UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

FÁBIO ISRAEL RAMOS ESMANHOTTO

UM MECANISMO CONTRA ATAQUES DO TIPO SYN FLOOD PARA FIREWALLS BASEADOS NO FRAMEWORK NETFILTER

CURITIBA

2025

FÁBIO ISRAEL RAMOS ESMANHOTTO

UM MECANISMO CONTRA ATAQUES DO TIPO SYN FLOOD PARA FIREWALLS BASEADOS NO FRAMEWORK NETFILTER

A SYN FLOOD MECHANISM FOR FIREWALLS BASED ON THE NETFILTER FRAMEWORK

Dissertação apresentada como requisito para obtenção do título de Mestre em Computação Aplicada do Programa de Pós-Graduação em Computação Aplicada da Universidade Tecnológica Federal do Paraná.
Orientador(a): Prof. Dr. Joilson Alves Junior

Coorientador(a): Prof. Dr. Daniel Fernando Pigatto

CURITIBA

2025

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Curitiba

FABIO ISRAEL RAMOS ESMANHOTTO

UM MECANISMO CONTRA ATAQUES DO TIPO SYN FLOOD PARA FIREWALLS BASEADOS NO FRAMEWORK NETFILTER

Trabalho de pesquisa de mestrado apresentado como requisito para obtenção do título de Mestre Em Computação Aplicada da Universidade Tecnológica Federal do Paraná (UTFPR). Área de concentração: Engenharia De Sistemas Computacionais.

Data de aprovação: 12 de Março de 2026

Dr. Joilson Alves Junior, Doutorado - Universidade Tecnológica Federal do Paraná

Dra. Ana Cristina Barreiras Kochem Vendramin, Doutorado - Universidade Tecnológica Federal do Paraná

Dr. Bruno Bogaz Zarpelao, Doutorado - Universidade Estadual de Londrina (Uel)

Dr. Marcos Eduardo Pivaro Monteiro, Doutorado - Universidade Tecnológica Federal do Paraná

Documento gerado pelo Sistema Acadêmico da UTFPR a partir dos dados da Ata de Defesa em 12/03/2026.

Dedico essa dissertação de mestrado à minha mãe, que sempre esteve ao meu lado me incentivando. Também dedico este trabalho aos meus orientadores que me guiaram com sabedoria durante esta jornada.

AGRADECIMENTOS

Agradeço aos meus orientadores, Professor Joilson Alves Junior e Professor Daniel Fernando Pigatto, por compartilhar seus conhecimentos e por me orientar e motivar nesta caminhada do conhecimento.

Agradeço a minha mãe, em especial pelo apoio e compreensão nos vários momentos que precisei estar focado e consequentemente ausente durante esta jornada.

Por fim, agradeço a todos os professores do PPGCA da Universidade Tecnológica Federal do Paraná, por todo o conhecimento e experiência que pude absorver ao longo das aulas do programa.

“Os países em todo o mundo tornaram-se suficientemente dependentes do ciberespaço para comunicações e controle do mundo físico; de uma forma que é definitivamente impossível separar-se dele.” (Zhao *et al.*, 2020).

RESUMO

Ataques de Negação de Serviço (DoS) representam uma ameaça significativa à disponibilidade de dispositivos de segurança em rede, como os *firewalls*, que se tornam alvos frequentes em razão de seu papel central na proteção de redes e serviços. Entre esses ataques, destaca-se o SYN Flood, que explora uma limitação no gerenciamento de conexões do *Transmission Control Protocol* (TCP) ao manter um grande número de conexões parcialmente estabelecidas, levando ao esgotamento dos recursos do sistema. Essa modalidade de ataque afeta de maneira particular os *firewalls* baseados no netfilter (subsistema do kernel Linux responsável pela filtragem e manipulação de pacotes). A presente pesquisa tem como objetivo desenvolver um mecanismo para a detecção e contenção de SYN Floods direcionados a *firewalls* baseados no netfilter. O mecanismo proposto contempla três etapas: (i) a formulação de um modelo matemático para definir um limiar de classificação de pacotes SYN suspeitos; (ii) a implementação de uma ferramenta para detectar ataques e (iii) a elaboração de um modelo matemático para estimar os recursos de hardware (CPU e memória) necessários para que a ferramenta suporte diferentes volumes de tráfego. A ferramenta, denominada *SYN Flood Interceptor for Netfilter* (SFINX), foi implementada na linguagem C, em ambiente GNU/Linux, como um Módulo Carregável do Kernel (LKM), e validada em um ambiente de testes simulado utilizando o *Graphical Network Simulator 3* (GNS3). Dois cenários de teste foram criados para avaliar a SFINX, a saber: (i) DoS Flood: ataque de negação de serviço com envio de pacotes SYN a uma taxa constante, sem variação na quantidade de pacotes por segundo; (ii) DoS Flood Controlado: ataque de negação de serviço com controle explícito da taxa de envio de pacotes por segundo. Os resultados das simulações demonstraram que o mecanismo foi eficaz na contenção de ataques do tipo SYN Flood, mantendo a disponibilidade do *firewall* mesmo sob uma taxa constante de envio superior a 160 mil pacotes SYN por segundo, em um ambiente com recursos limitados de hardware (3 GB de memória RAM e CPU de 2 GHz). Além disso, a ferramenta demonstrou resiliência em relação aos recursos de hardware disponíveis, permitindo, com base em modelagem matemática, a estimativa prévia da capacidade computacional necessária conforme a intensidade do tráfego SYN. Por exemplo, para filtrar um volume de 1.000 Mbps, são requeridos aproximadamente 3,28 GB de memória RAM e um processador de 2,48 GHz.

Palavras-chave: netfilter; tabela conntrack; negação de serviço; filtragem de pacotes; segurança de rede.

ABSTRACT

Denial-of-Service (DoS) attacks represent a significant threat to the availability of network security devices, such as firewalls, which become frequent targets due to their central role in protecting networks and services. Among these attacks, the SYN Flood stands out, exploiting a limitation in the Transmission Control Protocol (TCP) connection management by maintaining a large number of partially established connections, leading to the exhaustion of system resources. This type of attack particularly affects firewalls based on netfilter (a Linux kernel subsystem responsible for packet filtering and manipulation). This research aims to develop a mechanism for detecting and containing SYN Floods targeting netfilter-based firewalls. The proposed mechanism comprises three stages: (i) the formulation of a mathematical model to define a threshold for classifying suspicious SYN packets; (ii) the implementation of a tool to detect attacks and (iii) the development of a mathematical model to estimate the hardware resources (CPU and memory) required for the tool to support different traffic volumes. The tool, named *SYN Flood Interceptor for Netfilter* (SFINX), was implemented in the C language, in a GNU/Linux environment, as a Loadable Kernel Module (LKM), and validated in a simulated test environment using *Graphical Network Simulator 3* (GNS3). Two test scenarios were created to evaluate SFINX, namely: (i) DoS Flood: denial-of-service attack with the sending of SYN packets at a constant rate, without variation in the number of packets per second; (ii) Controlled DoS Flood: denial-of-service attack with explicit control of the packet sending rate per second. The simulation results demonstrated that the mechanism was effective in containing SYN Flood attacks, maintaining firewall availability even under a constant transmission rate exceeding 160,000 SYN packets per second, in an environment with limited hardware resources (3 GB of RAM and a 2 GHz CPU). Furthermore, the tool demonstrated resilience to available hardware resources, allowing, based on mathematical modeling, a preliminary estimate of the necessary computational capacity according to the intensity of SYN traffic. For example, to filter a volume of 1,000 Mbps, approximately 3.28 GB of RAM and a 2.48 GHz processor are required.

Keywords: netfilter; conntrack table; denial of service; packet filtering; network security.

LISTA DE ILUSTRAÇÕES

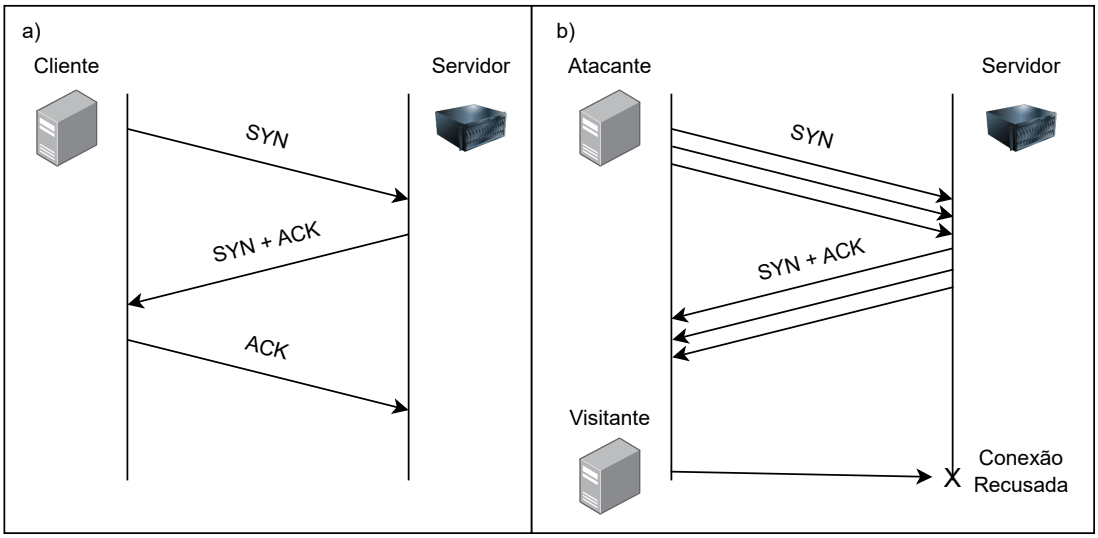
SUMÁRIO

1 INTRODUÇÃO

As redes de computadores tornaram-se um elemento essencial da sociedade moderna, sustentando desde atividades cotidianas até o funcionamento de infraestruturas críticas, como sistemas de energia elétrica, controle de tráfego aéreo, abastecimento de água, serviços hospitalares e sistemas bancários. Com essa crescente dependência, a disponibilidade dos serviços de rede passou a ser um requisito fundamental, tornando os ambientes computacionais especialmente sensíveis a ataques de negação de serviço DoS (Vedral, 2024).

O objetivo do DoS é tornar serviços indisponíveis por meio da exaustão deliberada de recursos computacionais, como capacidade de processamento, memória e largura de banda. O SYN Flood, um dos ataques de negação de serviço mais comuns, é uma técnica que explora o mecanismo de estabelecimento de conexões do TCP, conhecido como *three-way handshake*. No procedimento normal, o cliente envia um segmento SYN, o servidor responde com SYN + ACK e o cliente confirma com ACK, concluindo a conexão; no SYN Flood, o atacante envia repetidos segmentos SYN, impedindo a conclusão do *handshake* ao omitir deliberadamente o ACK final (Khan *et al.*, 2024). A Figura 1 (a) ilustra o procedimento normal de estabelecimento de uma conexão, enquanto a Figura 1 (b) mostra o ataque SYN Flood.

Figura 1 – (a) Procedimento Normal de Estabelecimento de Conexão (b) SYN Flood

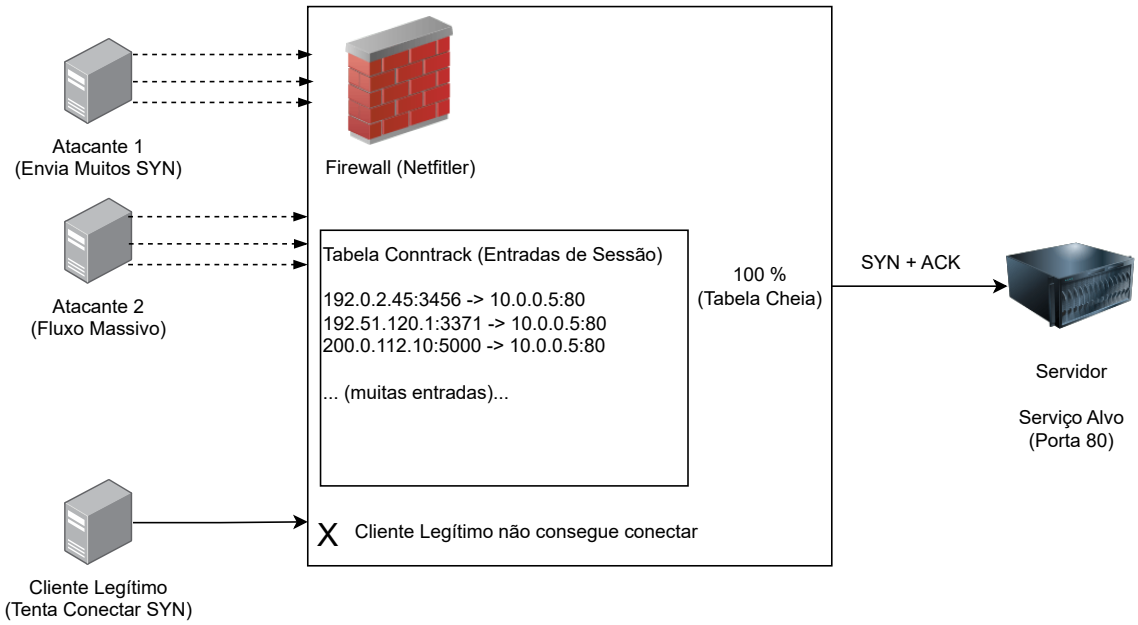


Fonte: Os Autores (2025).

Cada segmento SYN recebido por um *firewall* resulta na criação de uma entrada temporária em uma tabela de rastreamento de conexões, como a tabela conntrack, utilizada pelo netfilter (o *firewall* nativo do Linux). Esse processo tem como finalidade acompanhar o estado das conexões que atravessam o dispositivo, permitindo ao *firewall* distinguir pacotes pertencentes a comunicações já estabelecidas de novas tentativas. Para isso, o sistema armazena temporariamente informações como endereço de origem e destino, portas envolvidas e o estado atual do protocolo TCP (SYN enviado, SYN recebido, *handshake* incompleto, etc), até que a conexão seja finalizada ou expire após o tempo limite configurado. Em ataques do tipo SYN

Flood, um volume elevado de solicitações SYN falsas leva ao preenchimento acelerado dessa tabela com entradas incompletas, que permanecem ativas até que expire o tempo de retenção configurado. Quando o número de entradas atinge o limite máximo permitido, o *firewall* não consegue mais registrar novas conexões, inclusive as legítimas, resultando em perda de conectividade para usuários e serviços (Majkowski, 2020). A Figura 2 ilustra um ataque SYN Flood direcionado a um firewall, que está posicionado entre uma rede local e a Internet, demonstrando como o ataque sobrecarrega o dispositivo e impede a comunicação legítima.

Figura 2 – Ataque SYN Flood contra *Firewall* (conntrack cheio) - Cliente Legítimo Impedido



Fonte: Os Autores (2025).

Diversas técnicas têm sido usadas para mitigar ataques SYN Flood em *firewalls*. A limitação de taxa (*rate limiting*) controla o número de pacotes SYN aceitos, mas pode prejudicar usuários legítimos, especialmente em redes onde múltiplos dispositivos compartilham um mesmo IP. Filtragem por listas de bloqueios, mas sofre com IP spoofing e dificuldades para evitar falsos positivos (Bogdanoski; Suminoski; Risteski, 2012). Os *firewalls* com inspeção de estado e sistemas de prevenção de intrusão detectam padrões de ataque, mas podem ser sobrecarregados em ataques volumosos e exigem constante atualização (Aslam *et al.*, 2023).

Outra solução comum é aumentar a capacidade das tabelas de rastreamento de conexões, como a conntrack em *firewalls* Linux. Embora isso aumente a resistência do sistema, é uma medida paliativa, já que atacantes com recursos suficientes podem ultrapassar esses limites. Além disso, tabelas maiores podem degradar o desempenho do *firewall*, causando lentidão no processamento e aumento da latência para conexões legítimas (Majkowski, 2020). Esses problemas mostram a necessidade de abordagens mais eficientes para defender redes contra ataques SYN Flood.

Portanto, esta pesquisa tem como foco desenvolver um mecanismo para a detecção e mitigação de ataques SYN Flood direcionados a *firewalls* baseados no netfilter, considerando

que esses sistemas representam uma das soluções de proteção mais utilizadas mundialmente (Han *et al.*, 2021). O mecanismo proposto consiste na definição de um limiar para a classificação de pacotes SYN maliciosos, no desenvolvimento de uma ferramenta fundamentada nesse critério para identificação e contenção de ataques, e na estimativa dos recursos de hardware (CPU e memória) necessários para atender a diferentes volumes de tráfego.

Em ataques do tipo SYN Flood, é comum que a tabela conntrack registre um grande número de conexões nos estados UNREPLIED, SYN_SENT ou SYN_RECV, que indicam, respectivamente:

- **SYN_SENT:** O cliente envia o SYN e aguarda o SYN-ACK do servidor.
- **SYN_RECV:** O servidor envia o SYN-ACK e aguarda o ACK final do cliente.
- **UNREPLIED:** O cliente envia o SYN, mas se não houver resposta de SYN-ACK após um tempo, o estado muda de SYN_SENT para UNREPLIED.

No entanto, esses estados não devem ser interpretados automaticamente como um indicativo de ataque, uma vez que também pode ocorrer em situações legítimas, como em conexões temporariamente atrasadas por latência de rede. Diante disso, torna-se necessária a adoção de uma metodologia confiável para classificar conexões marcadas como UNREPLIED, SYN_SENT ou SYN_RECV, na tabela conntrack, distinguindo as legítimas das maliciosas. Assim, este trabalho investigou o tempo, em milissegundos (ms), que conexões legítimas normalmente levam para completar o ciclo do *three-way handshake* e, a partir dessa análise, propôs uma modelagem matemática capaz de caracterizar e diferenciar conexões maliciosas, como as geradas em ataques SYN Flood, das conexões legítimas. Além disso, uma ferramenta foi implementada com o propósito de utilizar os dados obtidos matematicamente para identificar e conter ataques SYN Flood por meio da manipulação direta da tabela conntrack.

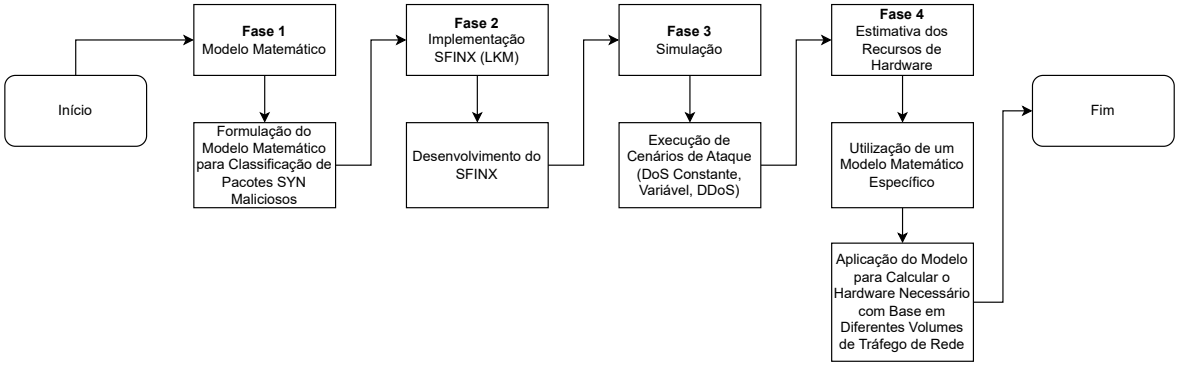
Em ambientes de redes, a estimativa do dimensionamento do hardware para *firewalls* é geralmente baseada na largura de banda do link de dados a ser protegido, uma vez que o *firewall* deve processar adequadamente o volume de tráfego que atravessa a rede, inspecionando cada pacote individualmente. Dessa forma, torna-se fundamental calcular a capacidade do *firewall* em PPS para assegurar seu desempenho eficaz. Neste contexto, este trabalho também focou em estimar a quantidade de memória RAM e capacidade de CPU necessárias para que o *firewall* suporte um determinado número de pacotes SYN por segundo. Para tal, a largura de banda foi convertida em PPS, possibilitando o dimensionamento matemático do hardware requerido para suportar eficientemente o tráfego previsto.

Por fim, as simulações mostraram que o mecanismo proposto controla eficazmente os ataques de SYN Flood, mantendo o *firewall* disponível mesmo sob alta taxa de pacotes SYN, além de ser escalável conforme a memória e CPU disponíveis.

2 METODOLOGIA

O trabalho será organizado em quatro fases principais, cada uma dividida em etapas específicas.

Figura 3 – Fluxograma - Metodologia



Fonte: Os Autores (2025).

Fase 1 – Modelo Matemático

Este trabalho investigou o tempo, em ms, que conexões legítimas normalmente levavam para completar o ciclo do *three-way handshake* e, a partir dessa análise, propôs uma modelagem matemática capaz de caracterizar e diferenciar pacotes SYN maliciosos dos legítimos.

Fase 2 – Implementação SFINX (LKM)

Na segunda fase, foi desenvolvido a ferramenta SFINX sob a forma de um LKM, implementado em linguagem C e integrado ao netfilter. A ferramenta teve como função interceptar pacotes SYN e aplicar o modelo matemático previamente desenvolvido, classificando as conexões como legítimas ou maliciosas e tomando decisões de descarte ou não. Durante o processo de implementação, foram adotadas estratégias de otimização voltadas ao uso eficiente de memória e ao processamento em tempo real, de modo a reduzir a sobrecarga do sistema.

Fase 3 – Simulações

A terceira fase consistiu na configuração de um ambiente de testes simulado no GNS3, no qual foram implementados diferentes cenários de ataque para avaliar o mecanismo proposto.

Fase 4 – Estimativa dos Recursos de Hardware

Por fim, a quarta etapa destinou-se à estimativa dos recursos de hardware necessários para o funcionamento da ferramenta em diferentes condições de tráfego. Para tal, foi desenvolvido e aplicado um modelo matemático que correlacionou o volume de pacotes processados à demanda de recursos computacionais, em especial o processamento da CPU e uso de memória RAM. Esse modelo permitiu prever os requisitos mínimos de infraestrutura para sua implantação em redes de diferentes portes. A análise dessa etapa foi fundamental para determinar a viabilidade prática da solução proposta, garantindo que seu uso fosse compatível com a realidade de ambientes computacionais heterogêneos, desde pequenas redes corporativas até infraestruturas de grande escala.

3 CONCLUSÃO

Este trabalho apresentou o desenvolvimento e a validação de um mecanismo para mitigação de ataques do tipo SYN Flood em *firewalls* baseados no framework netfilter. O mecanismo proposto foi estruturado em quatro partes: (i) análise estatística do tempo de estabelecimento do three-way handshake do protocolo TCP, resultando na modelagem matemática do comportamento de conexões legítimas; (ii) implementação da ferramenta SFINX (SYN Flood Interceptor for netfilter), desenvolvida em linguagem C sob a forma de um LKM; (iii) realização de simulações em ambiente controlado no GNS3; e (iv) elaboração de um modelo de dimensionamento de hardware para estimar o consumo de CPU e memória em função do volume de tráfego processado.

A análise estatística demonstrou que o tempo de estabelecimento de conexões legítimas segue uma distribuição Weibull, permitindo definir um limite superior de 100 ms para o handshake válido. Esse resultado fundamentou o limiar adotado pelo mecanismo para identificar conexões suspeitas. A ferramenta SFINX, ao interceptar pacotes SYN e aplicar esse modelo de classificação diretamente na camada de rastreamento de conexões (conntrack), mostrou-se capaz de detectar e descartar pacotes maliciosos antes que causem saturação da tabela de estados, reduzindo de forma expressiva a possibilidade de exaustão de recursos do sistema.

Os testes realizados contemplaram dois cenários distintos: DoS Flood e DoS Flood Controlado. Em todos eles, a solução demonstrou alta capacidade de contenção do ataque, mantendo a disponibilidade do *firewall* mesmo sob taxas superiores a 160 mil pacotes SYN por segundo, em ambiente com apenas 3 GB de RAM e CPU de 2 GHz. Além disso, o modelo de dimensionamento proposto possibilitou estimar, por exemplo, que para processar 1 Gbps de tráfego SYN seriam necessários cerca de 3,28 GB de RAM e 2,48 GHz de processamento total, o que reforça a aplicabilidade da solução em cenários de diferentes escalas.

As principais contribuições deste trabalho podem ser resumidas como:

- A definição de um modelo estatístico para caracterização de conexões legítimas TCP, servindo de base para a detecção de anomalias em tempo real;
- O desenvolvimento do módulo SFINX, integrado ao netfilter, com atuação direta sobre a tabela conntrack para mitigação preventiva de SYN Floods;
- A validação experimental da ferramenta em ambiente simulado de rede, demonstrando desempenho eficiente;
- A formulação de um modelo matemático de dimensionamento de hardware, permitindo prever a escalabilidade da solução conforme o volume de tráfego e recursos disponíveis;
- A proposta contribui, assim, para o avanço da segurança de redes em plataformas abertas, reforçando a importância de soluções baseadas em software livre e de fácil integração com arquiteturas existentes. Além da mitigação de SYN Floods, o estudo

estabelece um modelo metodológico que pode ser expandido para novas formas de ataques de negação de serviço.

Para trabalhos futuros, pretende-se estender o mecanismo SFINX para lidar com outros ataques baseados em estado no protocolo TCP e UDP, tais como:

- Mecanismos integrados de detecção e mitigação de ataques TCP, incluindo ACK Flood, FIN/PSH Flood e RST Flood;
- Defesas contra ataques UDP Flood em *firewalls* netfilter, utilizando abordagens probabilísticas e adaptativas.

REFERÊNCIAS

ASLAM, M. *et al.* A comprehensive review on detection and mitigation of TCP-SYN flooding DDoS attacks. **Electronics**, v. 12, n. 4, p. 993, feb 2023. ISSN 2079-9292.

BOGDANOSKI, M.; SUMINOSKI, T.; RISTESKI, A. Tcp-syn flooding attack in wireless networks. *In*: IEEE. 2012 INTERNATIONAL CONFERENCE ON INNOVATIONS IN INFORMATION TECHNOLOGY (IIT). 2012. **Anais [...]** [*S.l.*], 2012. p. 253–257.

HAN, B. *et al.* A survey on network function virtualization. **IEEE Communications Surveys Tutorials**, v. 23, n. 4, p. 2574–2603, 2021.

KHAN, Z. *et al.* A systematic review on denial of service attacks in cloud computing: research challenges and solutions. **Cluster Computing**, Springer v. 27, n. 2, p. 1389–1413, 2024.

MAJKOWSKI, M. Conntrack tales - one thousand and one flows. **Cloudflare Blog**, „, Apr 2020. Disponível em: https://blog.cloudflare.com/conntrack-tales-one-thousand-and-one-flows/.

VEDRAL, J. **Industrial and Laboratory Measuring Systems: Sensors, Distributed, Modular and Wireless Systems**. 1. ed. [*S.l.*]: River Publishers, 2024. (River Publishers Series in Energy Management).

ZHAO, J. *et al.* Timiner: Automatically extracting and analyzing categorized cyber threat intelligence from social data. **Computers Security**, v. 95„ p. 101867, 2020. ISSN 0167-4048. Disponível em: https://www.sciencedirect.com/science/article/pii/S0167404820301395.

APÊNDICE A – UM MECANISMO CONTRA ATAQUES DO TIPO SYN FLOOD PARA FIREWALLS BASEADOS NO FRAMEWORK NETFILTER

UM MECANISMO CONTRA ATAQUES DO TIPO SYN FLOOD PARA FIREWALLS BASEADOS NO FRAMEWORK NETFILTER

Fábio Israel Ramos Esmanhotto, Joilson Alves Junior, Daniel Fernando Pigatto, Marcos Eduardo Pivaro Monteiro

***Resumo*—Ataques de Negação de Serviço (DoS) representam uma ameaça relevante à disponibilidade de dispositivos de segurança, especialmente *firewalls*, frequentemente visados por seu papel central na proteção de redes. Entre eles, destaca-se o SYN Flood, que explora limitações do *Transmission Control Protocol* (TCP) ao manter inúmeras conexões parcialmente estabelecidas, causando esgotamento de recursos. Esse problema afeta particularmente *firewalls* baseados no netfilter, subsistema do kernel Linux responsável pela filtragem de pacotes. Esta pesquisa propõe um mecanismo para detecção e contenção de SYN Floods composto por três etapas: (i) definição de um modelo matemático para estabelecer um limiar de classificação de pacotes SYN suspeitos; (ii) implementação de uma ferramenta para detectar ataques; e (iii) modelagem para estimar requisitos de hardware (CPU e memória) conforme o volume de tráfego. A ferramenta, denominada SFINX, foi desenvolvida em C como Módulo Carregável do Kernel (LKM) em GNU/Linux e validada em ambiente simulado no *Graphical Network Simulator 3* (GNS3). Dois cenários foram avaliados: DoS Flood com taxa constante e DoS Flood Controlado com variação explícita da taxa de envio. Os resultados indicaram eficácia na contenção do ataque, mantendo a disponibilidade do *firewall* sob mais de 160 mil pacotes SYN por segundo em hardware limitado (3 GB de RAM e CPU de 2 GHz), além de demonstrar resiliência e permitir a estimativa prévia de recursos computacionais, como aproximadamente 3,28 GB de RAM e CPU de 2,48 GHz para filtrar 1.000 Mbps.**

***Index Terms*—Netfilter, Tabela Conntrack, Negação de Serviço, Filtragem de Pacotes, Segurança de Rede.**

I. INTRODUÇÃO

As redes de computadores tornaram-se um elemento essencial da sociedade moderna, sustentando desde atividades cotidianas até o funcionamento de infraestruturas críticas, como sistemas de energia elétrica, controle de tráfego aéreo, abastecimento de água, serviços hospitalares e sistemas bancários. Com essa crescente dependência, a disponibilidade dos serviços de rede passou a ser um requisito fundamental, tornando os ambientes computacionais especialmente sensíveis a ataques de negação de serviço [1].

O objetivo do DoS é tornar serviços indisponíveis por meio da exaustão deliberada de recursos computacionais, como capacidade de processamento, memória e largura de banda. O SYN Flood, um dos ataques de negação de serviço mais comuns, é uma técnica que explora o mecanismo de estabelecimento de conexões do TCP, conhecido como *three-way handshake*. No procedimento normal, o cliente envia um segmento SYN, o servidor responde com SYN + ACK e o cliente confirma com ACK, concluindo a conexão; no SYN Flood, o atacante envia repetidos segmentos SYN, impedindo a conclusão do *handshake* ao omitir deliberadamente o ACK final [2]. A Figura 1 (a) ilustra o procedimento normal de estabelecimento de uma conexão, enquanto a Figura 1 (b) mostra o ataque SYN Flood.

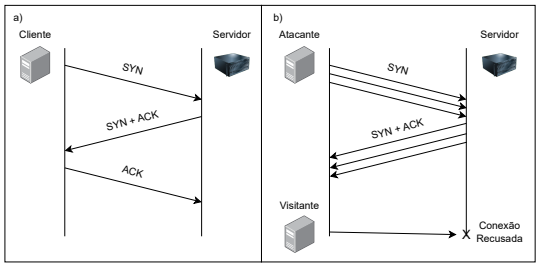


Figura 1. Conexões

Cada segmento SYN recebido por um *firewall* resulta na criação de uma entrada temporária em uma tabela de rastreamento de conexões, como a tabela conntrack, utilizada pelo netfilter (o *firewall* nativo do Linux). Esse processo tem como finalidade acompanhar o estado das conexões que atravessam o dispositivo, permitindo ao *firewall* distinguir pacotes pertencentes a comunicações já estabelecidas de novas tentativas. Para isso, o sistema armazena temporariamente informações como endereço de origem e destino, portas envolvidas e o estado atual do protocolo TCP (SYN enviado, SYN recebido, *handshake* incompleto, etc), até que a conexão seja finalizada ou expire após o tempo limite configurado. Em ataques do tipo SYN Flood, um volume elevado de solicitações SYN falsas leva ao preenchimento acelerado dessa tabela com entradas incompletas, que permanecem ativas até que expire o tempo de retenção configurado. Quando o número de entradas atinge o limite máximo permitido, o *firewall* não consegue mais registrar novas conexões, inclusive as legítimas, resultando em perda de conectividade para usuários e serviços [3]. A Figura 2 ilustra um ataque SYN Flood direcionado a um firewall, que esta posicionado entre uma rede local e a Internet, demonstrando como o ataque sobrecarrega o dispositivo e impede a comunicação legítima.

Diversas técnicas têm sido usadas para mitigar ataques SYN Flood em *firewalls*. A limitação de taxa (*rate limiting*) controla o número de pacotes SYN aceitos, mas pode prejudicar usuários legítimos, especialmente em redes onde múltiplos dispositivos compartilham um mesmo IP. Filtragem por listas negras bloqueia fontes suspeitas, mas sofre com IP spoofing e dificuldades para evitar falsos positivos [4]. Os *firewalls* com inspeção de estado e sistemas de prevenção de intrusão detectam padrões de ataque, mas podem ser sobrecarregados em ataques volumosos e exigem constante atualização [5].

Outra solução comum é aumentar a capacidade das tabelas de rastreamento de conexões, como a conntrack em *firewalls* Linux. Embora isso aumente a resistência do sistema, é uma

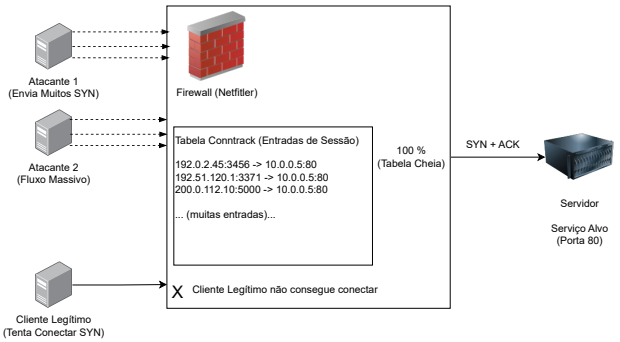


Figura 2. Conexões

medida paliativa, já que atacantes com recursos suficientes podem ultrapassar esses limites. Além disso, tabelas maiores podem degradar o desempenho do *firewall*, causando lentidão no processamento e aumento da latência para conexões legítimas [3]. Esses problemas mostram a necessidade de abordagens mais eficientes para defender redes contra ataques SYN Flood.

Portanto, esta pesquisa tem como foco desenvolver um mecanismo para a detecção e mitigação de ataques SYN Flood direcionados a *firewalls* baseados no netfilter, considerando que esses sistemas representam uma das soluções de proteção mais utilizadas mundialmente [6]. O mecanismo proposto consiste na definição de um limiar para a classificação de pacotes SYN maliciosos, no desenvolvimento de uma ferramenta fundamentada nesse critério para identificação e contenção de ataques, e na estimativa dos recursos de hardware (CPU e memória) necessários para atender a diferentes volumes de tráfego.

Em ataques do tipo SYN Flood, é comum que a tabela conntrack registre um grande número de conexões nos estados UNREPLIED, SYN_SENT ou SYN_RECV, que indicam, respectivamente:

- **SYN_SENT:** O cliente envia o SYN e aguarda o SYN-ACK do servidor.
- **SYN_RECV:** O servidor envia o SYN-ACK e aguarda o ACK final do cliente.
- **UNREPLIED:** O cliente envia o SYN, mas se não houver resposta de SYN-ACK após um tempo, o estado muda de SYN_SENT para UNREPLIED.

No entanto, esses estados não devem ser interpretados automaticamente como um indicativo de ataque, uma vez que também pode ocorrer em situações legítimas, como em conexões temporariamente atrasadas por latência de rede. Diante disso, torna-se necessária a adoção de uma metodologia confiável para classificar conexões marcadas como UNREPLIED, SYN_SENT ou SYN_RECV, na tabela conntrack, distinguindo as legítimas das maliciosas. Assim, este trabalho investiga o tempo, em milissegundos (ms), que conexões legítimas normalmente levam para completar o ciclo do *three-way handshake* e, a partir dessa análise, propõe uma modelagem matemática capaz de caracterizar e diferenciar conexões maliciosas, como as geradas em ataques SYN Flood, das conexões legítimas. Além disso, uma ferramenta foi implementada com o propósito de utilizar os dados obtidos matematicamente para identificar e conter ataques SYN Flood por meio da manipulação direta da tabela conntrack.

Em ambientes de redes, a estimativa do dimensionamento do hardware para *firewalls* é geralmente baseada na largura de banda do link de dados a ser protegido, uma vez que o *firewall* deve processar adequadamente o volume de tráfego que atravessa a rede, inspecionando cada pacote individualmente. Dessa forma, torna-se fundamental calcular a capacidade do *firewall* em PPS para assegurar seu desempenho eficaz. Neste contexto, este trabalho também focou em estimar a quantidade de memória RAM e capacidade de CPU necessárias para que o *firewall* suporte um determinado número de pacotes SYN por segundo. Para tal, a largura de banda foi convertida em PPS, possibilitando o dimensionamento matemático do hardware requerido para suportar eficientemente o tráfego previsto.

Por fim, as simulações mostraram que o mecanismo proposto controla eficazmente os ataques de SYN Flood, mantendo o *firewall* disponível mesmo sob alta taxa de pacotes SYN, além de ser escalável conforme a memória e CPU disponíveis.

II. Trabalhos Relacionados

Estudos recentes têm proposto diferentes abordagens para a detecção e mitigação de ataques de negação de serviço distribuídos (DDoS) em redes modernas, considerando a crescente complexidade dos padrões de tráfego e das infraestruturas de comunicação. Técnicas baseadas em aprendizado de máquina, ambientes de simulação e programabilidade da rede vêm sendo amplamente exploradas com o objetivo de aumentar a precisão da detecção e reduzir o impacto dos ataques.

Kumar *et al.* [7] apresentaram um framework baseado em Redes Generativas Adversariais (GANs) para a detecção e mitigação de ataques de baixa taxa em ambientes de Redes Definidas por Software (SDN), obtendo elevados índices de precisão por meio da análise temporal do tráfego. De forma semelhante, Salahuddin *et al.* [8] propuseram o mecanismo Chronos, baseado em autoencoders sensíveis ao comportamento temporal, demonstrando eficácia na identificação de ataques pouco perceptíveis. Apesar do bom desempenho, essas abordagens apresentam elevada dependência de grandes volumes de dados para treinamento e de recursos computacionais, o que pode limitar sua aplicação em ambientes com restrições operacionais.

Abordagens fundamentadas em simulação também desempenham papel relevante na avaliação de mecanismos de defesa. Meka *et al.* [9] propuseram um modelo integrado utilizando o simulador NS-3, combinando detecção, mitigação e rastreamento da origem dos ataques. Samatar *et al.* [10] utilizaram o Simulador Gráfico de Redes-3, do inglês *Graphical Network Simulator-3* (GNS3) para a construção de cenários realistas de ataques de inundação, avaliando mecanismos tradicionais de segurança, como listas de controle de acesso e segmentação por Rede Local Virtual, do inglês *Virtual Local Area Network* (VLAN). Complementarmente, Msaad *et al.* [11] analisaram o

impacto dos ataques sobre o desempenho da rede por meio de simulações controladas. Embora essas abordagens permitam experimentação sistemática, elas nem sempre refletem plenamente a dinâmica e a heterogeneidade das redes em produção.

Soluções baseadas na programabilidade do plano de dados também têm se destacado como alternativas promissoras para a mitigação em tempo real. Ilha *et al.* [12] apresentaram o Euclid, um sistema implementado em P4 capaz de realizar detecção e mitigação diretamente nos dispositivos de rede, reduzindo a latência e a dependência de controladores centralizados. No entanto, a adoção dessas soluções exige infraestrutura especializada e elevado nível de especialização técnica.

De forma geral, os trabalhos existentes concentram-se em aspectos específicos da defesa contra ataques DDoS, abordando isoladamente a precisão na detecção, a avaliação por simulação, a mitigação em tempo real ou a análise de impacto. Poucas propostas integram, de maneira abrangente, mecanismos de monitoramento contínuo, análise adaptativa do tráfego e resposta dinâmica, considerando simultaneamente a facilidade de implantação. Além disso, muitas soluções baseadas em aprendizado utilizam modelos estáticos e treinamento offline, limitando sua capacidade de adaptação a padrões de ataque em constante evolução.

Diante dessas limitações, este trabalho propõe uma abordagem integrada para a detecção e mitigação de ataques DDoS, baseada no monitoramento contínuo do tráfego, na análise comportamental dos fluxos e na aplicação dinâmica de políticas de defesa. A proposta busca reduzir a dependência de grandes volumes de dados para treinamento, minimizar o custo computacional e facilitar a implantação em ambientes heterogêneos, contribuindo para o avanço do estado da arte em sistemas de gerenciamento e segurança de redes.

III. Mecanismo Contra Ataques do Tipo SYN Flood para Firewalls Baseados no Framework Netfilter

Essa seção descreve o mecanismo proposto, apresentando sua arquitetura, seu fluxo de funcionamento e os principais componentes envolvidos na solução, bem como os critérios adotados para sua implementação e avaliação.

A. Modelo Matemático para a Classificação de Pacotes SYN Originados de Ataques SYN Flood

Um dos *firewalls* mais utilizados no mundo é o netfilter, especialmente em sistemas operacionais baseados em Linux [13]. Assim como outros sistemas de filtragem, o netfilter também é vulnerável a ataques de SYN flood. Para organizar e rastrear as conexões de rede, o netfilter utiliza a tabela conntrack, a qual é responsável por manter o estado das conexões ativas. O ataque de SYN flood explora justamente essa tabela, tentando saturá-la com conexões TCP semiabertas, isto é, situações em que o cliente envia um pacote SYN, o servidor responde com SYN-ACK, mas o terceiro pacote ACK nunca é enviado pelo cliente. Como a tabela conntrack possui um número limitado de entradas, essa sobrecarga pode levar ao esgotamento dos recursos internos do *firewall*, resultando na indisponibilidade completa do dispositivo [13].

Conexões no estado semiaberto são registradas como UNREPLIED, SYN_SENT ou SYN_RECV na tabela conntrack. No entanto, esses estados não devem ser interpretados de forma imediata como um indicativo de ataque, uma vez que também podem ocorrer em cenários legítimos. Entre os principais fatores que podem levar a esse comportamento estão a latência elevada na comunicação (como em conexões via satélite), a perda de pacotes, dispositivos com tempo de resposta mais lento, como equipamentos de IoT, e aplicações que operam com *timeouts* prolongados [14].

Por essa razão, surge a necessidade de adotar uma metodologia segura para classificar conexões marcadas como UNREPLIED, SYN_SENT ou SYN_RECV na tabela conntrack como provenientes de ataques do tipo SYN flood. Diante disso, este trabalho buscou investigar quanto tempo, em milissegundos (ms), uma conexão legítima leva para completar o ciclo completo do *three-way handshake* do protocolo TCP, composto pelas etapas SYN, SYN-ACK e ACK, e, a partir disso, caracterizar e diferenciar as conexões legítimas das maliciosas.

Para isso foram conduzidos estudos empíricos com o intuito de coletar dados reais e atualizados. O objetivo foi analisar o comportamento temporal do processo de *handshake* TCP e, a partir dessa análise, propor critérios capazes de distinguir conexões legítimas daquelas potencialmente oriundas de ataques do tipo SYN flood.

Para estimar o tempo necessário para o estabelecimento de uma conexão TCP via *three-way handshake*, foi adotada a seguinte metodologia: medir o Tempo de Ida e Volta, do inglês *Round Trip Time* (RTT) por meio do comando PING, que utiliza pacotes Protocolo de Mensagens de Controle da Internet, do inglês *Internet Control Message Protocol* (ICMP) para calcular o tempo total de ida e volta de pacotes até o destino. Com base nesse valor, aplicou-se o fator de ajuste 1,5 × RTT, considerando que o handshake TCP (SYN $\rightarrow$ SYN + ACK $\rightarrow$ ACK) envolve uma troca adicional de pacotes em relação ao RTT simples, e, portanto, seu tempo total tende a ser ligeiramente superior ao de uma requisição ICMP. Esse fator foi utilizado como uma aproximação empírica, reconhecendo-se que, embora ICMP e TCP operem em camadas distintas do modelo OSI [15], ambos estão sujeitos às mesmas condições de rede (como latência, perda e congestionamento), o que permite uma estimativa confiável do tempo necessário para o estabelecimento da conexão.

Os dados foram coletados por meio do envio de pacotes para Rede de Área Local, do inglês *Local Area Network* (LAN) e Rede de Longa Distância, do inglês *Wide Area Network* (WAN), tanto em âmbito nacional quanto internacional. A metodologia adotada consistiu na realização de 35 medições para cada tipo de rede. Nos casos em que os destinos eram redes WAN, os testes foram conduzidos a partir de três diferentes origens: uma rede doméstica (com largura de banda de 100 Mbps), uma rede empresarial (200 Mbps) e uma conexão via satélite (300 Mbps). Para os testes em rede local, a origem foi exclusivamente a própria LAN. A Figura 3 ilustra a metodologia adotada.

A seguir, são apresentadas as medidas de média, mediana, moda, amplitude, desvio padrão e coeficiente de variação para

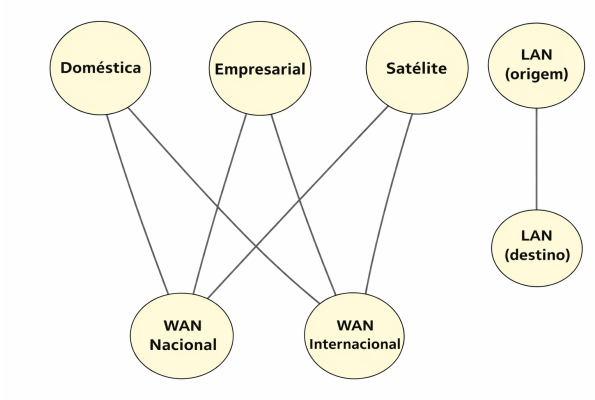


Figura 3. Conexões

cada um dos cenários avaliados.

Conforme pode ser visualizado na Tabela I, as 35 medições realizadas em ambiente de rede local apresentaram média, mediana e moda iguais a 1,5 ms, com desvio padrão nulo e coeficiente de variação de 0%, indicando total uniformidade nas respostas. Isso é coerente com o comportamento esperado em redes locais estáveis, sem interferência ou congestionamento.

Tabela I
LAN COM ORIGEM E DESTINO INTERNOS

Medida Estatística	Valor
Média	1,5 ms
Mediana	1,5 ms
Moda	1,5 ms
Amplitude (máx.–mín.)	0 ms
Desvio Padrão	0,00 ms
Coef. de Variação (%)	0,0

Conforme pode ser visualizado na Tabela II, os dados coletados para o tempo de resposta em uma rede WAN dentro do Brasil apresentam uma média de aproximadamente 32,6 ms, com um desvio padrão relativamente baixo de 1,56 ms, indicando pouca variação entre as medidas. O coeficiente de variação de 4,79% reforça essa estabilidade nos tempos de resposta. Esses valores estão dentro do esperado para conexões de internet de longa distância nacionais, onde fatores como a infraestrutura regional, roteamento e congestionamento podem influenciar, mas ainda permitem um desempenho estável para a maioria das aplicações. Essa consistência sugere que o estabelecimento do *three-way handshake* TCP em uma WAN brasileira ocorre em um intervalo de tempo previsível e confiável, o que é fundamental para garantir a eficiência das conexões e a detecção de eventuais anomalias, como ataques do tipo SYN Flood, que poderiam alterar significativamente esses tempos.

Como observado na Tabela III, os dados coletados para uma rede WAN internacional apresentam uma média de aproximadamente 51,14 ms, com um desvio padrão relativamente baixo de 2,11 ms, indicando pouca variação entre as medidas. O coeficiente de variação de 4,13% reforça essa estabilidade nos tempos de resposta. Esses valores estão dentro do esperado para conexões internacionais de longa distância, onde fatores

Tabela II
REDES WAN COM DESTINO NACIONAL

Medida Estatística	Valor
Média	32,6 ms
Mediana	33,0 ms
Moda	33,0 ms
Amplitude (máx.–mín.)	6 ms
Desvio padrão	1,56 ms
Coef. de variação (%)	4,79

como infraestrutura global, roteamento e latências físicas influenciam, mas ainda permitem um desempenho estável para a maioria das aplicações. Novamente essa previsibilidade é fundamental para garantir a eficiência das conexões e a detecção de eventuais anomalias, como ataques do tipo SYN Flood, que poderiam alterar esses tempos.

Tabela III
REDES WAN COM DESTINO INTERNACIONAL

Medida Estatística	Valor
Média	51,14 ms
Mediana	51,00 ms
Moda	49,5 ms
Amplitude (máx.–mín.)	9 ms
Desvio padrão	2,11 ms
Coef. de variação (%)	4,13

Os dados apresentados referem-se ao tempo de estabelecimento de conexões legítimas do protocolo TCP. Entretanto, esses dados foram obtidos por meio de amostras aleatórias e, portanto, não permitem a generalização do tempo médio com total confiança, visto que a média amostral é apenas uma estimativa da verdadeira média populacional e está sujeita a incertezas inerentes ao processo amostral. Dessa forma, optou-se por aplicar métodos estatísticos para modelar a distribuição probabilística dos dados, visando reduzir a incerteza na estimação da média populacional. O procedimento adotado consiste na identificação da função de densidade de probabilidade que melhor representa o comportamento dos tempos observados e, a partir dessa função, calcular a probabilidade de o tempo de estabelecimento exceder determinados limites de interesse.

Para identificar a distribuição que melhor se ajusta aos dados das redes, foram consideradas inicialmente as distribuições clássicas Normal, Exponencial e Log-Normal, amplamente utilizadas em análises de variáveis contínuas e de tempo. Em seguida, o conjunto de modelos candidatos foi ampliado para incluir também as distribuições Gamma, *Weibull* e Logística, com o objetivo de verificar possíveis ajustes mais adequados. Os critérios de seleção (AIC/BIC) e os KS indicaram que a distribuição *Weibull* resultou no melhor ajuste global aos dados, sendo, portanto, selecionada como a mais representativa. Essa abordagem assegura maior robustez metodológica, ao contemplar tanto modelos clássicos quanto alternativas mais flexíveis. A seguir, são apresentados os dados de ajuste para cada um dos cenários analisados: redes WAN com destino nacional, redes WAN com destino internacional e redes LAN com destino à própria rede local. A análise se inicia com os resultados obtidos a partir do envio de pacotes para redes WAN internacionais.

B. Redes WAN com Destino Internacional

A Tabela IV apresenta a comparação entre diferentes modelos de distribuição de probabilidade ajustados aos dados empíricos de tempo de estabelecimento do three-way handshake TCP. Os critérios utilizados para avaliação foram o Critério de Informação de Akaike, do inglês *Akaike Information Criterion* (AIC) e o Critério de Informação Bayesiano, do inglês *Bayesian Information Criterion* (BIC), sendo que valores menores indicam melhor qualidade de ajuste. O modelo *Weibull* apresentou o melhor desempenho geral, com os menores valores de AIC (146,47) e BIC (152,88), sendo, portanto, o mais adequado para representar a duração do *handshake*. O modelo Gamma teve desempenho semelhante, com ligeiramente maiores AIC (146,71) e BIC (153,12), indicando que também oferece um bom ajuste. As distribuições Log-Normal e Normal apresentaram ajustes aceitáveis, porém com qualidade inferior, refletida nos valores mais altos dos critérios. Já a distribuição Exponencial obteve os piores resultados, com AIC e BIC significativamente mais elevados (159,43 e 161,00, respectivamente), sendo inadequada para modelar os dados observados.

Tabela IV
COMPARAÇÃO DOS MODELOS COM BASE EM AIC E BIC

Modelo	AIC	BIC	Observação
Weibull	146,47	152,88	Melhor ajuste geral
Gamma	146,71	153,12	Ajuste semelhante ao Weibull
Log-normal	146,92	153,33	Ajuste aceitável
Normal	147,50	153,90	Ajuste inferior
Exponencial	159,43	161,00	Pior ajuste

O teste de KS (Tabela V) é um teste estatístico não paramétrico amplamente utilizado para avaliar a aderência de uma amostra a uma distribuição teórica contínua. Sua utilização é frequentemente combinada com outras métricas, como os critérios de informação AIC e BIC, para uma avaliação mais robusta da qualidade do ajuste. Dentre as distribuições analisadas a *Weibull* se mostrou a melhor candidata, com o menor AIC e o maior p-valor no teste KS. Gamma e Log-Normal também tiveram ajuste bom, mas ligeiramente piores que a *Weibull*. A Normal foi razoável. A Exponencial não se ajusta bem.

Tabela V
RESULTADOS DO TESTE DE KOLMOGOROV–SMIRNOV (KS)

Modelo	p-valor	Interpretação
Weibull	0,41	Aceita (melhor ajuste)
Gamma	0,32	Aceita
Log-normal	0,30	Aceita
Normal	0,19	Aceitável
Exponencial	0,002	Rejeitada

Convém destacar a formulação matemática da distribuição *Weibull*. A Função de Densidade de Probabilidade, do inglês *Probability Density Function* (PDF) descreve a forma da distribuição e permite calcular a probabilidade relativa de ocorrência dos valores. Na Fórmula 1 é apresentada a expressão geral da PDF da *Weibull* com seus parâmetros de forma (k) e escala (λ), conforme ajustados aos dados.

Função de Densidade da Weibull (PDF):

$$f(x;k,\lambda) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k}, & x \geq 0, \\ 0, & x < 0, \end{cases} \tag{1}$$

Onde:

- x = variável aleatória (tempo, resposta, etc.)
- $k > 0$ = parâmetro de forma (shape)
- $\lambda > 0$ = parâmetro de escala (scale)
- $f(x)$ = valor da densidade de probabilidade para x

Com base na estimação dos parâmetros pelo método de MLE, obteve-se a forma específica da função densidade de probabilidade da *Weibull* ajustada aos dados empíricos coletados (Fórmula 2). Essa função representa matematicamente o modelo que melhor descreve o comportamento observado, permitindo o cálculo de probabilidades associadas a diferentes intervalos de valores.

Parâmetros ajustados

- k (shape) $\approx$ **3,63**
- λ (scale) $\approx$ **52,24**

Função de densidade Weibull ajustada:

$$f(x) = \frac{3{,}63}{52{,}24}\left(\frac{x}{52{,}24}\right)^{2{,}63} \cdot e^{-\left(\frac{x}{52{,}24}\right)^{3{,}63}} \quad \text{para } x \geq 0 \tag{2}$$

A partir da função de distribuição ajustada aos dados empíricos, foi possível estimar a probabilidade de o tempo total do *three-way handshake* exceder 51,14 ms, valor máximo observado na amostra empírica (ver Tabela III). Considerando X como a variável aleatória que representa a duração dessa etapa da conexão TCP, obteve-se que a probabilidade de a variável aleatória X assumir um valor maior que 51,14 é de $39,63\%(P(X > 51,14) = 39,63\%)$, o que indica que 60,37% das conexões legítimas se completam em até 51,14 ms.

Mesmo assim, novos limites foram estabelecidos com base na função demonstrada na Fórmula 2. A Tabela VI apresenta os percentuais de observações que se encontram acima e abaixo de determinados limiares de latência.

Tabela VI
PERCENTUAIS QUE SE ENCONTRAM ACIMA E ABAIXO DA LATÊNCIA MÉDIA OBSERVADA

Latência (ms)	% Acima da Latência	% Abaixo da Latência
60	19,14%	80,86%
75	2,43%	97,57% (comportamento raro)
89	0,01%	99,99% (evento extremo)
100+	$\sim$0%	$\sim$100% (praticamente impossível)

A análise dos resultados evidencia que aproximadamente 19,14% das observações apresentam valores superiores a 60 ms, indicando que essa faixa ainda representa uma parcela relevante da distribuição. Entretanto, à medida que o limiar de latência é elevado, observa-se uma redução acentuada na frequência de ocorrência.

A partir do valor de 75 ms, apenas 2,43% das medições permanecem acima desse patamar, caracterizando eventos estatisticamente raros. Para latências superiores a 89 ms, a probabilidade de ocorrência torna-se praticamente residual, com apenas

0,01% dos registros acima desse limite, configurando eventos extremos do ponto de vista estatístico. Por fim, verifica-se que, para valores superiores a 100 ms, a probabilidade de ocorrência tende a zero, indicando que latências acima desse limite podem ser consideradas estatisticamente desprezíveis ou praticamente impossíveis.

Esses resultados estabelecem um limite superior estatisticamente fundamentado para a duração esperada de handshakes válidos, podendo o limiar de 100 ms ser utilizado como parâmetro para a identificação de comportamentos anômalos na rede, como aqueles característicos de ataques do tipo SYN flood.

Embora o limite determinado cientificamente já possa ser considerado um parâmetro aceitável, torna-se necessário estabelecer uma margem de tolerância adicional. Essa margem tem por objetivo contemplar eventuais atrasos introduzidos por elementos da rede, como enlaces satelitais, proxies com alto custo computacional, *firewalls* com DPI (*Deep Packet Inspection*) e outros componentes intermediários que podem impactar a latência da comunicação. A adoção dessa tolerância contribui para a redução de falsos positivos na detecção de anomalias, garantindo maior robustez ao mecanismo proposto.

Dessa forma, embora os dados empíricos indiquem que a quase totalidade das conexões legítimas se estabeleçam em até 100 ms, o limiar será estendido com base na seguinte expressão:

$$\text{Limiar final} = 100\text{ms} + \text{Margem extra}$$

Onde:

- A margem extra, definida como 900 ms, tem o objetivo de cobrir cenários de latência extrema, não representados na amostra observada, como conexões por satélite lentos, proxies lentos e *firewalls* com inspeção profunda [16]–[24].

Dessa forma, o valor final de 1000 ms é estabelecido como um limite superior absoluto, que não representa o comportamento típico das conexões legítimas, mas sim um limiar de segurança destinado a minimizar falsos positivos na detecção de anomalias em ambientes de rede com condições adversas. Esse parâmetro pode ser ajustado conforme o contexto operacional, conferindo flexibilidade ao modelo para diferentes cenários de aplicação. Ademais, é importante destacar que, em ataques do tipo SYN Flood, o pacote ACK final nunca é enviado, mantendo a conexão em estado pendente até o *timeout* do sistema, que pode ultrapassar 60 segundos. Portanto, a adoção do limiar de 1000 ms permite encurtar esse tempo de espera sem impactar negativamente os usuários legítimos.

Cabe destacar, ainda, que esta pesquisa é de fundamental importância por fornecer uma base estatística robusta. A modelagem dos dados empíricos por meio da distribuição *Weibull* possibilita a definição de parâmetros matematicamente fundamentados para a escolha desses limiares. Como evidência, constatou-se que mais de 99,9% das conexões legítimas se completam em até 100 ms, enquanto a probabilidade de uma conexão legítima exceder 1000 ms é praticamente nula. Sem o embasamento fornecido por esta pesquisa, a adoção de um limiar de 1000 ms seria apenas especulativa. Com os resultados obtidos, essa escolha passa a ser uma decisão fundamentada em evidências estatísticas, conferindo maior segurança e eficácia à identificação de comportamentos anômalos.

Além disso, a identificação da distribuição de probabilidade que melhor modela o comportamento do tempo de estabelecimento do *three-way handshake* do protocolo TCP, neste caso, a distribuição *Weibull*, representa uma contribuição significativa. Essa modelagem fornece uma base estatística que pode ser explorada pela comunidade acadêmica e técnica em diversos contextos, como o cálculo de probabilidades, simulações de tráfego de rede e o desenvolvimento de mecanismos de detecção de anomalias fundamentados em padrões de comportamento.

C. Redes WAN com destino nacional

A Tabela VII apresenta a comparação entre diferentes modelos de distribuição de probabilidade ajustados aos dados empíricos de tempo de estabelecimento do *three-way handshake* TCP. Foram utilizados como critérios de avaliação o AIC e o BIC, que permitem comparar a qualidade do ajuste penalizando a complexidade do modelo, sendo preferíveis os menores valores. O modelo *Weibull* apresentou o melhor desempenho, com os menores valores de AIC (128,35) e BIC (134,72), sendo considerado o mais adequado para representar a distribuição da duração do handshake. O modelo Gamma obteve resultados muito próximos (AIC = 128,60; BIC = 134,95), indicando ajuste semelhante ao *Weibull*. A distribuição Log-Normal apresentou valores um pouco mais altos (AIC = 129,05; BIC = 135,41), configurando-se como um ajuste aceitável. Já a Normal apresentou desempenho inferior (AIC = 130,12; BIC = 136,48), enquanto a Exponencial foi a pior candidata, com valores de AIC e BIC bastante elevados (142,80 e 145,10, respectivamente), sendo inadequada para modelar os dados.

Tabela VII
COMPARAÇÃO DOS MODELOS COM BASE EM AIC E BIC

Modelo	**AIC**	**BIC**	**Observação**
Weibull	128,35	134,72	Melhor ajuste geral
Gamma	128,60	134,95	Ajuste semelhante ao Weibull
Log-normal	129,05	135,41	Ajuste aceitável
Normal	130,12	136,48	Ajuste inferior
Exponencial	142,80	145,10	Pior ajuste

Como pode ser observado na Tabela VIII, nos resultados obtidos, a distribuição *Weibull* apresentou o melhor desempenho, com o maior $p - valor$ (0,37), indicando maior proximidade com os dados observados. As distribuições Gamma (p = 0,29) e Log-Normal (p = 0,27) também mostraram bom ajuste, ainda que ligeiramente inferiores ao da *Weibull*. A distribuição Normal apresentou resultado razoável (p = 0,15), sendo aceitável em alguns contextos. Já a Exponencial obteve $p-valor$ muito baixo (0,004), sendo considerada inadequada para representar os dados.

Tabela VIII
RESULTADOS DO TESTE DE KOLMOGOROV–SMIRNOV (KS)

Modelo	p-valor	Interpretação
Weibull	0,37	Aceita (melhor ajuste)
Gamma	0,29	Aceita
Log-normal	0,27	Aceita
Normal	0,15	Aceitável
Exponencial	0,004	Rejeitada

D. LAN com Origem e Destino Internos

Em alguns casos, os dados empíricos observados não apresentam qualquer variabilidade estatística, com todos os valores assumindo exatamente o mesmo resultado em todas as medições. Essa característica indica a presença de um fenômeno determinístico, no qual a variável de interesse não se comporta de forma aleatória, mas sim de maneira absolutamente previsível.

Matematicamente, tal comportamento é modelado por uma distribuição degenerada, também conhecida como distribuição determinística. Seja X uma variável aleatória, diz-se que ela segue uma distribuição degenerada em torno de um valor constante c se:

$$P(X - c) = 1 \tag{3}$$

Ou seja, a probabilidade de X assumir qualquer valor diferente de c é nula, e sua função de densidade de probabilidade é expressa como:

$$f(x) = \delta(x - c) \tag{4}$$

onde $\delta(x-c)$ representa a função delta de Dirac, que não é uma função no sentido tradicional, mas sim uma distribuição generalizada utilizada para descrever eventos pontuais com probabilidade total concentrada em um único valor.

Essa distribuição apresenta:

- Média igual a c;
- Variância igual a zero;
- Desvio padrão igual a zero;
- Coeficiente de variação igual a zero.

Sua utilidade está em representar sistemas totalmente previsíveis, sendo comum em cenários em que os resultados são controlados ou não estão sujeitos à aleatoriedade observável. Em contextos de redes, por exemplo, tempos de resposta que se mantêm invariavelmente constantes em todas as medições podem ser descritos por uma distribuição degenerada, evidenciando que não há ruído estatístico ou variabilidade introduzida por fatores externos.

E. SFINX: SYN FLOOD INTERCEPTOR FOR NETFILTER

A análise da tabela de rastreamento de conexões (conntrack) revelou um padrão recorrente de entradas permanecendo em estados pendentes como UNREPLIED, SYN_SENT ou SYN_RECV. Esses comportamentos resultam na saturação da tabela, comprometendo a disponibilidade do sistema e configurando um cenário típico de DoS.

Diante desse cenário, adotou-se uma abordagem de baixo nível para mitigar o consumo indevido de memória, atuando diretamente sobre o netfilter por meio de comandos do kernel. Essa estratégia permitiu aprimorar a eficiência da ferramenta SFINX na detecção e remoção de conexões maliciosas, especialmente aquelas relacionadas a ataques do tipo SYN Flood.

1) Funcionamento da SFINX: A adoção da linguagem C para o desenvolvimento da ferramenta fundamenta-se em sua capacidade de interagir de forma direta e eficiente com a biblioteca netfilter, além de facilitar a execução de chamadas ao kernel do sistema operacional. Diferentemente de linguagens de mais alto nível, que introduzem camadas adicionais de abstração e podem comprometer o desempenho ou a precisão no acesso a recursos críticos do sistema, o C possibilita maior controle sobre a gestão de memória e sobre as operações de baixo nível.

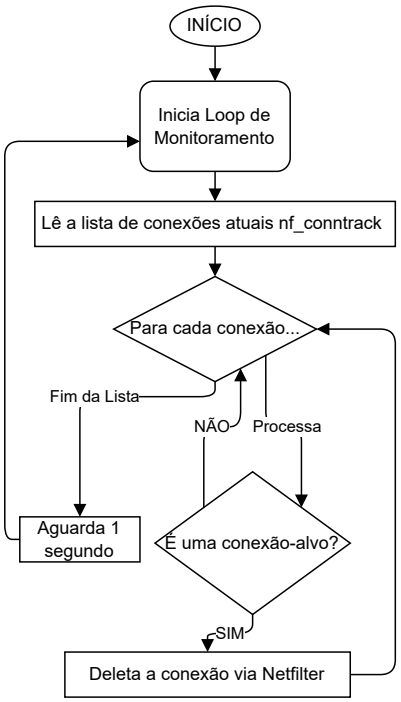


Figura 4. Fluxograma do Funcionamento

Assim, sua utilização mostra-se especialmente adequada em cenários que demandam desempenho, segurança e estabilidade, como no tratamento de ataques do tipo DoS e DDoS.

O fluxograma apresentado na Figura 4 ilustra o algoritmo de controle desenvolvido para o monitoramento e a manipulação das conexões registradas na tabela conntrack. Sua principal função consiste em inspecionar, em tempo real, a tabela de rastreamento de conexões mantida pelo subsistema netfilter e, a partir dessa análise, intervir de forma seletiva, removendo as conexões que atendam a critérios previamente definidos.

O bloco inicial marca o ponto de partida do processo, correspondendo à execução do módulo responsável por interagir com o subsistema de rede do kernel.

Em seguida, tem início o monitoramento periódico da tabela de conexões. O algoritmo é invocado a cada segundo, de forma contínua, garantindo vigilância constante sobre o estado da rede. A cada execução, o sistema realiza uma leitura completa da tabela conntrack, capturando o estado atual de todas as conexões ativas registradas pelo kernel.

Na sequência, é gerado uma captura da tabela de conexões (*snapshot*), a partir da qual o algoritmo inicia um processo iterativo. Cada entrada, isto é, cada conexão individual, é analisada de forma sequencial, assegurando que nenhuma conexão seja ignorada durante o processamento.

Durante essa análise, o algoritmo verifica o tempo de permanência de cada conexão na tabela. Caso a conexão esteja marcada com o status UNREPLIED, SYN_SENT ou SYN_RECV e permaneça na tabela por 1 segundo ou mais, ela é considerada maliciosa e elegível para remoção. Essa verificação temporal é fundamental para evitar a saturação da tabela por conexões pendentes, comum em cenários de ataques de inundação SYN.

A decisão de exclusão baseia-se, portanto, em critérios específicos, dentre os quais se destacam:

- **Status:** UNREPLIED, SYN_SENT ou SYN_RECV.
- **Tempo de permanência:** Igual ou superior a 1 segundo.

Quando uma conexão atende a essas condições, o algoritmo executa a ação principal: a remoção forçada da entrada correspondente da tabela nf_conntrack. Após a exclusão, o fluxo retorna ao ciclo de monitoramento, prosseguindo para a análise das demais conexões. Caso contrário, a conexão é mantida, sendo considerada legítima.

Esse processo cíclico garante que a tabela de rastreamento permaneça livre de entradas obsoletas ou potencialmente maliciosas, assegurando maior estabilidade e disponibilidade dos recursos do sistema frente a ataques que exploram a exaustão de estados pendentes.

Após a análise de todas as conexões presentes no snapshot, o algoritmo insere uma pausa deliberada. Essa etapa é essencial para assegurar a viabilidade do sistema em ambientes de produção, evitando sobrecarga desnecessária de recursos e permitindo que o monitoramento seja realizado de forma contínua e equilibrada.

Eficiência de Recursos: Se o *loop* fosse executado de forma contínua, sem qualquer intervalo (*tight loop*), ele consumiria 100% de um núcleo da CPU, uma vez que o processo estaria constantemente requisitando ao kernel a lista de conexões e iterando sobre ela. A introdução de uma pausa controlada (sleep(1)) evita esse problema, liberando recursos da CPU e permitindo que outros processos sejam executados normalmente.

Encerrado o período de espera, o fluxo retoma o monitoramento ao realizar uma nova leitura da tabela nf_conntrack, reiniciando todo o ciclo e assegurando, dessa forma, a vigilância contínua sobre o estado das conexões.

2) Função `VERIFICAR_ESTADO_CONEXAO`: No Algoritmo 1, é apresentado um filtro responsável por identificar, entre as linhas do arquivo de conexões, apenas aquelas que correspondem ao estado de conexão de interesse.

Algorithm 1 Função VERIFICAR_ESTADO_CONEXAO

```
Entrada:
    linha_de_texto
                ▷ Linha proveniente de /proc/net/nf_conntrack
        ▷ Estados de interesse: UNREPLIED, SYN_SENT, SYN_RECV
if linha_de_texto contém UNREPLIED ou linha_de_texto contém SYN_SENT ou linha_de_texto contém SYN_RECV then
    return Verdadeiro
else
    return Falso
end if
```

Ela possui a função de ler uma linha e retornar verdadeiro apenas se a palavra-chave UNREPLIED ou SYN_SENT ou SYN_RECV está presente. Essa etapa é importante pois é a primeira etapa de filtragem. O arquivo de conexões contém centenas ou milhares de linhas. Essa função permite que o programa ignore rapidamente as conexões saudáveis e foque apenas nas que são potencialmente maliciosas.

3) Função `EXTRAIR_INFORMACOES`: No Algoritmo 2 é apresentada essa função que, após uma linha ser identificada como possivelmente maliciosa, é utilizada para extrair os detalhes específicos dessa conexão.

Algorithm 2 Função EXTRAIR_INFORMACOES

```
Entrada:
    linha_texto, ip_origem
    porta_origem, ip_destino
    porta_destino, timeout
                ▷ Linha proveniente de /proc/net/nf_conntrack
if linha_texto contém os campos timeout, src, dst, sport, dport then
    Extrair timeout
    Extrair src → ip_origem
    Extrair dst → ip_destino
    Extrair sport → porta_origem
    Extrair dport → porta_destino
    return Sucesso
else
    return Falha
end if
```

Essa função realiza a varredura de cada linha de texto, extraindo cinco informações essenciais que identificam de forma única cada conexão: IP e porta de origem, IP e porta de destino, além do tempo de vida restante (*timeout*). Para remover uma conexão, não basta apenas detectar sua existência; é necessário informar ao kernel exatamente qual conexão deve ser eliminada. Dessa forma, essa função coleta a identidade completa da conexão, permitindo sua remoção precisa.

4) Função `DELETAR_ENTRADA_CONNTRACK`: No Algoritmo 3 é apresentado esta função que se comunica diretamente com o kernel do Linux para remover uma conexão específica.

Algorithm 3 Função DELETAR_ENTRADA_CONNTRACK

```
Entrada:
    ip_origem, porta_origem
    ip_destino, porta_destino
Criar socket AF_NETLINK
if falha na criação do socket then
    Exibir erro
    return Falha
end if
Construir mensagem Netlink (nlmsghdr)
Definir tipo da mensagem como DELETE
Definir flags de requisição e confirmação
Inserir identificadores do processo
Enviar mensagem ao kernel
if falha no envio then
    Exibir erro
    Fechar socket
    return Falha
end if
Fechar socket
return Sucesso
```

Essa função envia uma instrução formal ao kernel do Linux para remover uma entrada da tabela de rastreamento de conexões. Ela atua liberando os recursos associados às conexões com tempo maior que um segundo. A comunicação é realizada via Netlink, garantindo que as alterações no estado do *firewall* e no rastreamento de conexões do kernel sejam feitas de maneira correta.

5) Programa Principal: No Algoritmo 4 é apresentado este fluxo de controle que une todas as funções em um processo contínuo de monitoramento e limpeza.

Algorithm 4 Monitoramento e Limpeza da Tabela Conntrack

```
Declarar variáveis de conexão
Imprimir mensagem de inicialização
while verdadeiro do
    Abrir arquivo /proc/net/nf_conntrack
    if falha na abertura then
        Exibir erro
        Dormir por 1 segundo
        continue
    end if
    for all linha lida do arquivo do
        if VERIFICAR_ESTADO_CONEXAO(linha) then
            if EXTRAIR_INFORMACOES(linha) = Sucesso then
                if timeout ≥ 1 then
                    Chamar DELETAR_ENTRADA_CONNTRACK
                end if
            end if
        end if
    end for
    Fechar arquivo
    Dormir por 1 segundo
end while
```

O laço principal do algoritmo implementa um monitor contínuo, ou daemon, cujo propósito é a manutenção proativa da tabela de rastreamento de conexões do kernel Linux.

Operando em ciclo contínuo, a rotina adota sondagem com intervalo de um segundo *sleep(1)*, evitando contenção de CPU e consumo excessivo de recursos.

IV. Ambiente de Simulação

O GNS3 é uma ferramenta de simulação e emulação de redes que permite configurar e testar topologias de rede em um ambiente virtual. Ele combina dispositivos virtuais, como roteadores, switches e *firewalls*, com hosts simulados e, se necessário, com equipamentos reais, proporcionando um ambiente realista e flexível. Neste trabalho, o GNS3 foi utilizado para integrar e executar sistemas Linux reais dentro do ambiente de simulação, possibilitando a utilização do netfilter e a realização de testes relacionados à capacidade da tabela conntrack.

A. Cenários de Simulação

Esta seção apresenta os cenários desenvolvidos para avaliar o comportamento do mecanismo proposto.

A Figura 5 apresenta a topologia de rede utilizada para simular um ataque DoS. O cenário é composto por quatro elementos: Cliente, Máquina Atacada, *Firewall* (iptables) e Atacante, interligados por um Switch e conectados à Internet. O Cliente representa o usuário legítimo que acessa a Internet, simulando o tráfego normal. O Atacante gera o tráfego malicioso de negação de serviço, enviando pacotes para a máquina atacada. A máquina Iptables atua como um *firewall*, responsável por aplicar regras de filtragem e monitoramento das conexões, utilizando o módulo conntrack do netfilter.

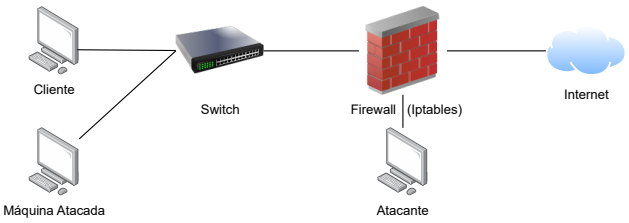


Figura 5. Topologia - DoS - GNS3

1) Cenário DoS Flood: ataque de negação de serviço com envio de pacotes SYN a uma taxa constante, sem variação na quantidade de pacotes por segundo: O primeiro cenário simulado representa um ataque de negação de serviço (DoS) baseado em inundação de pacotes SYN, com o objetivo de avaliar o comportamento do mecanismo proposto. Nesse contexto, o atacante envia um fluxo constante de pacotes SYN sem variação na quantidade de pacotes por segundo ao longo do tempo.

Para avaliar o comportamento do *firewall* e da tabela conntrack, foi empregado o utilitário hping3 [25] com a seguinte sintaxe **hping3 -V -d 120 -S -w 64 -p 445 -s 445 –flood –rate 25000 –rand-source 192.168.0.1**. Esse comando gera tráfego TCP com o flag SYN, carga de 120 bytes, janela TCP ajustada

para 64, porta de destino 445 e porta de origem definida para 445, emitindo pacotes em modo *flood* e com endereços de origem randomizados. Esse comportamento simula o envio de 25.000 pacotes por segundo.

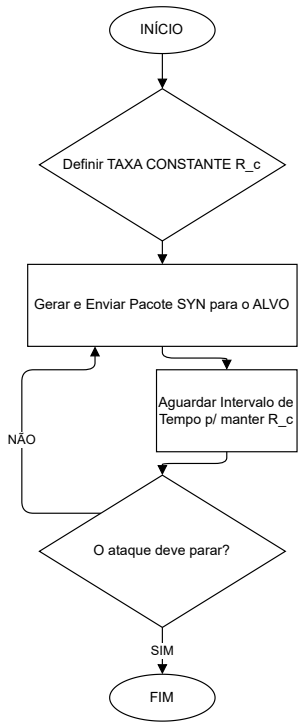


Figura 6. Fluxograma - Cenário I - DoS Flood

A Figura 6 representa o fluxo do cenário DoS Flood. O experimento inicia-se definindo a taxa fixa de envio (R_c). Em seguida, o gerador produz e envia pacotes SYN ao alvo e aguarda um intervalo calculado para manter no intervalo (R_c). O ciclo de envio e espera repete-se até que se verifique a condição de parada (tempo de ataque ou comando externo), quando o processo é encerrado.

Esse tipo de ataque explora o mecanismo de estabelecimento de conexão do protocolo TCP, no qual cada requisição SYN recebida pelo *firewall* demanda a alocação de recursos temporários na tabela de rastreamento de conexões. Como as respostas (SYN-ACK) da máquina atacada não são devidamente concluídas pelo atacante, as conexões permanecem em estado pendente UNREPLIED, SYN_SENT ou SYN_RECV, consumindo progressivamente a capacidade da tabela.

2) Cenário DoS Flood Controlado: ataque de negação de serviço com controle explícito da taxa de envio de pacotes por segundo: O segundo cenário simulado representa um ataque de negação de serviço também baseado na inundação de pacotes SYN, porém com a introdução de um controle explícito sobre a taxa de envio de pacotes por segundo. Diferentemente do cenário anterior, no qual a taxa de transmissão permanecia constante, neste caso o envio dos pacotes é ajustado de forma controlada, permitindo modular a intensidade do ataque conforme parâmetros pré-definidos.

Nesse cenário, o hping3 foi configurado com a seguinte sintaxe: **hping3 -V -d 120 -S -w 64 -p 445 -s 445 –flood –rate Variação-da-Taxa –rand-source IP-do-Alvo**. A Variação-da-Taxa foi testada em diferentes quantidades de pacotes por segundo (20000, 40000, 80000, 160000 e 320000 pacotes por segundo).

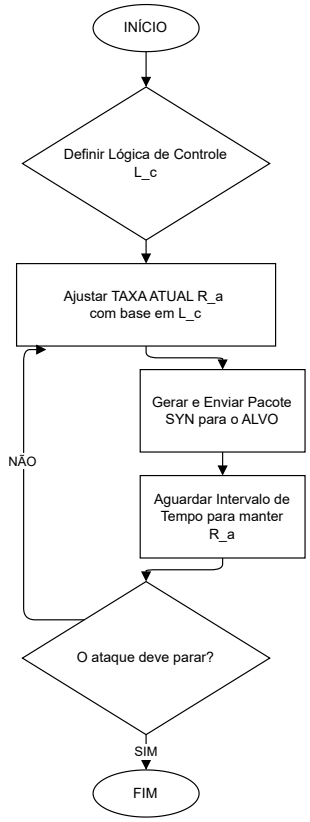


Figura 7. Fluxograma - Cenário II - DoS Flood Controlado

A Figura 7 apresenta o fluxo do cenário DoS Flood controlado. O processo inicia com a definição de uma lógica de controle (L_c), responsável por ajustar dinamicamente a taxa atual de envio (R_a) durante o ataque. Com base nessa lógica, o sistema gera e envia pacotes continuamente, com (R_a) sendo recalculada conforme (L_c), até que a condição de parada seja atingida. Esse modelo permite modular a intensidade do ataque ao longo do tempo, simulando diferentes quantidades de pacotes do segundo.

B. *Métricas*

O mecanismo foi avaliado com base nas seguintes métricas:

- **Pacotes por Segundo (PPS):** Número de pacotes recebidos pelo destino por segundo.
- **Pacotes por Minuto (PPM)**: Número de pacotes recebidos pelo destino por minuto.
- **Taxa de Sobrecarga do Processamento por Segundo (SPS):** Variação percentual no uso da CPU por segundo.
- **Taxa de Sobrecarga do Processamento por Minuto (SPM):** Variação percentual no uso da CPU por minuto.
- **Taxa da Sobrecarga de Memória RAM por Segundo (SMS):** Variação percentual no uso da memória RAM por segundo.
- **Taxa da Sobrecarga de Memória RAM por Minuto (SMM):** Variação percentual no uso da memória RAM por minuto.
- **Número de Conexões Não Estabelecidas (NCNE):** Quantidade de conexões que não foram completadas com sucesso, permanecendo no estado UNREPLIED, SYN_SENT ou SYN_RECV na tabela conntrack.
- **Taxa de Robustez (TR):** Percentual de uso da tabela conntrack em relação à sua capacidade máxima ao longo do tempo.

V. Resultados e Discussão

Nesta seção, são apresentados os resultados referentes às métricas descritas na Seção anterior, considerando os dois cenários de simulação propostos: DoS Flood e DoS Flood Controlado.

Todos os resultados apresentados nesta seção correspondem à media de 10 simulações. Para essas simulações, foi considerado um intervalo de confiança de 95%.

A. *Resultados para o cenário DoS Flood*

O cenário 1 consiste em um DoS baseado no envio de pacotes SYN a uma taxa constante, sem variação no número de pacotes por segundo. Nesse cenário observa-se a presença de um atacante que envia 25.000 pacotes SYN por segundo, um *firewall* netfilter pelo qual esses pacotes trafegam, uma máquina destinatária dos pacotes SYN e um cliente legítimo.

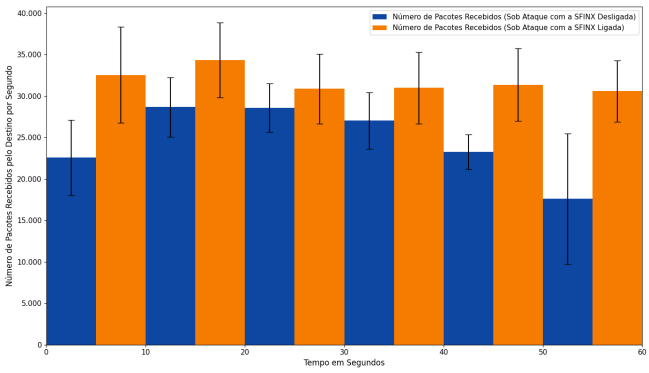


Figura 8. Pacotes por Segundo (PPS) - Destino

A Figura 8 apresenta os resultados do número de pacotes por segundo recebidos pelo destino durante 60 segundos de ataque. No cenário sem a ferramenta SFINX, observa-se uma redução progressiva dos pacotes SYN ao longo do tempo, decorrente do preenchimento da tabela *conntrack* do *firewall*, que passa a bloquear a passagem de novos pacotes. Em contraste, com a utilização da SFINX, o número de pacotes recebidos não apresenta comportamento constante, mas sim variações ao longo do tempo, reflexo do mecanismo de defesa que realiza a limpeza periódica da tabela, liberando espaço para a entrada de novos pacotes SYN.

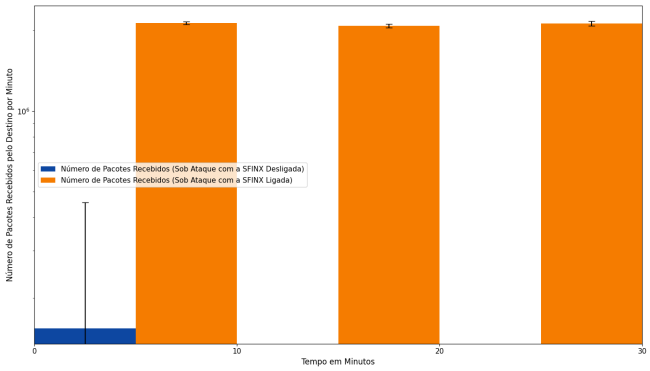


Figura 9. Pacotes por Minuto (PPM) - Destino

A Figura 9 mostra o número de pacotes por minuto recebidos pelo destino durante 30 minutos de ataque. Sem a utilização da ferramenta SFINX, o *firewall* entra em estado de DoS em aproximadamente 60 segundos (tempo apresentado na Figura 8). Esse comportamento é observado porque a tabela conntrack se enche e não consegue encaminhar novos pacotes. Com a ferramenta ativa, o destino recebe pacotes durante todo o período, evidenciando que o mecanismo implementado limpa e libera continuamente a tabela, permitindo a chegada de novos pacotes SYN, mesmo diante de um ataque volumoso, de aproximadamente 1.530.000 pacotes SYN por minuto.

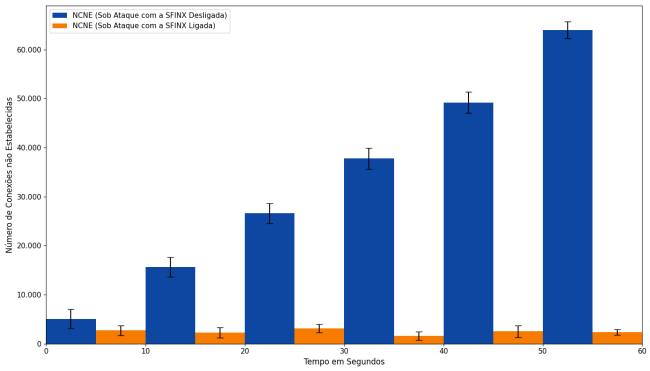


Figura 10. Número de Conexões Não Estabelecidas (NCNE)

A Figura 10 apresenta as conexões não estabelecidas armazenadas na tabela conntrack (estados SYN_SENT, SYN_RECV e UNREPLIED). Verificou-se que, em condições de ataque com a ferramenta desligada, o número de conexões aproxima-se do limite máximo de 65.000 entradas em 60 segundos. Com a SFINX ativada, esse valor mantém-se estável em torno de 500 entradas ao longo de todo o período analisado.

Dessa forma, a análise dos dados da conntrack evidencia que a ferramenta foi capaz de realizar a limpeza efetiva da tabela.

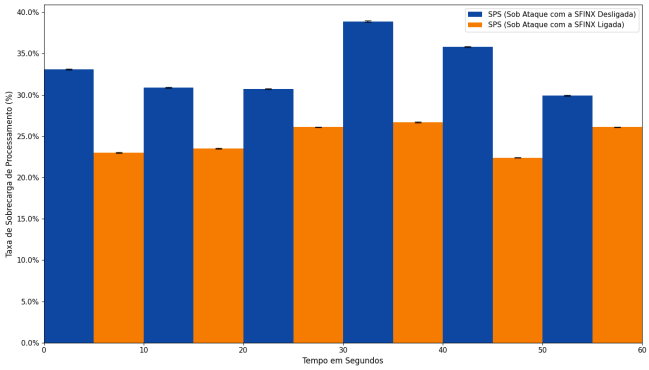


Figura 11. Taxa de Sobrecarga do Processamento (SPS) em Segundos

A Figura 11 apresenta a análise da utilização da CPU do *firewall*. Sob ataque e sem a SFINX, foram registrados picos de 38% de utilização em 60 segundos. Com a ativação do mecanismo proposto, o pico de utilização foi reduzido para 28%, o que representa uma diminuição na carga de processamento do *firewall* devido as remoções realizadas pelo mecanismo na tabela conntrack.

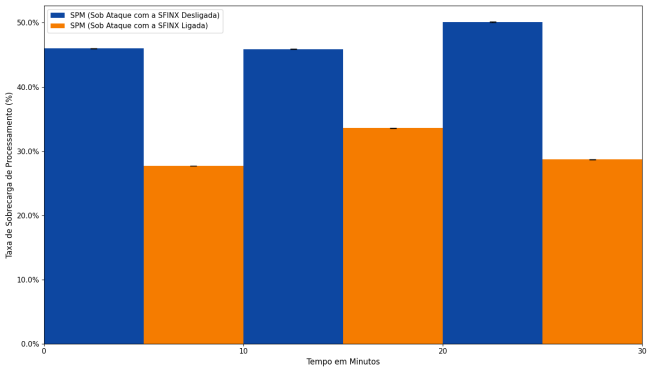


Figura 12. Taxa de Sobrecarga do Processamento (SPM) em Minutos

A Figura 12 demonstra que, ao longo dos 30 minutos, a utilização da CPU não se manteve estável, apresentando inicialmente um aumento na carga de processamento e, posteriormente, uma redução. Com a SFINX ativada, observa-se uma diminuição na carga do *firewall*, atribuída às remoções realizadas pelo mecanismo na tabela *conntrack*.

A Figura 13 revela um aumento no consumo de memória RAM quando o sistema está sob ataque, alcançando um pico de 30% com a SFINX ligada. Esse comportamento é atribuído ao comportamento padrão da SFINX, que ao detectar um ataque de SYN Flood, interage com a tabela de rastreamento de conexões do netfilter (conntrack) para remover as entradas correspondentes a conexões ilegítimas. Essa interação acarreta um aumento na necessidade de memória RAM, em comparação a SFINX desligada.

A Figura 14 revela um aumento no consumo de memória RAM quando o sistema está sob ataque. Esse comportamento é atribuído ao padrão de funcionamento da SFINX, que, ao

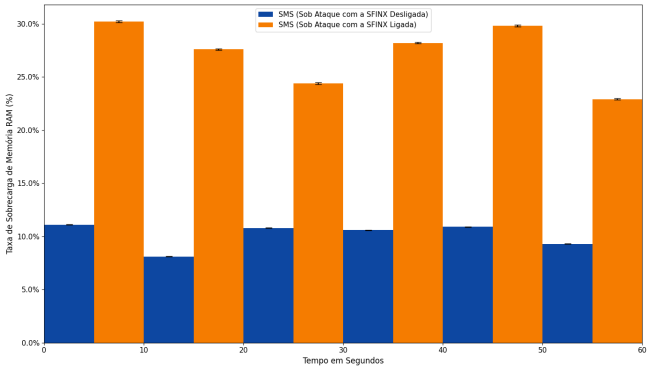


Figura 13. Taxa de Sobrecarga de Memória RAM (SMS) em Segundos

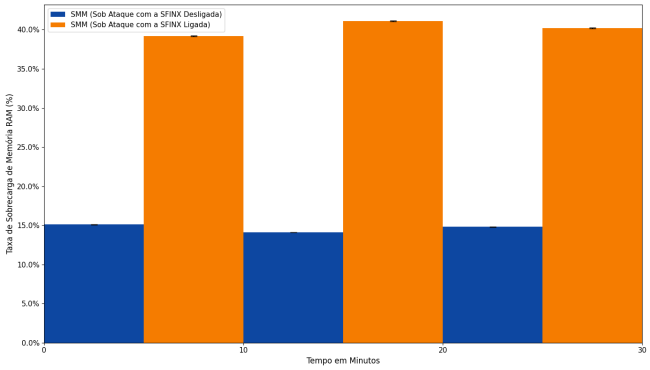


Figura 14. Taxa de Sobrecarga de Memória RAM (SMM) em Minutos

detectar um ataque de SYN Flood, interage com a tabela de rastreamento de conexões do netfilter (conntrack) para remover as entradas correspondentes a conexões ilegítimas. Essa interação acarreta um aumento na necessidade de memória RAM, em comparação com a SFINX desligada. Entretanto, esse comportamento não compromete o desempenho geral do *firewall*. Isso se deve à implementação do código, que se baseia em chamadas de sistema eficientes em nível de kernel, otimizando o uso de recursos.

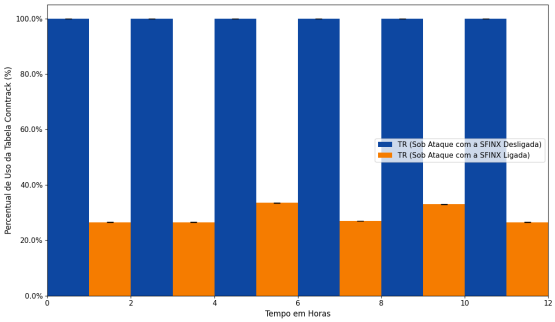


Figura 15. Taxa de Robustez (TR)

A Taxa de Robustez foi definida como a porcentagem de uso da tabela conntrack em relação à sua capacidade máxima ao longo do tempo de teste. Essa métrica permite avaliar a capacidade do *firewall* de manter operação estável durante ataques prolongados, evitando a saturação da tabela. Conforme

observado na Figura 15, o *firewall* manteve-se estável durante 12 horas de ataque, com uma taxa constante de 25.000 pacotes por segundo, totalizando mais de 1,1 bilhão de pacotes enviados. Além disso, sob ataque sem a ferramenta ativada, a tabela conntrack permaneceu saturada durante a maior parte do tempo, enquanto, com a SFINX ativada, a tabela apresentou uma porcentagem máxima de uso de aproximadamente 38%.

B. Resultados para o cenário DoS Flood Controlado

O cenário 2 consiste em um ataque de negação de serviço baseado no envio de pacotes SYN a uma taxa variável. Nesse cenário observa-se a presença de um atacante que envia 20.000 pacotes SYN por segundo inicialmente e vai dobrando até chegar em 320.000 pacotes por segundo no final das simulações, um *firewall* netfilter pelo qual esses pacotes trafegam, uma máquina destinatária dos pacotes SYN e um cliente legítimo.

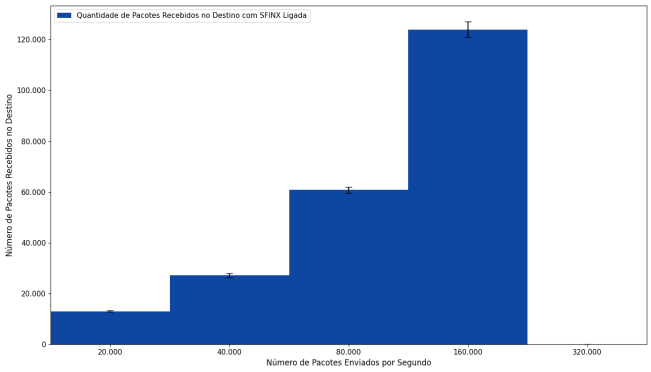


Figura 16. Quantidade de Pacotes Recebidos no Destino com SFINX Ligada

A Figura 16 apresenta a capacidade máxima do mecanismo durante um ataque de alta intensidade em um curto intervalo de tempo. Os dados indicam que, ao receber aproximadamente 320.000 pacotes no quinto segundo de simulação, após já ter recebido outros 300.000 pacotes (20.000 no primeiro segundo, 40.000 no segundo, 80.000 no terceiro e 160.000 no quarto), o sistema entra em estado de negação de serviço, mesmo com a SFINX ativada.

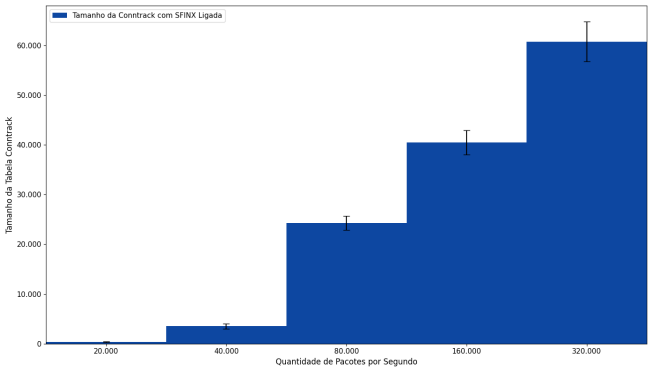


Figura 17. Tamanho da Conntrack com SFINX Ligada

A Figura 17 mostra que, ao receber 620.000 pacotes em cinco segundos, a tabela conntrack atinge seu limite de 65.536 entradas, indicando que a SFINX foi incapaz de manter a tabela desocupada nessa condição de teste.

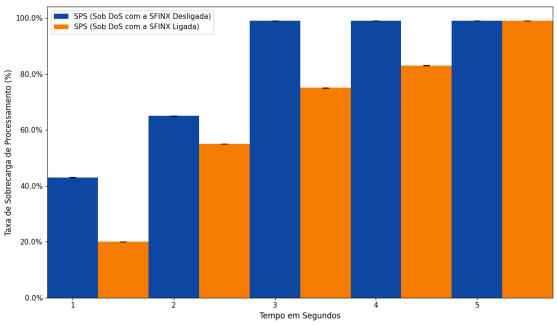


Figura 18. Taxa de Sobrecarga do Processamento (SPS) em Segundos

A Figura 18 apresenta a utilização da CPU durante cinco segundos, período em que o sistema entra em estado de negação de serviço, conforme ilustrado nas Figuras 16 e 17. Mesmo com a SFINX ativada, observou-se que o uso da CPU atingiu 100% durante o ataque de alta intensidade (620.000 pacotes em cinco segundos). Esse comportamento pode ser explicado pelo fato de que, embora a ferramenta atue removendo entradas não completadas da tabela conntrack, o processamento inicial de cada pacote SYN ainda é realizado pelo *firewall*. Assim, a SFINX reduz a saturação da tabela de conexões, mas não elimina a sobrecarga de processamento causada pelo volume extremo de requisições. Quando a CPU atinge 100%, o *firewall* não consegue mais executar todas as suas rotinas no tempo necessário, inclusive as da SFINX. Dessa forma, as tarefas de limpeza da tabela conntrack (como varredura e remoção de entradas incompletas) ficam atrasadas ou paralisadas, e a tabela volta a crescer até saturar. Portanto, constata-se que, sob essa condição, a ferramenta deixa de executar a limpeza das entradas da tabela conntrack, em razão da saturação total da capacidade de processamento do *firewall*. Cabe ressaltar, entretanto, que a saturação do processador ocorre em razão do processamento inicial de cada pacote SYN. O *firewall* precisa analisar o cabeçalho de cada pacote, verificando endereços IP, portas, regras e criando entradas na tabela conntrack, o que resulta em consumo elevado de CPU. Conforme ilustrado na Figura 11, quando há capacidade de processamento disponível, a SFINX é capaz de reduzir significativamente a carga de processamento do sistema. No entanto, a SFINX só consegue atuar, realizando a limpeza ou prevenindo o excesso de conexões, quando há CPU livre suficiente para executar suas rotinas. Em outras palavras, ela reduz a carga geral apenas enquanto o sistema ainda dispõe de recursos para operar adequadamente.

Em relação ao uso de memória, observou-se na Figura 19 que, no quinto segundo de simulação, o consumo atingiu aproximadamente 80% com a SFINX ativada e cerca de 40% sem a ferramenta. Esse aumento é atribuído à sobrecarga gerada pelas rotinas de monitoramento e limpeza implementadas pela SFINX, que demandam recursos adicionais para armazenar informações temporárias e gerenciar as conexões em tempo real.

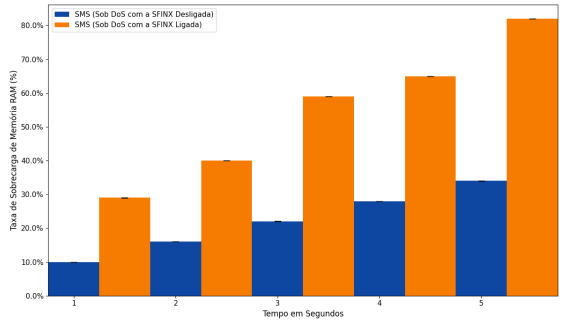


Figura 19. Taxa de Sobrecarga de Memória RAM (SMS) em Segundos

C. Estimativa de Hardware para Firewalls com Base na Largura de Banda e no Tráfego SYN

A demanda de RAM e CPU do *firewall* depende da largura de banda (bps) e do PPS processado. Com base nos dados empíricos da Tabela IX, foi possível derivar uma fórmula genérica para estimar a capacidade de hardware em diferentes tamanhos de link de dados. A Equação 5 baseia-se no comportamento observado da tabela de crescimento, levando em conta uma progressão sublinear de consumo de recursos. Uma progressão sublinear de consumo de recursos significa que, à medida que a carga de trabalho (neste caso, o número de pacotes por segundo) aumenta, o uso de recursos (como memória e CPU) não cresce de forma proporcional. Em outras palavras, o consumo de memória ou CPU cresce menos rapidamente do que o aumento da carga de pacotes.

Tabela IX
RESULTADOS DO DOS COM DOBRAMENTO DE PPS

PPS	RAM (%)	CPU (%)
10.000	20	15
20.000	29	20
40.000	40	55
80.000	59	75
160.000	65	83
320.000	82	99
640.000	95	100
1.280.000	100	100

Na Equação 5, utilizada estimar o uso de memória RAM e CPU em função do PPS é dada por:

$$\text{Uso de Recurso} = C \times \left(\frac{\text{PPS}}{\text{PPS}_{\text{ref}}} \right)^{n} \tag{5}$$

Onde:

1. Uso de Recurso: Percentual de memória RAM ou CPU consumido (em %).

2. PPS: O número de pacotes por segundo é a base para calcular os requisitos de hardware. Esse valor é determinado pela largura de banda do link, convertida para pacotes por segundo, levando em conta o tamanho médio do pacote.

A Equação 6, representa como esse cálculo é feito:

$$\text{PPS} = \frac{\text{Largura de Banda do Link (em bps)}}{\text{Tamanho do Pacote (em bits)}} \tag{6}$$

Exemplo: Para um link de 1 Gbps, com pacotes de 40 bytes (320 bits), o número de pacotes por segundo que esse link suporta é:

$$\text{PPS} = \frac{1.000.000.000 \text{ bps}}{320 \text{ bits por pacote}} = 3.125.000 \text{ pacotes por segundo} \tag{7}$$

3. C (Constante): Essa constante é determinada empiricamente a partir dos dados da tabela de dimensionamento. Para o caso de 10.000 PPS, por exemplo, a constante seria o valor de memória ou CPU para esse número de pacotes (20% de memória e 15% de CPU).

4. PPS_ref: O valor de referência de pacotes por segundo é utilizado para extrapolar os resultados. Neste estudo, é usado 10.000 PPS como referência. O PPS_ref é um valor de referência que é usado para calcular o crescimento dos recursos com base no número de pacotes. Pode ser qualquer valor da tabela que tenha dados de consumo de recursos medidos para um tráfego em específico. A escolha do valor de referência para pacotes por segundo (PPS_ref) é essencial para o dimensionamento adequado do hardware do *firewall*. Na análise empírica realizada (ver Tabela IX), observou-se o consumo relativo de memória RAM e CPU em um *firewall* com 3 GB de RAM e CPU de 2 GHz para diferentes taxas de pacotes por segundo. O consumo de recursos cresce de forma não linear conforme aumenta o número de pacotes processados. Assim, de forma resumida:

- Um PPS_ref menor resulta em estimativas de consumo de recursos mais altas para o mesmo volume de pacotes, garantindo uma margem de segurança no dimensionamento.
- Já um PPS_ref maior tende a subestimar o consumo de memória e CPU para o mesmo número de pacotes, podendo levar a um subdimensionamento do hardware.

Por isso, utilizar um PPS_ref baixo, como 10.000 pacotes por segundo, é uma escolha conservadora e realista para garantir que o *firewall* tenha capacidade suficiente para suportar o tráfego esperado, especialmente em cenários de ataques SYN Flood.

5. n (Exponente): O valor de n determina o padrão de crescimento. Com base nos dados da tabela, podemos observar que o crescimento não é linear, mas sublinear. Portanto, n é um valor entre 0 e 1. Valores menores que 1 refletem uma desaceleração do aumento de recursos à medida que os pacotes aumentam. Em termos simples, o $'n'$ controla o ritmo de crescimento dos recursos. Se o valor de $'n'$ for 0,7, por exemplo, isso significa que o consumo de recursos cresce menos rapidamente que o aumento no número de pacotes.

Neste trabalho, com o objetivo de quantificar o comportamento sublinear observado na Tabela IX, os dados foram ajustados a um modelo do tipo lei de potência. Para determinar os expoentes do modelo, aplicou-se regressão linear sobre os dados transformados por logaritmo (plano log-log), o que permitiu estimar o crescimento da memória RAM e da CPU de forma precisa. Com isso, obtiveram-se expoentes aproximados de 0,34 para a memória RAM e 0,40 para a CPU.

D. Cálculo dos Recursos de Hardware (RAM e CPU) com Base na Fórmula Proposta para Ataques de SYN Flood

A seguir, são apresentados os três passos para o cálculo dos valores de uso de memória RAM e capacidade de processamento da CPU, considerando um *firewall* operando em uma rede IPv4, na qual o tamanho total de um pacote SYN é de 54 bytes (14 bytes de Ethernet + 20 bytes de IPv4 + 20 bytes de TCP), em um link de 100 Mbps.

Passo 1: Determinando o Número de Pacotes por Segundo (PPS)

Para 100 Mbps:

- Largura de Banda: 100 Mbps = 100.000.000 bps.
- Tamanho do Pacote: 54 bytes = 432 bits.

$$\text{PPS} = \frac{100.000.000 \text{ bps}}{432 \text{ bits por pacote}} = 231.481 \text{ pacotes por segundo} \tag{8}$$

Passo 2: Usando a Fórmula de Cálculo do Uso de Recursos

Onde:

- PPS_ref = 10.000 (valor de referência da Tabela IX)
- RAM: C = 20% (valor da Tabela IX referente ao PPS_ref)
- CPU: C = 15% (valor da Tabela IX referente ao PPS_ref)
- PPS: 231.481 (calculado na Fórmula 8)
- n = 0,34 para o uso de memória RAM e 0,40 para o uso de CPU

Cálculo Uso de Memória RAM:

$$\text{Uso de Memória} = 20\% \times \left(\frac{231.481}{10.000}\right)^{0,34} \tag{9}$$

$$\text{Uso de Memória} \approx 20\% \times (23,15)^{0,34} \approx 20\% \times 2,91 \approx 58.2\% \tag{10}$$

Cálculo Uso de CPU:

$$\text{Uso de CPU} = 15\% \times \left(\frac{231.481}{10.000}\right)^{0,4} \tag{11}$$

$$\text{Uso de CPU} \approx 15\% \times 3,52 \approx 52,8\% \tag{12}$$

Passo 3: Interpretação dos Resultados

Para 100 Mbps:

- Uso de Memória é 58,2%, o que significa que o *firewall* precisaria de aproximadamente 42% a menos de memória do que o utilizado neste trabalho (esse trabalho usou 3,02 GB).
- Uso de CPU é de 52,8%, indicando aproximadamente 48% a menos de processador do que o usado neste trabalho (2 GHz).

A Tabela X apresenta os valores estimados de uso de memória RAM e CPU necessários para diferentes larguras de banda, considerando uma rede IPv4 com pacotes SYN de 54 bytes. Os cálculos adotam como referência o valor de PPS_ref = 10.000 (consumo de memória: 20% e de CPU: 15%), conforme dados empíricos apresentados na Tabela IX, obtidos a partir de um *firewall* com 3 GB de memória RAM e uma única CPU de 2 GHz.

Tabela X
ESTIMATIVAS DE CONSUMO DE RECURSOS

Link (Mb/s)	PPS Estimado	Mem. (%)	Mem. (GB)	CPU (%)	CPU (GHz)
100	231.481	58,20	1,75	52,80	1,06
200	462.963	73,60	2,21	69,60	1,39
300	694.444	82,69	2,48	81,24	1,62
400	925.926	89,07	2,67	90,11	1,80
500	1.157.407	94,06	2,82	97,46	1,95
600	1.388.889	98,14	2,94	104,05	2,08
700	1.620.370	101,59	3,05	109,93	2,20
800	1.851.852	104,51	3,14	115,07	2,30
900	2.083.333	107,07	3,21	119,78	2,40
1000	2.314.815	109,43	3,28	124,20	2,48

VI. CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho apresentou o desenvolvimento e a validação de um mecanismo para mitigação de ataques do tipo SYN Flood em *firewalls* baseados no framework netfilter. O mecanismo proposto foi estruturado em quatro partes: (i) análise estatística do tempo de estabelecimento do three-way handshake do protocolo TCP, resultando na modelagem matemática do comportamento de conexões legítimas; (ii) implementação da ferramenta SFINX (SYN Flood Interceptor for netfilter), desenvolvida em linguagem C sob a forma de um LKM; (iii) realização de simulações em ambiente controlado no GNS3; e (iv) elaboração de um modelo de dimensionamento de hardware para estimar o consumo de CPU e memória em função do volume de tráfego processado.

A análise estatística demonstrou que o tempo de estabelecimento de conexões legítimas segue uma distribuição Weibull, permitindo definir um limite superior de 100 ms para o handshake válido. Esse resultado fundamentou o limiar adotado pelo mecanismo para identificar conexões suspeitas. A ferramenta SFINX, ao interceptar pacotes SYN e aplicar esse modelo de classificação diretamente na camada de rastreamento de conexões (conntrack), mostrou-se capaz de detectar e descartar pacotes maliciosos antes que causem saturação da tabela de estados, reduzindo de forma expressiva a possibilidade de exaustão de recursos do sistema.

Os testes realizados contemplaram dois cenários distintos: DoS Flood e DoS Flood Controlado. Em todos eles, a solução demonstrou alta capacidade de contenção do ataque, mantendo a disponibilidade do *firewall* mesmo sob taxas superiores a 160 mil pacotes SYN por segundo, em ambiente com apenas 3 GB de RAM e CPU de 2 GHz. Além disso, o modelo de dimensionamento proposto possibilitou estimar, por exemplo, que para processar 1 Gbps de tráfego SYN seriam necessários cerca de 3,28 GB de RAM e 2,48 GHz de processamento total, o que reforça a aplicabilidade da solução em cenários de diferentes escalas.

As principais contribuições deste trabalho podem ser resumidas como:

- A definição de um modelo estatístico para caracterização de conexões legítimas TCP, servindo de base para a detecção de anomalias em tempo real;
- O desenvolvimento do módulo SFINX, integrado ao netfilter, com atuação direta sobre a tabela conntrack para

mitigação preventiva de SYN Floods;
- A validação experimental da ferramenta em ambiente simulado de rede, demonstrando desempenho eficiente;
- A formulação de um modelo matemático de dimensionamento de hardware, permitindo prever a escalabilidade da solução conforme o volume de tráfego e recursos disponíveis;
- A proposta contribui, assim, para o avanço da segurança de redes em plataformas abertas, reforçando a importância de soluções baseadas em software livre e de fácil integração com arquiteturas existentes. Além da mitigação de SYN Floods, o estudo estabelece um modelo metodológico que pode ser expandido para novas formas de ataques de negação de serviço.

Para trabalhos futuros, pretende-se estender o mecanismo SFINX para lidar com outros ataques baseados em estado no protocolo TCP e UDP, tais como:

- Mecanismos integrados de detecção e mitigação de ataques TCP, incluindo ACK Flood, FIN/PSH Flood e RST Flood;
- Defesas contra ataques UDP Flood em *firewalls* netfilter, utilizando abordagens probabilísticas e adaptativas.

REFERÊNCIAS

[1] J. Vedral, *Industrial and Laboratory Measuring Systems: Sensors, Distributed, Modular and Wireless Systems*, 1st ed., ser. River Publishers Series in Energy Management. River Publishers, 2024.

[2] Z. Khan, M. A. Al-Garadi, M. Al-Kahtani, and S. L. Mohammed, "A systematic review on denial of service attacks in cloud computing: research challenges and solutions," *Cluster Computing*, vol. 27, no. 2, pp. 1389–1413, 2024.

[3] M. Majkowski, "Conntrack tales - one thousand and one flows," *Cloudflare Blog*, Apr 2020. [Online]. Available: https://blog.cloudflare.com/conntrack-tales-one-thousand-and-one-flows/

[4] M. Bogdanoski, T. Suminoski, and A. Risteski, "Tcp-syn flooding attack in wireless networks," in *2012 International Conference on Innovations in Information Technology (IIT)*. IEEE, 2012, pp. 253–257.

[5] M. Aslam, A. U. Rehman, S. Tufail, and D. Kim, "A comprehensive review on detection and mitigation of TCP-SYN flooding DDoS attacks," *Electronics*, vol. 12, no. 4, p. 993, feb 2023.

[6] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "A survey on network function virtualization," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2574–2603, 2021.

[7] M. Anil Kumar, B. R. Killi, and E. Oki, "Generative adversarial networks based low-rate denial of service attack detection and mitigation in software-defined networks," *IEEE Transactions on Network and Service Management*, vol. 23, pp. 925–939, 2025.

[8] M. A. Salahuddin, V. Pourahmadi, H. A. Alameddine, M. F. Bari, and R. Boutaba, "Chronos: Ddos attack detection using time-based autoencoder," *IEEE Transactions on Network and Service Management*, vol. 19, no. 1, pp. 627–641, 2022.

[9] J. Meka, A. Jain, and N. Kumar, "A holistic approach to ddos mitigation: Leveraging ns-3 simulation and traceback for enhanced network resilience," in *2025 International Conference on Innovation in Computing and Engineering (ICE)*, 2025, pp. 1–6.

[10] A. F. Samatar, A. Hirsi, A. M. Omar, and M. J. Abdiaziz, "Investigating ddos attack mitigation strategies and simulation tools using gns3," in *2024 FORTEI-International Conference on Electrical Engineering (FORTEI-ICEE)*, 2024, pp. 142–147.

[11] M. A. Msaad, R. A. Saed, and A. M. Sllame, "A simulation based analysis study for ddos attacks on computer networks," in *2021 IEEE 1st International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering MI-STA*, 2021, pp. 756–761.

[12] A. d. S. Ilha, A. C. Lapolli, J. A. Marques, and L. P. Gaspary, "Euclid: A fully in-network, p4-based approach for real-time ddos attack detection and mitigation," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3121–3139, 2021.

[13] The Netfilter Project. (2025) The netfilter.org project. [Online]. Available: https://netfilter.org

[14] P. Neira Ayuso and F. Westphal, "netfilter: conntrack: do not renew entry stuck in tcp SYN_SENT state," Linux Kernel Mailing List, jul 2021.

[15] H. Zimmermann, "OSI Reference Model — The ISO Model of Architecture for Open Systems Interconnection," *IEEE Transactions on Communications*, vol. 28, no. 4, pp. 425–432, apr 1980.

[16] A. S. Tanenbaum, *Redes de Computadores*, 5th ed. São Paulo: Pearson, 2014.

[17] X. Chen, H. Kim, J. M. Aman, W. Chang, M. Lee, and J. Rexford, "Measuring TCP Round-Trip Time in the Data Plane," in *Proceedings of the Workshop on Secure Programmable Network Infrastructure*, 2020, pp. 35–41.

[18] S. Kumar and G. Xie, "A Measurement Study of TCP Performance over Wi-Fi and 3G Networks," in *Proceedings of the ACM SIGCOMM Workshop on Cellular Networks*, 2011, pp. 25–30.

[19] Y. Zhang, J. Li, and X. Wang, "Analysis of TCP SYN Flooding Attacks and Defense Mechanisms," *IEEE Transactions on Network and Service Management*, vol. 13, no. 4, pp. 890–901, 2016.

[20] L. Wang, Y. Chen, and Q. Liu, "TCP Behavior and RTT Dynamics in Cellular Networks," *IEEE Transactions on Mobile Computing*, vol. 16, no. 9, pp. 2600–2612, 2017.

[21] J. Qadir *et al.*, "6G–Satellite Network Integration: Challenges and Opportunities," *Technologies*, vol. 13, no. 6, p. 245, 2023.

[22] Internet Engineering Task Force (IETF), "BCP 28 – Satellite Communication Latency Guidelines," IETF, BCP 28, 2000. [Online]. Available: https://www.ietf.org/rfc/bcp/bcp28.html

[23] D. Zhang, J. Smith, and M. Lee, "Performance impact of deep packet inspection on network latency and throughput," *Journal of Network and Systems Management*, vol. 25, no. 3, pp. 555–574, 2017.

[24] X. Jin, Y. Wang, and L. Chen, "Understanding latency in proxy-based web security systems," *IEEE Transactions on Network and Service Management*, vol. 11, no. 4, pp. 456–468, 2014.

[25] S. a. Sanfilippo, "hping3: Network packet generator/analyzer," Software, Disponível em https://github.com/antirez/hping, 2011, acesso em: 10 de novembro de 2025.