Uma Comparação de Ferramentas Open Source de CLI para Geração de Ataques DDoS em Ambientes de Internet

Arthur Ferreira[2], **Carlos Pedroso**[1], **Agnaldo Batista**[1], **Aldri Santos**[1,2]

[1]Núcleo de Redes Sem-Fio e Redes Avançadas (NR2) – Dinf – UFPR – Brasil

[2]Center for Computational Security sCience (CCSC) – DCC – UFMG – Brasil

arthur.guimaraes.ferreira@dcc.ufmg.br, {capjunior, asbatista}@inf.ufpr.br, aldri@dcc.ufmg.br

Resumo. *Os ataques DDoS comprometem a disponibilidade de infraestruturas críticas de rede por meio de fluxos de dados massivos e coordenados. O uso de ferramentas de simulação de DDoS permite reproduzir esses comportamentos de forma controlada e segura, colaborando assim ao desenvolvimento de sistemas de segurança. Entretanto, há ainda uma carência de estudos comparativos que auxiliem na seleção de ferramentas quanto à fidelidade e ao desempenho. Este trabalho apresenta uma análise comparativa de sete ferramentas open source de linha de comando (CLI) voltadas à simulação de ataques DDoS nas camadas de rede e transporte. Essa avaliação experimental levou em conta a capacidade de geração de tráfego, a capacidade de customização e a aderência à taxonomia de Jelena Mirkovic para modelagem de anomalias. Os resultados alcançados apontam diferenças significativas entre as ferramentas, no qual a T50 favorece experimentos volumétricos, a Scapy a flexibilidade na modelagem de anomalias e a Trafgen um equilíbrio entre ambos aspectos.*

Abstract. *DDoS attacks compromise the availability of critical infrastructure through massive and coordinated data flows. The adoption of DDoS simulation tools enables the controlled and secure reproduction of these behaviors, supporting the development of security systems. Despite existing evaluations, there is a lack of systematized comparative data to aid in the selection of tools based on fidelity and performance. This work presents a comparative analysis of seven open source command-line interface (CLI) tools for simulating DDoS attacks at the network and transport layers. The evaluation among the tools took into account traffic generation capacity, customization capabilities, and adherence to Jelena Mirkovic's taxonomy for anomaly modeling. Results point out significant differences among the tools, with T50 supporting volumetric performance, Scapy prioritizing flexibility in anomaly modeling, and Trafgen offering a balanced compromise between both aspects.*

1. Introdução

Os Ataques Distribuídos de Negação de Serviço (*Distributed Denial of Service* — DDoS) geram impactos sobre a disponibilidade dos serviços, a integridade da infraestrutura de rede e a continuidade operacional dos sistemas. Os pesquisadores e desenvolvedores têm aprimorado continuamente as contramedidas, expandindo suas estratégias para mitigar danos e evitar prejuízos ainda maiores [Pakmehr et al. 2024]. No entanto, ainda existe

uma limitação significativa na disponibilidade de *datasets* atualizados que representem características realistas dos ataques, como a quantidade e o tamanho dos pacotes, além do número de *bots* envolvidos [Khanan et al. 2024]. Diante desse cenário, as ferramentas de simulação são opções viáveis para testar e validar, de forma segura e eficaz, os sistemas desenvolvidos para a contenção desses ataques [Dai et al. 2025]. Elas possibilitam a geração de diferentes tipos de tráfego e anomalias, muitas delas atuando nas camadas de rede, transporte ou aplicação. Assim, para proceder uma avaliação eficaz dos sistemas de defesa contra ataques DDoS, é fundamental selecionar ferramentas adequadas, capazes de reproduzir com fidelidade o comportamento de ameaças reais.

Embora a literatura apresente diversos trabalhos voltados à análise de ferramentas e geradores de tráfego [Anand et al. 2025], há uma divisão clara entre abordagens de escopo amplo e estudos mais restritos. Trabalhos como [Kaur et al. 2015, Behal and Kumar 2017] avaliam um grande conjunto de soluções com base em critérios majoritariamente objetivos, organizados em tabelas classificatórias e descrições sucintas. Esses estudos abrangem ataques DoS e DDoS nas camadas de rede, transporte e aplicação, incluindo ferramentas de linha de comando (CLI) e de interface gráfica (GUI) para diferentes sistemas operacionais. Contudo, essa ampla cobertura limita análises mais completas. Por outro lado, trabalhos como [Nagpal et al. 2015, Mahadev et al. 2016] e trabalhos mais recentes [Obaid 2020, Goel et al. 2023, Chamoli and Mittal 2024] adotam escopos mais restritos, avaliando um número reduzido de ferramentas. Ainda assim, direcionam parte significativa da análise a aspectos paralelos à caracterização das soluções, resultando em tabelas incompletas e avaliações igualmente superficiais. Os trabalhos recentes voltados à geração de *datasets* sintéticos [Poisson et al. 2024, Bistene et al. 2025] oferecem elevada flexibilidade de customização, porém operam de forma desacoplada e offline, além de não se concentrarem na geração específica de ataques DDoS [Alhijawi et al. 2022].

Este trabalho apresenta uma análise sistematizada de ferramentas e geradores de tráfego voltados à simulação de ataques de DDoS nas camadas de rede e transporte da Internet. O estudo foca em um conjunto de sete ferramentas *open source* que atuam via CLI no sistema operacional Linux, a fim de apoiar a tomada de decisão na seleção de ferramentas adequadas para a avaliação e validação de sistemas de segurança contra ataques DDoS. A abordagem combina uma classificação das ferramentas, adotando critérios consolidados na literatura, com testes de desempenho padronizados e análises comparativas da geração de tráfego. Além disso, avaliou-se a capacidade das ferramentas de modelar anomalias de tráfego conforme a taxonomia proposta por Jelena Mirkovic [Mirkovic 2025]. Os resultados evidenciam diferenças significativas entre as ferramentas, destacando que T50 prioriza desempenho volumétrico, a Scapy flexibilidade de anomalias e a Trafgen equilíbrio entre ambos. Eles mostram que a escolha da ferramenta impacta diretamente a fidelidade e a reprodutibilidade dos testes empregados.

O restante deste artigo está organizado da seguinte forma: A Seção 2 revisa os trabalhos relacionados. A Seção 3 apresenta a metodologia para a escolha das ferramentas avaliadas. A Seção 4 descreve a avaliação e os resultados. A Seção 5 conclui o trabalho.

2. Trabalhos Relacionados

Os trabalhos existentes na literatura concentram-se na identificação das funcionalidades oferecidas pelas ferramentas de geração de ataques DDoS e na categorização dos tipos de ataques que elas suportam. Esses estudos geralmente ignoram avaliações comparativas e

sistemáticas, e a análise da capacidade de geração controlada de anomalias de tráfego.

Em [Kaur et al. 2015], os autores propuseram uma análise sistemática voltada à caracterização de ferramentas de DDoS, organizando 36 soluções segundo critérios como tipo de interface, protocolos suportados, taxa de ataque, suporte a *IP spoofing* e sistema operacional. Entretanto, o escopo amplo com a análise de diversas ferramentas limita a profundidade das comparações. Apesar de fornecer uma visão abrangente do cenário, o estudo não realiza avaliações práticas de desempenho nem analisa a capacidade delas de gerar anomalias específicas. [Behal and Kumar 2017] estenderam esse estudo ao incluir geradores de tráfego na análise, buscando ampliar a compreensão do ecossistema e avaliar 59 ferramentas associadas a ataques DDoS. Nesse contexto, ferramentas como TCP-Replay e Trafgen, também utilizadas neste trabalho, passam a ser consideradas. No entanto, as principais limitações do estudo anterior permanecem, especialmente a ausência de avaliações comparativas de desempenho e de investigações sobre a capacidade de customização das ferramentas para a geração de anomalias de tráfego com fins experimentais. Assim como em [Kaur et al. 2015], o objetivo está em ferramentas empregadas em ataques reais, e não sobre simuladores projetados para testes reprodutíveis.

Outros trabalhos levaram em conta um conjunto menor de ferramentas, o que, em alguns casos, possibilita análises um pouco mais aprofundadas. Contudo, eles introduzem novas limitações, como um escopo que vai além da avaliação de ferramentas, a adoção de critérios mais subjetivos e a apresentação de tabelas de classificação incompletas. Por exemplo, em [Nagpal et al. 2015] e [Mahadev et al. 2016], os autores comparam 5 e 14 soluções respectivamente, mas eles ignoram avaliações quantitativas de desempenho, não exploraram a geração de anomalias e fazem análises superficiais, com foco em descrições. De modo semelhante, trabalhos mais recentes, como [Obaid 2020], [Goel et al. 2023] e [Chamoli and Mittal 2024], atualizam o estado da arte ao incorporar ferramentas e análises atualizadas. Entretanto, eles mantêm as mesmas limitações ao adotarem critérios de avaliação subjetivos, com foco em descrições, apresentarem tabelas comparativas deficientes e expandirem as discussões para temas além da análise sistemática das ferramentas. Embora [Chamoli and Mittal 2024] apresentem um estudo de caso com medições de tráfego gerado por diferentes soluções, a análise de desempenho é pouco sistematizada e não há uma investigação sobre a capacidade de geração controlada de anomalias. Além disso, como nos trabalhos apresentados anteriormente, o objetivo desse também está em ferramentas destinadas à execução de ataques reais, não em simuladores.

Os trabalhos recentes voltados à geração de *datasets* de tráfego para simulação oferecem elevada flexibilidade na customização dos dados gerados. Entretanto, tais abordagens atuam de maneira desacoplada e offline, além de não serem focadas na geração específica de ataques DDoS. Em [Poisson et al. 2024], os autores apresentam o gerador GothX, destinado à criação de tráfego legítimo e malicioso em redes IoT, oferecendo recursos como parametrização detalhada, execução automatizada de cenários e rotulação automática de *datasets*. De maneira complementar, [Bistene et al. 2025] apresentam uma revisão sistemática de geradores de tráfego utilizados em Cyber Ranges, os quais são classificados em abordagens baseadas em modelos, rastros e híbridas. Embora esse estudo não tenha como foco a simulação de DDoS, ele reconhece a capacidade de reprodução de ataques de *flood* como um critério relevante de validação, reforçando a importância de ferramentas capazes de gerar tráfego malicioso de forma controlada e reprodutível.

3. Metodologia da Classificação e Descrição das Ferramentas Selecionadas

Esta seção descreve a metodologia aplicada na seleção e classificação do conjunto de ferramentas de código aberto *open source* para a simulação de ataques DDoS em linha de comando (CLI), que atuam nas camadas de rede e transporte da Internet. O objetivo é fornecer uma base comparativa que auxilie pesquisadores na escolha de ferramentas adequadas para experimentos na área de segurança. Assim, foram selecionadas sete ferramentas, classificadas com base em critérios adotados em trabalhos anteriores [Kaur et al. 2015, Behal and Kumar 2017], conforme sumarizado na Tabela 1. Elas foram avaliadas quanto ao desempenho em simulações de ataques volumétricos e à capacidade de gerar anomalias de tráfego, de modo a evidenciar diferenças práticas relacionadas à eficiência, flexibilidade e adequação a distintos cenários experimentais. Ressalta-se que todos os testes empregaram exclusivamente IPv4, pois algumas ferramentas não suportam IPv6, o que poderia comprometer as análises. Além disso, foram utilizados apenas os códigos originais, sem recorrer à construção manual de pacotes em modo *raw*, interfaces interativas ou *scripts* em *Bash* para contornar limitações.

Tabela 1. Classificação inicial das ferramentas analisadas

Nome	Versão	Ano[6]	Tipo de ataques	Tráfego***	SO	Proto. IP	Spoof.	Botnet[1]	Instalação
BoNeSi	0.3.1	2018	DoS, DDoS	ICMP; UDP; TCP (HTTP)	Linux*	IPv4	Sim	Sim	GitHub
Hping3	3.0.0-alpha-2	2024	DoS, DDoS	ICMP; UDP; TCP	Linux; Unix-Like	IPv4	Sim	Sim[2]	Repo SO
Mausezahn	0.6.8	2023	DoS, DDoS	ICMP; UDP; TCP; ++	Linux	IPv4 e IPv6**	Sim	Sim[3]	GitHub
Scapy	2.6.1.dev89	2025	DoS, DDoS	ICMP; UDP; TCP; ++	Linux; Windows; MacOS; Unix	IPv4 e IPv6	Sim	Sim	PIP
Tcpreplay	4.4.4	2025	DoS, DDoS	ICMP; UDP; TCP; ++	Linux; MacOS; Unix	IPv4 e IPv6	Sim	Sim	Repo SO
Trafgen	0.6.8	2025	DoS, DDoS	ICMP; UDP; TCP; ++	Linux	IPv4 e IPv6	Sim	Sim[4]	GitHub
T50	5.8.7c	2025	DoS, DDoS	ICMP; UDP; TCP; ++	Linux*	IPv4	Sim	Não[5]	GitLab

Notas:

* Sistema operacional não explicitado; a ferramenta foi compilada e testada apenas em ambiente Linux neste trabalho.
** O suporte ao *IPv6* é informado pela ferramenta, porém o modo não apresentou funcionamento adequado no ambiente de testes.
*** "++" indica que a ferramenta suporta outros protocolos além dos citados na tabela.
[1] A coluna indica se a ferramenta é capaz de selecionar um número customizado de IPs para simular uma *botnet* com tamanho configurável (diferentemente da randomização completa de IPs avaliada na Seção 4.3, que gera um número indefinido de *bots*).
[2] Possível apenas por meio do modo de *script* TCL.
[3] Suporta apenas customização de IPs dentro de um intervalo.
[4] Suporta *botnet* por meio de arquivos de configuração extensos, com cada pacote definindo um IP diferente.
[5] Suporta somente a randomização completa de IPs para *spoofing*.
[6] Indica o ano da última atualização até o momento da pesquisa.

O escopo deste estudo foi delimitado por parâmetros objetivos, permitindo uma análise detalhada do desempenho e da capacidade de reprodução de anomalias. Diferente de abordagens com escopo amplo, que priorizam a diversidade de ferramentas em detrimento da profundidade técnica, esta pesquisa estabeleceu critérios rígidos de seleção. Levou-se em conta apenas ferramentas capazes de simular ataques DDoS nas camadas de rede e de transporte; aquelas que simulam apenas DoS ou atuam exclusivamente na camada de aplicação foram desconsideradas. Além disso, elas devem ser operadas por CLI, descartando-se aquelas que dependem de interface gráfica (GUI) ou de interação contínua

do usuário. Por fim, as ferramentas analisadas são *open source*, funcionam no sistema operacional Linux e possuem manutenção ativa ou versões recentes. Essas restrições refletem cenários práticos, nos quais exige o acionamento de geradores de tráfego por meio de comandos para disparar ataques ICMP, UDP ou TCP de forma automatizada e reprodutível [Farias Jr et al. 2023, Batista et al. 2025, Alalyan et al. 2025].

BoNeSi: realiza *IP Spoofing* a partir de um arquivo de endereços IP falsos, permitindo ao usuário configurar endereços de origem personalizados e o número de *zombies* da *botnet*. O módulo TCP da BoNeSi foi concebido para tráfego HTTP, mas permite gerar ataques na camada de transporte, como o *TCP SYN Flood*. Ela faz randomização automática de alguns campos nos pacotes, mas não viabiliza a personalização da maioria deles.

Hping3: utilizada para geração de tráfego e simulações de ataques DDoS, disponível nos repositórios oficiais das distribuições Linux. Ela suporta múltiplos modos de operação, incluindo um modo interativo baseado na sua API interna, bem como o envio direto via linha de comando. Neste trabalho, foram avaliadas duas formas de utilização: (i) o modo tradicional, com comandos simples[1], e (ii) o modo scriptável[2], que proporciona a construção e manipulação de pacotes via *scripts* Tcl, oferecendo maior controle sobre campos.

Mausezahn: integrante do kit *Netsniff-NG*, ela oferece dois modos de operação: (i) envio direto via CLI e (ii) modo interativo, que não foi avaliado por não fazer parte do escopo. Ela é capaz de gerar outros tipos de tráfego além dos testados.

Scapy: biblioteca do *Python* para a manipulação e formação de pacotes em camadas, que oferece um alto grau de customização para vários protocolos. Seu uso exige maior conhecimento técnico, pois o usuário precisa programar explicitamente a construção e o envio dos pacotes. Em contrapartida, essa flexibilidade, possibilitada pelo *Python*, permite modelar anomalias e cenários de ataque complexos, tornando a ferramenta uma das mais versáteis para criação de simulações altamente específicas. Possui uma função integrada ao Tcpreplay que faz envio rápido de pacotes: `sendpfast()`.

Tcpreplay: gerador de tráfego que realiza o *replay* de arquivos PCAP, disponível nos repositórios oficiais das distribuições Linux. Embora possa enviar pacotes, ele não os cria: seu propósito é retransmitir PCAPs produzidos por outras ferramentas ou extraídos de *datasets*. Neste trabalho, o Tcpreplay foi utilizado para reproduzir arquivos previamente formados com a Scapy. A ferramenta oferece opções úteis, como a execução em *loop*, *preload* do arquivo em memória, e viabiliza a simulação de diversos cenários.

Trafgen: integrante do kit *Netsniff-NG*, é um gerador de pacotes *multithreaded* capaz de distribuir o envio entre múltiplos núcleos de CPU. A customização dos pacotes é feita por meio de um arquivo de configuração, que pode utilizar tanto construções manuais quanto a sintaxe própria da ferramenta, mais concisa e flexível. Embora seu uso seja mais complexo do que o das demais, por requerer essa escrita específica, ela oferece um nível extremamente alto de personalização. Neste estudo, utilizou-se apenas sua sintaxe própria, mantendo-se o ajuste padrão de paralelização (distribuição automática pelos núcleos). Ela permite converter um PCAP em um arquivo de configuração próprio.

T50: suporta múltiplos protocolos e oferece dois modos de transmissão rápida: o modo *flood* padrão e o modo *turbo*, que executa dois processos paralelos para melhor desem-

[1]Exemplo: `sudo hping3 -1 192.168.100.2 --rand-source -d 0 --flood`
[2]Exemplo: `sudo hping3 exec arquivo.htcl`

penho. É focada na geração de pacotes formados apenas por cabeçalhos (sem *payload*), o que explica sua incapacidade de realizar testes que dependem de carga útil, como o cenário de *UDP Flood* com 512 bytes. Também possui o modo *Shuffle*, que transmite pacotes de todos os protocolos suportados de forma aleatória.

4. Avaliação de Desempenho e Geração de Anomalias

Esta seção apresenta a análise comparativa das ferramentas investigadas quanto ao desempenho em simulações de ataques DDoS volumétricos e à capacidade de geração de anomalias de tráfego. A metodologia adotada contempla execuções padronizadas de *flood* ICMP, UDP e TCP, bem como uma avaliação sistemática na qual cada uma delas foi testada quanto à capacidade de reproduzir cada anomalia definida neste estudo. Dessa forma, os experimentos permitem avaliar de maneira consistente tanto o potencial de desempenho volumétrico quanto o grau de flexibilidade na criação de diferentes padrões de tráfego. Também são descritos o ambiente de testes, os parâmetros usados nos experimentos volumétricos, as anomalias geradas por cada ferramenta e as tabelas comparativas que sintetizam os resultados obtidos. Os comandos, *scripts* e os demais arquivos relacionados às simulações e medições estão disponíveis no GitHub[3], assim como a descrição completa de todas as anomalias avaliadas neste estudo e os links para as ferramentas.

4.1. Ambiente de Testes

Na implementação de um cenário experimental capaz de replicar, com fidelidade, as condições de ataques nas camadas de rede e transporte, foi construída uma rede isolada para a realização dos experimentos, garantindo um ambiente controlado, evitando interferências externas e assegurando que os testes não causassem impactos em redes reais, como ilustrado na Figura 1. Assim, configurou-se uma rede interna com máquinas virtuais, com uso do *Oracle VirtualBox* (7.2.2), e foram criadas quatro *VMs*, cujos IPs foram atribuídos manualmente: uma atacante, uma vítima (alvo do ataque) e duas refletoras. Na máquina atacante, instalou-se cada ferramenta da forma indicada na Tabela 1 e, na máquina vítima, configurou-se o *Tshark* (4.2.2), versão em linha de comando do *Wireshark*, para captura e análise do tráfego. Nas máquinas refletoras, aplicaram-se as configurações aos testes de reflexão, apresentadas na Seção 4.3. A Tabela 2 apresenta as especificações das máquinas virtuais e a alocação de recursos conforme o papel de cada nó no experimento. A máquina vítima recebeu maior capacidade computacional para suportar a captura e o processamento de tráfego, já a máquina atacante utilizou recursos intermediários para sustentar cenários de *flood*. As máquinas refletoras empregaram configurações mínimas, pois sua função se limita à validação das anomalias de reflexão.

Tabela 2. Máquinas virtuais usadas no ambiente de testes

Característica	Atacante	Vítima	Máquinas de Reflexão
Memória RAM	8 GB	64 GB	2 GB / 1 GB
Processador	4 núcleos	20 núcleos	1 núcleo
Armazenamento	25 GB	25 GB	25 GB

Sistema Operacional: Ubuntu Server 24.04.3 LTS
Processador do hospedeiro: Intel® Core™ i7-14700 × 28
Memória RAM do hospedeiro: 128 GB

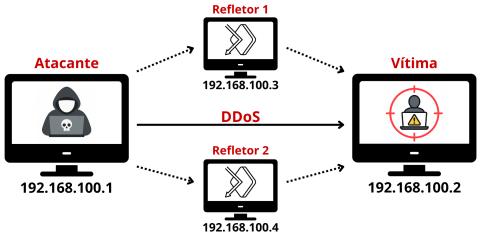

Figura 1. Ambiente de Testes

[3]https://github.com/arthurguimaraesferreira/ddos-sim-experiments-l3-l4

4.2. Testes de Desempenho Volumétrico (ICMP, UDP e TCP)

A fim de comparar as ferramentas quanto a capacidade de gerar tráfego em simulações de DDoS volumétrico, os testes foram padronizados entre os três protocolos ICMP, UDP e TCP com envio de pacotes diretamente da máquina atacante à máquina vítima. Buscou-se manter as execuções sob configurações e parâmetros tão semelhantes quanto possível para garantir a consistência nas comparações. Em alguns casos, certos ajustes em cabeçalhos e *payloads* não puderam ser igualados devido a limitações específicas de cada ferramenta. Ainda assim, as diferenças remanescentes impactam de forma mínima os resultados. Ao todo, o estudo realizou quatro testes de desempenho: um *ICMP Echo-Request Flood* (*ping flood*), dois testes de *UDP Flood* (com *payload* vazio e com *payload* de 512 bytes) e um teste de *TCP SYN Flood*. Essas configurações foram definidas com base nos parâmetros apresentados em [de Cámara et al. 2023, Choueiri et al. 2025], por serem simples, amplamente utilizadas em cenários reais e recorrentes em estudos recentes que realizam testes de DoS/DDoS com parâmetros equivalentes. Os campos menos críticos de cada protocolo foram, em sua maioria, mantidos nos valores padrão das ferramentas. Quando possível, padronizou-se os parâmetros-chave (por exemplo, *TTL* no ICMP e *Window* no TCP) ao adotar um valor de referência comum para melhor uniformizar as execuções.

A configuração experimental ajustou as ferramentas para transmitir pacotes à taxa máxima e com *IP spoofing* ativado (randômico ou por arquivo), de modo a simular ataques distribuídos. A fim de obter resultados consistentes, dez execuções independentes foram realizadas para cada ferramenta, considerando os quatro testes de desempenho, cada uma com duração de 60 segundos. O número de repetições deve-se ao fato de que todas as ferramentas apresentam variações de desempenho, seja em decorrência do sistema operacional, seja do próprio comportamento interno de cada ferramenta. Assim, múltiplas execuções permitem capturar essa variabilidade e produzir valores mais estáveis. Desse modo, foi possível calcular médias de desempenho, verificar a consistência entre envios do atacante e capturas na vítima, identificar erros inesperados e comparar, de forma confiável, os números de pacotes por segundo alcançados. Os resultados são apresentados nas Tabelas 3, 4, 5 e 6, com notas gerais sobre os testes abaixo.

Tabela 3. Resultados do teste ICMP Echo-Request

Ferramentas	Enviados	Capturados	Dif. Env./Cap.	Tempo (s)	PPS	Erros	Spoof. por arq.	Total Length
BoNeSi	5.475.293 ± 230.690	5.547.131 ± 230.228	**-71.838 ± 1.556**	59,9965	92.458 ± 3.836	0	S	60
Hping3	6.388.461 ± 1.222.345	6.387.587 ± 1.221.614	874 ± 2.764	59,9862	106.478 **± 20.334**	0	N	60
Hping3 (TCL)	N/D	5.302.684 ± 869.419	N/D	60,0006	88.376 ± 14.485	1	N	60
Mausezahn	N/D	4.806.775 ± 754.398	N/D	59,9977	80.113 ± 12.551	0	N	60
Scapy	N/D	372.821 ± 6.206	N/D	56,5887	**6.588 ± 97**	0	S	60
Tcpreplay	6.000.968 ± 98.169	6.000.969 ± 98.169	-1 ± 0	59,6037	100.682 **± 1.704**	0	S	60
Trafgen	13.416.815 ± 76.709	**13.412.413 ± 76.112**	4.402 ± 6.447	59,997	**223.551 ± 1.260**	0	N	60
T50	9.771.741 ± 2.362.257	9.770.797 ± 2.361.789	944 ± 2.078	59,9854	162.885 **± 39.370**	**4**	N	60
T50 (turbo)	23.199.621 ± 467.111	**23.139.169 ± 463.352**	**60.451 ± 32.641**	59,9911	**385.710 ± 7.704**	**9**	N	60

A Tabela 3 apresenta os resultados obtidos nos testes de desempenho de *ICMP Echo-Request Flood*. Primeiramente, observa-se que a Hping3 (TCL), Mausezahn e Scapy (com uso da função `sendpfast`) não informam o número de pacotes enviados nas simulações. Esse comportamento se repete com os protocolos UDP e TCP e, sem essa métrica, não é possível avaliar se há perdas significativas de pacotes. No caso da Hping3 (TCL), é possível implementar um contador manual no *script*. Contudo, essa abordagem

limita o desempenho de ataques *flood*. Outra limitação diz respeito à imprecisão do contador da BoNeSi, que deixa de contabilizar os pacotes enviados no último segundo do ataque quando ele é interrompido, resultando em uma discrepância entre os enviados e os capturados. Já algumas ferramentas, como Hping3 (padrão), Tcpreplay e T50 (padrão), apresentaram valores pequenos na coluna (Dif. Env./Cap.), os quais são atribuídos a imprecisões nos contadores internos e a perdas pontuais decorrentes do elevado volume de tráfego gerado. O mesmo ocorreu nos demais experimentos.

Quanto aos resultados de envio de tráfego, a T50 (turbo), que executa dois processos em paralelo, mostrou a maior taxa de pacotes por segundo. Embora ela tenha apresentado maior número de perdas de pacotes, esse valor representa menos de 0,3% dos enviados, e ela foi capaz de gerar um volume de tráfego significativamente elevado. Em contrapartida, a estabilidade durante os testes foi insatisfatória: para completar as 10 simulações bem-sucedidas, foram registrados 9 erros de execução, o que exigiu a realização de 19 execuções no total. Trafgen e Tcpreplay apresentaram menor desempenho em PPS, porém maior estabilidade, com menos erros, menores diferenças entre pacotes enviados e capturados e baixa variabilidade nos resultados. Em contraste, Hping3, Mausezahn e, principalmente, a T50 (padrão) exibiram alta variação nos valores de Enviados, Capturados e PPS, indicando maior instabilidade, conforme ilustrado na Figura 2. Por fim, a Scapy alcançou taxas de envio substancialmente inferiores às das demais, mesmo com o uso da função de envio rápido integrada ao Tcpreplay, tendo limitações de desempenho que se mantêm nos testes seguintes com UDP e TCP.

Tabela 4. Resultados do teste UDP Flood 0 bytes (*payload* vazio)

Ferramentas	Enviados	Capturados	Dif. Env./Cap.	Tempo (s)	PPS	Erros	Total Length	Porta Orig./Dest.
BoNeSi	7.031.547 ± 737.726	7.111.992 ± 740.363	-80.446 ± 11.774	60,0069	118.521 ± 12.349	1	60	Rand. / 50001
Hping3	6.894.312 ± 753.028	6.894.338 ± 753.040	-26 ± 33	59,9787	114.946 ± 12.557	0	60	**Incr.** / 50001
Hping3 (TCL)	N/D	5.022.581 ± 494.802	N/D	60,0023	83.707 ± 8.249	2	60	Rand. / 50001
Mausezahn	N/D	5.030.509 ± 945.196	N/D	60,0029	83.839 **± 15.760**	0	60	**0** / 50001
Scapy	N/D	576.471 ± 13.501	N/D	57,1664	**10.083 ± 203**	0	60	Rand. / 50001
Tcpreplay	5.870.921 ± 155.114	5.870.922 ± 155.114	-1 ± 0	58,6185	100.153 ± 2.568	0	60	Rand. / 50001
Trafgen	13.432.043 ± 48.735	13.412.998 ± 58.467	19.044 ± 17.495	59,9997	**223.551 ± 974**	0	60	Rand. / 50001
T50	9.722.535 ± 901.521	9.722.486 ± 901.607	49 ± 157	59,825	162.548 **± 15.353**	**7**	60	Rand. / 50001
T50 (turbo)	23.244.813 ± 267.301	**23.168.355 ± 258.469**	**76.458 ± 44.940**	59,9995	**386.141 ± 4.145**	**7**	60	Rand. / 50001

Tabela 5. Resultados do teste UDP Flood (*payload* 512 bytes)

Ferramentas	Enviados	Capturados	Dif. Env./Cap.	Tempo (s)	PPS	Erros	Total Length	Porta Orig./Dest.
BoNeSi	6.860.916 ± 312.846	6.946.199 ± 301.871	-85.283 ± 18.508	60,0036	115.763 ± 5.034	**3**	554	Rand. / 50001
Hping3	5.945.862 ± 429.746	5.943.460 ± 430.270	2.402 ± 2.841	59,9733	99.102 ± 7.178	0	554	**Incr.** / 50001
Hping3 (TCL)	N/D	4.558.059 ± 602.194	N/D	59,9988	75.969 ± 10.037	1	554	Rand. / 50001
Mausezahn	N/D	4.501.939 ± 761.229	N/D	60,0002	75.034 ± 12.703	0	554	**0** / 50001
Scapy	N/D	348.267 ± 8.704	N/D	56,2903	**6.187 ± 133**	0	554	Rand. / 50001
Tcpreplay	5.814.217 ± 146.104	5.812.349 ± 147.408	1.867 ± 2.801	58,1232	99.999 ± 2.445	0	554	Rand. / 50001
Trafgen	12.388.775 ± 96.668	**12.188.438 ± 91.426**	**200.337 ± 20.731**	59,9945	**203.159 ± 1.520**	0	554	Rand. / 50001
T50	–	–	–	–	–	–	–	–
T50 (turbo)	–	–	–	–	–	–	–	–

As Tabelas 4 e 5 apresentam os valores dos resultados obtidos nos testes de *UDP Flood* com *payloads* de 0 e 512 bytes, respectivamente. As limitações previamente discutidas quanto à contabilização do número de pacotes enviados por algumas ferramentas permanecem, incluindo a imprecisão observada da BoNeSi. Nos testes com carga vazia,

a T50 (turbo) voltou a se destacar com o maior valor de PPS, apesar de ter registrado um número elevado de erros de execução. Contudo, esse bom desempenho contrasta com o resultado verificado no teste com 512 bytes, visto que ela não é capaz de simular ataques UDP com *payload* não-nulo, sendo a única que não pôde executar esse cenário. Outra restrição relevante refere-se à customização das portas de origem: enquanto a padronização adotada nos experimentos foi a randomização, a Mausezahn não oferece suporte a esse recurso e a Hping3 (padrão), embora indique essa opção como *default*, na prática realiza apenas incrementos sequenciais. Esse mesmo aspecto também se manifestou nas avaliações TCP. Em geral, verifica-se uma redução moderada de PPS para todas as ferramentas ao comparar os testes UDP sem e com carga, resultado esperado em função do maior volume de dados transmitidos. Observa-se também um aumento significativo nas perdas de pacotes, com destaque para a Trafgen, que ainda assim manteve bom desempenho e baixa variabilidade nos resultados. Em contraste, BoNeSi, Hping3, Mausezahn e T50 (padrão) apresentaram elevados desvios-padrão. Além disso, destaca-se que o maior índice de perda constatado ao longo dos experimentos ocorreu com a Trafgen nos testes UDP com *payload*, ainda assim limitado a apenas 1,6%, seguido pela T50 (turbo), com 0,7% nos testes TCP. Esses números indicam que as discrepâncias entre enviados e capturados permanecem reduzidas mesmo nos piores cenários.

Tabela 6. Resultados do teste TCP SYN Flood

Ferramentas	Enviados	Capturados	Dif. Env./Cap.	Tempo (s)	PPS	Erros	Total Length	Porta Orig./Dest.	Flags
BoNeSi*	6.523.909 ± 1.029.022	6.564.243 ± 1.035.261	-40.334 ± 14.655	58,6831	111.862 **± 17.657**	**3**	**62/66/74/78**	Rand. / 50001	SYN
Hping3	6.595.783 ± 902.775	6.595.162 ± 903.139	621 ± 1.670	61,1574	108.018 **± 15.895**	0	60	**Incr.** / 50001	SYN
Hping3 (TCL)	N/D	4.887.545 ± 562279	N/D	62,5331	78.163 ± 8.923	**3**	60	Rand. / 50001	SYN
Mausezahn*	N/D	4.290.577 ± 549.463	N/D	62,2711	68.912 ± 8.816	0	**74**	**0** / 50001	SYN
Scapy	N/D	526.779 ± 13.099	N/D	56,97	**9.246 ± 207**	0	60	Rand. / 50001	SYN
Tcpreplay	5.926.294 ± 58.580	5.926.295 ± 58.580	-1 ± 0	58,5045	101.297 **± 1.081**	0	60	Rand. / 50001	SYN
Trafgen	13.522.498 ± 61.013	13.491.428 ± 65.361	31.070 ± 16.029	59,9917	**224.888 ± 1.084**	0	60	Rand. / 50001	SYN
T50	9.025.303 ± 1.884.236	9.025.304 ± 1.884.236	-1 ± 0	59,9869	150.455 **± 31.410**	1	**66**	Rand. / 50001	**SYN+**
T50 (turbo)	22.595.409 ± 1.282.716	**22.429.167 ± 1.236.676**	**166.242 ± 61.705**	59,9891	**373.887 ± 20.609**	0	**66**	Rand. / 50001	**SYN+**

Informações adicionais sobre os testes e tabelas:

Valores: os resultados apresentados nas colunas "Enviados", "Capturados", "Dif. Env./Cap.", "Tempo (s)" e "PPS" foram obtidos a partir da média das 10 simulações bem-sucedidas de cada ferramenta e incluem também os respectivos desvios-padrão amostrais (exceto a coluna "Tempo (s)").
Contagem de pacotes: as colunas "Enviados" e "Capturados" representam a média de pacotes transmitidos e capturados, respectivamente. A coluna "Dif. Env./Cap." corresponde à diferença entre esses valores ("Enviados – Capturados").
PPS: pacotes por segundo (calculado em cada execução com o número de Capturados dividido pelo Tempo(s)).
Spoof. por arq.: indica se a ferramenta usou um arquivo contendo endereços IP para realizar o *IP spoofing*. Esse mecanismo permite simular uma *botnet* com um número bem definido de *bots*, garantindo controle total sobre as origens dos pacotes. A coluna segue a mesma distribuição da tabela ICMP para os testes UDP e TCP.
(*) Extras (TCP): indica a presença de campos TCP adicionais transmitidos pelos pacotes, além dos campos configurados pelo usuário e além do conjunto padrão observado nas demais ferramentas. Os campos adicionais foram: *MSS*, *SACK_PERM*, *TSval*, *TSecr* e *WS*.
Flags "SYN+" no T50: o símbolo "+" indica que a ferramenta enviou flags extras não configuradas. Além da flag `SYN`, o T50 transmitiu também as flags `AE` (*Accurate ECN*) e `Reserved`, comportamento não observado nas demais ferramentas e não solicitado na configuração.
Scapy: foram gerados 10 mil pacotes em cada programa `.py`, posteriormente enviados em loop com `sendpfast`.
Tcpreplay: foram criados arquivos PCAP com 1 milhão de pacotes utilizando a Scapy, enviados em loop em cada teste.

Por fim, a Tabela 6 apresenta os resultados obtidos nos testes de *TCP SYN Flood*. A T50 (turbo) destacou-se novamente pelo elevado desempenho em termos de PPS. Entretanto, apesar da alta taxa de envio, ela apresentou limitações nos ajustes da simulação, uma vez que o cenário avaliado previa a ativação exclusiva da flag *SYN*, mas a ferramenta também habilita, de forma implícita e não configurável, as flags *AE* (*Accurate ECN*) e *Reserved*, além de utilizar um *payload* fixo de 12 bytes. Outras também registraram

restrições de padronização: a BoNeSi enviou pacotes com variação no tamanho total, enquanto a Mausezahn utilizou tamanho fixo de 74 bytes e manifestou a limitação já observada quanto à porta de origem, assim como a Hping3. Por outro lado, a Trafgen, de forma consistente com os experimentos anteriores, obteve alto desempenho, sem problemas de padronização, com pequena perda de pacotes de 0,2% dos enviados. De modo geral, a BoNeSi, a Hping3 (padrão), a Tcpreplay e a T50 (padrão) também alcançaram resultados satisfatórios. Diferentemente dos demais testes, a T50 (turbo) não apresentou erros de execução neste cenário, combinando alto desempenho com maior estabilidade. Ainda assim, observou-se leve aumento no desvio-padrão do PPS. Nesse contexto, BoNeSi, Hping3 (padrão) e T50 (padrão) apresentaram alta variabilidade, enquanto Tcpreplay e Trafgen se destacaram pelos baixos valores de desvio-padrão.

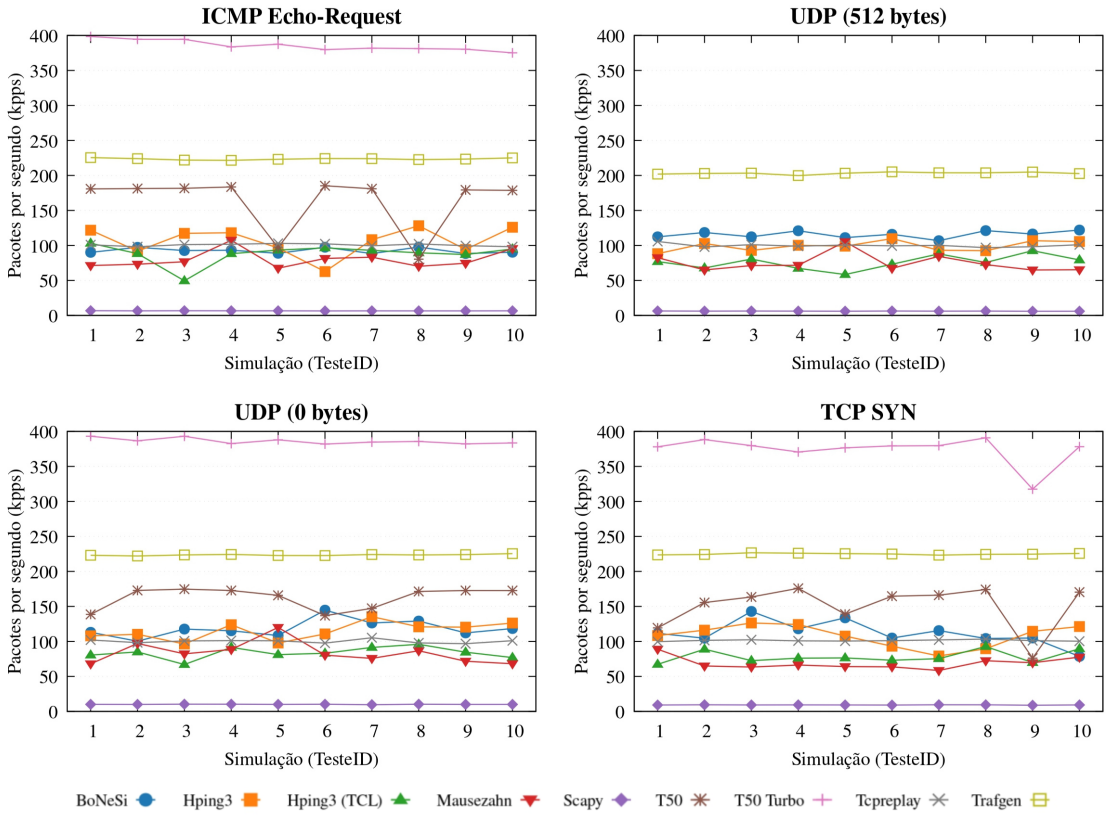

Figura 2. Desempenho quanto à taxa de envio das 10 simulações bem-sucedidas

A Figura 2 apresenta a taxa de pacotes por segundo (PPS), calculada como a razão entre pacotes capturados e o tempo total de captura, para cada ferramenta ao longo das 10 simulações bem-sucedidas de cada teste; isto permite melhor interpretação dos desvios-padrão apresentados nas Tabelas 3, 4, 5 e 6. Destaca-se a T50 (modo *turbo*), que alcançou valores significativamente superiores aos das demais, seguida pela Trafgen, que combina bom desempenho com alta consistência entre as execuções. Em contraste, a Scapy apresentou as menores taxas em todos os experimentos. Apesar do desempenho superior da T50 (modo *turbo*), esse resultado deve ser interpretado com cautela devido à elevada ocorrência de erros de execução, especialmente nos testes ICMP. As demais ferramentas apresentaram desempenhos próximos entre si, com variações pontuais de estabilidade, como observado na Hping3 em ICMP e na Mausezahn no teste UDP (512 bytes). Por fim, ressalta-se que a T50 não suporta o cenário de *UDP Flood* com *payload*.

4.3. Capacidade de Geração de Anomalias

A capacidade de customização das ferramentas é uma métrica de comparação importante. Com base na taxonomia de Jelena Mirkovic [Mirkovic 2025], a Tabela 7 indica, para cada anomalia, se a ferramenta é capaz de reproduzi-la. A análise incorporou diversos itens da lista da pesquisadora de forma integral, enquanto adaptou outros, descartou alguns por estarem fora do escopo e definiu novos a partir das capacidades de customização. O estudo avaliou 40 anomalias, sendo 7 ICMP, 6 UDP, 16 TCP, 6 relacionadas a portas e 5 gerais. Para cada entrada, verificou-se se a ferramenta é capaz de reproduzir o comportamento especificado. Por exemplo: a *UDP/TCP Port 0 (Dest)*, que tem como objetivo definir a porta de destino dos pacotes com o valor 0, foi executável por todas as ferramentas testadas, exceto a T50, que, ao definir "`--dport 0`", a randomiza. Várias anomalias podem ser combinadas para compor ataques mais complexos, como uma junção de *TCP Xmas*, *TCP SACK* e variação randômica da taxa de envio.

As simulações de reflexão utilizaram o ambiente de testes descrito na Seção 4.1, que incorporou duas *VMs* refletoras e demandou ajustes específicos para a execução das simulações. Assim, no *ICMP Smurf*, habilitou-se a resposta a requisições *Echo-Request* enviadas ao endereço de *broadcast*, enquanto, no *UDP Fraggle*, ativou-se o serviço de *Echo UDP*. Já no *TCP SYN Reflection (1)*, implementou-se, em uma das refletoras, um servidor *Python* simples capaz de responder a pacotes *SYN*. Como ilustração da reflexão, no *ICMP Smurf* o atacante envia pacotes *ICMP Echo-Request* ao endereço de *broadcast* da sub-rede (192.168.100.255), com o campo de origem forjado para o IP do alvo (192.168.100.2). Dessa forma, as máquinas refletoras retornam *Echo-Reply* diretamente à vítima e intensificam o volume de tráfego recebido.

A Tabela 7 apresenta os resultados da avaliação da capacidade das ferramentas em reproduzir as anomalias. Para padronizar os testes e avaliar exclusivamente as funcionalidades nativas, não foram utilizados *scripts* externos em *Bash*, uma vez que poderiam contornar algumas limitações. Executou-se todas as ferramentas em seu código original, sem modificações, e não empregou-se opções avançadas de construção manual de pacotes em modo *raw*. A Scapy, o Tcpreplay e a Trafgen se destacam pela ampla flexibilidade de customização. As duas primeiras implementaram todas as anomalias propostas, sendo que o Tcpreplay atua exclusivamente como reprodutor de ataques previamente construídos. A flexibilidade da Scapy se deve diretamente do uso do *Python*, que amplia sua capacidade de customização. A Trafgen deixou de gerar apenas três, entre elas a *TCP SYN Reflection*, na qual os pacotes foram criados e enviados corretamente, mas não houve reflexão do tráfego. A Hping3 também apresentou bom desempenho, com a ressalva de que muitas anomalias exigiram o uso do modo scriptável em Tcl, de maior complexidade.

As ferramentas Mausezahn, T50 e BoNeSi apresentaram limitações significativas, deixando de replicar anomalias consideradas simples, o que restringe suas aplicações em simulações de ataques complexos. As principais restrições da Mausezahn estão associadas à manipulação de portas, visto que ela permite apenas incrementos sequenciais, sem suporte à randomização, além de problemas na personalização de cargas e na fragmentação. A T50, por sua vez, tem como maior limitação a ausência de suporte a *payloads*, o que inviabiliza cenários que dependem desse recurso. Soma-se a isso a incapacidade de realizar ataques de reflexão, limitações na configuração de campos de cabeçalho e o fato de ser a única ferramenta que, embora realize *IP spoofing*, não permite alguma forma de definição explícita dos endereços IP de origem. Apesar do desempe-

nho superior em relação as demais em ataques volumétricos, essa característica não se reflete em flexibilidade. Por fim, a BoNeSi mostrou-se a mais limitada do conjunto analisado, com restrições em diversas categorias, especialmente na customização de portas de destino, no uso de *payloads* e na realização de ataques TCP, comprometendo sua aplicabilidade em simulações avançadas.

Tabela 7. Anomalias por Ferramenta

Anomalias	BoNeSi*	Hping3	Mausezahn*	Scapy	Tcpreplay**	Trafgen	T50
ICMP							
ICMP Echo-Request	✓	✓	✓	✓	✓	✓	✓
ICMP Type/Code	✗	✓	✓	✓	✓	✓	✓
ICMP Twinge/Trash	✗	✓[15]	✗	✓	✓	✓	✗
ICMP Smurf	✓	✓	✓	✓	✓	✓	✗[1]
ICMP Real Frag	✗	✓	✗	✓	✓	✓	✗
ICMP Fake Frag	✗	✗[16]	✗	✓	✓	✓	✗[17]
ICMP Ping of Death	✗	✗[16]	✗	✓	✓	✓	✗
UDP							
UDP Empty Payload	✓	✓	✓	✓	✓	✓	✓
UDP Custom Payload	✗	✓	✓	✓	✓	✓	✗
UDP Random Payload	✗	✓[15]	✗	✓	✓	✓	✗
UDP Real Frag	✗	✓	✗	✓	✓	✓	✗
UDP Fake Frag	✗	✗[16]	✗	✓	✓	✓	✗[17]
UDP Fraggle	✓[12]	✓	✓	✓	✓	✓	✗
TCP							
TCP Low-Rate (SYN Flood)	✗[4]	✓[15]	✗[4]	✓	✓	✗[4]	✗[4]
TCP SYN	✓	✓	✓	✓	✓	✓	✓[3]
TCP Custom Flags	✗	✓	✓	✓	✓	✓	✓[3]
TCP Random Flags	✗	✓[15]	✗	✓	✓	✓	✗
TCP Xmas (FIN, URG, PUSH)	✗	✓	✓	✓	✓	✓	✓[3]
TCP Erroneous Flags	✗	✓	✓	✓	✓	✓	✓
TCP NULL	✗	✓	✓	✓	✓	✓	✗
TCP ECE/CWR	✗	✓	✓	✓	✓	✓	✓[3]
TCP Window 0	✗	✓	✓	✓	✓	✓	✗
TCP Real Frag	✗	✓	✗	✓	✓	✓	✗
TCP Fake Frag	✗	✗[16]	✗	✓	✓	✓	✗[17]
TCP SYN Reflection (1)	✓[12]	✓[14]	✓	✓[14]	✓[14]	✗[2]	✗
TCP Land Attack (SYN)	✗[6]	✓	✓	✓	✓	✓	✓
TCP Malformed Packets	✗	✓	✗	✓	✓	✓	✗
TCP SACK	✗	✗[16]	✗	✓[11]	✓[11]	✓[11]	✓
TCP Snort Options	✗	✗[16]	✗	✓	✓	✓	✗
Gerais							
Flood ICMP/UDP/TCP	✓	✓	✓	✓	✓	✓	✓
UDP/TCP Custom Source Port	✗	✓	✓	✓	✓	✓	✓
UDP/TCP Custom Dest Port	✓	✓	✓	✓	✓	✓	✓
UDP/TCP 0 Port (Dest)	✓	✓	✓	✓	✓	✓	✗[9]
UDP/TCP Random Source Port	✓	✓[10]	✗[5]	✓	✓	✓	✓
UDP/TCP Random Dest Port	✗	✓[15]	✗[5]	✓	✓	✓	✓
UDP/TCP Range Ports	✗	✓[15]	✓	✓	✓	✓	✗
Diferentes Taxas de Envio (Fixas)	✓	✓	✓	✓	✓	✓	✓[7]
Taxa de Envio Random	✗	✓[15]	✓	✓	✓	✗	✗
Multi Protocol	✗	✓[15]	✗	✓	✓	✓	✓[8]
Random *IP Spoofing*	✗[13]	✓	✓	✓	✓	✓	✓

[1] Somente uma máquina da rede respondeu ao *broadcast*.
[2] Envia o pacote correto, mas não recebe resposta de reflexão.
[3] Flags *AE* e *Reserved* são ativadas mesmo sem configuração.
[4] Viável apenas com *scripts* externos.
[5] Apenas incrementa portas; não randomiza.
[6] Land Attack incompleto: IP igual, porta não.
[7] Apenas duas taxas: *flood* e turbo.
[8] Envia múltiplos protocolos além de ICMP/UDP/TCP.
[9] Porta 0 resulta em randomização.
[10] O *Help* da Hping3 indica randomização, mas a ferramenta apenas incrementa (modo padrão).
[11] Inclui *SACK* malformado, usado para testes com *Snort*.
[12] Não permite porta específica em ataques de reflexão.
[13] Suporta *IP Spoofing* somente por meio de um arquivo de *bots*.
[14] Poderia direcionar pacotes a múltiplos refletores.
[15] Funciona apenas via *script* TCL (.htcl).
[16] Testes padrão e TCL não funcionaram.
[17] `--frag-offset` não funcionou.
[*] Envia pacotes TCP com campos adicionais.
[**] PCAP para reprodução gerado via Scapy.

5. Conclusão

Este trabalho apresentou um estudo comparativo de sete ferramentas *open source* de CLI destinadas à simulação de ataques DDoS nas camadas de rede e de transporte, a fim de fornecer uma referência base que auxilie na seleção das ferramentas de geração de tráfego anômalo para diferentes cenários avaliativos de testes. A avaliação considerou tanto o desempenho em ataques volumétricos quanto a flexibilidade na geração de anomalias de tráfego associadas à negação de serviço. Os resultados indicam que a escolha da ferramenta mais apropriada depende das características específicas de cada experimento, uma vez que não existe uma solução universal. A T50, por exemplo, alcança as maiores taxas de envio no modo turbo, mas apresenta elevado número de erros de execução e limitações de customização. Em contrapartida, a Scapy oferece flexibilidade na geração de anomalias, à custa de um desempenho volumétrico inferior. A Trafgen, por sua vez, destacou-se como uma alternativa intermediária, com bom desempenho e ótima capacidade de geração de anomalias. Como trabalho futuro, pretende-se realizar uma análise comparativa de ferramentas CLI voltadas à simulação de ataques DDoS na camada de aplicação.

Agradecimento

O presente trabalho foi realizado com apoio do CNPq - processos nº 307752/2023-2, da CAPES - Código de Financiamento 001, e da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) - processo nº 2018/23098-0.

Referências

Alalyan, F., Awad, M., Jaafar, W., and Langar, R. (2025). Secure Distributed Federated Learning for Cyberattacks Detection in B5G Open Radio Access Networks. *IEEE Open Journal of the Communications Society*, 6:3067–3081.

Alhijawi, B., Almajali, S., Elgala, H., Bany Salameh, H., and Ayyash, M. (2022). A survey on DoS/DDoS mitigation techniques in SDNs: Classification, comparison, solutions, testing tools and datasets. *Computers and Electrical Engineering*, 99:107706.

Anand, N., Saifulla, M. A., Babu Ponnuru, R., Reddy Alavalapati, G., Patan, R., and Gandomi, A. H. (2025). Securing Software Defined Networks: A Comprehensive Analysis of Approaches, Applications, and Future Strategies Against DoS Attacks. *IEEE Access*, 13:64473–64515.

Batista, A., Pedroso, C., Brísio, S., Rodrigues, G., and Santos, A. (2025). Evaluating Robustness and Reliability of a C-NIDS for IoT Networks in Virtualized Environments. *IEEE Latin America Transactions*, 23(5):363–370.

Behal, S. and Kumar, K. (2017). Characterization and Comparison of DDoS Attack Tools and Traffic Generators: A Review. *Int. J. Netw. Secur.*, 19(3):383–393.

Bistene, J. V., dos Santos, A. F. P., das Chagas, C. E., and Salles, R. M. (2025). Modeling Network Traffic Generators for Cyber Ranges: A Systematic Literature Review. *Journal of Network and Systems Management*, 33(2):1–45.

Chamoli, S. and Mittal, V. (2024). DDoS Landscape: Insight into Attacks and Tools. In *2024 International Conference on Electrical Electronics and Computing Technologies (ICEECT)*, volume 1, pages 1–6.

Choueiri, S., Mazloum, A., Elizalde, S., Kfoury, E. F., and Crichigno, J. (2025). Volumetric DDoS Attack Detection and Mitigation using P4-DPDK. In *2025 IEEE Int. Black Sea Conference on Communications and Networking (BlackSeaCom)*, pages 1–6.

Dai, J., Dai, Z., and Thing, V. L. L. (2025). Cyber-Resilience Enhancement With Cross-Domain Software-Defined Network for Cyber-Physical Microgrids Against Denial of Service Attacks. *IEEE Transactions on Industrial Cyber-Physical Systems*, 3:273–284.

de Cámara, X. S., Flores, J. L., Arellano, C., Urbieta, A., and Zurutuza, U. (2023). Clustered federated learning architecture for network anomaly detection in large scale heterogeneous IoT networks. *Computers Security*, 131:103299.

Farias Jr, E. P., Jacinto Tavares, A. C., and Nogueira, M. (2023). A Runtime DDoS Attack Detection Technique Based on Stochastic Mathematical Model. In *2023 IEEE Latin-American Conference on Communications (LATINCOM)*, pages 1–6.

Goel, M., Singh, S., Garg, A., and Roy, N. R. (2023). Comparative Study of DDoS Attacks Tools and Their Analysis. In *2023 International Conference on IoT, Communication and Automation Technology (ICICAT)*, pages 1–8.

Kaur, H., Behal, S., and Kumar, K. (2015). Characterization and comparison of Distributed Denial of Service attack tools. In *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, pages 1139–1145.

Khanan, A., Abdelgadir Mohamed, Y., Mohamed, A. H. H. M., and Bashir, M. (2024). From Bytes to Insights: A Systematic Literature Review on Unraveling IDS Datasets for Enhanced Cybersecurity Understanding. *IEEE Access*, 12:59289–59317.

Mahadev, Kumar, V., and Kumar, K. (2016). Classification of DDoS attack tools and its handling techniques and strategy at application layer. In *2016 2nd International Conference on Advances in Computing, Communication, Automation (ICACCA) (Fall)*, pages 1–6.

Mirkovic, J. (2025). DDoS Attack List. `https://www.isi.edu/people-mirkovic/ddos-benchmarks/ddos-attack-list/`. Accessed: 07/10/2025.

Nagpal, B., Sharma, P., Chauhan, N., and Panesar, A. (2015). DDoS tools: Classification, analysis and comparison. In *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 342–346.

Obaid, H. S. (2020). Denial of service attacks: Tools and categories. *International Journal of Engineering Research & Technology (IJERT)*, 9(03):631–636.

Pakmehr, A., Aßmuth, A., Taheri, N., and Ghaffari, A. (2024). DDoS attack detection techniques in IoT networks: a survey. *Cluster Computing*, 27(10):14637–14668.

Poisson, M., Carnier, R., and Fukuda, K. (2024). GothX: a generator of customizable, legitimate and malicious IoT network traffic. In *Proceedings of the 17th Cyber Security Experimentation and Test Workshop*, CSET '24, page 65–73, New York, NY, USA. Association for Computing Machinery.