% | 10 | 0,00 | 100,00% |
| | 3 | $10^2$ | 100 | 0,00 | 100,00% | 100 | 0,00 | 100,00% |
| | 4 | $10^3$ | 1000 | 0,04 | 100,00% | 1000 | 0,04 | 100,00% |
| | 5 | $10^4$ | 9903 | 0,40 | 99,03% | 3813 | 0,15 | 38,13% |
| | 6 | $10^5$ | 100000 | 4,00 | 100,00% | 51398 | 2,06 | 51,40% |
| | 7 | $10^6$ | 588272 | 23,53 | 58,83% | 349636 | 13,99 | 34,96% |
| | 8 | $10^7$ | 601578 | 24,06 | 6,02% | 355466 | 14,22 | 3,55% |
| | 9 | $10^8$ | 602920 | 24,12 | 0,60% | 362528 | 14,50 | 0,36% |
| | 10 | $10^9$ | 598487 | 23,94 | 0,06% | 396594 | 15,86 | 0,04% |

(continua)

(conclusão)

| Ataque | Nível | | Saída (atacante) | | | Entrada (vítima) | | |
|---|---|---|---|---|---|---|---|---|
| | $L$ | PPS | PPS | Mbps | $\text{Eff}_{out}(\%)$ | PPS | Mbps | $\text{Eff}_{in}(\%)$ |
| **SYN** | 1 | 1 | 1 | ≤,01 | 100,00% | 1 | ≤,01 | 100,00% |
| | 2 | 10 | 10 | 0,00 | 101,00% | 10 | 0,00 | 100,00% |
| | 3 | $10^2$ | 100 | 0,00 | 100,00% | 100 | 0,00 | 100,00% |
| | 4 | $10^3$ | 1000 | 0,04 | 100,00% | 1000 | 0,04 | 100,00% |
| | 5 | $10^4$ | 7749 | 0,31 | 77,49% | 2304 | 0,09 | 23,04% |
| | 6 | $10^5$ | 94453 | 3,78 | 94,45% | 27126 | 1,09 | 27,13% |
| | 7 | $10^6$ | 602212 | 24,09 | 60,22% | 288989 | 11,56 | 28,90% |
| | 8 | $10^7$ | 610579 | 24,42 | 6,11% | 294039 | 11,76 | 2,94% |
| | 9 | $10^8$ | 606853 | 24,27 | 0,61% | 238008 | 9,52 | 0,24% |
| | 10 | $10^9$ | 608029 | 24,32 | 0,06% | 260691 | 10,43 | 0,03% |
| **SYN/ACK** | 1 | 1 | 1 | ≤,01 | 100,00% | 1 | ≤,01 | 100,00% |
| | 2 | 10 | 10 | 0,00 | 101,00% | 10 | 0,00 | 100,00% |
| | 3 | $10^2$ | 100 | 0,00 | 100,00% | 100 | 0,00 | 100,00% |
| | 4 | $10^3$ | 1000 | 0,04 | 100,00% | 1000 | 0,04 | 100,00% |
| | 5 | $10^4$ | 9653 | 0,39 | 96,53% | 3632 | 0,15 | 36,32% |
| | 6 | $10^5$ | 99700 | 3,99 | 99,70% | 36390 | 1,46 | 36,39% |
| | 7 | $10^6$ | 570748 | 22,83 | 57,07% | 285560 | 11,42 | 28,56% |
| | 8 | $10^7$ | 605383 | 24,22 | 6,05% | 295530 | 11,82 | 2,96% |
| | 9 | $10^8$ | 610706 | 24,43 | 0,61% | 287017 | 11,48 | 0,29% |
| | 10 | $10^9$ | 594410 | 23,78 | 0,06% | 350190 | 14,01 | 0,04% |
| **UDP** | 1 | 1 | 1 | ≤,01 | 100,00% | 1 | ≤,01 | 100,00% |
| | 2 | 10 | 10 | 0,00 | 100,00% | 10 | 0,00 | 100,00% |
| | 3 | $10^2$ | 100 | 0,00 | 100,00% | 100 | 0,00 | 100,00% |
| | 4 | $10^3$ | 1000 | 0,03 | 100,00% | 1000 | 0,03 | 100,00% |
| | 5 | $10^4$ | 10000 | 0,28 | 100,00% | 3990 | 0,11 | 39,90% |
| | 6 | $10^5$ | 100000 | 2,80 | 100,00% | 57598 | 1,61 | 57,60% |
| | 7 | $10^6$ | 603712 | 16,90 | 60,37% | 409886 | 11,48 | 40,99% |
| | 8 | $10^7$ | 612532 | 17,15 | 6,13% | 423534 | 11,86 | 4,24% |
| | 9 | $10^8$ | 607583 | 17,01 | 0,61% | 422977 | 11,84 | 0,42% |
| | 10 | $10^9$ | 609430 | 17,06 | 0,06% | 452655 | 12,67 | 0,05% |