

NetLab Educacional: Desenvolvimento de um Software de Análise de Redes para o Ensino de Informática

Yuri Gonçalves Pavão, João Carlos de Carvalho e Silva Ribeiro¹, Michel Michelon¹

¹Instituto Federal de Educação, Ciência e Tecnologia Farroupilha
Campus Uruguaiana

yuri.04986@aluno.iffar.edu.br, joao.ribeiro@iffarroupilha.edu.br,

michel.michelon@iffarroupilha.edu.br

Resumo. *O ensino de redes exige aproximar teoria e prática. Este artigo apresenta o NetLab Educacional, uma ferramenta em Python que integra análise de tráfego, visualização de topologia, motor pedagógico e servidor web para práticas de segurança. O sistema fornece explicações em três níveis para 14 eventos de rede, com inspeção profunda restrita a HTTP/1.x em texto claro. A validação técnica, realizada em laboratório com 37 computadores, incluiu captura contínua de 30 minutos (711.603 pacotes) sem interrupção ou degradação perceptível da interface, evidenciando seu potencial como recurso educacional.*

Abstract. *Teaching computer networks requires bridging theory and practice. This paper presents NetLab Educacional, a Python-based tool integrating traffic analysis, topology visualization, a pedagogical engine, and a web server for security exercises. The system provides three levels of explanations for 14 network events, with deep inspection restricted to cleartext HTTP/1.x. Technical validation in a lab with 37 machines included a continuous 30-minute capture (711,603 packets) with no interruption or perceptible interface degradation, highlighting its potential as an educational resource.*

1. Introdução

O ensino de redes em cursos técnicos enfrenta um desafio recorrente: a distância entre a teoria de sala de aula e a prática necessária para consolidar o aprendizado. Protocolos como TCP/IP, DNS, ARP e HTTP são descritos de forma abstrata, e sua compreensão profunda costuma ocorrer apenas quando o estudante observa o comportamento real dos pacotes [Aguiar et al. 2025]. Conforme [Lins 2013], a internet tornou-se o principal contexto para o ensino de redes, tornando indispensável o uso de ferramentas que conectem teoria e prática.

A principal ferramenta disponível é o Wireshark [Wireshark Foundation 2026], amplamente usado na área profissional. Contudo, segundo a Teoria da Carga Cognitiva [Sweller 1988], ele impõe sobrecarga cognitiva extrínseca ao iniciante ao exibir informações em formato técnico de alto nível, sem contexto educativo.

Nesse contexto, este trabalho apresenta o NetLab Educacional, um software desktop de análise de redes voltado ao ambiente pedagógico de Informática. O sistema captura tráfego em tempo real, transforma os dados brutos em eventos compreensíveis e gera explicações didáticas sobre o que ocorreu, como o protocolo funciona e quais

as implicações envolvidas. A inspeção profunda de payload é restrita a HTTP/1.x em texto claro (portas 80, 8080 e 8000); HTTPS, HTTP/2 e HTTP/3 sobre QUIC não são inspecionados por serem cifrados. O código-fonte está em <https://github.com/Yurigonpav/NetLab-Educacional>.

2. Trabalhos Relacionados

As ferramentas usadas no ensino de redes e segurança dividem-se em três grupos. Analisadores de protocolos, como o Wireshark [Wireshark Foundation 2026], capturam tráfego real, mas não explicam o que o estudante observa. Simuladores e emuladores, como o Cisco Packet Tracer [Cisco Systems 2026] e o GNS3 [GNS3 Technologies 2026], permitem montar topologias, porém o primeiro usa tráfego simulado e nenhum oferece aplicações vulneráveis integradas. Aplicações web intencionalmente vulneráveis, como o DVWA [DVWA 2026] e o WebGoat [OWASP Foundation 2026], permitem praticar ataques, mas não mostram o tráfego que eles geram. A Tabela 1 resume essa comparação; o NetLab diferencia-se por reunir, em uma única ferramenta, captura real, explicação automática, topologia e servidor vulnerável com tráfego explicado em tempo real.

Tabela 1. Comparação entre o NetLab Educacional e ferramentas existentes

Ferramenta	Tráf. real	Explica	Topol.	Serv. vuln.	Cód. aberto
Wireshark	Sim	Não	Não	Não	Sim
Packet Tracer	Não (sim.)	Parcial	Sim	Não	Não
GNS3	Sim	Não	Sim	Não	Sim
DVWA	Não	Não	Não	Sim	Sim
WebGoat	Não	Não	Não	Sim	Sim
NetLab	Sim	Sim	Sim	Sim	Sim

3. Método

O desenvolvimento foi orientado pela questão: é possível integrar, em uma única plataforma de código aberto, captura de tráfego, motor pedagógico estruturado e servidor HTTP para testes, operando de forma estável em laboratório? Os objetivos foram: (a) desenvolver e descrever a arquitetura do software; (b) implementar um servidor HTTP educacional com vulnerabilidades intencionais; (c) realizar validação funcional e técnica em laboratório; e (d) discutir contribuições e limitações, delimitando a validação pedagógica para trabalhos futuros.

O desenvolvimento seguiu o modelo iterativo e incremental [Sommerville 2011]: requisitos e revisão bibliográfica; definição de tecnologias e arquitetura; implementação incremental dos módulos (captura, analisador, motor pedagógico, painéis e servidor HTTP), com teste isolado antes da integração; e validação em laboratório. A plataforma-alvo é Windows 10 e 11. As tecnologias foram Python 3.11+, PyQt6 6.7+, Scapy 2.7+, PyQtGraph 0.13+, SQLite, manuf 1.1+, psutil 6.0+ e Npcap 1.82+.

A validação funcional baseou-se em critérios por módulo: captura, operação contínua por 30 minutos sem interrupção; motor pedagógico, geração correta das explicações nos três níveis para os 14 eventos; servidor HTTP, execução das oito classes de vulnerabilidades com captura simultânea; topologia, identificação de dispositivos via ARP sweep complementada pela tabela ARP do sistema. O capturador impõe limite de proteção de 800 pacotes/s em Ethernet (400 em Wi-Fi), teto de segurança e não taxa

mínima; o comportamento nesse limite não foi avaliado. Os testes foram manuais e observacionais.

4. Arquitetura e Implementação

O NetLab foi projetado sob o princípio de que o processamento de pacotes nunca deve bloquear a interface gráfica, o que determinou uma arquitetura em três camadas com comunicação assíncrona via filas. Uma QThread dedicada à captura isola o processamento da thread principal do Qt; o buffering usa um deque limitado que descarta os elementos mais antigos, mantendo o consumo de memória estável. A Figura 1 apresenta o fluxo arquitetural completo, da captura na interface de rede até a renderização nas quatro abas.

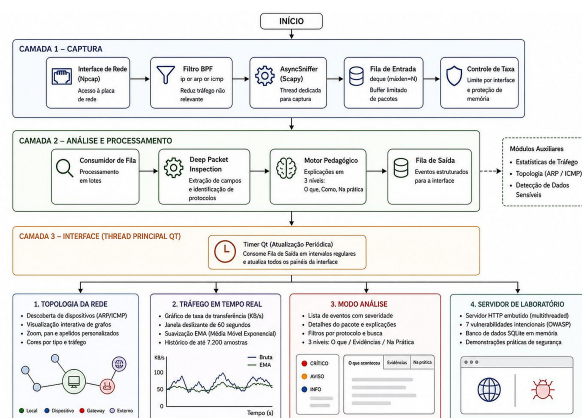


Figura 1. Fluxograma de funcionamento do sistema

O módulo de captura utiliza o AsyncSniffer da Scapy com filtro BPF para reduzir tráfego irrelevante; o modo promíscuo é habilitado em Ethernet e desabilitado em Wi-Fi por limitação dos drivers do Windows. O decodificador extrai MAC, IP, portas, flags TCP, campos ARP e payloads HTTP, organizando-os em um dicionário de evento. A descoberta ativa combina ARP sweep em múltiplas rodadas com varredura ICMP paralela, repetida periodicamente.

O motor pedagógico recebe o dicionário e retorna HTML em três níveis, gerado a partir de modelos textuais pré-definidos combinados dinamicamente com os campos extraídos do pacote, sem processamento de linguagem natural, garantindo determinismo, latência previsível e independência de conexões externas. O Nível 1 (O Que Aconteceu) contextualiza o evento em linguagem acessível; o Nível 2 (Evidências) exibe os campos técnicos brutos com destaque para campos sensíveis; o Nível 3 (Na Prática) descreve boas práticas, riscos e vetores de ataque, incluindo, para HTTP, um hexdump dos primeiros 1024 bytes. Ao todo, o motor cobre 14 eventos de rede.

A inspeção profunda ocorre em dois estágios e aplica-se apenas a HTTP/1.x em texto claro (80, 8080, 8000): identificação por porta com extração de método, URL e cabeçalhos, seguida da comparação dos campos POST a uma lista de mais de 50 nomes sensíveis e à busca de assinaturas de SQL Injection e XSS; padrões suspeitos elevam o evento à severidade crítica. HTTPS, HTTP/2 e HTTP/3 sobre QUIC (UDP/443) são apenas classificados por porta, pois o conteúdo é cifrado. O servidor HTTP embutido,

multithreaded, usa SQLite em memória e implementa oito classes de vulnerabilidades alinhadas ao OWASP Top 10 [OWASP Foundation 2021].

A interface gráfica, em PyQt6, organiza-se em quatro abas. A aba Topologia (Figura 2) implementa um visualizador de grafos em QPainter, com zoom, pan, seleção de nó e apelido por duplo clique; o tamanho do nó reflete o volume de tráfego e a cor identifica o tipo de dispositivo. A aba Tráfego em Tempo Real (Figura 3) exibe um gráfico com janela deslizante de 60 s em KB/s, buffer de até 7200 amostras, suavização por média móvel exponencial e crosshair interativo. A aba Modo Análise (Figura 4), núcleo da proposta pedagógica, combina lista de eventos e painel de detalhe com as três abas de explicação, além de filtros por protocolo, busca textual e glossário com mais de 60 termos. A quarta aba, Servidor de Laboratório, controla o servidor HTTP e exibe requisições e alertas em tempo real.

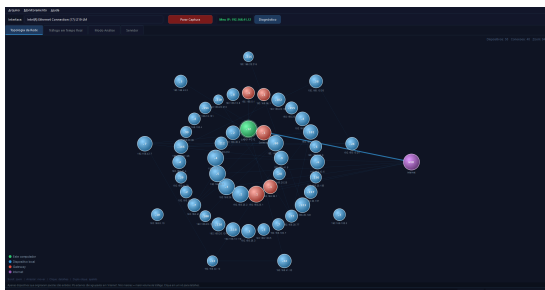


Figura 2. Aba de Topologia



Figura 3. Aba de Tráfego em Tempo Real

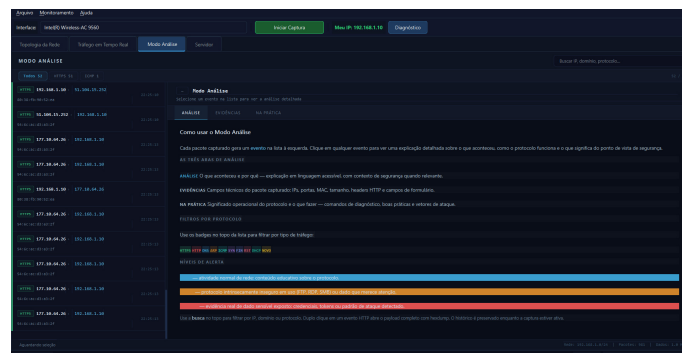


Figura 4. Aba do Modo Análise

5. Resultados de Validação

Os testes foram realizados no laboratório de informática do IFFar, Campus Uruguaiana, com 37 computadores Intel Core i5 de 14ª geração, 16 GB de RAM e rede Ethernet. O módulo de captura foi validado com sessão contínua de 30 minutos em rede ativa com múltiplos usuários, processando 711.603 pacotes e 365,54 MB (média de aproximadamente 395 pacotes/s, abaixo do limite de proteção de 800), sem descartes nem erros reportados pelos contadores da interface de rede e sem degradação perceptível da interface. O valor de 800 pacotes/s é teto de segurança do capturador, não taxa mínima exigida; o desempenho nesse limite não foi medido e é apontado como trabalho futuro.

O módulo de topologia identificou dispositivos até o limite configurado de 50 hosts simultâneos, e a complementação pela tabela ARP do sistema mostrou-se efetiva para hosts que não responderam ao ARP sweep ativo. O motor pedagógico gerou explicações nos três níveis para todos os 14 eventos suportados, com campos dinâmicos preenchidos com dados reais, verificado por inspeção manual, e sem atraso perceptível. A detecção automática de dados sensíveis em requisições POST operou conforme projetado, elevando a severidade do evento e destacando o campo comprometido.

O servidor HTTP educacional permitiu demonstrar as oito classes de vulnerabilidades; em especial, o bypass de autenticação por SQL Injection e a execução de código via XSS refletido foram reproduzidos com sucesso em navegador externo. A integração com a captura permitiu observar em tempo real o tráfego HTTP/1.x gerado, incluindo credenciais em texto puro, tornando visível ao estudante a consequência do uso de protocolos sem criptografia.

A arquitetura assíncrona em três camadas mostrou-se adequada ao contexto de laboratório, sem degradação durante a sessão de 30 minutos. O descarte automático do deque foi adotado em vez do bloqueio da fila produtora, priorizando a responsividade. O sistema apresenta, contudo, restrições: opera exclusivamente em Windows 10 e 11 com o driver Npcap; em redes comutadas sem port mirroring, a captura restringe-se ao tráfego destinado ao próprio host; HTTPS, HTTP/2 e HTTP/3 não são inspecionados; IPv6 e Wi-Fi com WPA-Enterprise não foram implementados; e descartes por fila cheia ou pelo limitador de taxa ainda não são contabilizados individualmente. Além disso, o motor pedagógico exige manutenção manual dos modelos ao incorporar novos protocolos.

Cabe delimitar o escopo: este trabalho é um estudo de desenvolvimento e validação técnica, não de eficácia pedagógica. A fundamentação em aprendizagem por descoberta guiada [Bruner 1961], currículo em espiral [Bruner 1960] e Aprendizagem Baseada em Problemas [Barrows 1986] orienta as decisões de design, sem verificação empírica neste artigo; a validação do impacto sobre a aprendizagem requer protocolo específico, com grupo de participantes, instrumento de coleta e aprovação em Comitê de Ética em Pesquisa, configurando trabalho futuro prioritário.

6. Conclusão

Este trabalho apresentou o desenvolvimento e a validação técnica do NetLab Educacional, um software desktop de análise de redes com motor pedagógico integrado, voltado a cursos técnicos em Informática, que reúne visualizador de topologia, gráfico de tráfego em tempo real, painel de análise com motor pedagógico e servidor HTTP. A validação funcional confirmou a estabilidade da interface durante captura contínua de 30 minutos, o correto funcionamento do motor pedagógico para os 14 eventos suportados e a efetividade das demonstrações de segurança com o servidor embutido. Fundamentado na aprendizagem por descoberta guiada e na Aprendizagem Baseada em Problemas, o sistema oferece condições concretas para transformar conceitos abstratos em experiências observáveis, hipótese a ser verificada em estudos futuros com usuários. Como trabalhos futuros, destacam-se: teste de carga próximo ao limite de captura, com contabilização de descartes; exportação para PCAP; suporte a IPv6; geração de relatórios em PDF; e aplicação do System Usability Scale [Brooke 1996] com estudantes.

Disponibilidade do código

O código-fonte do NetLab Educacional (versão 5.0, avaliada neste artigo) está disponível em <https://github.com/Yurigonpav/NetLab-Educacional>.

Declaração sobre o uso de IA generativa

Ferramentas de inteligência artificial generativa foram utilizadas como apoio ao desenvolvimento de partes do código-fonte e à revisão textual deste artigo. Os autores revisaram, testaram e validaram todo o material produzido e assumem total responsabilidade pelo conteúdo apresentado.

Referências

- Aguiar, R. K. B. et al. (2025). Métodos de ensino e aprendizagem em redes de computadores: uma revisão sistemática da literatura. *RENOTE: Revista Novas Tecnologias na Educação*, 23(1):161–171.
- Barrows, H. S. (1986). A taxonomy of problem-based learning methods. *Medical Education*, 20(6):481–486.
- Brooke, J. (1996). SUS: a “quick and dirty” usability scale. In Jordan, P. W., Thomas, B., Weerdmeester, B. A., and McClelland, I. L., editors, *Usability Evaluation in Industry*, pages 189–194. Taylor & Francis, London.
- Bruner, J. S. (1960). *The Process of Education*. Harvard University Press, Cambridge, MA.
- Bruner, J. S. (1961). The act of discovery. *Harvard Educational Review*, 31(1):21–32.
- Cisco Systems (2026). Cisco packet tracer. <https://www.netacad.com/cisco-packet-tracer>. Acesso em: out. 2026.
- DVWA (2026). Damn vulnerable web application (DVWA). <https://github.com/digininja/DVWA>. Acesso em: out. 2026.
- GNS3 Technologies (2026). GNS3. <https://www.gns3.com/>. Acesso em: out. 2026.
- Lins, B. F. E. (2013). A evolução da internet: uma perspectiva histórica. *Cadernos ASLEGIS*, (48):11–45.
- OWASP Foundation (2021). OWASP Top 10:2021. <https://owasp.org/Top10/>. Acesso em: out. 2026.
- OWASP Foundation (2026). OWASP WebGoat. <https://owasp.org/www-project-webgoat/>. Acesso em: out. 2026.
- Sommerville, I. (2011). *Engenharia de Software*. Pearson Prentice Hall, São Paulo, 9 edition.
- Sweller, J. (1988). Cognitive load during problem solving: effects on learning. *Cognitive Science*, 12(2):257–285.
- Wireshark Foundation (2026). Wireshark. <https://www.wireshark.org/>. Acesso em: out. 2026.