HackInSDN: Uma Arquitetura Flexível, Incremental e Portável para Experimentação em Cibersegurança

Italo V. S. Brito[1], Talita R. Pinheiro[1], Mayara R. E. Santos[1], Raquel S. M. Santos[1], Gustavo Gomes[2], Henrique Q. S. Sampaio[1], Allan E. S. Freitas[2], Leobino N. Sampaio[1]

[1] Instituto de Computação – Universidade Federal da Bahia (UFBA)
Salvador, BA – Brasil

[2]Instituto Federal da Bahia (IFBA)
Salvador, BA – Brasil

{italovalcy,talitapinheiro,mayara.rodrigues,raquelsms,henrique.scoppetta,
leobino}@ufba.br, gustavo7@gmail.com, allan@ifba.edu.br

Abstract. *Testbed environments for experimenting and teaching cybersecurity with real-world characteristics and isolated resources can be an alternative to address recent security threats and challenges. Traditional experimentation tools, datasets, and existing testbeds usually provide few cybersecurity specialized resources or unrealistic scenarios, which pose challenges for building near to production experiments with component reuse. This paper presents HackInSDN, an architecture based on programmable testbed infrastructures for teaching and experimentation in cybersecurity. HackInSDN incorporates monitoring tools, offensive security, benign traffic generation, network programmability, anomaly detection, and threat intelligence capabilities into a modular and scalable platform. A proof of concept was developed and integrated into a Kubernetes cluster. Use cases demonstrate the benefits of the proposed architecture.*

Resumo. *O uso de ambientes de ensino e experimentação em cibersegurança com características do ambiente real, porém com recursos isolados dos sistemas de produção, pode representar uma alternativa para tratar as ameaças e desafios de segurança recentes. Ferramentas de experimentação tradicionais, datasets e ambientes de testbed existentes, geralmente possuem poucos recursos de cibersegurança ou características diferentes dos cenários reais, o que impõe desafios para a construção de experimentos realistas e com reúso de componentes. Este artigo apresenta o HackInSDN, uma arquitetura baseada no uso de infraestruturas programáveis de testbeds para ensino e experimentação em cibersegurança que incorpora recursos de monitoramento, segurança ofensiva, geração de tráfego benigno, programabilidade de redes, detecção de anomalias e inteligência de ameaças em uma infraestrutura modular e escalável. Uma prova de conceito foi desenvolvida e integrada a um cluster Kubernetes, e casos de uso demonstram os benefícios da arquitetura proposta.*

1. Introdução

Em resposta à contínua evolução e complexidade das ameaças de cibersegurança, tem-se tornado imprescindível o desenvolvimento de novos mecanismos de proteção, construção de soluções inovadoras e formação consistente de profissionais. Ambientes de teste

para familiarizar-se com ameaças recentes como *ransomware*, para testar vulnerabilidades em sistemas e protocolos ou para validar recursos de defesa tipicamente são escassos, limitados em termos de escalabilidade ou complexos no que tange à preparação dos cenários de cibersegurança [Rahouti and Xiong 2019]. Ambientes para experimentação em cibersegurança precisam oferecer características que permitam ao experimentador reproduzir ameaças, fazer o monitoramento granular, avaliar técnicas de detecção e resposta a incidentes e promover a mitigação de ataques [Rahouti and Xiong 2019], tudo isso sem riscos para sistemas e dados reais [Rahouti et al. 2021, Rahouti and Xiong 2019], favorecendo a prática de *hacking* ético, e garantindo aspectos de reprodutibilidade e repetitividade científica. Além disso, é quase imperativo a disponibilização de recursos de IA que permitam reproduzir o uso de técnicas de aprendizado de máquina para potencializar a detecção de ameaças e promover a resposta automatizada a incidentes [Jonas et al. 2023].

Pesquisadores em cibersegurança atualmente fazem uso de *datasets* publicamente disponíveis ou ambientes com restrição de recursos para avaliar mecanismos de ataque e defesa através de simulação [Gemmer et al. 2023]. Uso de ambientes experimentais de larga escala poderia beneficiar estas pesquisas e fornecer cenários realísticos para investigação. Por outro lado, universidades e centros de formação tipicamente fornecem recursos limitados de *hardware* para experimentação, impondo grande desafio para estudantes e pesquisadores na realização de laboratórios avançados de segurança [Rahouti and Xiong 2019]. Já ambientes de emulação convencionais, como Mininet[1] e Containerlab[2], ou ambientes de experimentação em redes de propósito geral [Gomez et al. 2023] necessitam ser adaptados pelo experimentador para realizar testes em cenários específicos, com poucas possibilidades de reuso para novos experimentos.

Outra alternativa são os chamados *Cyber Ranges* – plataformas de simulação de ataques para capacitação em cibersegurança – que oferecem recursos específicos de segurança para experimentação, treinamentos práticos e competições em segurança [Yamin et al. 2020, Chouliaras et al. 2021]. Estes ambientes, no entanto, geralmente não fornecem recursos avançados de programabilidade de rede e perfis de tráfego que se assemelhem ao uso convencional da rede para validação dos experimentos. Ganham força, portanto, os ambientes de *testbed* especializados em cibersegurança como o MENTORED [Wangham et al. 2024] e o SPHERE [Mirkovic et al. 2024]. Ambos fornecem recursos para experimentação em cibersegurança, seja no contexto de IoT/DDoS (MENTORED) ou para demandas emergentes de privacidade e cibersegurança (SPHERE), beneficiando um subconjunto de pesquisadores e estudantes de cibersegurança.

Este trabalho apresenta a arquitetura HackInSDN, que visa oferecer recursos para experimentação em cibersegurança através de uma infraestrutura flexível, incremental e portável que beneficia um amplo conjunto de usuários. A HackInSDN é flexível devido ao baixo acoplamento dos seus módulos, o que permite a utilização de sistemas reais de diferentes fabricantes e versões. Seus módulos podem ser combinados de acordo com as necessidades do experimentador, favorecendo o crescimento incremental – incorporando novos módulos sob demanda –, e reuso de componentes para composição de laboratórios de experimentação, facilitando a produção de práticas de ensino. Os recursos da HackInSDN facilitam os experimentos em segurança a partir de tópicos que vão além dos

[1]http://mininet.org
[2]http://containerlab.dev

sistemas de detecção de intrusão, incorporando outras temáticas, tais como a detecção de novidades e anomalias, apoiados pela Inteligência Artificial (IA), contenção e filtragem de ataques dinâmicos, ferramentas de simulação de ataques e geração de tráfego benigno, base de dados para inteligência de ameaças, dentre outros. Através destes recursos, a HackInSDN possibilita a reprodução de diversos cenários relacionados à cibersegurança.

As principais contribuições deste trabalho são: (i) o projeto de uma arquitetura flexível para cenários que favorecem *hacking* ético; (ii) o desenvolvimento de um ecossistema de aplicações reais que podem ser incorporadas de acordo com a necessidade dos usuários finais; (iii) a construção de um ambiente que permite portabilidade e escalabilidade, mesclando recursos de ambientes de emulação, equipamentos reais e *cluster* de infraestrutura (e.g., Kubernetes). A viabilidade técnica da arquitetura proposta é demonstrada através da construção de um protótipo chamado Dashboard HackInSDN[3] que permite instanciar os diversos componentes da arquitetura HackInSDN, de forma conjunta ou separada, para construção de laboratórios de experimentação e ensino de cibersegurança e redes, utilizando ambientes com servidores físicos (*bare metal*) ou virtuais. Em particular, o Dashboard HackInSDN está em pleno funcionamento utilizando a infraestrutura de *cluster* nacional Kubernetes da Rede Nacional de Ensino e Pesquisa - RNP[4]. O Dashboard já tem sido utilizado por professores de algumas universidades no Brasil em fase piloto e também será utilizado para formações de temas específicos no projeto Hackers do Bem[5]. Neste artigo serão apresentados dois laboratórios disponíveis no Dashboard para demonstrar os componentes da arquitetura HackInSDN: (a) detecção de ataques de varredura, força bruta e negação de serviço; e (b) orquestração, execução e mitigação de ataques de negação de serviço distribuídos.

Este artigo está organizado da seguinte forma: a Seção 2 apresenta os trabalhos relacionados; a Seção 3 descreve a Arquitetura HackInSDN, seus módulos e modelo de dados utilizado nos casos de uso; a Seção 4 expõe as implementações necessárias no desenvolvimento dos módulos; a Seção 5 apresenta exemplos de casos de uso com a utilização da arquitetura proposta; e a Seção 6 apresenta as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

A construção de ambientes para experimentação e ensino em cibersegurança já foi alvo de investigações anteriores. Em [Rahouti and Xiong 2019] os autores apresentam metodologias para produção de roteiros de experimentação e aprendizagem, utilizando máquinas virtuais pré-instaladas e distribuídas aos alunos como ambiente de experimentação. Em [Rahouti et al. 2021] os autores revisitam o tema e propõem o uso de nuvens de infraestrutura para hospedagem das máquinas virtuais, citando ainda o uso do *testbed* GENI[6] como alternativa. Já [Vykopal et al. 2021] também faz uso de máquinas virtuais, locais ou na nuvem, porém cujas configurações são realizadas através de sistemas como *Ansible* a partir de *templates* YAML.

Uma iniciativa que merece destaque no contexto de capacitação em cibersegurança é o projeto SEED[7] [Du 2011], sigla em inglês para *SEcurity EDucation*. O

[3] https://dashboard.hackinsdn.ufba.br/
[4] https://ajuda.rnp.br/servico-de-testbeds
[5] https://hackersdobem.org.br
[6] https://www.geni.net
[7] https://seedsecuritylabs.org

projeto teve início em 2002, financiado em mais de 1,3 milhão de dólares pela Fundação Nacional de Ciência dos Estados Unidos (NSF), com o objetivo de construir laboratórios práticos para ensino de cibersegurança em diversos tópicos: laboratórios de *blockchain*, criptografia, segurança de hardware, computação móvel, segurança de redes, segurança de *software* e segurança *web*. Com mais de 40 laboratórios disponíveis e milhares de instituições usando-os nos seus cursos de segurança, o projeto continua evoluindo a partir de colaborações da comunidade e incorporando recursos de apoio, como o emulador SEED [Du et al. 2022] que consiste em uma biblioteca em Python para emular diversos elementos da Internet (i.e., sistemas autônomos, roteadores, redes, *hosts*, pontos de troca de tráfego, etc.) favorecendo a execução de laboratórios práticos de redes, BGP, DNS, *Blockchain*, *Botnet*, *Dark-net*, entre outros. Cada laboratório do projeto SEED inclui roteiros, com passo a passo, exercícios e texto de apoio, além da plataforma de execução que dá suporte aos laboratórios. Em termos da plataforma, disponibiliza-se uma máquina virtual que deve executar no computador do estudante ou em recursos da instituição usuária. O projeto não tem como objetivo oferecer uma plataforma compartilhada, especializada e escalável para execução dos laboratórios em nuvem.

Nesse sentido, dois projetos que oferecem ambientes de experimentação especializados em cibersegurança para uso compartilhado são o *testbed* MENTORED [Wangham et al. 2024] e o projeto SPHERE [Mirkovic et al. 2024]. O *testbed* MENTORED [Gemmer et al. 2023, Wangham et al. 2024] é referenciado como um *framework* escalável para a experimentação de ataques DDoS em ambientes de IoT. O *framework* usa tecnologias como *Kubernetes* e *Open vSwitch* para a criação de um *testbed*, e apresenta dois casos de uso com cenários de ataque de DDoS simples e em escala a fim de confirmar a viabilidade da proposta. Já o projeto SPHERE [Mirkovic et al. 2024] é um financiamento da NSF para criar uma infraestrutura comum de experimentação e ensino em cibersegurança e privacidade que incorpora recursos de capacidade computacional em geral, sistemas embarcados, GPU, arquitetura para emulação de sistemas de controle industrial, dispositivos de redes programáveis como placas NetFPGA e nós IoT. O projeto SPHERE foi criado em 2023 a partir das experiências do *testbed* DETERlab (sigla em inglês para Laboratório de Pesquisa Experimental em Tecnologias de Defesa), que oferece recursos de experimentação para academia, indústria e governo, viabilizando pesquisas nas áreas de análise comportamental e tecnologias de defesa para ataques de DDoS, ataques de *botnets*, criptografia, detecção de padrões de comprometimento e desenvolvimento de protocolos de armazenamento tolerantes a intrusão [Gomez et al. 2023].

Outros trabalhos nessa linha fazem parte da dimensão das plataformas de *Cyber Range* (CR), que podem ser descritas como ambientes de simulação de ataques e experimentação para capacitação em cibersegurança, oferecendo recursos específicos de segurança para treinamentos práticos, competições em segurança, exercícios de defesa cibernética e simular cenários de ataques em geral [Stamatopoulos et al. 2024, Yamin et al. 2020, Chouliaras et al. 2021]. As plataformas de CR tipicamente buscam oferecer práticas hiper-realistas, provendo atividades de cibersegurança e exercícios online com características de sistemas reais e cenários da vida real [Stamatopoulos et al. 2024]. A revisão sistemática apresentada por Stamatopoulos et al. 2024 aborda alguns aspectos arquiteturais das diversas plataformas de CR existentes, incluindo desde o público-alvo das plataformas, setor, domínio de aplicação, tecnologia utilizada, geolocalização e, finalmente, cenários de aplicação (desafios pré-configurados, níveis de dificuldade, etc).

Trabalhos anteriores [Yamin et al. 2020, Chouliaras et al. 2021] também apresentam características das plataformas de CR, porém ampliando a análise para incluir *testbeds* de cibersegurança e com foco nas ferramentas utilizadas, cenários disponíveis e recursos de experimentação e avaliação disponíveis em cada plataforma.

À luz dos trabalhos previamente mencionados, identificação dos desafios, cenários de aplicação e design das soluções, propõem-se o desenvolvimento da arquitetura HackInSDN, composta por módulos e recursos de cibersegurança que suportam a criação e execução escalável e flexível de práticas de segurança ofensiva e defensiva.

3. Arquitetura HackInSDN

A HackInSDN provê um ambiente de *software* composto por módulos funcionais para gerenciamento de cenários de experimentação (topologias, componentes de segurança como *firewalls*, IDS, IPS, WAF, entre outros, e serviços de rede mais comuns como HTTP, DNS, *E-mail*, etc.), monitoramento, detecção de anomalias, simulador de tráfego benigno e adversário, além de base de conhecimento para inteligência de ameaças. Tais funcionalidades são mediadas por um orquestrador de redes que disponibiliza informações de topologia, gestão de fluxos, estatísticas de tráfego e permite espelhamento adaptativo e contenção de ataques.

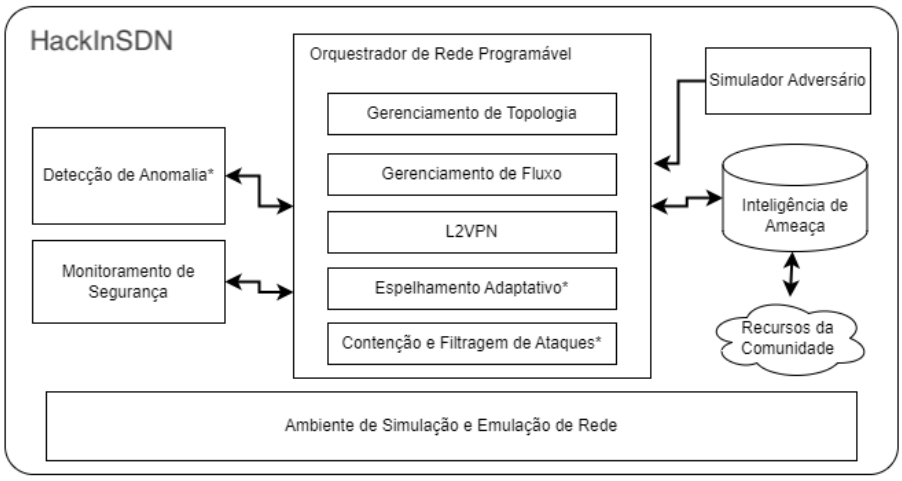


Figura 1. Módulos funcionais da arquitetura HackInSDN.

3.1. Módulos da Arquitetura

A Figura 1 ilustra os principais módulos da HackInSDN[8], detalhados a seguir:

- **Orquestrador de rede programável**: Trata-se de um elemento central da arquitetura que possui o papel de prover os serviços de gerenciamento e provisionamento da rede, e integração dos demais componentes. O orquestrador possui como principais funcionalidades: i) espelhamento adaptativo, ii) Contenção e filtragem de ataques; iii) gerenciamento de topologia e fluxos; iv) criação de VPN camada 2 (L2VPN). O espelhamento adaptativo permite a captura do tráfego dos serviços provisionados pelo orquestrador para ser usada pelo módulo de Monitoramento

[8]Componentes marcados com asterisco foram desenvolvidos para o projeto.

de Segurança. Já a Contenção e filtragem de ataques permite aplicação de regras de controle e mitigação para conter um ataque detectado pelos outros módulos.

- **Monitoramento de segurança**: Este componente tem como função a detecção de ataques a partir do casamento de padrões de tráfego e heurísticas customizadas para identificar tráfego malicioso. O casamento de padrões se dá a partir de um conjunto de regras (assinaturas) carregadas em um Sistema de Detecção de Intrusão (IDS). Já as heurísticas customizadas utilizam características contextuais do tráfego para correlação de eventos e aplicação de políticas, culminando na identificação de ataques complementar à técnica baseada em assinatura.
- **Detecção de anomalia**: Este módulo recebe dados de telemetria de rede, extrai características de interesse, e aplica algoritmos de Aprendizagem de Máquina para gerar Alertas de Novidades. Os Alertas de Novidades são analisados pelo Operador de Redes para confirmação de anomalias. Alertas de novidades previamente analisados e confirmados são automaticamente classificados como anomalias.
- **Inteligência de ameaça**: o objetivo deste módulo é fornecer uma base de dados com informações estratégias de segurança que podem ser compartilhadas com outras organizações e consumidas pelos módulos de Contenção de ataques para bloqueio preventivo de acessos maliciosos. Informações de reputação de IP, repositórios de URL maliciosas, listas de bloqueio anti-*spam*/anti-*phishing*, assinaturas de arquivo, são alguns casos de uso que são abordados na HackInSDN.
- **Simulador adversário**: Este módulo provê ferramentas de segurança ofensiva, simulação de ataques, e geração de tráfego benigno para análise da eficiência e acurácia dos demais módulos da arquitetura. As ferramentas de segurança ofensiva são organizadas de acordo com o tipo de ataque: levantamento de informações, negação de serviço, *web pentest*, ataques a senhas, dentre outros. Já a geração de tráfego benigno visa reproduzir o perfil de tráfego considerado normal ou legítimo de uma organização.
- **Ambiente de simulação e emulação de rede**: Por fim, o módulo de emulação de rede fornece um ambiente completo para experimentação, incluindo a criação e visualização da topologia do experimento, interface para execução de comandos nos elementos da topologia, simulação de serviços de rede típicos (e.g., HTTP, *E-mail*, DNS, SQL, dentre outros) e instanciação de recursos de redes e segurança como *firewalls*, WAF, IDS, *switches*, roteadores, *hosts*, etc.

A interação entre esses módulos pode acontecer de diferentes formas. Por exemplo, é possível ter um processo que se inicia com o módulo **Simulador Adversário**, simulando ataques em determinada VLAN. O tráfego é analisado para a identificação de anomalias conforme métricas previamente estabelecidas no módulo **Detecção de Anomalias**. Diante da identificação de uma anomalia, alertas de novidades são enviados ao **Orquestrador da rede**. Este, por sua vez, confirma a identificação de ataque e aplica regras de contenção, que pode envolver uma ação de bloqueio, redirecionamento de tráfego para quarentena ou ação de *rate-limit*. Por fim, a base de **Inteligência de Ameaças** registra informações do bloqueio realizado para futuras ações.

3.2. Modelo de Dados

O modelo de dados aplicado na arquitetura HackInSDN especifica os mecanismos de transporte para comunicação entre os componentes, formato dos dados, ciclo de vida das

requisições, códigos de retorno, e requisitos de escalabilidade, segurança e flexibilidade. Foram avaliados protocolos baseados em REST/JSON, gRPC, XML, Netconf/YAML, optando-se pelo uso de REST/JSON devido a flexibilidade e vasta gama de *softwares* para validação das APIs.

Foram definidas APIs para os seguintes componentes: requisições ao módulo de Espelhamento Adaptativo; requisições ao módulo de Contenção e Filtragem de Ataques; notificações de alertas do módulo de Monitoramento de Segurança; geração de alertas e requisições de inserção de IoCs do módulo de Inteligência de Ameaças; e requisições de mitigação via BGP Flowspec.

4. Implementação da arquitetura HackInSDN

Uma prova de conceito para avaliar a viabilidade da arquitetura HackInSDN foi materializada através do *software* Dashboard HackInSDN, adotando um modelo de desenvolvimento ágil e guiado por testes, com foco na compatibilidade com os ambientes de *testbed* da RNP e no reuso de códigos e sistemas existentes. As próximas subseções detalham a implementação de cada um dos módulos que envolveram esta abordagem, culminando, por fim, no Dashboard.

4.1. Orquestrador de rede programável

O Kytos-ng[9] é a plataforma de orquestração SDN que foi usada como base para o desenvolvimento do HackInSDN, pois incorpora uma solução escalável, flexível e programável. Dois módulos foram desenvolvidos para o Kytos-ng como parte da HackInSDN: espelhamento de tráfego adaptativo e contenção de ataques. O módulo de espelhamento de tráfego baseia-se na criação de fluxos de mais alta prioridade para copiar/espelhar os pacotes de interesse para uma interface de rede local ou remota. A criação dos fluxos de espelhamento pode ser customizada com filtros que permitem adaptar a granularidade do monitoramento de forma dinâmica, a partir dos alertas de segurança.

Já o módulo de contenção e filtragem de ataques opera de três maneiras: bloqueio, redirecionamento e limitação de banda. Na contenção por *bloqueio*, fluxos de alta prioridade são criados com instrução de descarte de tráfego. Na contenção por *redirecionamento*, o tráfego é desviado para uma interface de rede específica, permitindo o emprego de técnicas de quarentena de *hosts* ou estratégias de defesa por movimentação do alvo[10]. Por fim, na contenção por *limitação de banda*, aplica-se uma restrição de banda para que apenas uma porção do tráfego seja encaminhada, reduzindo assim a volumetria do ataque.

4.2. Monitoramento de Segurança

O monitoramento de segurança foi implementado através das ferramentas Zeek[11] e Suricata[12]. O Zeek é um analisador de tráfego que pode ser utilizado como monitor de segurança de rede, auxiliando na detecção e análise de tráfego malicioso. Já o Suricata é uma ferramenta de IDS, capaz de gerar alertas a partir de assinaturas, as quais são baseadas em características comumente associadas a padrões de tráfego malicioso.

[9] https://github.com/kytos-ng/
[10] https://www.dhs.gov/archive/science-and-technology/csd-mtd
[11] https://zeek.org
[12] https://suricata.io

As ferramentas Zeek e Suricata foram escolhidas por sua popularidade na comunidade de segurança e diversidade de regras disponíveis publicamente para tipos de ataque mais comuns. A arquitetura HackInSDN permite, entretanto, integração com quaisquer outras ferramentas de monitoramento de segurança que possam ser instanciadas como contêineres, por exemplo, soluções proprietárias/comerciais, soluções baseadas em P4, BPF, etc.

A configuração das assinaturas do Suricata foi baseada em regras padrões da ferramenta, bem como em um conjunto de regras *opensource* do projeto *Emerging Threats*[13] e regras desenvolvidas especificamente para o HackInSDN para ilustrar a utilização de heurísticas próprias para detecção de certos ataques, como varredura de portas e quebra de autenticação por força bruta. Já a configuração do Zeek consistiu no emprego de *scripts* para análise do comportamento da rede com foco nas consultas DNS e no cálculo da entropia dos nomes para identificação de túneis DNS ou domínios gerados aleatoriamente – tipicamente utilizados por *botnets* para controle remoto de máquinas infectadas. Em ambos os casos – Suricata e Zeek – o usuário do HackInSDN (experimentador ou aluno) tem à disposição exemplos práticos e funcionais sobre como aprimorar tais ferramentas para seus casos de uso específicos.

4.3. Detecção de Anomalias

O módulo de Detecção de Anomalias consiste no uso de técnicas de Inteligência Artificial e Aprendizagem de Máquina para detecção de ataques desconhecidos (e.g., ataques *zero-day* ou sem um padrão sistemático de tráfego), e para lidar com tráfego criptografado, aplicando heurísticas nos cabeçalhos dos pacotes para identificar anomalias [Seufert et al. 2024]. As análises são baseadas em dados de telemetria de rede por fluxo e por pacote. Para o primeiro, a telemetria é coletada no plano de controle dos equipamentos, sendo aplicadas técnicas de amostragem do tráfego para não degradar o desempenho, e processada através de ferramentas para sFlow [Wang et al. 2004]. Para o segundo grupo, foi adotada a telemetria em banda (INT, do inglês *In-band Network Telemtry* [Tan et al. 2021]), cuja coleta de dados ocorre no plano de dados, beneficiando-se da capacidade de comutação do dispositivo para coleta granular de dados.

A partir dos dados de telemetria, algumas das características são extraídas: volumetria de tráfego em pacotes por segundo, *bits* por segundo e fluxos por segundo; estatísticas de IPs de origem e destino do tráfego, portas de origem e destino TCP/UDP e protocolo, *flags* TCP; e medições de rede, tamanho dos pacotes, como intervalo entre pacotes e intervalo entre fluxos. A partir dessas informações, os modelos de aprendizagem de máquina são treinados e posteriormente utilizados para predição sobre anomalias. A execução dos modelos de aprendizagem de máquina se dá através da biblioteca *scikit-learn*[14]. Esta biblioteca possui um vasto conjunto de algoritmos de ML/IA, que permite ao experimentador/aluno avaliar diferentes análises (e.g. *K-nearest neighbor, Random Forest, SVM*, etc). Embora existam outras bibliotecas com melhor desempenho que o *scikit-learn* [Seufert et al. 2024], sua adoção foi importante na arquitetura HackInSDN por viabilizar a execução de diferentes algoritmos e facilidade no cálculo de métricas de desempenho dos modelos.

[13] `https://rules.emergingthreats.net`
[14] `https://scikit-learn.org`

4.4. Base de inteligência de ameaças

Mecanismos de inteligência de ameaça e compartilhamento de informações de segurança são essenciais para defesa e pesquisa em segurança atualmente. É muito difícil imaginar, por exemplo, a análise de *malware* de forma totalmente isolada, sem suporte de uma base de conhecimento de outros pesquisadores que possa contribuir para o entendimento do comportamento do *malware* ou caracterização de uma família de códigos maliciosos. A base de inteligência de ameaças adotada na HackInSDN foi o MISP[15]. O MISP suporta integração com diversas fontes de dados (*feeds*), algumas organizações inclusive publicam Indicadores de Comprometimento (IoCs). No HackInSDN, alguns dos padrões de ataque coletados até o momento são os do tipo *Malware*, *Phishing*, *Input Capture*, *Exploit Public-Facing Application*, e outros. Os indicadores de comprometimento coletados foram relevantes na etapa de detecção de intrusão, executada pelo módulo de Monitoramento de Segurança.

O MISP possui uma API REST que viabiliza a consulta, criação e modificação dos modelos de dados, além da integração com outras ferramentas. A biblioteca usada para acessar a plataforma MISP por meio de sua API é a PyMISP[16], a qual permite a manipulação de eventos, atributos e o gerenciamento da instância MISP. Para efetuar a importação e exportação de dados na instância MISP da HackInSDN, foi utilizada a biblioteca PyMISP, visando os benefícios do processo de automação da plataforma.

4.5. Simulador adversário

O objetivo do módulo simulador adversário é funcionar como gerador de tráfego especializado em cibersegurança, provendo meios para que o usuário possa simular diferentes tipos de ataques, utilizar ferramentas de segurança ofensiva e gerar tráfego benigno simulando perfis de tráfego real. Nessa perspectiva, foi desenvolvida a ferramenta SecFlood[17], que funciona de forma totalmente integrada com os demais componentes da HackInSDN.

Para simulação de ataques, o Secflood inclui ferramentas específicas de *pentest* de aplicações *web*, ataques ao sistema de autenticação, negação de serviço, enumeração de informações, varredura de vulnerabilidades, entre outros. Em contrapartida, a geração de tráfego benigno permite a utilização de modelos de tráfego previamente configurados, como modelo de tráfego IMIX[18] (*Internet MIX*) e EMIX[19] (*Enterprise MIX*), ou ainda o uso de *datasets* próprios, que podem ser customizados com os parâmetros de rede específicos do experimento e volumetria desejada. O perfil de tráfego IMIX consiste em pacotes de variados tamanhos (64 até 1500 bytes) e diferentes protocolos (e.g., IPv4, IPv6, TCP, UDP, protocolos de VPN). Já o perfil de tráfego EMIX contempla uma mescla de tráfego tipicamente observado em empresas e conexões domésticas de Internet, incluindo tráfego HTTP/HTTPS, vídeo *streaming*, e-mail, DNS, tráfego não classificado, etc.

4.6. Ambiente de simulação e emulação de rede

Este módulo é responsável pela preparação da topologia de rede, instanciação de serviços e adição de recursos específicos de segurança para execução em diferentes plataformas:

[15] `https://www.misp-project.org`
[16] `https://github.com/MISP/PyMISP`
[17] `https://github.com/hackinsdn/secflood`
[18] `https://en.wikipedia.org/wiki/Internet_Mix`
[19] `https://bit.ly/cisco-whitepaper-emix`

ambientes virtuais, *containers Docker* e *pods* do *Kubernetes*, além de equipamentos físicos eventualmente disponíveis no *testbed*. Nesse contexto, foi desenvolvida a ferramenta Mininet-Sec [Brito and Sampaio 2024], que possui elementos de cibersegurança para composição da topologia (Firewall, IDS, filtro *web*, etc.), oferece suporte nativo a emular diversos serviços de rede para testes de segurança (e.g., servidor *web*, servidor de *e-mail*, DNS, FTP, NTP) e ainda incorpora melhorias na usabilidade do ambiente através de uma interface *web* enriquecida para gerenciamento da topologia e execução de comandos nos nós. O Mininet-Sec permite ainda a integração da topologia emulada com recursos externos como *pods Kubernetes*, *containers Docker*, máquinas virtuais e equipamentos físicos, através de túneis VXLAN e L2TP.

4.7. Dashboard HackInSDN

O Dashboard agrega todos os componentes da arquitetura HackInSDN e oferece recursos para criação e execução de experimentos que façam uso dos módulos de forma integrada ou independente para casos de uso variados. A prova de conceito do Dashboard baseia-se no *testbed* RNP (Cluster *Kubernetes*) como ambiente de execução, podendo também ser executado em outros ambientes *Kubernetes* e facilmente adaptado para usar APIs de outros ambientes. O Dashboard HackInSDN oferece uma interface web simples e flexível que permite ao professor cadastrar laboratórios e disponibilizar guias de aprendizagem, enquanto os alunos podem executar os laboratórios em qualquer lugar e a qualquer horário, tendo como requisito apenas um navegador *web*. A Figura 2 ilustra de forma geral a arquitetura do Dashboard HackInSDN. Já a Figura 3 ilustra o Dashboard com a visualização de estatísticas do ambiente (3a) e execução de laboratórios (3b).

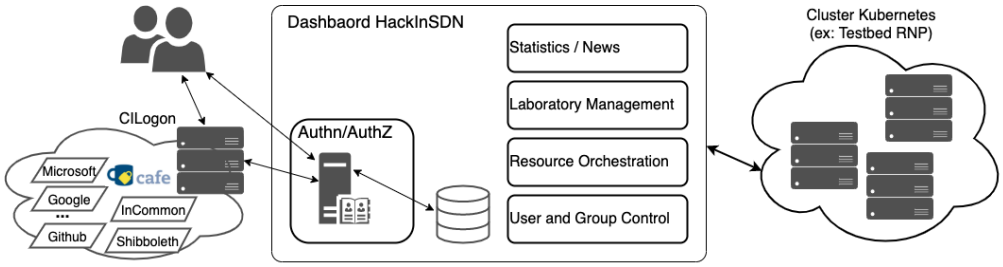


Figura 2. Arquitetura Dashboard HackInSDN.

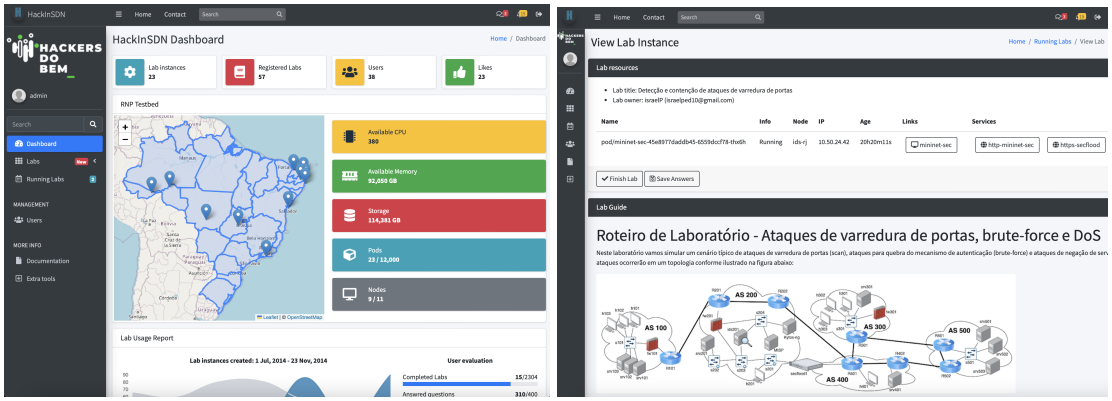
O Dashboard permite autenticação federada de usuários, através da plataforma CILogon, autorização de usuários baseada em grupos, criação e execução de laboratórios com roteiros de atividade interativos, e orquestração geral dos recursos disponíveis no *testbed*. Os roteiros de atividade interativos oferecem recursos para que o professor forneça um passo a passo para o aluno executar o laboratório, enquanto apresenta questionários diversificados ao longo do roteiro que ajudam a avaliar o progresso do aluno/experimentador. Tais questionários podem ser usados ainda em competições e desafios de segurança, derivando métricas para estratégias de *gamificação*.

5. Casos de uso

A avaliação experimental da proposta se dá através de casos de uso e experimentos realizados utilizando o ambiente desenvolvido.

5.1. Detecção de ataques de varredura, força bruta e negação de serviço

Alguns ataques que muito comumente afetam as organizações são a varredura de portas e serviços, negação de serviços e ataques de força bruta contra os mecanismos de autenticação. Foi desenvolvido um laboratório no Dashboard HackInSDN para esse cenário. A Figura 3b ilustra a visualização da topologia do laboratório, um cenário típico em *hosts* na Internet, onde participam de ataques orquestrados de varredura e força bruta contra uma organização, ao passo que dispositivos comprometidos são usados para gerar grande volume de tráfego, tendo como alvo um serviço a ser atacado. A partir da execução de um ataque de força bruta contra um servidor SSH, alertas de novidades são gerados pelo Suricata e enviados para o orquestrador da rede.



(a) Página inicial (b) Execução de Lab

Figura 3. Interface web do Dashboard HackInSDN

A partir da análise desse caso de uso é possível entender a cronologia e relação dos componentes da HackInSDN na execução, identificação e tratamento destes ataques:

1. A instanciação do cenário de teste, configuração de topologia de rede, criação de serviços de conectividade e configuração da rede constituem o primeiro passo na execução do experimento (tarefas apoiadas pelo Mininet-Sec e Kytos-ng).
2. Inicialmente o operador de segurança habilita o espelhamento de tráfego (módulo Espelhamento Adaptativo) para permitir o monitoramento de ataques.
3. O Sistema de detecção de intrusão (módulo Monitoramento de Segurança) recebe e processa o tráfego espelhado em relação a assinaturas configuradas e heurísticas customizadas de análise do perfil de tráfego (e.g., usando o Zeek), gerando como saída alerta de atividade maliciosa. Neste processo, é possível também integrar o MISP e incluir IPs maliciosos na lista de assinaturas.
4. Alertas de atividade maliciosa são convertidos em contenção especializada no tipo de ataque (módulo de Contenção e Filtragem de Ataques). Por exemplo, o operador de segurança pode habilitar a contenção de ataques de varredura e força bruta com base no **bloqueio** da conexão, ao passo que aplica controles de **limitação de banda** para ataques de negação de serviço simples.
5. Além da contenção a partir dos alertas de atividade maliciosa, as informações do ataque podem ser catalogadas (módulo de Inteligência de Ameaça) para formação de uma base de conhecimento e correlação de eventos, ou para compartilhamento de reputação de IP, DNS, hashes de arquivos, entre outras

6. Por fim, para avaliar o funcionamento do arcabouço citado acima, bem como realizar provas de conceito de ataques específicos, o módulo Simulador Adversário pode ser configurado para geração de tráfego benigno e ataques.

5.2. Orquestração, execução e mitigação de ataques de DDoS

Os ataques de negação de serviço distribuídos (DDoS) são caracterizados pela ocorrência de tráfego volumétrico que visa indisponibilizar um recurso ou seus componentes adjacentes [Gemmer et al. 2023] a partir de múltiplas origens. Comumente, ataques de DDoS ocorrem em fases que incluem desde o recrutamento de máquinas que executarão o ataque (*hosts* infectados com vírus/*bot*), orquestração e controle dos *hosts* e execução propriamente dita. A arquitetura HackInSDN incorpora recursos que propiciam experimentar e estudar todas essas fases de um ataque DDoS, desde a orquestração até a execução do ataque, acrescido, claro, do monitoramento/detecção e mitigação.

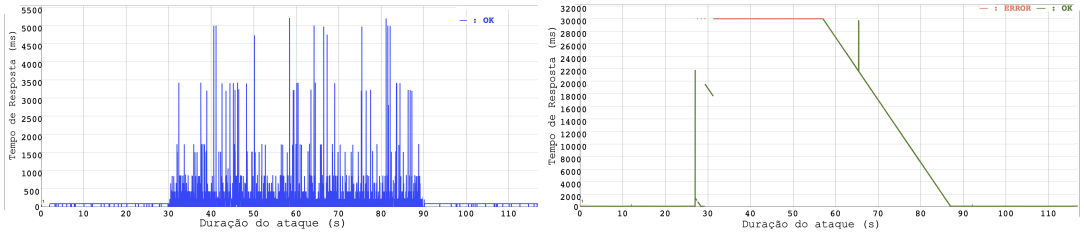
Neste caso de uso, o monitoramento e detecção dos ataques de DDoS são baseados na coleta de estatísticas de fluxos de rede via protocolo sFlow e, em seguida, no processamento destes dados a partir de algoritmos de aprendizagem de máquina. O módulo de Emulação de Redes oferece suporte à configuração de sFlow nos *switches* de uma topologia. Para o processamento dos relatórios sFlow e geração de estatísticas, pode-se utilizar diversos *softwares*, em particular este caso de uso faz referência ao sFlow-RT, uma ferramenta *opensource* que permite customizações nas métricas de estatísticas dos fluxos de rede. A partir dessas estatísticas, o módulo de Detecção de Anomalia faz o cálculo do grau de entropia dos IPs de origem e destino com base no cálculo da entropia de Shannon [Shannon 1948], e incorpora esse resultado às demais métricas para análise dos algoritmos de aprendizagem de máquina (Seção 4.3).

As Figuras 4 e 5 apresentam estatísticas do experimento de ataque DDoS executado de duas maneiras: ataques de volumetria executados com o comando HPING[20] e ataques de *slow HTTP* executados com o comando *SLOWLORIS*[21]. É possível observar que o ataque com HPING possui uma volumetria de rede muito maior (5a), e impacta na latência das requisições legítimas (4a), ao passo que o ataque com SLOWLORIS possui uma volumetria muito menor (5b) e um impacto muito maior na latência da aplicação (4b). Isso ocorre pois o ataque de SLOWLORIS envia poucos pacotes em intervalos longos para ocupar os recursos disponíveis e inviabilizar o atendimento de novas conexões, ao passo que o ataque com HPING envia um grande volume de dados para ocupar o canal de transmissão. Em ambos os casos, a arquitetura HackInSDN permite a aplicação de estratégias de simulação dos ataques, monitoramento, detecção e contenção.

6. Conclusões e Trabalhos Futuros

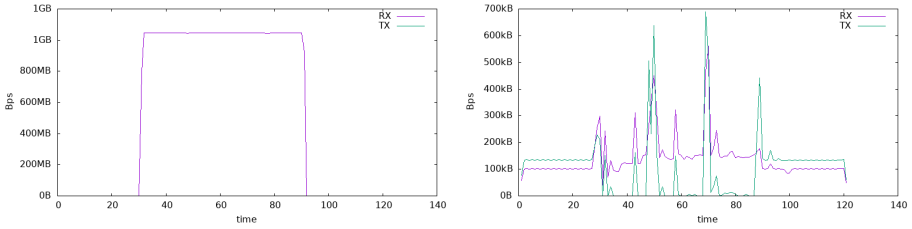
Este artigo apresentou o desenvolvimento de uma arquitetura que explora a segurança cibernética em ambientes de *testbeds*. Para isso, foram levantadas as principais tecnologias relacionadas ao seu desenvolvimento. HackInSDN apresenta um pacote de ferramentas com um ambiente mais robusto e completo para experimentos em tópicos avançados de segurança por meio da programabilidade de redes. Estudantes podem se beneficiar dessa plataforma a partir da execução de laboratórios práticos de segurança e redes utilizando um ou mais componentes da arquitetura. Em particular, quando utilizada em

[20] `https://www.kali.org/tools/hping3/`
[21] `https://github.com/gkbrk/slowloris`



(a) HPING (b) Slowloris

Figura 4. Latência de requisições legítimas em cenário sobre ataques de DDoS



(a) HPING (b) Slowloris

Figura 5. Volumetria de rede em cenário sobre ataque DDoS

conjunto com o *testbed* RNP, os estudantes podem utilizar um ambiente com recursos de experimentação semelhantes àqueles encontrados em produção. A plataforma é altamente extensível e permite integração de componentes de software e hardware de diferentes naturezas, por exemplo: switches P4, hardware de geração de tráfego, contêineres com ferramentas diversas inclusive soluções comerciais, entre outras.

Em trabalhos futuros, investigaremos novas regras de contenção/mitigação de ataques, e.g. aplicação de regras probabilísticas para identificação de tráfego malicioso. Além disso, realizaremos novos experimentos com casos de uso relacionados a equipes de defesa (*Blue teams*) e de ataque (*Red teams*).

Agradecimentos

Os autores agradecem o apoio da Rede Nacional de Ensino e Pesquisa (RNP), do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e da Fundação de Amparo à Pesquisa do Estado da Bahia (FAPESB).

Referências

Brito, I. V. S. and Sampaio, L. N. (2024). Mininet-sec: plataforma de experimentação para segurança cibernética em redes programáveis. *Salão de Ferramentas - SBRC*.

Chouliaras, N., Kittes, G., Kantzavelou, I., Maglaras, L., Pantziou, G., and Ferrag, M. A. (2021). Cyber ranges and testbeds for education, training, and research. *Applied Sciences*, 11(4):1809.

Du, W. (2011). Seed: hands-on lab exercises for computer security education. *IEEE Security & Privacy*, 9(5):70–73.

Du, W., Zeng, H., and Won, K. (2022). Seed emulator: An internet emulator for research and education. In *Proceedings of the 21st ACM Workshop on Hot Topics in Networks*, pages 101–107.

Gemmer, D. D., Meyer, B. H., Mello, E. R. d., Schwarz, M., Wangham, M. S., and Nogueira, M. (2023). A Scalable Cyber Security Framework for the Experimentation of DDoS Attacks of Things. In *NOMS 2023-2023 IEEE/IFIP*, pages 1–7.

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.

Jonas, D., Yusuf, N. A., and Zahra, A. R. A. (2023). Enhancing Security Frameworks with Artificial Intelligence in Cybersecurity. *International Transactions on Education Technology*, 2(1):83–91.

Mirkovic, J., Kocoloski, B., and Balenson, D. (2024). Enabling reproducibility through the sphere research infrastructure. *login Usenix Mag.*

Rahouti, M. and Xiong, K. (2019). A Customized Educational Booster for Online Students in Cybersecurity Education. In *CSEDU (2)*, pages 535–541.

Rahouti, M., Xiong, K., and Lin, J. (2021). Leveraging a cloud-based testbed and software-defined networking for cybersecurity and networking education. *Engineering Reports*, 3(10):e12395.

Seufert, M., Dietz, K., Wehner, N., Geißler, S., Schüler, J., Wolz, M., Hotho, A., Casas, P., Hoßfeld, T., and Feldmann, A. (2024). Marina: Realizing ML-Driven Real-Time Network Traffic Monitoring at Terabit Scale. *IEEE Transactions on Network and Service Management*.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423.

Stamatopoulos, D., Katsantonis, M., Fouliras, P., and Mavridis, I. (2024). Exploring the architectural composition of cyber ranges: A systematic review. *Future Internet*, 16(7):231.

Tan, L., Su, W., Zhang, W., Lv, J., Zhang, Z., Miao, J., Liu, X., and Li, N. (2021). In-band network telemetry: A survey. *Computer Networks*, 186:107763.

Vykopal, J., Čeleda, P., Seda, P., Švábenskỳ, V., and Tovarňák, D. (2021). Scalable learning environments for teaching cybersecurity hands-on. In *2021 IEEE Frontiers in Education Conference (FIE)*, pages 1–9. IEEE.

Wang, M., Li, B., and Li, Z. (2004). sflow: Towards resource-efficient and agile service federation in service overlay networks. In *24th International Conference on Distributed Computing Systems, 2004. Proceedings.*, pages 628–635. IEEE.

Wangham, M. S., Meyer, B. H., Gemmer, D. D., de Santana, K. G., Frank, L. R., de Campos, L. E. F., de Mello, E. R., and Schwarz, M. F. (2024). Testbeds para pesquisa experimental em cibersegurança: Da teoria à prática. *Minicursos do Simpósio Brasileiro de Redes de Computadores - SBRC*.

Yamin, M. M., Katt, B., and Gkioulos, V. (2020). Cyber ranges and security testbeds: Scenarios, functions, tools and architecture. *Computers & Security*, 88:101636.