FirewallTester: desenvolvimento de ferramenta para automação de testes e validação de regras de *firewalls*

Luiza Batista Basseto[1], **Luiz Arthur Feitosa dos Santos**[1]

[1]Departamento de Computação – Universidade Tecnológica Federal do Paraná (UTFPR)
Campo Mourão – PR – Brasil

luizabatista@alunos.utfpr.edu.br, luizsantos@utfpr.edu.br

Abstract. *Protecting network infrastructure is directly related to the proper definition of filtering rules, a detailed technical process that can be compromised by operational failures. Therefore, this text presents FirewallTester, a technical-pedagogical tool that applies principles of software test automation to the dynamic validation of firewall rules. The solution integrates GNS3 and Docker containers to execute test cases in high-fidelity environments, allowing for the systematic verification of security policies. Evaluation with cybersecurity students indicated high acceptability, with the SUS index obtaining a score of 77.11 points, highlighting its potential as a tool to support technical training in the field.*

Resumo. *A proteção da infraestrutura de rede está diretamente relacionada à definição adequada das regras de filtragem, processo técnico detalhado que pode ser comprometido por falhas operacionais. Assim, o presente texto apresenta o FirewallTester, uma ferramenta técnico-pedagógica que aplica princípios de automação de testes de software à validação dinâmica de regras de firewall. A solução integra GNS3 e contêineres Docker para executar casos de teste em ambientes de alta fidelidade, permitindo a verificação sistemática de políticas de segurança. A avaliação com discentes de cibersegurança indicou elevada aceitabilidade, com o índice SUS obtendo uma avaliação de 77,11 pontos, evidenciando seu potencial como ferramenta de apoio à formação técnica na área.*

1. Introdução

A complexidade inerente à configuração de *firewalls* é um fator crítico na segurança de redes, onde a definição de políticas de segurança eficazes exige a manipulação precisa de regras de filtragem, um desafio sofisticado que exige expertise comprovada [Voronkov et al. 2017]. Diante dessa complexidade operacional e do risco associado a configurações inadequadas, torna-se essencial que a formação de futuros profissionais seja estruturada de modo a capacitá-los para organizar, sistematizar e validar o processo de elaboração de políticas de *firewall* [Lorusso et al. 2025]. Trata-se de um domínio fundamentalmente desafiador, que demanda não apenas domínio conceitual, mas também habilidades práticas de análise e verificação, pois é altamente suscetível a falhas de lógica [Wool 2010]. Nesse sentido, a recorrência destes problemas, não se trata apenas de má formação teórica dos profissionais, mas também a limitação de ferramentas disponíveis para auxiliar na validação de regras de *firewalls* implementadas e em seu funcionamento [Liu 2012].

Considerando este cenário, observa-se uma lacuna significativa entre as ferramentas amplamente utilizadas no ensino de redes e as demandas reais de formação prática em cibersegurança, especialmente no que se refere à configuração e validação de *firewalls*. Ferramentas atuais como o Cisco Packet Tracer e o *Graphical Network Simulator-3* (GNS3), permitem a construção de cenários realistas, porém não oferecem mecanismos nativos para a definição e execução automatizada de testes funcionais sobre políticas de *firewall* [Allison 2022, Gomez et al. 2023]. Como consequência, a verificação do comportamento do tráfego depende fortemente da experiência do usuário, tornando a validação prévia um ponto crítico de falha humana [Louro et al. 2024].

Diante desse contexto, este artigo apresenta o FirewallTester [Santos and Basseto 2025], uma ferramenta de código aberto que propõe uma nova abordagem para a prática e ensino de administração de *firewalls* ao transpor paradigmas de testes de software para o domínio da segurança de redes. Incorporando também ao seu escopo, o realismo de simulação do GNS3 e a flexibilidade de contêineres Docker, a solução busca viabilizar a validação automática, contínua e dinâmica de regras. Além do aspecto técnico, a ferramenta também facilita o processo de configuração, reduzindo a complexidade inicial para usuários iniciantes ao fornecer *feedback* imediato e automatizado, contribuindo para potencializar a assimilação dos conceitos envolvidos.

2. Arquitetura do FirewallTester

O FirewallTester foi desenvolvido como uma solução de automação para a validação dinâmica de políticas de segurança, assegurando a conformidade entre a implementação das regras de *firewall* e o planejamento lógico inicial. Para viabilizar a execução de testes em cenários de alta fidelidade técnica, a ferramenta opera integrada a um ecossistema de virtualização e emulação (ver Figura 1), composto pelos seguintes elementos:

- GNS3: atua como motor de orquestração da topologia de rede, simplificando a criação de redes para iniciantes e permitindo a replicação de infraestruturas empresariais reais de forma segura. Embora o FirewallTester possa ser adaptado para operar em redes externas ao GNS3, inclusive em infraestruturas físicas, o simulador é um facilitador estratégico por permitir testes complexos com maior facilidade do que em ambientes de produção.
- Docker: fornece suporte de virtualização leve, em nível de Sistema Operacional. O FirewallTester possui dependência direta desta tecnologia para operar, utilizando APIs do Docker para a execução de comandos nos *hosts* do cenário; adicionalmente, a realização dos testes baseia-se em ferramentas contidas em uma imagem Docker específica, que será detalhada posteriormente.
- FirewallTester: centraliza a lógica operacional do sistema em duas funções principais: (i) automatizar o ciclo de vida dos testes de conectividade para assegurar que a política de segurança esteja operando conforme o planejado; (ii) abstrair a complexidade de gerência da infraestrutura, unificando em uma única interface gráfica o controle de estados dos *hosts* e a inserção dinâmica de regras nos *firewalls*.

Em síntese, o GNS3 provê a infraestrutura de conectividade para os contêineres Docker, enquanto o FirewallTester atua como a camada de controle superior. Essa integração utiliza APIs do Docker para gerenciar fluxos de tráfego e regras de *firewall* na topologia, consolidando um ambiente automatizado para validação de segurança. A Figura

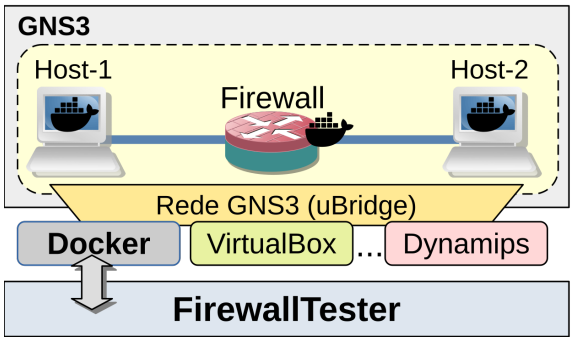

Figura 1. Ecossistema da ferramenta FirewallTester

2 detalha essa interligação e a arquitetura interna do sistema, estruturada nos seguintes módulos:

- Gerenciar Hosts: responsável por extrair dados operacionais dos ativos da rede, em sua maioria contêineres Docker, coletando IDs, nomes de *host* e endereços IP. Essas informações são persistidas em uma base de dados local (base Serviços na Figura 2) e exibidas na interface do FirewallTester para o monitoramento em tempo real dos nós; adicionalmente, o módulo controla os serviços internos de cada dispositivo, permitindo ao usuário visualizar quais portas e protocolos estão ativos para operarem como alvos nos fluxos de validação de testes.
- Gerenciar Regras de Firewall: provê um ambiente gráfico para a edição e aplicação direta de regras de *firewall* nos *hosts firewalls* do cenário de rede. Embora o foco principal seja a manipulação de regras *iptables*, a ferramenta é compatível com outras soluções baseadas no *netfilter*, como o *nftables*. O módulo evita a necessidade de interação manual via terminal, permitindo listar regras ativas, editá-las em um editor integrado e salvá-las no *host* hospedeiro para garantir a persistência das configurações.
- Testar Firewall: motor central de validação que executa testes utilizando fluxos baseados em protocolos TCP, UDP ou ICMP, definindo origem, destino, porta e a ação esperada (Liberado ou Bloqueado). O diagnóstico é apresentado via interface gráfica por meio de um sistema de cores, no qual o verde confirma o sucesso em testes de liberação de tráfego, o azul valida a eficácia no bloqueio de fluxos indesejados e o vermelho sinaliza falha no teste, ou seja, indica que o comportamento obtido diverge do planejado nos testes.
- Controle Docker: atua como camada de abstração que traduz as ações da interface em comandos operacionais executados nos contêineres. Ele é responsável por tarefas como a coleta de endereços IP, a ativação de serviços de rede e a aplicação de regras de *firewall* nos nós que atuam como tal; o módulo interpreta as saídas brutas do Docker e devolve apenas os dados estruturados e relevantes para os demais componentes do sistema.
- Servidor/Cliente: são softwares integrados às imagens Docker que compõem os contêineres do cenário, atuando como a ponte de comunicação entre o Firewall-Tester e os *hosts* do GNS3. O componente Servidor é responsável por responder a requisições ICMP ou manter portas TCP/UDP que simulam serviços de rede; em contrapartida, o Cliente inicia as tentativas de comunicação com os servidores e reporta o resultado de sucesso ou falha para processamento do FirewallTester.

Todo o tráfego gerado entre esses agentes é transmitido pela infraestrutura de rede do GNS3, conforme indicado pela linha azul tracejada na Figura 2.

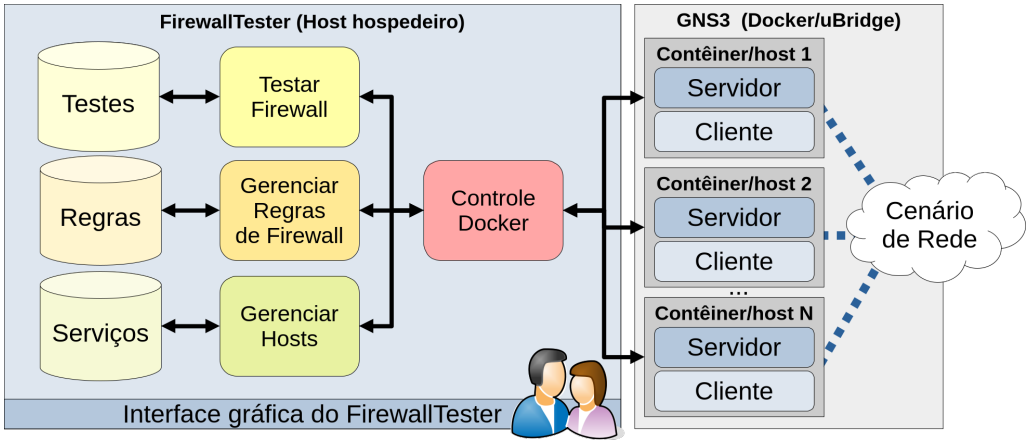

Figura 2. Arquitetura da ferramenta FirewallTester

Embora o FirewallTester utilize imagens Docker como padrão, é possível validar fluxos roteados por outros tipos de *firewalls*, como aqueles baseados em *appliances* Dynamips. A restrição atual reside na impossibilidade de testar serviços originados ou destinados ao próprio dispositivo de filtragem, uma vez que a ferramenta não controla sistemas proprietários ou externos ao ecossistema Docker. Todavia, a validação de tráfego roteado por esses nós permanece funcional, o que permite utilizar o FirewallTester para auditar a conformidade de diferentes arquiteturas de segurança.

2.1. Evolução

A ferramenta foi integralmente desenvolvida em Python, tendo seu protótipo inicial implementado por meio da biblioteca de interface gráfica Tkinter (ver Figura 3 no Anexo – Seção 5), com o objetivo de fornecer uma interface básica e responsiva que permitisse aos alunos interagir com a ferramenta fora do ambiente de terminal, tornando a utilização mais acessível para iniciantes. Após a aplicação e avaliação do protótipo em uma turma de graduação, constatou-se a necessidade de evoluir a interface gráfica para um modelo mais intuitivo, moderno e responsivo, culminando na migração para a biblioteca de interface gráfica PyQt5, que oferece maior flexibilidade estrutural e suporte à expansão de funcionalidades (ver Figura 5 na Seção 5).

Além da mudança na biblioteca de interface gráfica, houve também uma reestruturação arquitetural do projeto, com o objetivo de aumentar a modularidade e alinhar o desenvolvimento aos princípios da engenharia de software. A versão inicial adotava uma arquitetura monolítica, na qual grande parte da lógica da aplicação, da manipulação de dados e da interface gráfica encontrava-se concentrada em poucos arquivos, caracterizando um *anti-pattern* conhecido como "*God Class*" [Fowler 2018]. Tal abordagem contrariava princípios fundamentais, como o Princípio da Responsabilidade Única e a Separação de Conceitos, uma vez que um único componente acumulava responsabilidades distintas, como consequência, a manutenibilidade e a escalabilidade eram comprometidas [Pressman 2010].

Dessa forma, a refatoração promoveu uma transição de um modelo monolítico para uma arquitetura modular, com alta coesão e baixo acoplamento, sendo cada módulo responsável por um domínio específico [Pressman 2010]. Como resultado, o projeto apresentou uma evolução significativa em termos de qualidade de software, tornando-se melhor estruturado e escalável, e com uma interface mais amigável. Essa reorganização contribuiu diretamente para a manutenibilidade, testabilidade e escalabilidade da ferramenta, facilitando possíveis expansões e a incorporação de novas funcionalidades. O código-fonte da ferramenta está disponível em: `https://github.com/luizabasseto/firewallTester` para fins de reprodutibilidade e colaboração técnica [Santos and Basseto 2025].

2.2. Funcionamento da ferramenta

O FirewallTester foi avaliado em ambiente baseado em virtualização e contêineres. Para sua execução, recomenda-se a seguinte configuração:

- **Hardware:**
  - CPU: 4 núcleos (mínimo), 8 núcleos (recomendado);
  - Memória RAM: 8 GB (mínimo), 16 GB ou superior (recomendado);
  - Armazenamento: 20 GB livres (mínimo), 50 GB ou mais (recomendado, preferencialmente SSD);
  - Virtualização: suporte a VT-x/AMD-V habilitado na BIOS.
- **Software:**
  - Python 3.10 ou superior;
  - Docker (para execução dos contêineres);
  - GNS3 (incluindo dependências como *ubridge* e *libvirt*);
  - VirtualBox (opcional), utilizado para execução da máquina virtual pré-configurada da ferramenta.

2.2.1. Demonstração de funcionamento

Para ilustrar a lógica de funcionamento do FirewallTester e como utilizá-lo, apresenta-se a seguir um cenário de validação baseado na topologia da Figura 1, no qual são definidos os seguintes testes: Teste 1: Host-1 pode acessar o Host-2 na porta TCP/80 (HTTP); Teste 2: Host-1 não pode acessar o Host-2 na porta TCP/22 (SSH); e Teste 3: Host-2 pode acessar o Host-1 via SSH.

A operação inicia-se pela montagem da topologia de rede no GNS3 (ver Figura 4 do Anexo na Seção 5) e a configuração básica de rede (IP, *gateway* e DNS) nos *hosts*, sendo que os *hosts* devem utilizar a imagem Docker disponível no Docker Hub[1].

Para simplificar a integração das imagens Docker no GNS3, o GitHub do projeto[2] disponibiliza arquivos de importação de *appliances* do GNS3. Alternativamente, há uma VM pré-configurada com todo o ecossistema do FirewallTester, que inclusive já inicializa com o cenário deste exemplo. Na mesma página do GitHub do projeto, há um vídeo demonstrativo que detalha a execução dos passos realizados neste exemplo, que também são descritos a seguir.

[1] `https://hub.docker.com/r/luizarthur/cyberinfra/tags`
[2] `https://github.com/luizabasseto/firewallTester`

Vale ressaltar que, para fins didáticos, recomenda-se que o instrutor forneça os cenários de rede prontos. Essa prática permite que alunos iniciantes foquem exclusivamente no aprendizado das regras de *firewall*, eliminando a complexidade da montagem e configuração inicial da infraestrutura de rede.

Após a configuração e ativação da rede no GNS3, utiliza-se a aba *Firewall Rules* (ver Figura 5 do Anexo) do FirewallTester para a definição dos testes. Nela, especificam-se os *hosts*, o protocolo (TCP, UDP ou ICMP), a porta e o resultado esperado (Liberado ou Bloqueado). No caso do ICMP, a ferramenta permite validar a conectividade via `ping` com nós externos, possibilitando testar inclusive a comunicação direta com *hosts* da Internet. As regras adicionadas compõem uma tabela que suporta edição, exclusão e exportação, garantindo a repetibilidade das validações.

Na aba *Hosts* (ver Figura 6 no Anexo), o administrador monitora as interfaces e os IPs de cada dispositivo. É possível gerenciar as portas em execução, garantindo que os serviços alvos dos testes estejam ativos; a interface dispõe ainda de comandos para iniciar ou interromper todos os serviços da rede simultaneamente em todos os *hosts* do cenário.

A aba *Firewall Rules* (ver Figura 7) permite redigir e enviar regras ao *firewall* de forma amigável, ideal para iniciantes, sem impedir a gestão direta pelos terminais do GNS3. Nela, é possível resetar políticas pré-existentes com um clique para evitar interferências, consultar as regras ativas, além de salvar e carregar configurações para uso posterior.

Assim, ao implementar a regra `iptables -A FORWARD -p tcp --dport 22 -j DROP` na aba *Firewall Rules* e depois ao executar os testes estabelecidos na aba *Firewall Rules*, o FirewallTester exibe os resultados da Figura 8 na Seção 5, sendo que: o primeiro teste é marcado em verde (*Pass* na coluna *Result*), confirmando o acesso HTTP do Host-1 ao Host-2; o segundo teste, em azul (*Pass*), valida o bloqueio SSH pretendido entre esses *hosts*. Já o terceiro teste é exibido em vermelho (*Fail*), pois a regra genérica bloqueou o fluxo SSH do Host-2 para o Host-1 indevidamente, sinalizando uma falha de conformidade na política de segurança planejada.

Com o diagnóstico da ferramenta, o administrador identifica a inconsistência na regra e pode refiná-la, incluindo, por exemplo, o IP de destino do Host-2: `iptables -A FORWARD -d 192.168.2.2 -p tcp --dport 22 -j DROP`. Com essa alteração, ao reexecutar as validações, todos os resultados estarão em conformidade com o planejado, retornando sucesso. Desta forma, o FirewallTester otimiza esse processo ao substituir a complexidade do terminal por uma interface amigável e *feedback* visual imediato, permitindo um ciclo de teste contínuo até que a implementação atinja plena conformidade com o planejamento lógico, garantindo agilidade e precisão na correção de falhas em regras de segurança de *firewalls*.

3. Experimentos e resultados

A validação do FirewallTester foi realizada com 96 discentes da disciplina de cibersegurança da Universidade Tecnológica Federal do Paraná (UTFPR) ao longo de 2025, considerando as versões protótipo (2025/1) e revisada (2025/2) e tinha como objetivo avaliar tanto a funcionalidade da metodologia proposta quanto sua eficácia pedagógica no ensino de políticas de *firewall*. Os dados foram coletados por meio de formulário eletrônico, anonimizados conforme a Lei Geral de Proteção dos Dados (LGPD) e disponibilizados publicamente no repositório Zenodo sob o DOI: https://doi.org/10.5281/zenodo.18197977.

A avaliação do FirewallTester foi conduzida mediante um questionário fundamentado em métodos mistos, integrando indicadores quantitativos e qualitativos para mensurar a usabilidade e a eficácia pedagógica da ferramenta. Para a análise de usabilidade, foi aplicado o questionário *System Usability Scale* (SUS), que avalia a satisfação do usuário com o software [Martins et al. 2015] e para avaliação do ganho de aprendizagem, foi aplicado uma metodologia de autoavaliação baseada no modelo de auto eficácia percebida [Chou and Jones 2018], por meio da qual os estudantes avaliaram o próprio ganho de aprendizagem relacionado à compreensão e aplicação de regras de *firewall* de maneira direta e indireta.

Os resultados obtidos indicam uma percepção favorável em todos os sentidos, conforme apresentado na Tabela 1 observa-se uma evolução nos indicadores de usabilidade, com um aumento de 2,01% na média, sugerindo que os ajustes implementados na interface produziram efeitos positivos na experiência de interação com a ferramenta.

Tabela 1. Resultados de usabilidade, aprendizado e avaliação integrada

Semestre	Usabilidade		Aprendizado		Desempenho
	Média	DP	Média	DP	Global
2025/1	3,98	0,83	4,50	0,72	4,24
2025/2	4,06	0,87	4,25	0,66	4,15
Anual	4,02	0,85	4,38	0,69	4,20

DP – Desvio Padrão.
Desempenho Global – média aritmética simples entre Usabilidade e Aprendizado.

Embora tenha ocorrido pequena redução na média de aprendizado em 2025/2, os valores permaneceram elevados e com baixa dispersão (DP < 1), indicando consistência nas respostas. Destaca-se que 89,58% dos participantes não possuíam experiência prévia com *firewalls*, reforçando o potencial da ferramenta para apoiar usuários iniciantes.

No semestre de 2025/2, o FirewallTester obteve índice SUS médio de 77,11 pontos, classificando-se como um sistema de boa/excelente usabilidade segundo a literatura [Bangor et al. 2008, Martins et al. 2015], com a confiabilidade confirmada pelo alfa de Cronbach ($\alpha = 0,82$), indicando alta consistência interna. De forma complementar, os resultados do modelo TAM corroboram a percepção positiva quanto à facilidade de uso e utilidade, com relatos apontados na (Tabela 2) destacando a clareza do *feedback* visual e a organização dos testes, contribuindo para facilitar a interpretação dos testes e a identificação de inconsistências.

4. Conclusão[3]

As ferramentas tradicionais deixam uma lacuna significativa na administração de regras de *firewall*, sem obter sucesso em equilibrar fundamentação teórica consistente e prática operacional realista. Nesse cenário, o FirewallTester consolida-se como uma solução primordial para este cenário ao integrar, de forma inédita, a interatividade pedagógica com o rigor técnico da emulação, ao aplicar paradigmas de testes de software diretamente à

[3]Este trabalho contou com o suporte de ferramentas de IA exclusivamente para a revisão linguística e readequação estilística do texto.

Tabela 2. Relatos qualitativos a respeito da facilidade e utilidade da ferramenta.

Categoria (TAM)	**Relatos de Participantes**
Facilidade de Uso (PEOU)	*"[A ferramenta] apresentou de forma atraente com as cores indicando cada estado da legenda, além de ter uma tabela com a parte de esperado e resultado."*
Utilidade Percebida (PU)	*"Foi essencial, ver a origem, destino, qual protocolo, qual porta o esperado e o resultado."*
PU e *Feedback*	*"Sim, o feedback foi claro. As cores para indicar sucesso ou falha facilitaram muito a interpretação dos resultados e ajudaram a entender rapidamente se o comportamento estava correto."*

infraestrutura de rede e utilizando de tecnologias como o GNS3 e Docker para execução de cenários realistas, oferecendo uma abordagem estruturada para análise e validação de regras.

A validação realizada com estudantes demonstrou o potencial da ferramenta, pois os resultados indicam que a ferramenta apresenta comportamento consistente tanto do ponto de vista técnico quanto de usabilidade. Apresentando todos os índices acima da média (escala 1-5), com alta taxa de aceitação pelos usuários (SUS de 77,11) e métricas estáveis de desempenho e aprendizado. Esses resultados evidenciam a convergência das avaliações positivas entre os discentes de que a ferramenta é capaz de apoiar a validação sistemática de políticas de segurança, reduzindo erros operacionais e facilitando a análise de cenários de rede.

Como perspectivas futuras, prevê-se a ampliação da avaliação técnica da ferramenta, incluindo testes em maior escala e comparação com outras soluções de validação de redes. Além disso, considera-se a integração de agentes baseados em inteligência artificial para suporte à criação e interpretação de regras, potencializando tanto o uso prático quanto educacional da ferramenta.

Dessa forma, o FirewallTester busca reduzir o abismo entre a teoria de segurança e a prática operacional. Como ferramenta técnico-pedagógica em evolução, possui potencial de aplicação tanto em contextos acadêmicos quanto profissionais, preparando especialistas para desafios reais de redes. Ao apoiar processos de validação e análise de políticas de segurança, a solução fortalece a convergência entre a formação acadêmica e as demandas de mercado, garantindo maior agilidade e precisão na gestão de infraestruturas críticas. Adicionalmente, os experimentos técnicos realizados reforçam o potencial da ferramenta como solução prática para validação de políticas de *firewall*.

Referências

Allison, J. (2022). Simulation-based learning via cisco packet tracer to enhance the teaching of computer networks. In *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 1*, ITiCSE '22, page 68–74, New York, NY, USA. Association for Computing Machinery.

Bangor, A., Kortum, P. T., and Miller, J. T. (2008). An empirical evaluation of the system usability scale. In *International Conference on Usability and Internationalization*, volume 5094 of *Lecture Notes in Computer Science*, pages 63–70. Springer.

Chou, T.-S. and Jones, J. (2018). Developing and evaluating an experimental learning environment for cyber security education. In *Proceedings of the 19th Annual SIG Conference on Information Technology Education*, SIGITE '18, page 92–97, New York, NY, USA. Association for Computing Machinery.

Fowler, M. (2018). *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Professional, 2nd edition.

Gomez, J., Kfoury, E. F., Crichigno, J., and Srivastava, G. (2023). A survey on network simulators, emulators, and testbeds used for research and education. *Computer Networks*, 237:110054.

Liu, A. X. (2012). Firewall policy change-impact analysis. *ACM Transactions on Internet Technology (TOIT)*, 11(4):1–24.

Lorusso, R., Maci, A., and Coscia, A. (2025). Gollum: Guiding configuration of firewall through augmented large language models. In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence - Volume 1: ICAART*, pages 489–496. INSTICC, SciTePress.

Louro, B., Abreu, R., Cabral Costa, J., F. Sequeiros, J. a. B., and M. Inácio, P. R. (2024). Analysis of the capability and training of chat bots in the generation of rules for firewall or intrusion detection systems. In *Proceedings of the 19th International Conference on Availability, Reliability and Security*, ARES '24, New York, NY, USA. Association for Computing Machinery.

Martins, A. I., Rosa, A. F., Queirós, A., Silva, A., and Rocha, N. P. (2015). European portuguese validation of the system usability scale (sus). *Procedia Computer Science*, 67:293–300. Proceedings of the 6th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion.

Pressman, R. S. (2010). *Software Engineering: A Practitioner's Approach*. McGraw-Hill, 7th edition.

Santos, L. A. F. and Basseto, L. B. (2025). firewalltester. `https://github.com/luizabasseto/firewallTester`. GitHub repository. Acesso em 24 dez. 2025.

Voronkov, A., Iwaya, L. H., Martucci, L. A., and Lindskog, S. (2017). Systematic literature review on usability of firewall configuration. *ACM Computing Surveys (CSUR)*, 50(6):1–35.

Wool, A. (2010). Trends in firewall configuration errors: Measuring the holes in swiss cheese. *IEEE Internet Computing*, 14(4):58–65.

5. Anexos

Esta seção anexa telas do ecossistema da proposta, composto por GNS3 com *hosts* em contêineres Docker (Figura 4) e pelas interfaces do FirewallTester (demais figuras). A evolução da ferramenta é evidenciada pela Figura 3, que ilustra o protótipo original em Tkinter, em contraste com as demais imagens que detalham a versão atual já desenvolvida

utilizando PyQt5, sendo essas: a gestão de testes (Figura 5), o controle de *hosts* e serviços (Figura 6), a edição e aplicação de regras de *firewall* (Figura 7) e, por fim, a tela de gestão de testes novamente, mas com o *feedback* visual (Figura 8) dos testes realizados no exemplo deste trabalho.

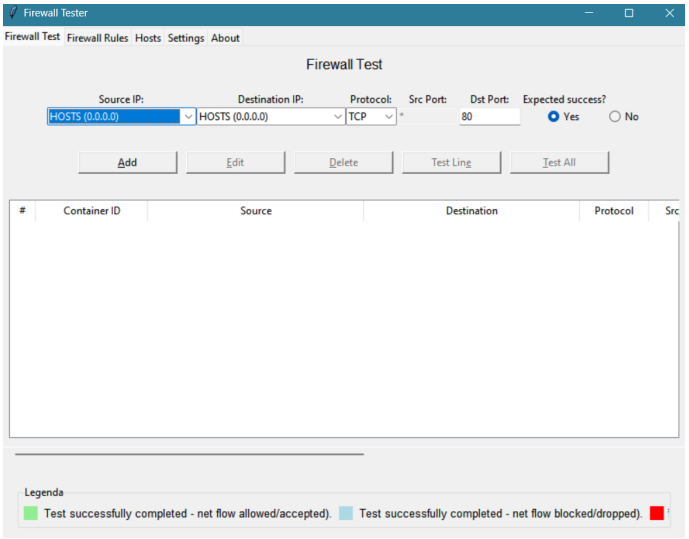

Figura 3. Interface do protótipo inicial desenvolvida em Tkinter.

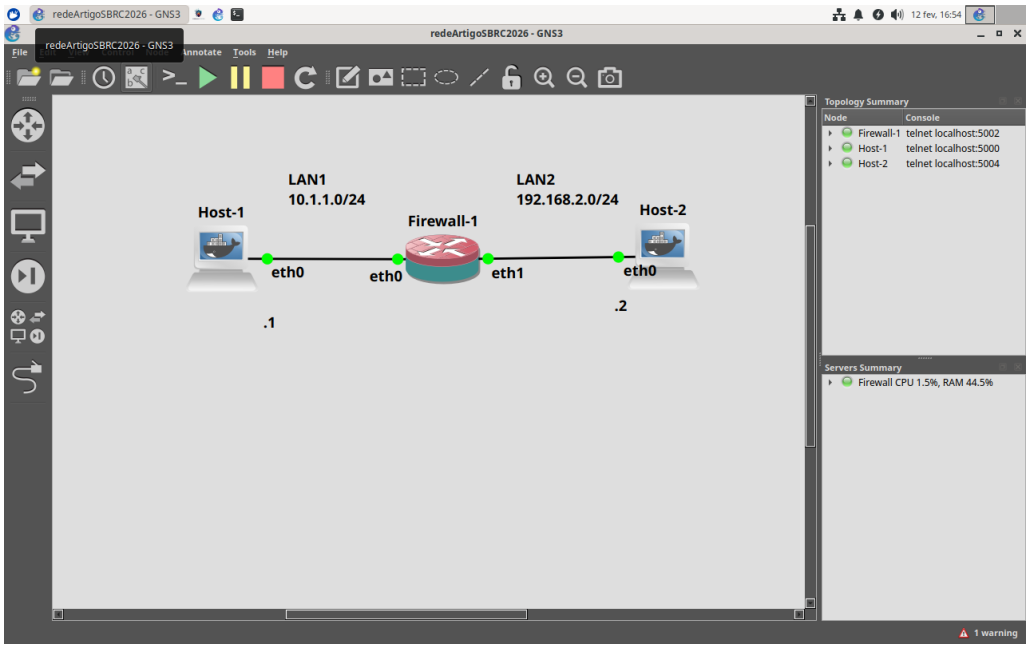

Figura 4. Topologia de rede do estudo de caso no ambiente GNS3.

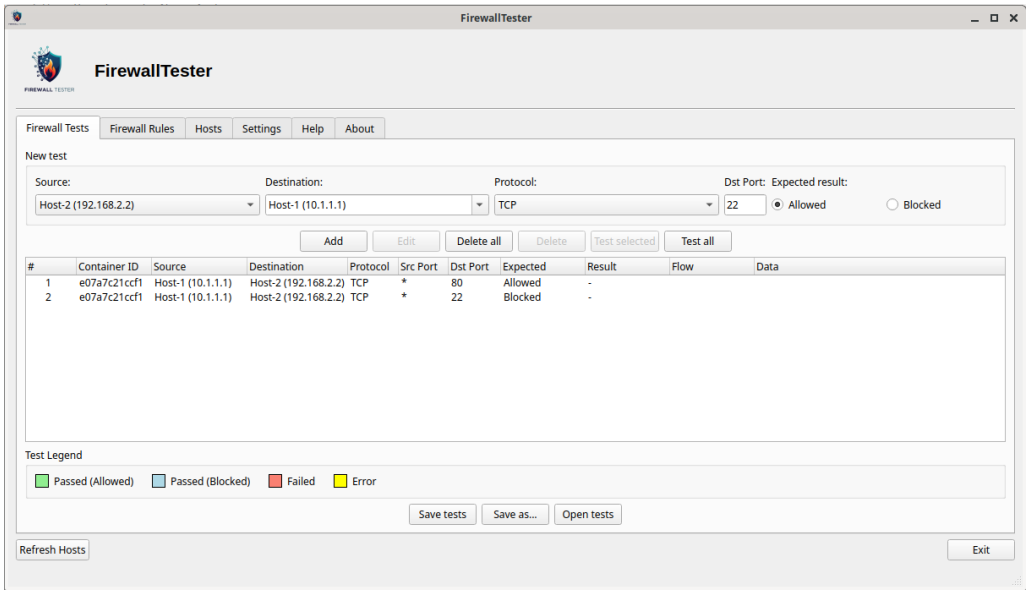

Figura 5. Aba *Firewall Tests* para gerenciamento de regras de *firewall*.

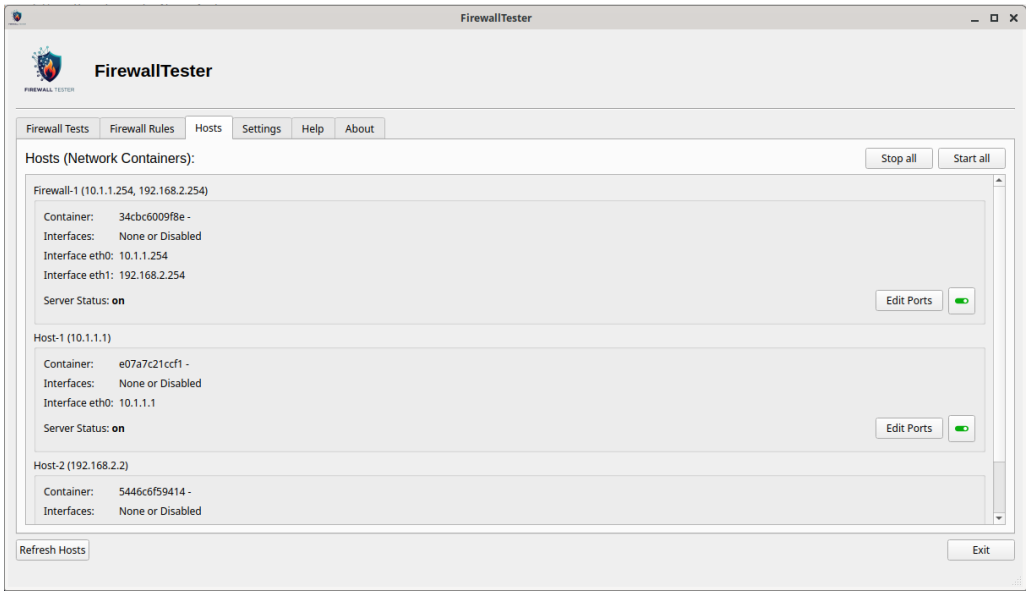

Figura 6. Aba *Hosts*, com informações de *hosts* e controle de serviços de rede ativos.

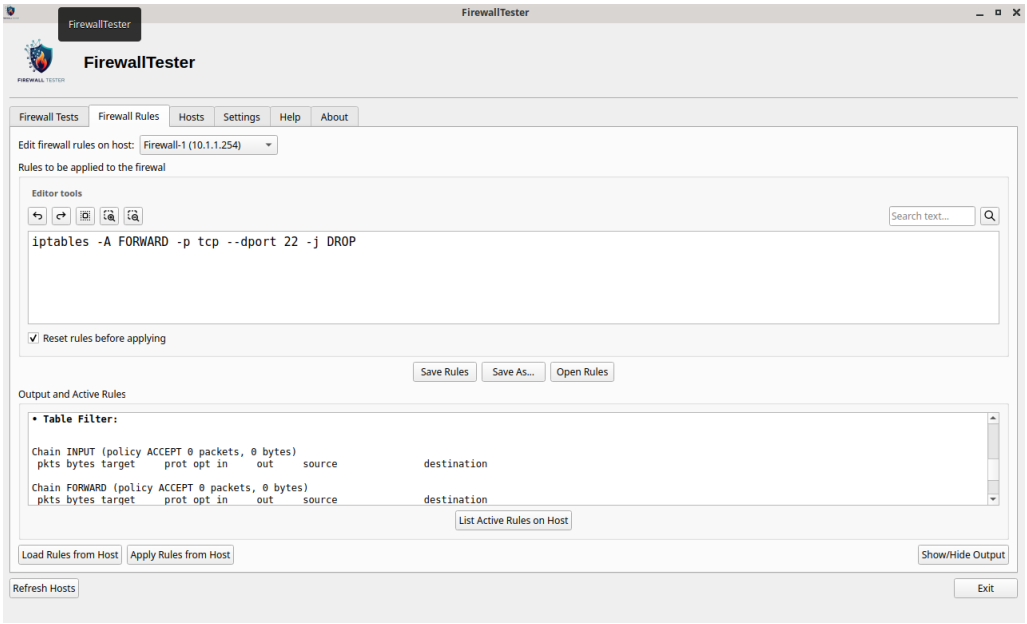

Figura 7. Aba *Firewall Rules* para edição e aplicação de regras de *firewall*.

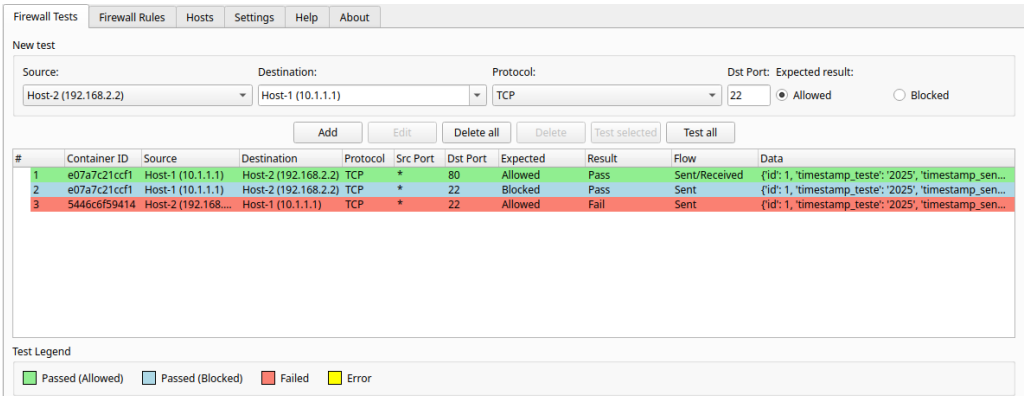

Figura 8. Resultado visual no FirewallTester dos testes do exemplo.