PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

ADILSON GALIANO FILHO

UM MODELO DINÂMICO DE DETECÇÃO DE INTRUSÃO DE REDE PARA AMBIENTES IMPLANTADOS ATRAVÉS DE INFRAESTRUTURA COMO CÓDIGO

CURITIBA
2025

ADILSON GALIANO FILHO

UM MODELO DINÂMICO DE DETECÇÃO DE INTRUSÃO DE REDE PARA AMBIENTES IMPLANTADOS ATRAVÉS DE INFRAESTRUTURA COMO CÓDIGO

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Informática da Pontifícia Universidade Católica do Paraná, como requisito parcial para obtenção do título de Mestre em Informática.

Orientador: Prof. Dr. Eduardo Kugler Viegas
Coorientador: Prof. Dr. Altair Olivo Santin

CURITIBA

2025

Dedico este trabalho a minha amada esposa, Luciana Correa Hoefel Galiano e ao meu amado filho, Emmanuel Correa Hoefel Galiano, pelo amor, apoio e pela compreensão exercidos de forma paciente e incansável em todas as etapas deste Mestrado em Informática.

AGRADECIMENTOS

Demonstro aqui a minha sincera gratidão ao meu orientador, Prof. Dr. Eduardo Kugler Viegas e ao meu coorientador, Prof. Dr. Altair Olivo Santin, que, com serenidade, sabedoria e dedicação, mostraram-me caminhos, apontaram meus erros e me encorajaram nos momentos difíceis acreditando em minha capacidade durante toda a jornada de realização deste trabalho.

Manifesto também meu profundo agradecimento aos amigos que sempre estiveram prontos a me ajudar, especialmente aos Professores Me. Edson Ossamu Kageyama, Me. Fellipe Medeiros Veiga, Me. Jhonatan Geremias, Me. Miguel Gustavo Rodrigues, Me. Pedro Horchulhack, Dr. Roger Robson dos Santos e, principalmente, ao Me. Juliano Sartori Langaro, por sua parceria fiel nos estudos em sala de aula e nos laboratórios de pesquisa.

Por fim, agradeço aos meus coordenadores profissionais, Diogo Xavier Saes e Rogério Sotero Wansson, pela oportunidade de realizar este Mestrado em Informática.

RESUMO

A natureza dinâmica dos ambientes provisionados por Infraestrutura como Código (IaC) impõe desafios significativos, tais como evolução contínua da infraestrutura, dificuldade na definição de padrões normais e treinamento e atualização contínua dos modelos para os Sistemas de Detecção de Intrusão de Rede (NIDS) baseados em Aprendizado de Máquina (AM). A evolução da IaC dificulta a manutenção da precisão da detecção, uma vez que os modelos de AM enfrentam dificuldades para se adaptar a rápidas mudanças nos padrões de rede e às novas técnicas de ataque. Sendo assim, este trabalho propõe uma nova estrutura de NIDS baseada em seleção de características multiobjetivo e classificação dinâmica, projetada para lidar com o comportamento não estacionário das infraestruturas provisionadas por IaC. A seleção de atributos multiobjetivo aprimora a capacidade de generalização do modelo durante o treinamento, permitindo uma adaptação mais eficaz às mudanças contínuas dos ambientes IaC. A classificação dinâmica complementa essa abordagem ao selecionar, de forma ativa, o subconjunto mais adequado de classificadores na fase de inferência, garantindo uma adaptação contínua ao estado atual da infraestrutura. Avaliações experimentais realizadas em um testbed realista, provisionado por IaC com mais de 19 configurações, demonstram melhorias expressivas de até 0,31 no F1-Score, em comparação com abordagens tradicionais.

Palavras-chave: Infraestrutura como Código; Detecção de Intrusão de Rede; Aprendizado de Máquina; Seleção de Características; Classificação Dinâmica.

ABSTRACT

The dynamic nature of environments provisioned by Infrastructure as Code (IaC) poses significant challenges, such as continuous infrastructure evolution, difficulty in defining normal patterns, and the need for continuous training and updating of models, for Network Intrusion Detection Systems (NIDS) based on Machine Learning (ML). The evolution of IaC complicates the maintenance of detection accuracy, as ML models struggle to adapt to rapid changes in network patterns and emerging attack techniques. This work proposes a novel NIDS framework based on multi-objective feature selection and dynamic classification, designed to handle the non-stationary behavior of infrastructures provisioned by IaC. Multi-objective feature selection enhances the model's generalization capability during training, allowing for more effective adaptation to the continuous changes in IaC environments. Dynamic classification complements this approach by actively selecting the most suitable subset of classifiers during the inference phase, ensuring continuous adaptation to the current state of the infrastructure. Experimental evaluations conducted in a realistic testbed, provisioned by IaC with over 19 configurations, demonstrate significant improvements of up to 0.31 in F1-Score compared to traditional approaches.

Keywords: Infrastructure as Code; Network Intrusion Detection; Machine Learning; Feature Selection; Dynamic Classification.

LISTA DE ALGORITMOS

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AB-TRAP	Activity-Based Threat Recognition and Analysis Platform
ACK	Acknowledgment
AM	Aprendizagem de Máquina
API	Application Programming Interface
ARM	Azure Resource Manager
CAAS	Containers as a Service
CDN	Content Delivery Network
CICIDS	Canadian Institute for Cybersecurity Intrusion Detection System
CMS	Content Management System
COVID-19	Coronavirus Disease 2019
CPU	Central Unit Processor
CVE	Common Vulnerabilities and Exposures
DBAAS	Database as a Service
DEVOPS	Development and Operations
DDOS	Distributed Denial of Service
DIRB	Directory Buster
DNS	Domain Name System
DOS	Denial of Service
FAAS	Function as a Service
GCP	Google Cloud Platform
HCL	Hashcorp Configuration Language
HIDS	Host Intrusion Detection System
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IAAS	Infrastructure as a Service
IAM	Identity and Access Management
IAC	Infrastructure as Code
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IPC	Inter-Process Communication
ISS	Information Systems Security

KNN	K-Nearest Neighbors
MNT	Mount
NET	Network
NFS	Network File System
NIDS	Network Intrusion Detection System
NIKTO	Análise de Vulnerabilidades em Servidores Web
NIST	National Institute of Standards and Technology
NMAP	Network Mapper
NSL-KDD	New System Level Knowledge Discovery in Databases
PAAS	Platform as a Service
PCAP	Packet Capture
PCN	Protected Core Networking
PID	Process Identifier
RQ	Research Question
RFC	Request for Comments
SAAS	Software as a Service
SIEM	Security Information and Event Management
SMB	Server Message Block
SQL	Structured Query Language
SSH	Secure Shell
SYN	Synchronize
TCPDUMP	Análise de Pacotes em Redes
TI	Tecnologia da Informação
UDP	User Datagram Protocol
UNSW-NB15	University of New South Wales Network-Based 15
USER	User
VLAN	Virtual Local Area Network
VPC	Virtual Private Cloud
VPN	Virtual Private Network
WAPITI	Análise de Vulnerabilidades em Aplicações Web
WPSCAN	Análise de Vulnerabilidades em Sites WordPress
XSS	Cross-site scripting
YAML	Yet Another Markup Language

SUMÁRIO

CAPÍTULO 1

INTRODUÇÃO

O capítulo 1 deste trabalho é organizado, em subseções, da seguinte forma: 1.1 Contexto; 1.2 Motivação; 1.3 Objetivos, 1.3.1, Objetivo geral, 3.2 Objetivos específicos; 1.4 Contribuições e 1.5 Estrutura do documento.

1.1 CONTEXTO

Em 2023, o mercado global de Infraestrutura como Código (IaC) foi estimado em US$ 847 milhões, com previsão de crescimento anual de 24,4% até 2030 (GRAND VIEW RESEARCH, 2024). Esse avanço reflete a adoção crescente da computação em nuvem e das práticas de integração de desenvolvimento e operações (DevOps), focadas na otimização da gestão de infraestrutura de Tecnologia da Informação (TI). Ao mesmo tempo, a computação em nuvem segue uma trajetória de expansão acelerada estimada em US$ 602,31 bilhões, com projeção de 21,2% ao ano até 2030 (GRAND VIEW RESEARCH, 2024). A migração das empresas para a nuvem busca maior agilidade, flexibilidade e redução de custos pela eliminação de arquiteturas físicas.

Nesse cenário, a utilização de contêineres e orquestração ganha destaque, pois gerencia múltiplos contêineres de forma automatizada. Além disso, a arquitetura de microsserviços complementa a nuvem ao fornecer maleabilidade para aplicativos modernos, sobretudo no crescente ecossistema de dispositivos móveis que demandam aplicações resilientes e escaláveis. A IaC é o processo de gerenciamento e provisionamento da infraestrutura de TI, por meio de arquivos de configuração legíveis por máquina, reduzindo a necessidade de configurações manuais (NIST, 2021).

“Nos últimos anos, o provisionamento de infraestrutura de TI era realizado predominantemente de forma manual, uma tarefa custosa e demorada, que frequentemente exige supervisão contínua por parte dos operadores” (RAHMAN, 2019). À medida que a complexidade das infraestruturas de TI aumenta, a demanda por técnicas de provisionamento automatizado de recursos torna-se cada vez mais evidente. Nesse contexto, a IaC abriu caminho

para automatizar a tarefa de provisionamento de infraestrutura na qual o software será implantado (OPDEBEECK, 2023).

Na prática, a IaC permite a configuração automática de dependências do sistema e o provisionamento de instâncias locais e remotas para facilitar a implantação contínua de acordo com os requisitos do serviço (KONJAANG, 2021). Isso é realizado por meio do uso de scripts de provisionamento (códigos) que atuam como modelos de configuração para as tarefas de provisionamento de serviços, possibilitando que organizações de TI reduzam significativamente o tempo de implantação. Devido às vantagens proporcionadas pela automação do provisionamento de infraestrutura, várias tecnologias de IaC surgiram nos últimos anos, incluindo a Chef (PROGRESSCHEF, 2024), Puppet (PERFORCE, 2024) e Terraform (HASHICORP, 2024) para infraestruturas privadas, além do AWS CloudFormation (SERVICES, 2024) e Azure Resource Manager (MICROSOFT, 2024) para nuvens públicas.

A natureza dinâmica dos ambientes provisionados por IaC representa um desafio para Sistemas de Detecção de Intrusão (IDS), sobretudo para Sistemas de Detecção de Intrusão de Rede (NIDS) baseados em Aprendizado de Máquina (AM). A rápida evolução da infraestrutura dificulta a manutenção da precisão na detecção, uma vez que as mudanças contínuas alteram padrões de tráfego e introduzem novas técnicas de ataque.

1.2 MOTIVAÇÃO

Proteger infraestruturas provisionadas por IaC é um desafio significativo devido à natureza dinâmica das configurações de infraestrutura (THAKKAR, 2024). Essas configurações mudam com base no script IaC fornecido, exigindo que as soluções de segurança se adaptem de maneira eficaz a essas variações (OPDEBEECK, 2023). Por outro lado, apesar da automação do provisionamento de infraestrutura promovida pelo IaC, a configuração das soluções de segurança ainda frequentemente requer intervenção manual. Isso inclui configurar regras de firewall, políticas de controle de acesso, Redes Locais Virtuais (VLANs) e NIDS, entre outras medidas.

Como resultado, existe uma lacuna entre a automação das tarefas de provisionamento de infraestrutura habilitadas por técnicas de IaC e a automação das configurações de soluções de segurança necessárias para proteger a infraestrutura recém-provisionada (SAAVEDRA, 2022). Essa lacuna dificulta a capacidade de abordar preocupações de segurança em ambientes de IaC em rápida mudança de forma eficaz. Entretanto, automatizar configurações de segurança

com base no código IaC apresenta desafios significativos, especialmente para soluções NIDS (CHERFI, 2024).

Grande parte da literatura foca na concepção de modelos de detecção de intrusão baseados em comportamento, geralmente depende da construção de um modelo de AM comportamental (MOLINA-CORONADO, 2020). Essas abordagens assumem uma configuração de ambiente estática como referência, dificultando a adaptação à natureza em constante mudança das infraestruturas provisionadas por IaC (RAHMAN, 2023). Na prática, a configuração da infraestrutura deve ser conhecida previamente para treinar e avaliar o modelo de AM de forma adequada.

Como consequência, quando a configuração da infraestrutura muda, por exemplo, devido a um novo script IaC, o NIDS baseado em AM projetado anteriormente se torna pouco confiável para implementação em produção. Tipicamente, esse problema só pode ser resolvido por meio de atualizações do modelo, um processo que frequentemente exige vários dias ou até semanas para ser concluído. Esse atraso representa uma incitação para a manutenção da detecção eficaz de intrusões em ambientes altamente dinâmicos (CATILLO, 2023).

Desenvolver NIDS baseados em AM para ambientes dinâmicos é particularmente desafiador, pois os modelos de AM são tipicamente otimizados para maximizar a precisão da detecção em um ambiente de treinamento estático e predefinido (SANTOS, 2024). Essa suposição inerente de design limita sua capacidade de se adaptar às mudanças contínuas características das infraestruturas dinâmicas implantadas com IaC (MOLINA-CORONADO, 2020). Essa limitação frequentemente resulta em modelos de detecção de intrusão que generalizam inadequadamente os dados de treinamento, levando a problemas de desempenho em cenários do mundo real.

Consequentemente, esses modelos enfrentam dificuldades para detectar, de maneira eficaz, novos comportamentos ou comportamentos em evolução, situação geralmente observada em infraestruturas provisionadas por IaC. Na prática, os sistemas de detecção de intrusão tendem a alcançar altas precisões de detecção em ambientes que se assemelham ao conjunto de dados de treinamento. No entanto, seu desempenho geralmente se degrada significativamente quando exposto a mudanças comportamentais ou variações desconhecidas no ambiente operacional (WANG, 2021).

Por outro lado, desenvolver um NIDS confiável baseado em AM para infraestruturas provisionadas por IaC exige que o modelo projetado considere efetivamente comportamentos não observados durante a fase de treinamento (RONG, 2022). O desenvolvimento de um NIDS

baseado em AM que seja generalizável tem sido o foco de numerosos estudos nos últimos anos, mas sua aplicação em ambientes de IaC permanece amplamente negligenciada. Em tais ambientes, construir um conjunto de dados de treinamento que reflita com precisão o comportamento da infraestrutura a ser implantada não é facilmente viável devido à natureza dinâmica das configurações de TI, que mudam com base no script IaC e resultam em mudanças correspondentes no comportamento do ambiente (ANDRESINI, 2021).

A implantação de uma nova configuração de serviço altera o comportamento do tráfego de rede, impactando as operações normais e as atividades potenciais de atacantes. Em configurações tradicionais, mudanças no comportamento do tráfego de rede levam a NIDS baseados em AM pouco confiáveis, necessitando de atualizações de modelo para remediação eficaz (VIEGAS, 2021). No entanto, em infraestruturas provisionadas por IaC, o comportamento do ambiente só é revelado após a conclusão da tarefa de provisionamento.

Portanto, para que os NIDS baseados em AM permaneçam confiáveis nesses ambientes dinâmicos, as abordagens propostas devem se adaptar aos novos comportamentos do ambiente em tempo real, sem depender de atualizações frequentes do modelo. Isso é particularmente importante, dada as ambições e atrasos associados à construção de conjuntos de dados de treinamento atualizados para ambientes de IaC em constante evolução.

Os NIDS tradicionais baseados em AM assumem que o comportamento do ambiente implantado é estacionário e não muda ao longo do tempo. Na prática, a literatura presume que o comportamento do tráfego de rede é inteiramente conhecido durante o treinamento e permanece inalterado após o sistema ser implantado em produção. Caso contrário, as taxas de erro do sistema aumentarão em comparação com as observadas durante a fase de teste, levando a um sistema não confiável (OLÍMPIO, 2023). Contudo, desenvolver um NIDS baseado em AM que possa se adaptar a comportamentos de tráfego de rede potencialmente desconhecidos não é facilmente alcançável, particularmente em ambientes de IaC.

Em contrapartida, os pesquisadores frequentemente negligenciam as capacidades de generalização de seus modelos projetados em favor de alcançar maiores precisões de detecção durante os testes (WANG, 2021). Essa abordagem frequentemente resulta em sobreajuste, em que os modelos são ajustados ao comportamento de um único ambiente e apresentam desempenho insatisfatório quando expostos a mudanças no ambiente. Consequentemente, apesar da crescente adoção de infraestruturas provisionadas por IaC nos últimos anos, persiste uma lacuna de pesquisa no desenvolvimento de NIDS baseados em AM capazes de lidar com

as mudanças no comportamento do ambiente causadas pelos códigos de automação (scripts) de configuração de IaC.

1.3 OBJETIVOS

1.3.1 Objetivo geral

O objetivo geral deste trabalho é propor um novo NIDS baseado em AM adaptado para infraestruturas provisionadas por IaC, visando aprimorar a capacidade de generalização e a detecção de comportamentos inesperados. Para tanto, a proposta contempla técnicas de seleção de características e mecanismos de adaptação de classificadores, assegurando maior robustez em cenários dinâmicos introduzidos pelos scripts de configuração.

1.3.2 Objetivos específicos

O objetivo geral se desdobrará nos seguintes objetivos específicos:

I. Criar um ambiente de IaC flexível para a geração de cenários de intrusão, com base nos datasets e modelos destes, para a validação da proposta apresentada;
II. Projetar um IDS adaptável que utilize técnicas de aprendizagem de máquina para identificar comportamentos anômalos em infraestruturas configuráveis;
III. Validar como o classificador dinâmico se ajusta e responde a alterações na infraestrutura, medindo a precisão e a taxa de falsos positivos nos diferentes cenários de experimentos para a entrega da proposta;

1.4 CONTRIBUIÇÕES

As principais contribuições do trabalho são:

- Um novo NIDS baseado em AM para infraestruturas provisionadas por IaC. O modelo proposto aborda a generalização do mesmo na fase de treinamento e seleciona ativamente os classificadores durante a fase de inferência para detecção de novidades. A proposta melhora o F1-Score em até 0,31;

- Um novo conjunto de dados de intrusão provisionado por IaC, publicamente disponível, com 19 configurações diferentes, gerado com 100 clientes normais para 5 serviços, bem como 14 comportamentos diferentes de atacantes.

Devido às contribuições deste trabalho, A dissertação resultou na submissão de um artigo intitulado *A Dynamic Network Intrusion Detection Model for Infrastructure as Code Deployed Environments*, no Journal of Network and Systems Management.

1.5 ESTRUTURA DO DOCUMENTO

O Capítulo 2 é composto pela fundamentação teórica da pesquisa. No Capítulo 3, são elencadas as bibliografias relacionadas ao projeto. O Capítulo 4 detalha a proposta aplicada para a realização deste projeto. No Capítulo 5, são apresentados o protótipo e o testbed. Já o capítulo 6 aponta os resultados dos experimentos já apresentados. E, por fim, no Capítulo 7, apresenta-se a conclusão, a qual traz o resumo dos principais achados e as contribuições científicas do trabalho.

CAPÍTULO 2

FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, apresenta-se a fundamentação teórica sobre o sistema de detecção de intrusão escolhido para a arquitetura proposta, assim como os demais elementos presentes no estudo. Inicia-se com a computação em nuvem e a conteinerização na subseção 2.1. A subseção 2.2 aborda os fundamentos da IaC. Na subseção 2.3, define-se o que é um IDS. Em seguida, na subseção 2.4, discutem-se os desafios da AM para a IaC. Ao final, na subseção 2.5, são realizadas algumas considerações sobre este capítulo.

2.1 COMPUTAÇÃO EM NUVEM E CONTEINERIZAÇÃO

A computação em nuvem está relacionada a um modelo de fornecimento de serviços de TI em que recursos como processamento, armazenamento e rede são disponibilizados de forma virtualizada, flexível e sob demanda. Essa abordagem possibilita que os usuários gerenciem esses recursos de maneira remota, dispensando a necessidade de se manter uma infraestrutura física complexa e custosa. O conceito de computação em nuvem remonta às décadas de 1960 e 1970, período em que surgiu a ideia de computação em grade, cujo objetivo era facilitar o compartilhamento de recursos entre universidades, instituições e centros de pesquisa.

No entanto, a transformação mais significativa ocorreu nos anos 1990, com a expansão da internet comercial, que criou as condições necessárias para o surgimento das nuvens computacionais no formato que conhecemos hoje. Um marco importante nesse processo aconteceu em 2006, quando a Amazon lançou o Amazon Web Services (AWS) inaugurando a era moderna da computação em nuvem. Em seguida, companhias como Google, Microsoft e IBM também aderiram a esse modelo, acelerando sua adoção em larga escala.

As nuvens computacionais possuem características que as diferenciam de outros tipos de serviços de TI, tais como:

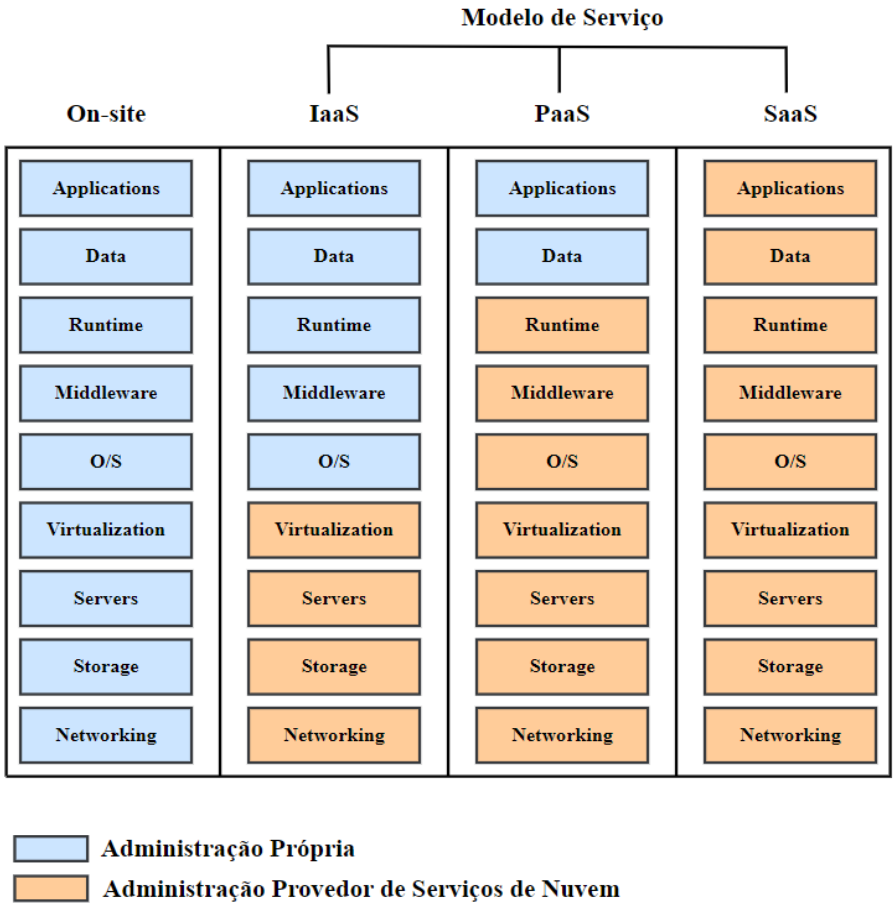
- Elasticidade: permite o ajuste rápido e dinâmico dos recursos conforme a demanda, garantindo maior eficiência;

- Pay-as-you-go: modelo de pagamento baseado no consumo real dos recursos, proporcionando um controle mais eficaz sobre os custos;
- Virtualização: abstração dos recursos físicos, viabilizando a alocação dinâmica de máquinas virtuais e o compartilhamento eficiente de recursos;
- Autosserviço: possibilita aos usuários provisionarem e gerenciarem seus próprios recursos sem a necessidade de intervenção do provedor;
- Alta disponibilidade: garante que os serviços estejam sempre acessíveis, minimizando falhas e interrupções.

Os benefícios das nuvens computacionais incluem a redução de custos com infraestrutura, flexibilidade na gestão de recursos, escalabilidade, resposta rápida às demandas e simplificação do gerenciamento de TI. Os serviços de computação em nuvem podem ser oferecidos de diversas formas. Por meio da norma SP-800-145 (NIST, 2011) o NIST define três modelos de arquitetura de nuvem com base nos recursos fornecidos ao cliente:

1. Infraestrutura como Serviço (IaaS): o cliente pode provisionar recursos computacionais como processamento, armazenamento, rede e infraestrutura física, tendo a liberdade de instalar e executar qualquer software, incluindo sistemas operacionais e aplicações, mas não tem controle sobre a infraestrutura subjacente. Ainda assim, esse cliente tem controle sobre o sistema operacional, armazenamento, as aplicações e, possivelmente, configurações de rede específicas, como firewalls.
2. Plataforma como Serviço (PaaS): o provedor oferece uma plataforma em nuvem para o desenvolvimento e implantação de aplicações usando linguagens de programação e ferramentas suportadas. Nesse caso, o cliente tem controle sobre as aplicações que desenvolve e implanta, mas não gerencia a infraestrutura subjacente, como redes, servidores e sistemas operacionais. Em alguns casos, o cliente PaaS pode se tornar um provedor de SaaS para seus próprios usuários.
3. Software como Serviço (SaaS): o provedor disponibiliza aplicações que rodam em sua infraestrutura de nuvem e são acessadas pelo cliente por meio de uma interface simples, como um navegador web (por exemplo: webmail). Diante disso, o cliente não gerencia a infraestrutura adjacente, nem controla as funcionalidades da aplicação, exceto alguns parâmetros de configuração permitidos. A seguir, a Figura 1 ilustra os recursos oferecidos por cada um dos modelos On-site, IaaS, PaaS e SaaS.

Figura 1 – Modelo de referência para modelo de serviço



Fonte: adaptado de Red Hat, 2024.

Os serviços de nuvem também podem ser classificados de acordo com a propriedade e o uso compartilhado dos recursos entre diferentes clientes, resultando nos seguintes modelos de implantação:

a. Nuvem privada: a infraestrutura é dedicada a uma única organização, podendo ser gerida por ela ou por terceiros. Essa infraestrutura pode estar localizada dentro ou fora da organização, mas oferece maior controle sobre os recursos e menor exposição ao risco de ambientes multiusuários.
b. Nuvem híbrida: combina dois ou mais modelos de nuvem (pública, privada ou comunitária), que operam como entidades independentes, mas são interligados por tecnologias que permitem a interoperabilidade entre eles.
c. Nuvem pública: a infraestrutura e os recursos computacionais são oferecidos ao público em geral por meio da internet e são gerenciados por um provedor externo. Assim, os serviços são disponibilizados fora da organização cliente.

d. Nuvem comunitária: a infraestrutura de nuvem é provisionada para uso exclusivo por uma comunidade específica de consumidores de organizações, que compartilham interesses em comum. Ela pode ser de propriedade, gerenciada e operada por uma ou mais das organizações da comunidade, por um terceiro, ou por uma combinação destes, e pode existir dentro ou fora das instalações.

A conteinerização é uma técnica de virtualização que permite executar aplicativos e processos isolados em um sistema operacional compartilhado, portanto, os recursos e bibliotecas disponibilizados pelo sistema operacional. O isolamento é a garantia de que um contêiner não é capaz de acessar informações de outro, bem como do sistema operacional. No entanto, o sistema operacional percebe o conjunto de processos executados dentro de um contêiner como processos convencionais, escalonando-os como quaisquer outros.

Algumas das primeiras tentativas de se executar microsserviços em ambientes Unix ocorreram no início dos anos 2000 com o chroot. O chroot permitia alterar o caminho de um processo e seus processos filhos. Embora não exista isolamento real com o chroot e a alteração seja apenas da árvore de diretórios, este serviu de base para os namespaces. No Unix, os contêineres são criados a partir de namespaces (LINUX, 2024) e CGroups (LINUX, 2024). Os namespaces são recursos do kernel Linux, implementados desde a versão 2.4.19 do kernel, que permitem isolar diferentes aspectos do sistema operacional dentro do contêiner, como processos, rede, sistema de arquivos e identidades de usuário e grupo.

Cada namespace cria um ambiente isolado para esses recursos no contêiner, garantindo que eles não possam ver ou interagir com recursos fora desse contêiner. Dessa forma, cria-se uma camada de abstração para aquele recurso: dentro do contêiner, o conteúdo dessa camada é visto como todo recurso disponível. Os tipos de namespaces usados na conteinerização são:

- PID Namespace: namespace de processos, criado em 2002, que cria um ambiente isolado para processos em execução no contêiner, garantindo que eles não possam ver ou interagir com processos fora deste.
- NET Namespace: namespace de rede, criado em 2002, que isola a pilha de rede do contêiner, de modo que o contêiner tenha seu próprio endereço IP, as portas, a tabela de roteamento, entre outros recursos de rede.
- MNT Namespace: namespace de sistema de arquivos, criado em 2002, o qual permite que o contêiner tenha sua própria visão do sistema de arquivos, isolado do sistema de arquivos host.

- USER Namespace: namespace de usuário, criado em 2004, que isola as identidades de usuário e grupo dentro do contêiner, para que este não possa ver ou modificar usuários ou grupos fora do contêiner.
- IPC Namespace: namespace de comunicação entre processos, criado em 2008, que isola a comunicação entre processos dentro do contêiner.

Esses recursos garantem que os contêineres sejam isolados do resto do sistema operacional, permitindo que aplicativos e processos sejam executados em um ambiente controlado e com previsibilidade. Já o cgroups é um mecanismo do kernel Linux para limitar, medir e isolar o uso de recursos do sistema, como CPU, memória, E/S de disco e rede. Cada cgroup pode ser configurado com limites e políticas específicas para o uso desses recursos pelos processos dentro do contêiner.

2.2 INFRAESTRUTURA COMO CÓDIGO

A Infraestrutura como Código (IaC, do inglês Infrastructure as Code) é uma abordagem para automação de infraestrutura baseada em práticas de desenvolvimento de software. Ela enfatiza rotinas consistentes e repetíveis para provisionar e modificar sistemas e suas configurações. As alterações são feitas no código, e a automação é usada para testar e aplicar essas modificações aos sistemas (MORRIS, 2021). A IaC ganhou destaque com o surgimento da computação em nuvem, mas muitos dos seus princípios e práticas podem ser aplicados também a outros ambientes, como infraestruturas virtualizadas e hardware físico.

Morris (2021) observa que plataformas de virtualização podem ser configuradas para criar e destruir servidores de forma programática, semelhante ao que ocorre na nuvem. No caso do hardware físico, embora o provisionamento seja feito manualmente, sua configuração e atualização podem ser realizadas com ferramentas de gerenciamento baseadas em IaC. Há três recursos essenciais fornecidos por uma plataforma de infraestrutura: computação, armazenamento e rede. Diferentes plataformas combinam e empacotam esses recursos de diferentes maneiras.

Os recursos de computação são responsáveis pela execução de código. No seu nível mais básico, a computação refere-se ao tempo de execução em um núcleo de CPU de um servidor físico. As plataformas de infraestrutura em nuvem oferecem formas avançadas de computação, adaptadas a diferentes necessidades. As máquinas virtuais operam em hipervisores gerenciados pela plataforma, utilizando servidores físicos de forma eficiente. Além disso,

servidores físicos, ou bare metal, podem ser provisionados sob demanda para cargas de trabalho que exigem acesso direto ao hardware.

Clusters de servidores consistem em grupos de instâncias, sejam máquinas virtuais ou servidores físicos, geridos como uma unidade para maior escalabilidade e redundância. A conteinerização, por meio de Containers as a Service (CaaS), permite a execução de aplicações em ambientes isolados, utilizando imagens padronizadas, como Docker. Essa tecnologia facilita a implantação ágil e escalável de aplicativos. Clusters de hospedagem de aplicativos gerenciam a execução de vários aplicativos em um conjunto de servidores, oferecendo infraestrutura automatizada para esses serviços.

Finalmente, a Função como Serviço (FaaS), também conhecida como computação sem servidor (serverless), permite a execução de código sob demanda, ativado por eventos ou agendamentos, sem a necessidade de manter servidores permanentemente ativos, encerrando a execução ao completar a tarefa. Muitos sistemas dinâmicos exigem soluções de armazenamento, como volumes de disco, bancos de dados e repositórios centrais para arquivos. Mesmo que um aplicativo não utilize armazenamento diretamente, vários serviços que ele depende necessitam de recursos de armazenamento, como para guardar imagens de computação, incluindo snapshots de máquinas virtuais e imagens de contêiner.

Uma plataforma verdadeiramente dinâmica gerencia e fornece armazenamento para aplicativos de forma transparente, diferindo dos sistemas de virtualização clássicos, nos quais é necessário especificar explicitamente qual armazenamento físico alocar e anexar a cada instância de computação. As plataformas de infraestrutura em nuvem oferecem vários recursos de armazenamento projetados para diferentes finalidades. O armazenamento em bloco permite anexar volumes de disco virtual a servidores ou contêineres, funcionando como discos locais, ideal para dados persistentes.

O armazenamento de objetos é utilizado para armazenar e acessar arquivos de forma distribuída, ideal para dados não estruturados. Sistemas de arquivos em rede permitem o compartilhamento de volumes entre múltiplas instâncias de computação, usando protocolos como NFS ou SMB, facilitando a colaboração entre servidores. Já o armazenamento de dados estruturado é oferecido por meio de bancos de dados gerenciados como serviço (DBaaS), suportando sistemas como MySQL, PostgreSQL e SQL Server, com gestão automatizada e definição via código.

Esses recursos fornecem flexibilidade e escalabilidade para diferentes necessidades de armazenamento em ambientes em nuvem. Além disso, qualquer recurso de armazenamento

pode ser criptografado para permitir o armazenamento seguro de senhas, chaves e outras informações sensíveis que poderiam ser exploradas por invasores para obter acesso privilegiado a sistemas e recursos. Um serviço de gerenciamento de segredos é projetado especificamente para ajudar na administração desses tipos de recursos.

Uma plataforma de infraestrutura em nuvem oferece uma variedade de serviços e construções de rede essenciais para gerenciar sistemas distribuídos. Os blocos de endereços de rede, como VPCs (AWS) e Redes Virtuais (Azure e GCP), são estruturas que agrupam recursos e controlam o roteamento de tráfego. Eles podem ser subdivididos em sub-redes ou VLANs e conectados a data centers para melhorar a disponibilidade e o controle.

Nomes (DNS) mapeiam endereços IP, rotas definem o tráfego permitido entre blocos de rede e gateways controlam o fluxo de entrada e saída desses blocos. O balanceamento de carga distribui conexões entre recursos e os proxies transformam ou redirecionam essas conexões conforme regras definidas. Gateways de API atuam como proxies especializados em gerenciar autenticação e limites de uso de APIs. VPNs conectam diferentes blocos de endereços geograficamente, simulando uma única rede, enquanto conexões diretas criam links dedicados entre redes em nuvem e locais físicos, como data centers. Regras de firewall controlam o tráfego entre diferentes partes da rede. Filas de mensagens assíncronas permitem a comunicação eficiente entre processos e caches distribuem dados para reduzir latência, como ocorre em CDNs, que espalham conteúdo geograficamente via HTTP/S.

Por fim, a malha de serviços gerencia, de forma descentralizada, a conectividade entre partes de sistemas distribuídos, garantindo controle e eficiência na comunicação interna dos serviços. Para gerenciar efetivamente a infraestrutura, os conceitos da IaC são divididos em três categorias distintas, cada uma com seu próprio conjunto de software especializado, (WANG, 2022).

Essas categorias, representadas pela Figura 2, são:

a. O Gerenciamento de Configuração, que envolve a automação do processo de configuração e administração de software e serviços em servidores. Isso abrange a instalação de programas, a definição de parâmetros e a aplicação de atualizações. Exemplos de ferramentas que realizam essas tarefas incluem Ansible, Chef e Puppet.
b. A construção de imagens envolvendo a criação de templates de máquina que podem ser utilizados para a rápida inicialização de novos servidores. Esse processo inclui a geração de uma imagem base de um sistema operacional e a captura dessa imagem como um modelo, permitindo sua reutilização para criar novas instâncias ou servidores.

Ferramentas como HashiCorp Packer e Docker são exemplos de softwares para construção de imagens.

c. O provisionamento de infraestrutura, que se refere à automação no fornecimento e gerenciamento de servidores físicos e virtuais, além de dispositivos de armazenamento e rede. Esse processo inclui a configuração física dos servidores, a instalação do sistema operacional e a configuração da rede. Exemplos de ferramentas que realizam esse tipo de tarefa incluem Terraform, TOSCA e Cloudify.

Figura 2 – Categorias da IaC: Gerenciamento de Configuração (Ansible, Chef, Puppet), Construção de Imagens (Packer, Docker) e Provisionamento de Infraestrutura (Terraform, TOSCA, Cloudify)

Fonte: adaptado de WANG, 2022.

2.3 DEFINIÇÃO DE UM IDS

Os Sistemas de Detecção de Intrusão (IDS, do inglês Intrusion Detection Systems) são ferramentas aplicadas no campo da segurança da informação, comumente comparados na literatura a sistemas de segurança física, como os utilizados em instituições bancárias. Em um banco, além de cofres e trancas, que protegem fisicamente os ativos, é comum a instalação de câmeras de segurança e alarmes que monitoram continuamente o ambiente. Esses sistemas de monitoramento não são responsáveis por impedir fisicamente um intruso de acessar os cofres, mas funcionam como uma camada adicional de proteção, alertando as equipes de segurança assim que uma atividade suspeita ou uma tentativa de invasão é detectada.

Analogamente, um IDS monitora os sistemas e redes computacionais, procurando sinais de atividades maliciosas. Assim como as câmeras de vigilância e alarmes de um banco emitem alertas quando identificam tentativas de violação de segurança, o IDS gera notificações para os administradores de segurança de TI quando detecta comportamentos anômalos, permitindo uma resposta rápida e eficiente.

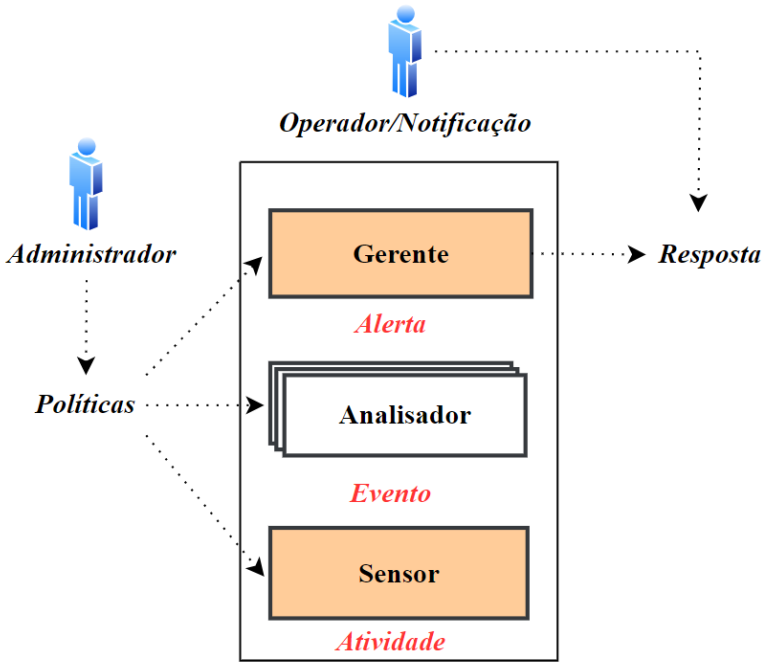
De acordo com a definição da norma SP-800-94 (NIST, 2007), a detecção de intrusão é o processo de monitoramento de eventos que ocorrem em uma rede ou sistema de computadores, seguido da análise desses eventos para identificar possíveis intrusões. Essas

intrusões representam tentativas de comprometer a confidencialidade, integridade ou disponibilidade do sistema ou de contornar seus mecanismos de segurança. Nesse sentido, o IDS age como um sistema de vigilância digital, projetado para detectar e alertar sobre atividades potencialmente prejudiciais, sem necessariamente bloquear o ataque de imediato, mas permitindo que os responsáveis tomem medidas corretivas.

O diagrama da RFC-4766 (WOOD, 2007), na Figura 3, oferece uma perspectiva técnica sobre a arquitetura de um IDS detalhando como ele coleta dados, analisa padrões de tráfego e gera alertas com base em algoritmos de detecção. Em ambientes dinâmicos e em constante mudança, como os proporcionados por IaC, a capacidade de adaptação de um IDS é importante. Ele precisa ser capaz de ajustar suas táticas de monitoramento à medida que a infraestrutura se reconfigura, garantindo uma vigilância contínua e eficaz.

Assim, o IDS atua como uma camada crítica de segurança em sistemas computacionais modernos, oferecendo uma linha de defesa capaz de detectar e mitigar ameaças em tempo real, tal como um sofisticado sistema de segurança em uma instituição bancária previne tentativas de roubo ao alertar sobre atividades suspeitas, permitindo, assim, que ações rápidas sejam tomadas antes que danos mais sérios ocorram.

Figura 3 – Diagrama da RFC-4766 que apresenta a arquitetura técnica de um IDS destacando a coleta de dados, análise de padrões de tráfego e geração de alertas baseada em algoritmos de detecção



Fonte: adaptado de RFC-4766.

Antes de gerar um alerta, o analisador do IDS precisa de uma estratégia para determinar se está ocorrendo ou não um ataque. Na literatura é comum encontrar dois tipos de IDS ao se classificar por estratégia de análise: os que são baseados em mau uso e os que são baseados em anomalia. No IDS baseado em mau uso (misuse) aplica-se uma técnica em que padrões conhecidos são usados para detectar atividades maliciosas. Os padrões correspondentes aos ataques conhecidos são chamados de assinaturas. Como ilustra a Figura 3, o analisador recebe eventos do sensor e faz consultas a uma base de assinaturas para determinar se existe uma correspondência com algum ataque (LYNDON, 2004).

Grande parte dos IDS de mercado fazem a combinação de IDS baseados em redes e assinaturas. Alguns exemplos destes são (KRAWCZYK, 2007): o Snort e o ISS Event Policy. Uma das vantagens desse método é que não consome tantos recursos como o que é baseado em anomalia. Além disso, o processamento dos padrões de ataques pode ser otimizado e o número de falsos positivos pode ser controlado apenas com ajustes (MANDUJANO, 2004).

A exigência de um bom nível de conhecimento de quem cria esses padrões é uma das desvantagens. Outra, é o padrão de ataque, que pode se assemelhar a uma atividade normal e gerar muitos falsos positivos. Entretanto, a principal desvantagem está na necessidade de atualizar constantemente esses padrões para que possam acompanhar os novos tipos de ameaças (MANDUJANO, 2004). Conhecido como Anomaly based IDS ou Behavior based IDS, seu objetivo é detectar atividades com um comportamento incomum em um host ou em uma rede.

Assim como ilustra a Figura 3, o que for diferente de uma atividade normal, ou legítima, pode ser detectado como um ataque. Assim sendo, esse tipo de IDS utiliza perfis que são construídos com dados históricos coletados do ambiente monitorado. Esses dados são usados para medir e monitorar as atividades para saber se estas estão ou não fora do normal (BACE, 2001). Hofmeyer *et al.* (1998), em seu trabalho, apresenta o método de IDS baseado em anomalias, no qual desenha o perfil de chamadas de sistemas e cria métricas para determinar se uma atividade é normal ou anormal.

Um projeto chamado MINDS (Minnesota Intrusion Detection System) foi implementado e comparado ao método baseado em assinatura do Snort (ERTÖZ *et al*., 2004). MINDS é um sistema baseado em mineração de dados, data mining, em que o módulo de detecção é alimentado por ferramentas de fluxo de rede, como o NetFlow. O IDS baseado em anomalias pode detectar novos ataques pelo simples desvio de comportamento e sem a necessidade de ter o conhecimento detalhado da intrusão. Os próprios ataques detectados poderão ser tratados como assinaturas para serem utilizados em um IDS baseado em mau uso.

Por outro lado, o IDS baseado em anomalias tem as seguintes desvantagens: por detectar atividades anormais, qualquer modificação no comportamento legítimo do host ou da rede pode gerar muitos falsos alertas; e a coleta de dados para criar o perfil de comportamento pode se tornar difícil e extensiva (BACE, 2001). Os métodos de detecção baseados em aprendizagem de máquina estabelecem modelos explícitos ou implícitos de padrões categorizados. Nesse método, é comum determinar o comportamento normal por meio de treinamento realizado sobre uma base com dados rotulados.

No caso de IDS baseado em mau uso, o treinamento deve ser feito sobre uma base de ataques conhecidos. Muitas vezes, a aplicação de aprendizagem de máquina pode coincidir com as técnicas estatísticas, mas esse método tem a capacidade de mudar a sua estratégia de execução de acordo com novas informações que são adquiridas. A seguir alguns métodos de aprendizagem em IDS:

I. Os algoritmos genéticos (genetic algorithms) são categorizados como técnicas de solução aproximada, por intermédio de busca e otimização, e fazem parte de uma classe particular de algoritmos evolucionários por utilizarem técnicas inspiradas em biologia evolucionária (tais como herança, mutação, seleção e recombinação). Um exemplo é o trabalho de (LI, 2004) que aplica a técnica de algoritmos genéticos como NIDS. Entre as vantagens estão a robustez e a flexibilidade no método de busca que converge para uma solução de múltiplas direções, mesmo quando não há um conhecimento prévio do comportamento do sistema. Porém, sua desvantagem está no alto consumo de recursos envolvidos (GARCIA-TEODORO *et al*., 2009).

II. O agrupamento de dados (clustering) é uma técnica que funciona agrupando elementos similares (clusters), dada uma similaridade ou distância de medida entre esses elementos. Ao fazer o agrupamento, alguns elementos podem não se encaixar em grupo algum, estes são elementos isolados ou outliers. Para a detecção de intrusão, os elementos outliers podem representar um ataque ou uma anomalia. As abordagens de IDS nessa área variam de acordo com o quanto um outlier pode representar um ataque. Um exemplo é o trabalho de Portnoy *et al*. (2001) que aplica clustering para detecção de intrusão. Uma vantagem dessa técnica é o fato de determinar a ocorrência de intrusão, a partir de dados puros de auditoria, reduzindo o esforço necessário para ajustar o IDS (GARCIA-TEODORO *et al*., 2009). Outra vantagem está em obter bons resultados a partir de treinamento não supervisionado. Entre as desvantagens, a principal está na dependência do comportamento do ambiente monitorado.

Os métodos de coleta de informações mais comuns de se encontrar na literatura de IDS são baseados: em hospedeiro (HIDS) e em rede (NIDS). O IDS baseado em hospedeiro, também conhecido como HIDS (Host-based Intrusion Detection System), é um sistema que monitora um único host para detectar atividades suspeitas. O HIDS normalmente coleta informações de duas maneiras: por meio de pistas de auditoria do sistema operacional e por logs do sistema. As pistas de auditoria geralmente são geradas pelo kernel e, portanto, são mais detalhadas e protegidas do que os logs do sistema. Já os logs de sistema são menores e mais fáceis de compreender (BACE, 2001).

Um IDS baseado em rede (NIDS, do inglês Network-based Intrusion Detection System) é uma solução que visa monitorar e analisar o tráfego de rede com o objetivo de identificar atividades maliciosas ou anômalas. Esse tipo de sistema atua coletando informações diretamente da rede, seja por meio da interceptação de pacotes ou da observação passiva de fluxos de dados, para detectar possíveis tentativas de ataque. A instalação de um NIDS pode ocorrer em dois modos distintos: inline ou passivo.

No modo inline, o NIDS é implementado em um ponto de intersecção do tráfego, operando como um dispositivo de ponte (bridge). Nesse contexto, o sistema intercepta diretamente o fluxo de dados da rede, capturando e analisando os pacotes em tempo real para identificar intrusões, sendo capaz de responder de maneira imediata a ameaças detectadas. Esse modelo permite uma atuação mais proativa, uma vez que o NIDS pode bloquear ou mitigar ataques enquanto os pacotes ainda estão em trânsito. Por outro lado, no modo passivo, o NIDS é conectado a um switch ou hub que fornece cópias dos pacotes transmitidos na rede, sem interferir diretamente no fluxo de dados.

Essa abordagem, embora menos intrusiva, tem a limitação de que o sistema não pode interferir diretamente nas comunicações em tempo real, mas, ainda assim, oferece uma capacidade robusta de detecção ao realizar a análise dos pacotes capturados. Independentemente do modo de instalação, o que define um IDS como um NIDS é o fato de seu componente sensor ser responsável pela coleta dos pacotes de rede, utilizando essas informações para identificar comportamentos anômalos que possam sinalizar um ataque ou uma atividade maliciosa. Assim, o NIDS se destaca por ser uma ferramenta essencial na proteção de redes ao monitorar continuamente o tráfego e, com base em suas análises, identificar tentativas de comprometimento da segurança.

2.4 DESAFIOS DA APRENDIZAGEM DE MÁQUINA PARA A INFRAESTRUTURA COMO CÓDIGO

Na construção de um sistema de detecção de intrusão baseado em aprendizado de máquina, a seleção e o tratamento dos dados desempenham um papel fundamental. Segundo Zhang *et al*. (2022), uma abordagem amplamente adotada envolve, primeiramente, a escolha do conjunto de dados que será utilizado pelo modelo. Devido à complexidade inerente aos dados de detecção de intrusão em redes e à ampla variação na magnitude dos atributos entre diferentes características, o processo de pré-processamento dos dados é essencial para garantir a qualidade do modelo. Esse pré-processamento geralmente inclui a engenharia de recursos, que abrange a extração e a seleção das características mais relevantes.

Após a construção de um conjunto de dados adequado, o próximo passo descrito por Zhang *et al*. (2022) envolve a escolha de um algoritmo de AM apropriado, que será utilizado para construir o modelo de classificação. Esse modelo é, então, treinado e validado com o objetivo de garantir sua precisão na detecção de intrusões, equilibrando a complexidade do ambiente com a capacidade de identificar ameaças reais.

Esse processo estruturado permite que o sistema se adapte às especificidades de diferentes ambientes de rede, oferecendo uma abordagem eficaz para a detecção de intrusões. Além disso, a complexidade dos ambientes da IaC introduz obstáculos relacionados à diversidade e ao volume dos dados. Uma vez que a origem pode ser de diferentes fontes, como arquivos de configuração estruturados e logs não estruturados, exigindo técnicas avançadas de pré-processamento e extração de características. Modelos robustos precisam ser capazes de lidar com dados incompletos e inconsistentes, típicos em infraestruturas automatizadas.

A segurança é outro aspecto crítico. As infraestruturas automatizadas ampliam a superfície de ataque, e os modelos de AM devem ser capazes de detectar anomalias e resistir a ataques adversários, nos quais invasores tentam manipular a configuração da infraestrutura para evitar a detecção. O modelo de AM precisa ser resiliente e adaptável a essas ameaças sofisticadas, como discutido por (ALZOUBI, Y.I. *et al*., 2024). Assim, é necessário atingir cenários de reconfigurações de regras e validações de segurança.

CAPÍTULO 3

TRABALHOS RELACIONADOS

Neste capítulo, são descritos alguns artigos relacionados aos temas necessários para a compreensão deste trabalho, o quais estão separados por subseções, sendo 3.1 Infraestrutura como Código, 3.2 Sistema de Detecção de Intrusão e Aprendizado de Máquina e 3.3 Considerações Finais.

3.1 INFRAESTRUTURA COMO CÓDIGO

O artigo de Abbas, Garg (2024) apresentou uma análise sobre a integração de tecnologias emergentes com Infrastructure as Code (IaC), destacando como a automação e padronização facilitam a agilidade e resiliência em ambientes distribuídos. A pesquisa apontou o impacto da IoT, edge computing, DevOps e machine learning, e como essas tecnologias permitiram flexibilidade para operar com requisitos dinâmicos e de baixa latência. Os resultados mostraram que IaC melhorou a consistência e segurança em operações distribuídas. A principal contribuição foi oferecer práticas para enfrentar barreiras de escalabilidade e segurança, sendo essencial para infraestruturas automatizadas.

O artigo de Mehdi, Walia (2023) mostrou um estudo sobre a adoção do Terraform, uma ferramenta da IaC que gerencia recursos em várias plataformas de nuvem. Utilizou-se uma abordagem declarativa com a HashiCorp Configuration Language (HCL), o Terraform destacou-se pela simplicidade e legibilidade. Esse estudo mostrou como o Terraform conecta facilmente a diversos serviços de nuvem, unificando a gestão de infraestrutura. Sua capacidade de simplificar a configuração e gerenciamento de recursos resultou em operações mais eficientes, economizando tempo e minimizando erros humanos, abordando a complexidade na gestão de infraestruturas distribuídas.

O artigo de Putra, Nurwa (2022) trouxe uma investigação sobre a implementação da IaC para automatizar a segurança e monitoramento de redes, especialmente devido ao aumento de crimes cibernéticos durante a pandemia de COVID-19. Usando ferramentas como Ansible, a pesquisa integra configuração e monitoramento de segurança com IDS, honeypots e SIEM. A

validação, por meio de teste de aceitação do usuário, mostrou uma taxa de aceitação de 81,05%, indicando a viabilidade e facilidade de uso. O estudo resolveu a necessidade de soluções de segurança eficientes em ambientes de trabalho remoto, destacando a importância de uma gestão de segurança robusta em um contexto dinâmico.

O artigo de Lopes *et al*. (2023) descreveu a criação de um laboratório virtual para experimentação com funções de cibersegurança em redes de coalizão, utilizando o conceito de Protected Core Networking (PCN). Por intermédio da IaC com Terraform e AWS, o estudo emulou redes multinacionais para investigar a reconfiguração automatizada de funções de segurança. Os resultados dos experimentos demonstram a viabilidade da automação em redes dinâmicas, com ênfase na segurança federada e políticas de controle transnacionais. A pesquisa solucionou o problema da gestão ágil e segura de redes multinacionais, sugerindo melhorias em cenários de roteamento baseado em riscos e monitoramento federado para incidentes de segurança.

3.2 SISTEMA DE DETECÇÃO DE INTRUSÃO E APRENDIZADO DE MÁQUINA

Em seu trabalho, Pathak e Shrivas (2024) exploraram o uso de várias técnicas de AM no desenvolvimento de IDS, utilizando o conjunto de dados CICIDS-2017 como base para avaliação. Eles propuseram um modelo de ensemble que combina os algoritmos Random Forest e Bagging Classifier, buscando melhorar a precisão na identificação de ataques em redes. O modelo alcançou uma taxa de acurácia de 99,85% ao empregar 77 atributos do dataset e superou a maioria dos classificadores individuais, como Decision Tree e KNN, exceto pelo Gaussian Naïve Bayes. A contribuição do estudo está em demonstrar como a combinação de algoritmos aumenta a eficiência de detecção, solucionando o problema de baixa precisão em classificações isoladas e sugerindo futuras melhorias, como a redução da dimensionalidade e otimizações no tempo de resposta.

O estudo de Horchulhack, Santin, Viegas (2024) propôs um método inovador para atualizar modelos de detecção de intrusões utilizando stream learning, buscando reduzir as instâncias necessárias para atualização e os custos computacionais. A técnica utiliza instâncias rejeitadas na classificação para atualizações incrementais e rotulagem automática de eventos a partir de repositórios públicos. Com base em um banco de dados de 2,6 TB, o modelo demonstrou manter alta acurácia por até três meses, reduzindo falsos positivos em 12% e rejeitando 8% das instâncias, consumindo apenas 3,2% do tempo de processamento, comparado

a métodos tradicionais. A solução abordou eficientemente o problema de rotulagem de eventos, reduzindo custos computacionais sem comprometer a precisão.

O trabalho de Das *et al*. (2022) apontou uma solução de segurança cibernética baseada em AM para a detecção de intrusões, utilizando um framework supervisionado combinado com técnicas de seleção de características em modelos ensemble. O estudo comparou diversos modelos de AM e métodos de seleção de características, buscando criar um mecanismo de detecção genérico que ofereça alta precisão e baixa taxa de falsos positivos. Com base em datasets como NSL-KDD, UNSW-NB15 e CICIDS-2017, a pesquisa alcançou uma taxa de detecção de 99,3% e FPR de 0,5%, superando outras soluções existentes. O trabalho resolveu a complicação de detectar ataques em evolução, melhorando a precisão e reduzindo o tempo de treinamento dos modelos.

O artigo de Bertoli *et al*. (2021) apresentou o AB-TRAP, um framework em cinco etapas para o desenvolvimento de IDS) adaptativos utilizando machine learning. O framework abrangeu desde a criação de datasets de tráfego atualizado até a implementação e avaliação de modelos de AM em redes locais e ambientes de internet. Os resultados indicaram alta precisão na detecção de ataques de varredura de portas TCP, com baixa sobrecarga computacional. O AB-TRAP ofereceu uma solução prática e adaptativa para proteger redes complexas, como IoT e sistemas ciberfísicos, abordando a evolução das ameaças e a necessidade de adaptação a diferentes ambientes.

A pesquisa de Viegas *et al*. (2018) propôs uma solução de detecção de intrusões baseada em AM para redes de alta velocidade, focando na classificação em tempo real de grandes volumes de tráfego e utilizou aprendizado incremental para ajustar os modelos conforme mudanças no comportamento do tráfego. O sistema, chamado BigFlow, foi projetado para escalar até 10 Gbps em um cluster de hardware comum. A principal inovação do BigFlow foi sua capacidade de rejeitar classificações de baixa confiança, reduzindo falsos positivos e permitindo a adaptação contínua do modelo, sem recomeçar o treinamento do zero. Testado em um dataset real de um ano de tráfego, o sistema demonstrou alta precisão e baixo custo de armazenamento e treinamento. Esse trabalho resolveu dificuldades de escalabilidade e adaptação a novos tipos de ataques, o que é importante para redes onde o tráfego evolui rapidamente.

O artigo de Molina-Coronado *et al*. (2020) trata-se de uma revisão de metodologias para NIDS com foco no processo de Knowledge Discovery in Databases (KDD). Ele enfatizou as fases de coleta, pré-processamento, transformação e mineração de dados, propondo uma nova

taxonomia para os métodos de detecção baseados em mineração de dados. O estudo destacou as limitações na análise de tráfego de rede e na evolução das ameaças, além de revisar os mecanismos de avaliação dos NIDS. Os resultados mostraram que as pesquisas anteriores frequentemente utilizam métricas inadequadas e carecem de mecanismos para atualizar detectores conforme surgem novas ameaças. Em suma, o trabalho abordou problemas de escalabilidade e adaptação de NIDS, sugerindo direções futuras para a pesquisa na área.

O artigo de Khalid, Abdullah, Sefer (2024) propôs uma abordagem híbrida para sistemas de detecção de intrusão, utilizando algoritmos de AM de máquina como Random Forest, Gradient Boosting Machines e Redes Neurais para melhorar a precisão e resiliência contra ataques avançados. Com uma taxa de detecção de intrusões de 96% usando o dataset CIC-IDS2017, a pesquisa destacou a escalabilidade e a resistência a ataques sofisticados. Além de abordar falhas em IDS tradicionais, especialmente na detecção baseada em assinaturas, o estudo propôs um modelo mais eficiente para identificar ameaças em tempo real e sugere técnicas de seleção de características e aprendizado profundo para lidar com grandes volumes de dados, melhorando a segurança das infraestruturas digitais.

O estudo de Akhtar *et al*. (2023) abordou a necessidade de aprimorar IDS enfrentando barreiras de precisão e taxa de falsos positivos diante da evolução das ameaças cibernéticas. A proposta envolveu o uso de algoritmos robustos de AM, como Random Forest, Gradient Boosting e um classificador genético, para melhorar a detecção de intrusões em redes. Usando um dataset de tráfego de rede, a abordagem atingiu alta precisão e baixa taxa de erros, superando técnicas tradicionais de IDS. A principal contribuição foi o desenvolvimento de um modelo robusto que combina técnicas de ensemble learning, otimizadas geneticamente, eficiente na gestão de ruídos e outliers nos dados de rede.

3.3 CONSIDERAÇÕES FINAIS

O desenvolvimento de um IDS adaptável para um ambiente da IaC específico representa um avanço na proteção de aplicações. Este trabalho integra um modelo de AM que identifica padrões anômalos em cenários de rede variáveis, ajustando-se continuamente às alterações da infraestrutura. A partir da análise realizada com base nos artigos elencados na Tabela 1, entende-se que, embora Abbas, Garg (2024) enfatizem a automação e padronização, sua pesquisa não abordou a criação de datasets exclusivos para cada ambiente, um aspecto que o trabalho supera, permitindo validações offline simplificadas.

O estudo de Putra, Nurwa (2022) discutiu a complexidade dos dados em IaC, mas não explorou as técnicas avançadas necessárias para lidar com dados incompletos e inconsistentes, uma lacuna que esta pesquisa busca preencher. Enquanto Pathak, Shrivas (2024) e Horchulhack, Santin, Viegas (2024) focam na precisão de algoritmos e eficiência dos modelos, suas abordagens carecem de uma adaptação contínua às mudanças nas topologias de rede.

Trabalhos como os de Das *et al*. (2022) e Bertoli *et al*. (2021) lidam com a evolução das ameaças, mas não propõem um mecanismo que ajuste dinamicamente as regras de detecção. A contribuição principal deste trabalho é fornecer um IDS que, além de detectar ameaças, ajusta suas configurações em resposta a novas intrusões, oferecendo uma abordagem que se adapte à segurança em ambientes da IaC, o que não é abordado de forma abrangente nas pesquisas existentes.

Tabela 1 – Trabalhos relacionados comparados ao trabalho proposto

AUTORES	**CONSIDERA IAC**	**IDS DINÂMICO**	**UTILIZA AM**	**CONSIDERA NOVOS CENÁRIOS**
Pathak, Shrivas, (2024)	Não	Não	Sim	Não
Horchulhack, Santin, Viegas, (2024)	Não	Não	Sim	Sim
Das *et al*., (2022)	Não	Não	Sim	Não
Bertoli *et al*., (2021)	Não	Não	Sim	Sim
Viegas *et al*., (2018)	Não	Não	Sim	Sim
Abbas, Garg, (2024)	Sim	Não	Sim	Sim
Mehdi, Walia, (2023)	Sim	Não	Não	Não
Putra, Nurwa, (2022)	Sim	Não	Não	Sim
Lopes *et al*., (2023)	Sim	Não	Não	Sim
Molina-Coronado *et al*., (2020)	Não	Não	Sim	Sim
Khalid, Abdullah, Sefer, (2024)	Não	Não	Sim	Sim
Akhtar *et al*., (2023)	Não	Não	Sim	Sim
Presente proposta	Sim	Sim	Sim	Sim

Fonte: o autor, 2024.

CAPÍTULO 4

PROPOSTA DE UM MODELO DE CLASSIFICAÇÃO DINÂMICA

Para abordar os resultados da generalização causados pelo comportamento dinâmico de infraestruturas implantadas por IaC em NIDS baseados em AM, é proposto um modelo de classificação dinâmica implementado por meio de um modelo de seleção de recursos. A operação do modelo é ilustrada na Figura 4 e é implementada em duas fases: Seleção de Recursos Multiobjetivos e Classificação Dinâmica. A separação em duas fases permite ao modelo melhorar sua capacidade de adaptação a mudanças, reduzindo falsos positivos e falsos negativos e garantindo uma detecção mais eficiente de ameaças em ambientes dinâmicos. A Seleção de Recursos Multiobjetivos visa encontrar um espaço de recursos que otimize a precisão do classificador e melhore a generalização.

Para isso, enquadra-se a seleção de recursos como uma tarefa de otimização multiobjetivo, na qual se espera que o classificador aumente a precisão ao antecipar o comportamento do ambiente IaC, ao mesmo tempo em que aprimora a generalização. Na prática, medimos a generalização avaliando a precisão da detecção de eventos de rede não vistos durante a fase de treinamento, uma complicação comum em infraestruturas implantadas por IaC. A premissa central é aprimorar a generalização da classificação durante a fase de treinamento como uma tarefa de seleção de recursos. A generalização é alcançada por meio da seleção otimizada de recursos, garantindo que o modelo aprenda a identificar padrões essenciais do ambiente IaC, sem ajustar-se firmemente a um conjunto específico de dados, permitindo a detecção de novas variações de tráfego.

Isso resulta em um classificador capaz de abordar o comportamento não estacionário inerente às infraestruturas implantadas por IaC. O objetivo da Classificação Dinâmica é lidar com a classificação de novos comportamentos de ambiente durante a fase de inferência em infraestruturas implantadas por IaC. Para alcançar isso, propomos uma abordagem de seleção de classificador dinâmico, em que o subconjunto de classificadores usados para inferência é ativamente escolhido com base no comportamento atual do ambiente.

Isso permite que o modelo detecte novos comportamentos gerados pela natureza dinâmica dos scripts da IaC quando implantados em produção. A dinâmica refere-se à capacidade dos scripts de modificar continuamente a infraestrutura, impactando o tráfego de rede e exigindo que o NIDS seja adaptável para detectar novos comportamentos sem comprometer sua precisão. Como resultado, a abordagem não apenas aprimora a generalização, impulsionada pelo processo de seleção de recursos, mas também melhora a detecção de novos comportamentos de tráfego de rede. Isso abre caminho para a implementação de NIDS baseados em AM nas infraestruturas implantadas por IaC nos ambientes de produção. As subseções a seguir descrevem melhor a implementação desta proposta, incluindo os módulos que a compõem.

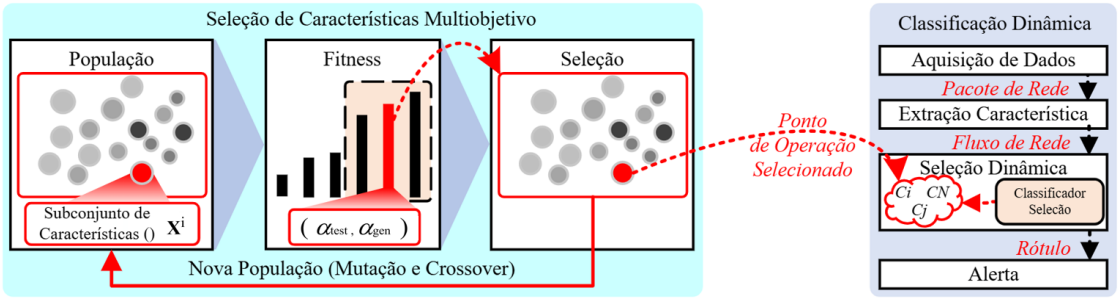
4.1 CLASSIFICAÇÃO DINÂMICA

Implementar um NIDS baseado em AM confiável nas infraestruturas implantadas por IaC é desafiador devido à natureza dinâmica do ambiente resultante (ALKHAFAJI, N., VIANA, T., AL-SHERBAZ, 2024). Isso apresenta dificuldades para NIDS tradicionais baseados em AM, que não são bem adaptados para lidar com o comportamento não estacionário do tráfego de rede. Para resolver isso, o modelo proposto enquadra a inferência como uma tarefa de classificação dinâmica. O objetivo é selecionar ativamente o subconjunto de classificadores mais adequados para classificar o comportamento atual do ambiente.

A visão geral da classificação é ilustrada na Figura 4 e segue uma operação tradicional de NIDS baseado em AM. Ela começa com a coleta de pacotes de rede pelo módulo de Aquisição de Dados. O comportamento dos pacotes de rede coletados é então extraído por um módulo de Extração de Características, gerando um fluxo de rede. Esse fluxo de rede é classificado por um módulo de Seleção Dinâmica, que gera alertas conforme necessário.

Para realizar a classificação, enquadra-se isso como uma tarefa de seleção dinâmica. Seja $x \in \mathrm{R}^{D}$ um vetor de características de dimensão D e $y \in \{n, a\}$, onde n representa um evento normal e a representa eventos rotulados como ataque. O modelo visa aprender um conjunto dinâmico de classificadores, cada um modelando a distribuição probabilística preditiva $p\ (y|x)$ sobre os rótulos verdadeiros. Para alcançar esse objetivo, construímos um conjunto de classificadores de tamanho N (Figura 4, Ci).

Figura 4 – Visão geral do modelo proposto de seleção dinâmica de classificadores para infraestruturas implantadas com IaC



Fonte: o autor, 2024.

A seleção de características multiobjetivo visa aumentar as capacidades de generalização dos classificadores resultantes. A classificação dinâmica seleciona ativamente o subconjunto de classificadores para abordar a detecção de novidades na fase de implantação da IaC. O classificador dinâmico, então, encontra um subconjunto de classificadores do conjunto que melhor se adapta à tarefa de classificação com base no evento de rede fornecido x. Para atingir esse objetivo constrói-se uma região de competência D a partir de um conjunto de testes rotulado.

A região de competência compreende o conjunto de eventos β do conjunto de testes que foram corretamente classificados por cada classificador do conjunto. Na prática, D representa eventos corretamente classificados por cada classificador. Na fase de inferência, o objetivo do classificador dinâmico é encontrar um subconjunto de classificadores K, de forma que os eventos da região de competência D sejam semelhantes ao evento a ser classificado x. Assim, identifica-se o subconjunto de classificadores com base na Equação 1:

(1)

$$\min_{\beta \in \mathcal{D}} \mathrm{k} \sqrt{\sum_{i=1}^{n} (\beta_i - x_i)^2}$$

onde k denota o número de classificadores a serem selecionados, β o evento da região de competência D, x o evento atual do ambiente e n o número de características. Como resultado, k classificadores são selecionados com base na menor distância Euclidiana em relação ao evento atualmente avaliado. Lembrando que a região de competência D contém eventos

corretamente classificados. Assim, selecionam-se ativamente os classificadores com base em uma medida de similaridade com essas classificações corretas anteriores.

Algoritmo 1 – Representação Algoritmo Genético

Algoritmo 1 Algoritmo Genético para resolver Eq. 4.

Inicialize a população P com soluções aleatórias
Avalie a aptidão de P com base na Eq. 2 e Eq. 3
while a condição de parada não for atendida **do**
    Selecione os pais de P com base na aptidão (Eq. 2 e Eq. 3)
    $child \leftarrow Crossover$(pais)
    Mutação($child$, $Mrate$)
    Anexe $child$ à descendência
    Avalie a aptidão da descendência (Eq. 2 e Eq. 3)
    Substitua as soluções menos aptas em P pela descendência
    Atualize $best\_solution$ se uma melhor for encontrada
end while

Fonte: o autor, 2024.

Algoritmo 2 – Representação Seleção Dinâmica de Classificadores – Fase de Inferência

Algoritmo 2 Seleção Dinâmica de Classificadores - Fase de Inferência

Require:
Vetor de características x ▷ Evento a ser classificado
Região de competência $\mathcal{D}$ ▷ Região de competência de eventos previamente classificados corretamente
Conjunto $\{C_1, C_2, \ldots, C_N\}$ ▷ Conjunto de classificadores
Número de classificadores a serem selecionados k
procedure ClassificadorDinâmico(x, $\mathcal{D}$, $\{C_1, C_2, \ldots, C_N\}$, k)
    Inicialize uma lista vazia $\mathcal{S}$ para armazenar as distâncias de cada classificador
    for cada evento x_i in $\mathcal{D}$ **do** ▷ Calcular a distância do evento à região de competência
        Calcule a distância d from x_i para x
        Adicione d to $\mathcal{S}$
    end for
    Classifique os classificadores $\{C_1, \ldots, C_N\}$ com base em suas menores distâncias em $\mathcal{S}$
    Return top-k classificadores $\{C_1, C_2, \ldots, C_k\}$ com as menores distâncias ▷ Selecionar classificadores mais próximos ao evento
end procedure

Fonte: o autor, 2024.

4.2 SELEÇÃO DE CARACTERÍSTICAS MULTIOBJETIVO

NIDS baseados em AM projetados para infraestruturas provisionadas por IaC devem generalizar o comportamento do ambiente de treinamento. Esse requisito decorre da variabilidade nas infraestruturas implantadas por IaC, que mudam conforme o script de configuração fornecido. Essa variabilidade apresenta uma tarefa árdua para a implementação eficaz desses modelos em configurações dinâmicas. Para enfrentar isso, aborda-se a fase de

treinamento de Classificação Dinâmica como uma tarefa de seleção de características multiobjetivo, conforme ilustrado na Figura 4.

Na prática, o objetivo é otimizar a precisão da classificação no ambiente de treinamento e considerar um subconjunto de atividades normais e de ataque não encontradas durante a fase de treinamento. Para alcançar esse objetivo, a tarefa de seleção de características multiobjetivo visa encontrar um espaço de características que minimize a taxa de erro no ambiente de treinamento por meio da Equação 2:

(2)

$$\alpha_{test}(h, x_i, \mathcal{D}_{test}) = erro(h(x_i, \mathcal{D}_{test}))$$

onde erro denota uma função que mede a taxa de erro do sistema de classificação *h*, usando um espaço de características xi, em um conjunto de teste D$test$. Aqui, o conjunto de teste D$test$ contém um conjunto de amostras com um comportamento semelhante ao inicialmente esperado da infraestrutura IaC a ser implantada. Portanto, ele inclui o conjunto de amostras normais e de ataque que se espera serem geradas a partir do script de configuração inicial da IaC. Para medir as capacidades de generalização do modelo projetado, utiliza-se a Equação 3:

(3)

$$\alpha_{gen}(h, x_i, \mathcal{D}_{gen}) = erro(h(x_i, \mathcal{D}_{gen}))$$

onde erro também denota uma função que mede a taxa de erro do sistema de classificação *h*, usando um espaço de características xi, em um conjunto de teste Dgen. Em contraste, Dgen contém um conjunto de amostras normais e de ataque geradas usando novos serviços e variantes de ataque. Como resultado, ele inclui amostras que não se espera serem geradas a partir da versão atual do script IaC. Assim, essa abordagem de seleção de características multiobjetivo visa resolver a Equação 4:

(4)

$$\arg\min_{x_0,\ldots,x_n} \alpha_{test}(h, x_i, \mathcal{D}_{test})$$

$$e$$

$$\arg\min_{x_0,\ldots,x_n} \alpha_{gen}(h, x_i, \mathcal{D}_{gen})$$

onde $\{x0, ..., xn\}$ denota todas as variações de espaços de características, h o modelo de classificação dinâmica, $atest$ o objetivo de erro de teste, detalhado na Equação 2, e $agen$ o objetivo de erro de generalização detalhado na Equação 3. Como resultado, o objetivo é encontrar um espaço de características que reduza simultaneamente a taxa de erro do classificador dinâmico para eventos já vistos ($atest$) e para novos eventos ($agen$).

Dado que buscar em todo o espaço de características não é viável, se resolve a Equação 4 por meio de uma implementação de busca genética. O Algoritmo 1 apresenta uma visão geral da implementação de busca genética neste modelo proposto. Ele começa iniciando um conjunto de população P com espaços de características aleatórios e calcula sua aptidão usando as Equações 2 e 3. Em seguida, prossegue para selecionar os indivíduos mais aptos com base em sua aptidão associada para realizar o cruzamento e mutação, gerando uma nova população. Os indivíduos menos aptos são substituídos pela nova descendência, e o procedimento é repetido por N gerações.

4.3 DISCUSSÃO

Lidar com o comportamento dinâmico de ambientes implantados por IaC é uma tarefa árdua para os NIDS baseados em AM atuais. As dificuldades identificadas durante o estudo foram o comportamento dinâmico dos ambientes IaC, a necessidade de seleção dinâmica de classificadores e o ajuste do módulo de classificação ao ambiente atual. Em comparação com dificuldades anteriores, o modelo apresenta elementos semelhantes aos já mencionados, mas enfatiza a seleção dinâmica de classificadores, que não foi destacada diretamente nessas ações anteriores. Esse aspecto reforça a necessidade de um NIDS adaptável e que ajuste seu comportamento conforme o ambiente IaC evolui. À luz disso, este modelo proposto estrutura a detecção de intrusões como uma seleção dinâmica de classificadores construída por meio de uma operação de seleção de características. O primeiro objetivo é selecionar ativamente o subconjunto de classificadores que deve ser utilizado na tarefa de classificação, ajustando o comportamento do módulo de classificação com base no comportamento atual do ambiente implantado por IaC.

O segundo objetivo é selecionar o subconjunto de características que minimize simultaneamente a taxa de erro do classificador dinâmico no conjunto de teste e no conjunto de generalização (Equação 4) . Assim, melhora-se a capacidade de generalização do modelo, considerando as variações no script IaC utilizado. Como resultado, este modelo proposto abre caminho para a implementação confiável de NIDS baseados em AM nas infraestruturas IaC.

CAPÍTULO 5

PROTÓTIPO E TESTBED

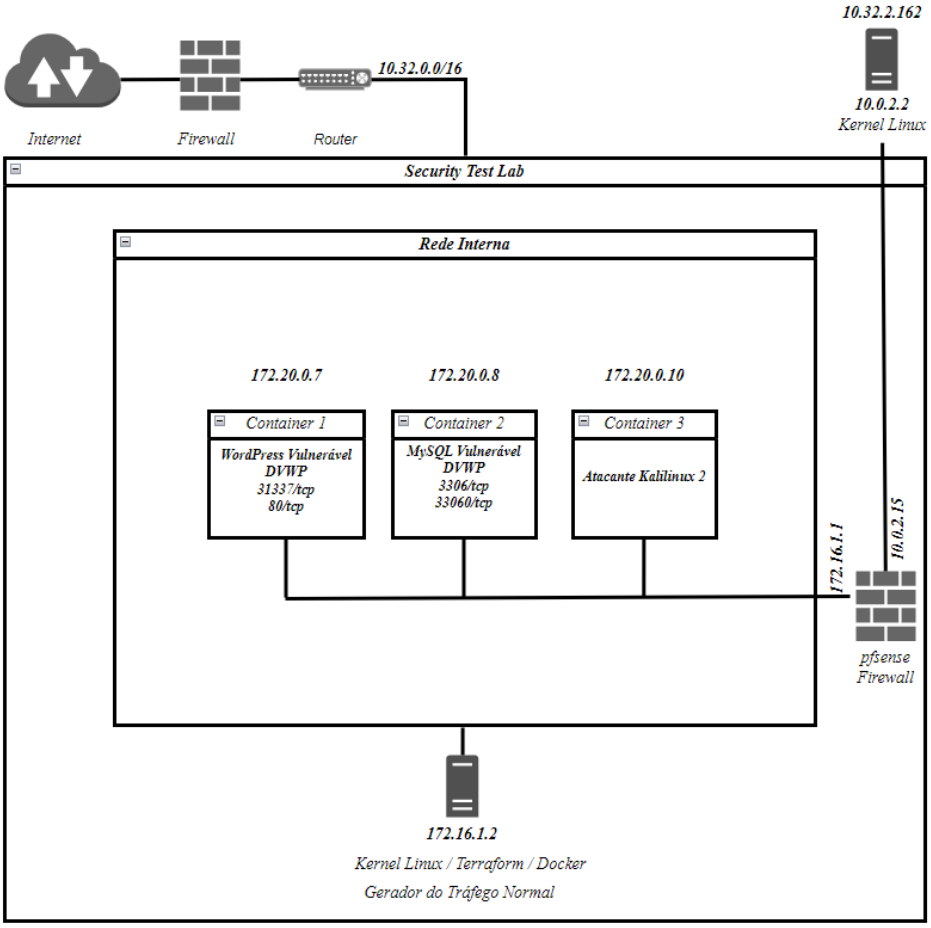
Neste capítulo, é descrito o cenário utilizado para implementar uma IaC com foco na criação de um testbed destinado a um IDS baseado em AM, descrevendo a arquitetura, o cenário de validação, as técnicas de captura de tráfego e o processo de geração do dataset. A abordagem visa evitar possíveis sobreajustes, empregando uma infraestrutura para treino/teste e outra distinta para validação, garantindo assim que o modelo aprenda padrões gerais. No testbed, são simulados cenários controlados para o treinamento, enquanto a infraestrutura de validação utiliza um ambiente diferente e mais realista, assegurando maior robustez do IDS. Por fim, definem-se serviços mínimos (web e banco de dados) no testbed e serviços mais complexos (DNS, HTTP, SMTP, SNMP, SSH, Kubernetes e Docker) no ambiente de validação.

5.1 ARQUITETURA DO TESTBED

A arquitetura de rede escolhida como base para o testbed segue um modelo tradicional, composto por um cliente que acessa um serviço web, sustentado por um banco de dados e por mecanismos de segurança, como um firewall. A diferença significativa nesse cenário é a ênfase na detecção de ameaças internas, que são notoriamente mais difíceis de identificar por já possuírem acesso à rede. Essa representação está disposta na Figura 5. A proposta substitui a proteção perimetral convencional por um IDS distribuído, implementado em cada nó da arquitetura (por exemplo, o servidor web e o banco de dados).

Essa abordagem visa monitorar e identificar padrões anômalos que possam indicar atividades maliciosas, comuns a ataques internos, como roubo de dados ou atividades de sabotagem. Dessa forma, o IDS proposto cobre potenciais brechas de segurança que não seriam identificadas por sistemas baseados em perímetro, garantindo uma defesa mais eficaz contra ameaças complexas e internas.

Figura 5 – Arquitetura de rede tradicional para o testbed, composta por cliente, serviço web, banco de dados e mecanismos de segurança como firewall



Fonte: o autor, 2024.

5.2 CENÁRIO DE VALIDAÇÃO

Para que este trabalho fosse conduzido, foi necessário configurar um cenário de validação e testes para garantir que as condições refletissem o comportamento dinâmico esperado. A arquitetura do ambiente de validação foi estruturada em três etapas principais: inicialmente, as aplicações e o atacante foram implantados, na sequência, simulou-se o tráfego normal para estabelecer um fluxo de operações legítimas e, finalmente, procedeu-se à captura dos dados de rede para análise detalhada. A escolha de uma versão vulnerável do WordPress permitiu a criação de um ambiente controlado, onde foram experimentadas vulnerabilidades específicas, que incluíram a CVE-2020-8771, que explora falhas de autenticação no plugin Time Capsule; a CVE-2020-8772, com ausência de verificação de autorização no plugin

InfiniteWP Client; a CVE-2020-10564, que explora o componente directory traversal no plugin File Upload; além da CVE-2019-9978, uma falha de XSS no plugin Social Warfare.

A experimentação dessas e de outras falhas de segurança descritas resultou em um conjunto de dados que possibilitou a investigação de padrões de tráfego malicioso e a avaliação da eficácia do IDS em detectar os comportamentos de ataque. Com isso, o trabalho permitiu validar a capacidade do IDS em adaptar-se dinamicamente e identificar ataques específicos em uma infraestrutura vulnerável, abordando o problema de proteção eficiente para IaC, que apresenta variações únicas devido à sua natureza mutável e configurável.

Na configuração inicial do ambiente de testes, foram implantadas aplicações essenciais para simular um cenário real, com serviços distribuídos utilizando IaC. As aplicações WordPress, executando em servidor web Apache e Banco de Dados MySQL, que foram configurados para reproduzir um ambiente típico de produção, garantindo que as solicitações HTTP/HTTPS fossem corretamente processadas, e que o armazenamento e gerenciamento de dados ocorressem de forma integrada e eficiente. O uso de Terraform e Docker para o realizar o deploy dessas aplicações assegurou consistência e facilidade na replicação do ambiente.

Em paralelo, um ambiente controlado de ataque foi configurado usando uma máquina Kali Linux. Essa máquina dispõe de ferramentas para realizar ataques cibernéticos, permitindo explorar vulnerabilidades e gerar tráfego malicioso de modo a avaliar a resposta do ambiente às tentativas de intrusão. A estrutura incluiu um firewall pfSense para controle adicional de acesso, fornecendo uma rede isolada, onde o deploy das aplicações e do atacante pôde ser gerenciado sem impactar o ambiente acadêmico real.

O ambiente de teste isolado solucionou a necessidade de proteção da infraestrutura de produção contra riscos que poderiam comprometer a segurança dos dados do ambiente real, permitindo uma validação de segurança controlada e reproduzível. A resume a configuração e o respectivo propósito, para cada um dos componentes contidos na arquitetura.

Tabela 2 – Componentes do ambiente virtual: contêineres e máquinas virtuais, com a virtualização sendo suportada por tecnologias como Docker, Oracle VirtualBox e Microsoft Hyper-V

COMPONENTE	TIPO DE SERVIÇO	VIRTUALIZADOR
Contêiner 1	Servidor Web	Docker
Contêiner 2	Banco de Dados	Docker
Contêiner 3	Teste de Penetração	Docker
pfSense	Firewall	Oracle VirtualBox
Ubuntu Server 1	Terraform/Docker Server	Oracle VirtualBox
Ubuntu Server 2	Servidor Web/Virtualizador	Microsoft Hyper-V

Fonte: o autor, 2024.

A carga de trabalho, ou workload, nesse ambiente foi planejada para simular o comportamento de um sistema corporativo em produção, incluindo interações reais de usuários e solicitações de rede, a fim de testar o funcionamento esperado das aplicações implantadas. Compreendeu um fluxo constante de requisições ao WordPress, consultas ao banco de dados MySQL e processamento de dados pelo servidor web Apache, garantindo uma representação do volume e da intensidade de operações típicas de um ambiente distribuído.

Para refletir as condições de carga reais, o Kali Linux foi utilizado para gerar tentativas de exploração e tráfego malicioso controlado, o que adicionou complexidade ao workload e permitiu uma avaliação robusta das respostas de segurança. Esse cenário promoveu a simulação de uma carga mista, onde solicitações legítimas coexistiam com tentativas de ataque, possibilitando a análise do comportamento das aplicações e dos sistemas de defesa. Esse workload simulou efetivamente o cenário de operação de um sistema corporativo, oferecendo uma base sólida para avaliar o impacto de diferentes tipos de interação e validar a robustez do ambiente de segurança configurado.

Com o intuito de analisar o comportamento da rede e avaliar a eficácia dos mecanismos de segurança implementados, foi realizada uma captura detalhada do tráfego no testbed da arquitetura proposta. Esse procedimento envolveu a execução de 12 tipos distintos de ataques e varreduras controladas, visando gerar tráfegos específicos para uma análise precisa. Inicialmente, medidas preventivas foram adotadas para proteger a integridade dos dados e do ambiente de teste. Arquivos críticos foram copiados para locais seguros fora das máquinas utilizadas, eliminando o risco de perda de informações relevantes.

Além disso, as máquinas que executavam bots foram reiniciadas, assegurando que quaisquer atividades prévias não interferissem nos novos experimentos. Para cada tipo de ataque ou varredura, foi efetuada uma nova captura de tráfego, armazenada em um arquivo PCAP nomeado conforme o teste em execução. Essa segmentação facilitou a organização e a análise posterior dos dados coletados. As capturas foram realizadas utilizando o comando do tcpdump, *tcpdump -i nome-da-interface -w nome-do-arquivo.pcap*, ferramenta reconhecida pela precisão na monitoração de tráfegos de rede.

Os testes foram conduzidos individualmente, permitindo controle rigoroso sobre cada atividade e evitando sobreposições que pudessem comprometer a qualidade dos dados. Adicionalmente, outliers foram removidos dos arquivos PCAP de ataque para garantir que a análise estatística dos dados fosse precisa e representativa do tráfego gerado pelas atividades específicas.

A lista de ataques e varreduras executados inclui:

- Varredura SYN: identificação de portas abertas por meio do envio de pacotes SYN;
- Varredura Full Connect (Nmap): estabelecimento completo de conexões TCP para detecção de serviços ativos;
- Varredura Stealth (Nmap): técnicas que evitam o estabelecimento completo de conexões, reduzindo a possibilidade de detecção;
- Varredura ACK (Nmap): mapeamento de regras de firewall para determinar portas filtradas;
- Varredura UDP (Nmap): identificação de serviços ativos em portas UDP;
- Descoberta de Host com ICMP Echo (Nmap): detecção de hosts ativos por meio de pacotes ICMP Echo Request;
- Varredura de Vulnerabilidades (Nmap): busca por vulnerabilidades conhecidas em serviços e aplicações;
- Ataque de Negação de Serviço (DoS/DDoS) (Pyflooder): simulação de ataques para sobrecarregar os recursos do sistema;
- Ataques de Força Bruta (Hydra): tentativas de acesso não autorizado a serviços como MySQL e SSH, utilizando múltiplas combinações de credenciais;
- Brute Force de Diretório (Dirb): descoberta de diretórios e arquivos ocultos em servidores web;

- Scan de Vulnerabilidades CMS (WordPress) (Wpscan): identificação de falhas de segurança em sites baseados no WordPress; e
- Scan de Vulnerabilidades Web (Nikto e Wapiti): varredura de aplicações web em busca de vulnerabilidades comuns.

Durante cada teste, o tráfego de rede foi monitorado diretamente no ambiente de execução e registrado em arquivos PCAP específicos. A captura era interrompida após a conclusão de cada atividade para assegurar que apenas os dados relevantes fossem armazenados no arquivo PCAP correspondente. Para análise posterior, utilizou-se o comando *tcpdump -r nome-do-arquivo.pcap | grep endereco-ip-do-alvo-do-ataque* para ler e filtrar pacotes específicos que continham o endereço IP do alvo. Essa metodologia permitiu coletar o tráfego de rede de forma estruturada e abrangente, fundamental para o estudo das interações entre os ataques e os mecanismos de defesa no ambiente proposto.

A remoção dos outliers nos PCAP de ataque garantiu a consistência e precisão dos dados analisados. A aplicação de técnicas e ferramentas consagradas na área de segurança da informação assegurou a relevância e a validade dos resultados obtidos. A captura eficaz de dados é fundamental para o sucesso de um NIDS baseado em AM, pois determina a qualidade e a representatividade dos dados utilizados para treinar os algoritmos. Para criar um ambiente de rede que refletisse um cenário corporativo realista, implementa-se um ambiente controlado com três contêineres executando um sistema web que recebia operações legítimas.

Aproximadamente 100 bots foram programados para realizar ações legítimas aleatórias no sistema web em intervalos curtos, aumentando o volume de dados e introduzindo variabilidade no tráfego. Simultaneamente, realizam-se ataques de força bruta no ambiente para coletar dados de eventos maliciosos em meio ao tráfego normal. Utilizam-se ferramentas de captura de pacotes para registrar todo o tráfego de rede, gerando arquivos PCAP com eventos legítimos coletados ao longo de 12 horas e eventos de ataque incorporando os ataques simulados, garantindo a sincronização temporal dos eventos capturados.

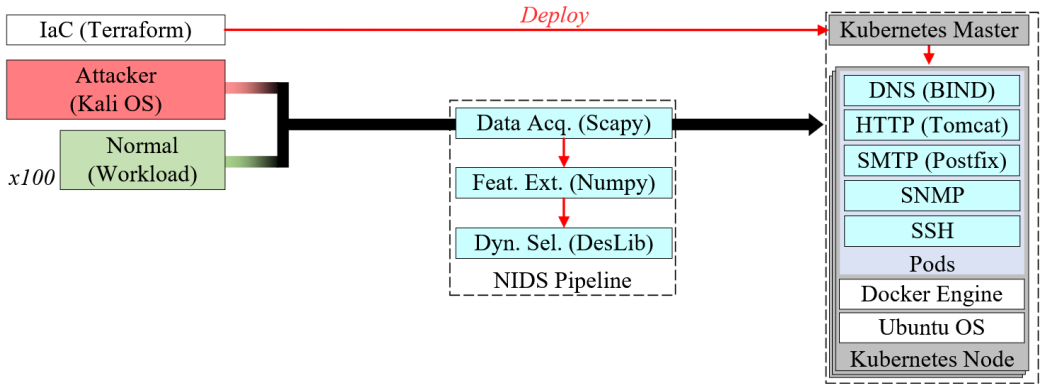
Após a captura inicial, aplicam-se técnicas de pré-processamento e filtragem de dados para preparar os dados para a extração de características. O pacotes irrelevantes para a detecção de ataques de força bruta foram removidos, focando em protocolos e portas específicos que são comumente alvos desse tipo de ataque. Também se balanceia o conjunto de dados para equilibrar a proporção entre eventos legítimos e maliciosos, evitando que o algoritmo desenvolva um viés durante o treinamento. Utiliza-se, então, o Packet Trait Analyzer, uma

ferramenta desenvolvida para extrair características significativas dos eventos de rede capturados.

As técnicas de extração incluíram análise de fluxos de rede, cálculo de estatísticas temporais e identificação de padrões de requisição, visando detectar indicadores potenciais de ataques de força bruta. Para assegurar a eficácia das técnicas de captura, realizam-se testes de validação. Aplicam-se as técnicas de captura em um ambiente de rede operacional para verificar se os dados coletados eram representativos e úteis para a detecção de intrusões. Avaliam-se as taxas de falsos positivos e negativos gerados pelo sistema, ajustando os parâmetros de captura e filtragem para otimizar a precisão.

As técnicas de captura foram refinadas continuamente com base no *feedback* dos testes, garantindo que o sistema permanecesse eficaz diante de novas ameaças e padrões de tráfego.

Figura 6 – Visão geral da implementação do protótipo em uma infraestrutura de validação



Fonte: o autor, 2024.

Implementa-se um protótipo da proposta em uma infraestrutura provisionada por IaC, executada em um ambiente privado. A Figura 6 apresenta uma visão geral da implementação do modelo proposto. Considera-se um ambiente da IaC implementado por meio do Terraform (HASHICORP, 2024), que implanta scripts gerados em um cluster Kubernetes (KUBERNETES, 2024) v.1.31. O Kubernetes implanta os scripts IaC associados como Pods, utilizando o Docker (DOCKER, 2024) v.24.0. Consideram-se cinco serviços normais diferentes, descritos a seguir:

- DNS: um servidor de Serviço de Nomes de Domínio (DNS) implementado por meio do BIND v9.11. O tráfego de rede normal é gerado por meio de consultas de resolução de nomes executadas em direção ao servidor.

- HTTP: um servidor web implementado por meio do Apache Tomcat v11.0, hospedando os 500 principais sites listados pela Alexa. O tráfego de rede normal é gerado por uma carga de trabalho que consulta aleatoriamente os sites hospedados.
- SMTP: um servidor de e-mail implementado por meio do Postfix v3.9. O tráfego de rede normal é gerado enviando e-mails de tamanhos aleatórios, variando de 100 a 1.000 bytes.
- SNMP: um servidor de Protocolo Simples de Gerenciamento de Rede (SNMP) hospedado no Ubuntu. O tráfego de rede normal é gerado consultando aleatoriamente a árvore MIB.
- SSH: um servidor de Shell Seguro (SSH) implementado por meio do openssh-server v9.3 em um contêiner Ubuntu 22.04. O tráfego de rede normal é gerado executando aleatoriamente um comando de uma lista de 100 comandos em intervalos aleatórios.

Para gerar tráfego de rede normal realista, implantam-se 100 contêineres que executam cargas de trabalho de acordo com o serviço implementado, em intervalos aleatórios de 0 a 4 segundos de periodicidade. Também se gera 14 categorias de comportamentos de ataque por meio do Kali OS. Para a geração de ataques, variou-se a frequência e a taxa de transferência. Implementou-se o NIDS baseado em AM proposto (Figura 6) como um serviço isolado que continuamente avalia o tráfego de rede gerado neste testbed.

Nesse caso, o tráfego de rede gerado é continuamente adquirido por um módulo de Aquisição de Dados implementado usando a API Scapy v2.6. O comportamento dos pacotes de rede coletados é extraído usando a API numpy v1.26. A Tabela 3 lista as características do fluxo de rede extraídas do protótipo. Extraímos 31 características considerando o agrupamento de IP e serviços em um intervalo de 60 segundos. As características do fluxo de rede extraídas são a entrada para o modelo de seleção dinâmica de classificadores implementado usando a API DesLib v0.4.

Tabela 3 – Conjunto de características extraídas no nível de rede em um intervalo de janela de tempo de 60 segundos para cada comunicação entre cliente e servidor

#	Característica	Descrição
1	Total Fwd Pkts	Número total de pacotes enviados do cliente para o servidor
2	Total Fwd Vol	Volume total de dados (em bytes) enviados do cliente para o servidor
3	Total Bwd Pkts	Número total de pacotes enviados do servidor para o cliente
4	Total Bwd Vol	Volume total de dados (em bytes) enviados do servidor para o cliente
5	Fwd Pkt Len Std	Desvio padrão do comprimento dos pacotes enviados do cliente para o servidor
6	Bwd Pkt Len Max	Comprimento máximo dos pacotes enviados do servidor para o cliente
7	Bwd Pkt Len Std	Desvio padrão do comprimento dos pacotes enviados do servidor para o cliente
8	Fwd IAT Mean	Tempo médio de intervalo entre pacotes enviados do cliente para o servidor (em milissegundos)
9	Fwd IAT Max	Tempo máximo de intervalo entre pacotes enviados do cliente para o servidor (em milissegundos)
10	Fwd IAT Std	Desvio padrão do tempo de intervalo entre pacotes enviados do cliente para o servidor (em milissegundos)
11	Bwd IAT Max	Tempo máximo de intervalo entre pacotes enviados do servidor para o cliente (em milissegundos)
12	Bwd IAT Std	Desvio padrão do tempo de intervalo entre pacotes enviados do servidor para o cliente (em milissegundos)
13	Duration	Duração total do fluxo de rede (em milissegundos)
14	Active Min	Tempo mínimo em que o fluxo esteve ativo (em milissegundos)
15	Active Mean	Tempo médio em que o fluxo esteve ativo (em milissegundos)
16	Active Max	Tempo máximo em que o fluxo esteve ativo (em milissegundos)
17	Active Std	Desvio padrão do tempo em que o fluxo esteve ativo (em milissegundos)
18	Idle Min	Tempo mínimo em que o fluxo esteve ocioso (em milissegundos)
19	Idle Mean	Tempo médio em que o fluxo esteve ocioso (em milissegundos)
20	Idle Max	Tempo máximo em que o fluxo esteve ocioso (em milissegundos)
21	Idle Std	Desvio padrão do tempo em que o fluxo esteve ocioso (em milissegundos)
22	SFlow Fwd Pkts	Número amostrado de pacotes enviados do cliente para o servidor
23	SFlow Fwd Bytes	Número amostrado de bytes enviados do cliente para o servidor
24	SFlow Bwd Pkts	Número amostrado de pacotes enviados do servidor para o cliente
25	SFlow Bwd Bytes	Número amostrado de bytes enviados do servidor para o cliente
26	FPSH Count	Número de flags PUSH em pacotes enviados do cliente para o servidor
27	BPSH Count	Número de flags PUSH em pacotes enviados do servidor para o cliente
28	FURG Count	Número de flags URGENT em pacotes enviados do cliente para o servidor
29	BURG Count	Número de flags URGENT em pacotes enviados do servidor para o cliente
30	Total FHLen	Comprimento total do cabeçalho dos pacotes enviados do cliente para o servidor
31	Total BHLen	Comprimento total do cabeçalho dos pacotes enviados do servidor para o cliente

Fonte: o autor, 2024.

A Tabela 4 apresenta as estatísticas do ambiente de teste gerado. Foram executadas 10 diferentes configurações de serviço, cada uma por 30 minutos, a fim de simular um cenário realista de configuração IaC. Na prática, o comportamento do ambiente, considerando tanto as variantes de ataque quanto as normais, varia a cada implantação, levando a uma representação realista das dinâmicas do ambiente implantado com IaC. Como resultado, o ambiente de teste gerado cria uma situação realista da IaC que pode variar o comportamento do ambiente com base na configuração do script utilizado.

Tabela 4 – Estatísticas do ambiente de teste gerado em relação aos fluxos de rede e pacotes

Amb.	Comportamento (Ferramenta)	Fluxos de Rede	Pacotes de Rede
D_{test}	HTTP	516.5k	5760
	SNMP	2.7k	5760
	ACK Scan	13.2M	26.4M
	DDoS	734.8k	66.7M
	ICMP Echo Discover	13.2M	26.4M
	UDP Scan	51.8k	98.1k
D_{gen}	DNS	49.1k	5760
	SMTP	286.8k	5760
	SSH	18.2k	5760
	Brute Force DIRB	481.5k	32.9M
	CMS Scan	1.1M	48.8M
	Full Connect Scan	13.2M	26.4M
	MySQL Brute Force	70.2k	39.5M
	Nikto	1.6M	111.3M
	Scan Vuln	13.7M	53.5M
	SSH	12.2k	472.5k
	Stealth Scan	13.2M	26.4M
	SYN Scan	13.1M	26.4M
	Wapiti	1.4k	83.9k

Fonte: o autor, 2024.

5.3 PROCESSO DE CRIAÇÃO E CARACTERIZAÇÃO DO DATASET

A proteção de infraestruturas provisionadas por IaC enfrenta barreiras significativas devido à natureza dinâmica das configurações que os scripts IaC trazem. Essas configurações estão em constante mudança, exigindo que as soluções de segurança, como os Sistemas de Detecção de Intrusão de Rede baseados em AM, se adaptem de forma eficaz a essas mudanças. No entanto, a automação proporcionada pelo IaC ainda não se estende plenamente às configurações de segurança, que frequentemente requerem intervenção manual em elementos como firewalls, controles de acesso e VLANs.

Essa discrepância cria uma lacuna entre o provisionamento automatizado e a segurança da infraestrutura, dificultando a resposta a preocupações de segurança em ambientes que evoluem rapidamente. A automatização das configurações de segurança com base no código IaC apresenta obstáculos, especialmente ao considerar soluções de NIDS. A maioria dos estudos existentes concentra-se em modelos de detecção de intrusão baseados em comportamento, que dependem da construção de modelos de AM treinados em ambientes estáticos.

Esses modelos assumem que a configuração da infraestrutura é conhecida antecipadamente e permanece inalterada, o que não é o caso em ambientes provisionados por IaC. Quando um novo script IaC modifica a infraestrutura, o NIDS baseado em AM

previamente projetado torna-se pouco confiável para implantação, pois não consegue lidar com as novas configurações e comportamentos resultantes. Considerar a dinâmica do ambiente durante o desenvolvimento de um NIDS baseado em AM é essencial, mas desafiador.

Modelos treinados em conjuntos de dados estáticos frequentemente não generalizam bem para cenários reais, enfrentando dificuldades para detectar comportamentos novos ou em evolução. Em ambientes tradicionais, mudanças no tráfego de rede podem levar a um desempenho degradado dos NIDS, exigindo atualizações constantes dos modelos. Entretanto, em infraestruturas implantadas com IaC, o comportamento do ambiente só é plenamente revelado após o provisionamento, tornando impraticável a atualização contínua dos conjuntos de dados de treinamento.

Diante dessas ações, o processo de criação e caracterização do dataset para o NIDS proposto envolve abordagens que visam melhorar a generalização e a adaptabilidade do modelo. Inicialmente, a seleção de características multiobjetivo é utilizada para identificar um subconjunto de atributos que simultaneamente aprimora a precisão e a capacidade de generalização do sistema. Essa estratégia considera, já na fase de construção do modelo, as potenciais mudanças comportamentais introduzidas pelos scripts IaC, permitindo que o NIDS seja treinado com uma visão mais abrangente dos possíveis cenários de tráfego.

Adicionalmente, a seleção dinâmica de classificadores é implementada para que o sistema possa escolher, de forma ativa, o conjunto mais apropriado de algoritmos de classificação com base no comportamento atual do ambiente IaC implantado. Essa adaptabilidade é fundamental para mitigar a degradação na precisão que ocorreria devido às mudanças nas configurações da infraestrutura. Ao ajustar dinamicamente os classificadores utilizados, o NIDS mantém sua eficácia, mesmo diante de comportamentos de rede não previstos durante o treinamento.

A caracterização do dataset, portanto, não se limita a um único conjunto de dados estático, mas engloba uma diversidade de configurações e comportamentos resultantes dos diferentes scripts IaC. Isso envolve a coleta e a análise de tráfego de rede sob diversas condições operacionais, capturando tanto as operações normais quanto as atividades potenciais de invasores. O dataset é construído para refletir a variabilidade intrínseca dos ambientes IaC, fornecendo ao modelo de AM uma base sólida para aprender e generalizar a partir de múltiplos cenários.

Ao final, o processo de criação e caracterização do dataset para o NIDS adaptado a infraestruturas provisionadas por IaC requer uma abordagem inovadora, que considera a

natureza não estacionária desses ambientes. Ao focar na seleção de características que promovem a generalização e na adaptabilidade dinâmica dos classificadores, o modelo proposto busca superar as limitações dos NIDS tradicionais. Dessa forma, é possível desenvolver uma solução de detecção de intrusão eficaz e confiável capaz de acompanhar as rápidas mudanças impostas pelos scripts de configuração em ambientes modernos de TI.

CAPÍTULO 6

RESULTADOS

Neste capítulo, a apresentação dos experimentos realizados tem como objetivo responder às seguintes Questões de Pesquisa (RQs):

- RQ1. Qual é o impacto das mudanças de comportamento em IaC na acurácia de NIDS tradicionais baseados em AM?
- RQ2. A técnica proposta de seleção de características melhora as capacidades de generalização do modelo?
- RQ3. Qual é o desempenho de acurácia do modelo proposto em ambientes provisionados por IaC?

As subseções a seguir descrevem detalhadamente o desempenho do modelo, incluindo os aspectos de construção do modelo e sua performance no novo conjunto de dados.

6.1 CONSTRUÇÃO DO MODELO

Avaliou-se a proposta de Seleção Dinâmica de Classificadores implementada com uma seleção dinâmica de classificadores (Algoritmo 2), especificamente o k-Nearest Oracle-Eliminate (KNORA-E). Foram utilizados 5 vizinhos para estimar a região de competência e o algoritmo k-Nearest Neighbor (kNN) para o cálculo da distância (Equação 1). Os parâmetros foram definidos empiricamente, e nenhuma influência significativa nos resultados foi observada ao variá-los. Os classificadores foram implementados utilizando a API DESLib v.0.4.

O desempenho do modelo proposto com classificadores tradicionais amplamente usados, baseados em ensembles de AM também foi comparado. Os classificadores de ensemble, especificamente Random Forest (RF) e Bagging (Bag), foram implementados com 100 árvores de decisão como base, utilizando o critério gini como métrica de qualidade de divisão dos nós. O classificador Decision Tree (DT) também utilizou o critério gini, sem limite de profundidade máxima. Por fim, o classificador Multilayer Perceptron (MLP) foi implementado com 256 neurônios ocultos, função de ativação Relu e otimizador adam.

Os classificadores tradicionais de ensemble foram implementados utilizando a API scikit-learn v0.24. O conjunto de dados original (Tabela 4) foi dividido em conjuntos de treinamento, teste e validação, cada um composto por 40%, 30% e 30%, respectivamente, de cada comportamento. Além disso, gerou-se dois conjuntos de dados distintos (Tabela 4) com base nessas divisões, conforme segue:

- Dtest: conjunto de dados usado para fins de treinamento. Contém tráfego de rede normal, de DNS, HTTP e SMTP, e tráfego de rede de ataque, como ACKScan, Bruteforce DIRB, CMSScan, DDoS e Full ConnectScan.
- Dgen: conjunto de dados usado para avaliar as capacidades de generalização. Contém tráfego de rede normal de SNMP e SSH e os ataques de rede restantes. Esses serviços e ataques não são usados no treinamento.

Portanto, todos os classificadores selecionados são treinados utilizando o conjunto de treinamento de Dtest, avaliados usando o conjunto de teste de Dtest e também, de Dgen. O objetivo é permitir a avaliação da capacidade de generalização do modelo para gerar um comportamento semelhante ao que seria evidenciado em infraestruturas provisionadas por IaC. Nesse caso, o NIDS baseado em AM é submetido a um ambiente de treinamento específico e deve ser capaz de generalizar o comportamento sem retreinamento, mesmo com variações no script IaC utilizado.

Foram avaliados os classificadores selecionados utilizando as seguintes métricas de desempenho de classificação:

- Verdadeiro Positivo (True Positive – TP): número de amostras de ataque corretamente classificadas como ataque.
- Verdadeiro Negativo (True Negative – TN): número de amostras normais corretamente classificadas como normais.
- Falso Positivo (False Positive – FP): número de amostras normais incorretamente classificadas como ataque.
- Falso Negativo (False Negative – FN): número de amostras de ataque incorretamente classificadas como normais.

A F-Measure foi calculada de acordo com a média harmônica dos valores de Precision e Recall, considerando amostras de ataque como positivas e amostras normais como negativas, conforme mostrado na Equação 5.

(5)

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F - Measure = 2 * \frac{Precision * Recall}{Precision + Recall}$$

6.2 NIDS BASEADOS EM AM NAS INFRAESTRUTURAS PROVISIONADAS POR IAC

O primeiro experimento tem como objetivo responder à RQ1 e investigar o desempenho em acurácia de NIDS tradicionais baseados em AM nas infraestruturas provisionadas por IaC. Para alcançar esse objetivo, os classificadores tradicionais de AM foram treinados usando o subconjunto Dtest e a avaliou-se seu desempenho em todo o conjunto de dados. A Tabela 5 apresenta o desempenho dos classificadores selecionados quando submetidos a variações em ambientes da IaC. É evidente que, embora todos os modelos mantenham altas taxas de detecção em ambientes semelhantes aos utilizados durante o treinamento (Dtest), seu desempenho cai significativamente ao serem avaliados em ambientes desconhecidos com novos serviços e padrões de ataque (Dgen).

Tabela 5 – Acurácia de classificação dos classificadores selecionados no conjunto de dados da IaC e acurácia de detecção reportada na taxa de Verdadeiros Positivos (TP) para eventos de ataque e de Verdadeiros Negativos (TN) para eventos normais

Amb.	**Comportamento**	**Detecção de Acurácia**				
		Random Forest	**Decision Tree**	**Bagging**	**Multilayer Perceptron**	***Ours***
$\mathcal{D}_{test}$	ACK Scan	1.000	1.000	1.000	0.325	1.000
	DDoS	1.000	1.000	1.000	0.415	1.000
	ICMP Echo Discover	1.000	1.000	1.000	0.325	1.000
	UDP Scan	1.000	1.000	0.999	0.999	1.000
	HTTP (*Normal*)	1.000	1.000	0.999	1.000	1.000
	SNMP (*Normal*)	1.000	0.996	0.996	0.995	0.999
$\mathcal{D}_{gen}$	MySQL Brute Force	0.493	0.495	0.491	0.466	0.944
	Nikto	0.004	0.000	0.006	0.103	0.066
	Scan Vuln	0.962	0.962	0.962	0.332	0.964
	SSH Brute Force	0.999	0.999	0.999	0.984	0.997
	Stealth Scan	1.000	1.000	1.000	0.407	1.000
	SYN Scan	1.000	1.000	1.000	0.311	1.000
	Brute Force DIRB	0.000	0.000	0.006	0.113	0.067
	Wapiti	0.192	0.290	1.000	0.290	0.033
	CMS Scan	0.000	0.000	1.000	0.042	0.012
	Full Connect Scan	1.000	1.000	0.032	0.140	1.000
	DNS (*Normal*)	0.364	0.000	0.000	0.969	1.000
	SMTP (*Normal*)	1.000	1.000	0.998	0.998	1.000
	SSH (*Normal*)	0.964	0.990	0.990	0.995	1.000
Média		0.736	0.723	0.725	0.524	0.794

Fonte: o autor, 2024.

Isso demonstra a capacidade limitada dos NIDS tradicionais baseados em AM de se generalizarem para cenários reais e dinâmicos. Por exemplo, o classificador Random Forest (RF) alcançou desempenho perfeito no ambiente Dtest, com taxas médias de Verdadeiros Negativos (TN) e Verdadeiros Positivos (TP) de 1,0 e 1,0 respectivamente. No entanto, no ambiente Dgen, representando novas condições geradas por variações nos scripts IaC, o mesmo classificador apresentou um declínio significativo de desempenho, com taxas médias de TN e TP de apenas 0,77 e 0,56, correspondendo a quedas de 0,23 e 0,44, respectivamente.

Essa diferença acentuada evidencia a incapacidade do classificador de se adaptar ao comportamento dinâmico da rede introduzido pela natureza evolutiva das implementações em IaC. Essas limitações destacam a necessidade de desenvolver novas abordagens que enfrentem esses cenários, garantindo desempenho consistente mesmo em condições ambientais dinâmicas. O segundo experimento visa responder à RQ2, investigando como o modelo proposto de seleção de características pode ser usado para melhorar as capacidades de generalização do sistema.

Para isso, implementou-se o modelo proposto como uma seleção de características baseada em wrapper utilizando o Algoritmo Genético de Classificação Não Dominada II (NSGA-II) (KALYANMOY, 2002), por meio da API pymoo (ver Algoritmo 1). No experimento, o NSGA-II utilizou uma população de 100 indivíduos, 100 gerações, taxa de cruzamento de 0,3 e probabilidade de mutação de 0,1. A seleção multiobjetivo visou diminuir os valores de αtest (Equação 2) e αgen (Equação 3), seguindo o Algoritmo 1. Utilizando o proposto modelo de classificação dinâmica, medimos os objetivos resultantes para cada indivíduo avaliado.

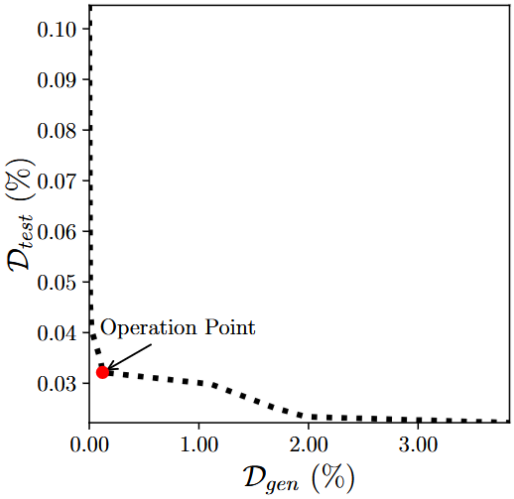
Nesse caso, avaliou-se a Seleção Dinâmica implementada com o classificador KNORA-E, utilizando um pool de bagging com 100 estimadores e 5 vizinhos (Algoritmo 2). A Figura 7 apresenta a curva de Pareto do modelo de seleção de características baseado em wrapper, destacando o equilíbrio entre a acurácia no teste (Dtest) e as capacidades de generalização (Dgen). Os resultados demonstram que melhorar a generalização para lidar com variações em infraestruturas provisionadas por IaC frequentemente requer tolerar taxas de erro ligeiramente mais altas no ambiente de treinamento.

Apesar desse compromisso, a abordagem melhora significativamente a generalização com impacto mínimo na acurácia do treinamento. Por exemplo, o modelo atinge uma taxa de erro de generalização tão baixa quanto 0,01%, mantendo uma taxa de erro de treinamento de apenas 0,1%. Essa flexibilidade permite que operadores de rede escolham modelos que atendam

às suas necessidades específicas, priorizando a acurácia de teste ou capacidade de generalização. Essa adaptabilidade torna o modelo proposto uma solução confiável para a implantação de NIDS baseados em AM nas infraestruturas reais provisionadas por IaC, reduzindo os requisitos de retreinamento e melhorando a eficiência operacional.

Para responder à RQ3, investigou-se como a classificação dinâmica com seleção de características pode melhorar a acurácia do modelo resultante. Para isso, foi selecionado um ponto de operação médio que fornece os melhores valores médios em relação aos objetivos da proposta (Figura 7, Ponto de Operação – Operation Point). É importante notar que o ponto de operação deve ser definido a critério do operador, podendo priorizar maior capacidade de generalização para lidar com ambientes mais dinâmicos, mesmo com impacto na acurácia do modelo.

Figura 7 – Curva de Pareto do mecanismo proposto de seleção de características multiobjetivo



Fonte: o autor, 2024.

A Tabela 5 apresenta a acurácia de detecção alcançada pelo modelo no ponto de operação selecionado (Ours). Os resultados mostram que o modelo proposto oferece melhorias substanciais na acurácia de detecção em ambos os ambientes de treinamento e generalização. Especificamente, o modelo alcança uma acurácia média de detecção de 0,794, superando o desempenho de métodos tradicionais como Random Forest, Decision Tree, Naive Bayes e Multilayer Perceptron, com melhorias de 0,058, 0,071, 0,056 e 0,270, respectivamente.

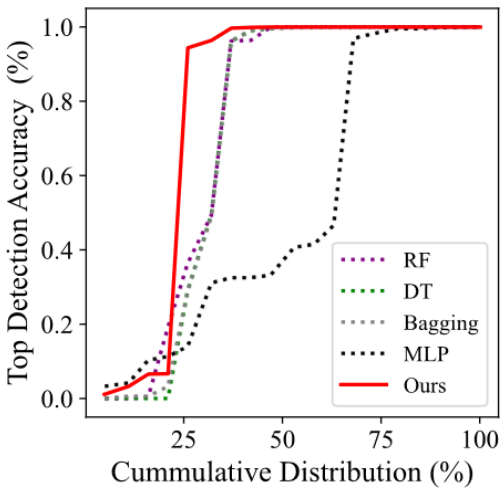
Analisou-se ainda como o modelo proposto melhora a acurácia de classificação para cada ataque avaliado no testbed. A Figura 8 – Curva de Pareto do mecanismo proposto de seleção de características multiobjetivo, apresenta a Função de Distribuição Cumulativa

(Cumulative Distribution Function – CDF) dos classificadores selecionados para os 14 ataques gerados no ambiente IaC. Os resultados indicam que o modelo apresenta melhorias significativas na acurácia de detecção para todos os ataques avaliados. Por exemplo, a abordagem atinge uma taxa de TP de 90% para 11 dos 14 ataques, superando os métodos tradicionais.

Por comparação, os classificadores Random Forest, Decision Tree, Naive Bayes e Multilayer Perceptron alcançam uma taxa de TP de 90% para apenas 9, 9, 9 e 5 ataques, respectivamente. Essa melhoria pode ser atribuída à sinergia entre o processo proposto de seleção de características multiobjetivo (Algoritmo 1) e o mecanismo de seleção dinâmica de classificadores (Algoritmo 2) empregado durante a fase de inferência. A precisão aprimorada em um espectro mais amplo de cenários de ataque demonstra a robustez e a adaptabilidade do modelo à natureza dinâmica das infraestruturas provisionadas por IaC.

Esses resultados destacam a importância de técnicas personalizadas de seleção de características e classificação adaptativa para alcançar uma detecção de intrusão confiável em ambientes tão complexos e variáveis.

Figura 8 – Curva de Pareto do mecanismo proposto de seleção de características multiobjetivo

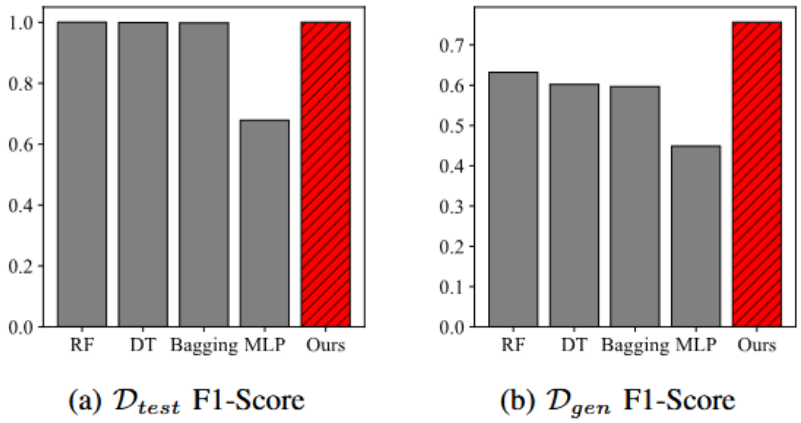


Fonte: o autor, 2024.

A Figura 9 compara as pontuações médias de F1 entre os modelos avaliados. Os resultados mostram que o modelo melhora substancialmente as capacidades de generalização em comparação com abordagens tradicionais (Figura 9b), ao mesmo tempo em que alcança melhorias leves nas acurácias do ambiente de treinamento (Figura 9a). Notavelmente, o modelo alcança um aumento no F1-Score de até 0,31 no conjunto de generalização, destacando sua

capacidade de detectar efetivamente comportamentos conhecidos e novos em ambientes dinâmicos provisionados por IaC.

Figura 9 – Pontuação F1 dos classificadores selecionados nos conjuntos de dados de treinamento e generalização



(a) $\mathcal{D}_{test}$ F1-Score (b) $\mathcal{D}_{gen}$ F1-Score

Fonte: o autor, 2024.

Essa melhoria decorre da sinergia entre o mecanismo de seleção dinâmica de classificadores e a estratégia de seleção de características multiobjetivo, que, em conjunto, otimizam a adaptabilidade do modelo a cenários de infraestrutura diversos. Esses resultados enfatizam o potencial da abordagem para viabilizar a implementação confiável de NIDS baseados em AM nas infraestruturas provisionadas por IaC, enfrentando os desafios impostos por sua natureza dinâmica e não estacionária.

6.3 DISCUSSÃO

Os experimentos realizados evidenciam os desafios enfrentados por NIDS baseados em AM nos ambientes dinâmicos provisionados por IaC. Tais como Queda na acurácia em ambientes desconhecidos (Dgen), limitações na capacidade de generalização dos classificadores tradicionais, necessidade de balancear a acurácia de treinamento e generalização, redução da necessidade de retreinamento em ambientes dinâmicos e escalabilidade e adaptação automática para NIDS em IaC. Enquanto os classificadores tradicionais, como Random Forest e Multilayer Perceptron, apresentaram alto desempenho em ambientes conhecidos (Dtest), suas taxas de acurácia caíram significativamente em ambientes desconhecidos (Dgen), refletindo limitações na capacidade de generalização. O modelo proposto, integrando seleção de características multiobjetivo (NSGA-II) e seleção dinâmica de

classificadores (KNORA-E), demonstrou superioridade ao melhorar a taxa de detecção em novos cenários, com ganhos de até 0,31 no F1-Score no conjunto de generalização e aumento na precisão de detecção para uma gama mais ampla de ataques. Assim sendo, a proposta mitiga diretamente quatro dos cinco desafios apresentados e abre caminho para pesquisas sobre escalabilidade e adaptação automática. Mesmo o modelo apresentando um avanço na detecção de intrusões em ambientes dinâmicos de IaC, ele enfrenta aqueles relacionados à qualidade dos dados, escalabilidade, adaptação em tempo real e robustez contra ataques adversariais. Mitigar essas ameaças pode exigir abordagens híbridas ou técnicas avançadas de aprendizado contínuo.

Além disso, o mecanismo mostrou-se adaptável, permitindo balancear a acurácia de treinamento e generalização com impacto computacional mínimo. Esses resultados destacam a relevância de abordagens personalizadas para NIDS em infraestruturas IaC, reduzindo a necessidade de retreinamento e aumentando a confiabilidade em ambientes complexos e não estacionários, enquanto abrem caminhos para estudos sobre escalabilidade e adaptação automática.

CAPÍTULO 7

CONCLUSÃO

Este trabalho apresenta a eficácia de um IDS adaptável, especialmente desenvolvido para proteger infraestruturas dinâmicas configuradas por meio da IaC. Por meio de um modelo AM, o sistema foi projetado para responder de forma dinâmica às variações comportamentais introduzidas pela própria natureza dos scripts IaC. Para abordar a generalização em redes não estacionárias, a pesquisa integrou uma abordagem de seleção de características multiobjetivo, capaz de otimizar simultaneamente a precisão do sistema e a capacidade de adaptação do classificador.

Essa metodologia visa aprimorar a generalização durante a fase de construção do modelo, considerando as potenciais mudanças de comportamento induzidas pelos scripts de configuração. Adicionalmente, a proposta apresentou uma seleção dinâmica de classificadores, permitindo o ajuste dos mesmos em tempo real de acordo com o ambiente IaC implementado. Essa seleção dinâmica assegura uma detecção eficiente de novos padrões de comportamento de rede que surgem da variabilidade dos scripts IaC, prevenindo a degradação da precisão ao adaptar os classificadores de acordo com o comportamento atual da infraestrutura.

Os experimentos realizados em um ambiente de teste realista, composto por 100 contêineres e múltiplas variações de comportamento de ataque, confirmam a eficácia do modelo em aprimorar tanto a generalização quanto a precisão na detecção. Este avanço representa um passo significativo na proteção de ambientes dinâmicos configurados por IaC, demonstrando a viabilidade de NIDS baseados em AM para ambientes de produção. A capacidade de adaptação contínua e a detecção de novos comportamentos conferem ao modelo proposto uma relevância prática para a segurança de infraestruturas consideradas modernas.

Proteger infraestruturas provisionadas por IaC requer análise e tratamento próprios para os respectivos cenários resultantes, devido à própria natureza dinâmica desses ambientes. A infraestrutura passa por mudanças contínuas impulsionadas pelos scripts de configuração fornecidos, levando a comportamentos de ambiente em constante evolução. Essa dinamicidade inerente dificulta a manutenção de uma segurança consistente e torna os NIDS baseados em AM tradicionais pouco confiáveis nesses contextos. Para enfrentar esses desafios, este trabalho

propõe um novo framework de NIDS baseado em AM, especificamente projetado para infraestruturas implantadas com IaC.

A abordagem incorpora dois componentes principais: seleção de características multiobjetivo e seleção dinâmica de classificadores. A seleção de características multiobjetivo tem como foco identificar um subconjunto de características que melhore as capacidades de generalização do modelo, enquanto a seleção dinâmica de classificadores escolhe proativamente o subconjunto de classificadores mais adequado para se adaptar ao comportamento atual do ambiente. Experimentos realizados em um testbed de intrusão desenvolvido para IaC demonstram a viabilidade e a eficácia da abordagem proposta. Os resultados mostram melhorias significativas na precisão de detecção em relação aos métodos tradicionais, abrindo caminho para NIDS baseados em AM mais confiáveis em ambientes dinâmicos e não estacionários da IaC.

Como trabalho futuro, vislumbra-se estender o modelo para incluir atualizações não supervisionadas após a fase de provisionamento da IaC. Essa estratégia permitirá a adaptação contínua do sistema às mudanças dinâmicas do ambiente, reforçando a robustez da detecção em cenários não estacionários. Além disso, planeja-se investigar métodos de aprendizagem contínua e transferência para ampliar a capacidade de identificação de novos padrões de ataque. A integração de técnicas de validação online também poderá ser considerada, visando refinar o modelo em tempo real e fortalecer a segurança de infraestruturas provisionadas por IaC.

REFERÊNCIAS

ABBAS, S.I.; GARG, A. Integrating Emerging Technologies with Infrastructure as Code in Distributed Environments. **2024 3rd International Conference on Applied Artificial Intelligence and Computing (ICAAIC)**, Salem, India, 2024.

ALKHAFAJI, N., VIANA, T., AL-SHERBAZ, A. Integrated Genetic Algorithm and Deep Learning Approach for Effective Cyber-Attack Detection and Classification in Industrial Internet of Things (IIoT) Environments. **Arabian Journal for Science and Engineering**, out. 2024. Disponível em: https://doi.org/10.1007/s13369-024-09663-6. Acesso em: 12 nov. 2024.

ANDRESINI, G. *et al*. Insomnia: Towards concept-drift robustness in network intrusion detection, **in Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security.** ser. CCS 21. ACM, Nov. 2021. Disponível em: http://dx.doi.org/10.1145/3474369.3486864. Acesso em: 08 nov. 2021.

AKHTAR, M.A. *et al*. Robust genetic machine learning ensemble model for intrusion detection in network traffic. **Sci Rep** 13, 17227. 2023.

ALZOUBI, Y.I., MISHRA, A.; TOPCU, A.E. Research trends in deep learning and machine learning for cloud computing security. **Artif Intell** Rev 57, 132. 2024.

BACE, R.; MELL, P. Intrusion Detection Systems, Special Publication (NIST SP). **National Institute of Standards and Technology**, Gaithersburg, MD. Disponível em: https://doi.org/10.6028/NIST.SP.800-31. Acesso em: 29 set. 2024.

BERTOLI, G. De C. *et al*. An End-to-End Framework for Machine Learning-Based Network Intrusion Detection System. **in IEEE Access**, vol. 9, pp. 106790-106805. 2021.

BLAISE, A. *et al*. Detection of zero-day attacks: An unsupervised port-based approach. **Computer Networks**, v. 180, p. 107391. 2020.

CATILLO M.; PECCHIA, A.; VILLANO, U. Machine learning on public intrusion datasets: Academic hype or concrete advances in nids? **in 2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S)**. IEEE, Jun. 2023. Disponível em: http://dx.doi.org/10.1109/DSN-S58398.2023.00038. Acesso em: 30 set. 2024.

CHERFI, S.; LEMOUARI, A.; BOULAICHE, A. Mlp-based intrusion detection for securing iot networks. **Journal of Network and Systems Management**, vol. 33, no. 1, Dec. 2024. Disponível em: http://dx.doi.org/10.1007/s10922-024-09889-7. Acesso em: 26 dez. 2024.

CSA Research Group. **Guia de segurança para áreas críticas focado em computação em nuvem**. Disponível em: cloudsecurityalliance.org/guidance. Acesso em: 30 ago. 2024.

DAS, S. *et al*. Network Intrusion Detection and Comparative Analysis Using Ensemble Machine Learning and Feature Selection. **in IEEE Transactions on Network and Service Management**, vol. 19, no. 4, pp. 4821-4833, Dec. 2022.

DOCKER. **Docker develop faster**. Disponível em: https://www.docker.com/. Acesso em: 10 jan. 2024.

ELZARIDI, K.; KUERNAZ, S. Integration Between Network Intrusion Detection and Machine Learning Techniques to Optimizing Network Security. **Babylonian Journal of Networking**. 2024.

ERTÖZ, L. *et al*. **Minnesota Intrusion Detection System**. MIT Press. 2004.

GAO, X. *et al*. An Adaptive Ensemble Machine Learning Model for Intrusion Detection. **IEEE Access**, v. 7, p. 82512–82521. 2019.

GARCIA-TEODORO, P. *et al*. Anomaly-based Network Intrusion Detection: Techniques, Systems and Challenges. **Computers & Security**, 28(1-2):18–28. 2009.

HASHICORP. **Terraform Deliver Infrastructure as Code**. Disponível em: https://www.terraform.io/. Acesso em: 10 jan. 2024.

HOFMEYR, S. A.; FORREST, S.; SOMAYAJI, A. Intrusion Detection Using Sequences of System Calls. **Journal of Computer Security**, 6(3):151–180. 1998.

HORCHULHACK, P.; SANTIN, A.; VIEGAS, E. Atualização Confiável dos Modelos de Detecção de Intrusão Baseada em Aprendizagem de Máquina. **in Anais Estendidos do XXIV Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais**, São José dos Campos/SP, pp. 17-24. 2024.

HORCHULHACK, P. Atualização Confiável dos Modelos de Detecção de Intrusão Baseada em Aprendizagem de Máquina 2023. 75 f. Dissertação (Mestrado em Informática) **Programa de Pós-Graduação em Informática, Pontifícia Universidade Católica do Paraná**, Curitiba, 2023.

KALYANMOY, D. *et al*. A fast and elitist multiobjective genetic algorithm: NSGA-II. **IEEE Transactions on Evolutionary Computation**, vol. 6, no. 2, pp. 182–197. 2002.

KHALID, M.; ABDULLAH, E.; SEFER, K. Integration Between Network Intrusion Detection and Machine Learning Techniques to Optimizing Network Security. **Deleted Journal**, 2024:57-68. 2024.

KONJAANG, J. K.; Xu L. Meta-heuristic approaches for effective scheduling in infrastructure as a service cloud: A systematic review, **Journal of Network and Systems Management**. vol. 29, no. 2, Jan. 2021. Disponível em: http://dx.doi.org/10.1007/s10922-020-09577-2. 2021. Acesso em: 30 jan. 2021.

KRAWCZYK, B.; MINKU, L. L.; GAMA, J. *et al*. Ensemble learning for data stream analysis. **A survey. Information Fusion**, v. 37, p. 132–156. 2017.

KUBERNETES. **Kubernetes Open-source**. Disponível em: https://kubernetes.io/. Acesso em: 15 jan. 2024.

LI, W. Using Genetic Algorithm for Network Intrusion Detection. **United States Department of Energy Cyber Security Group 2004 Training Conference**. 2004.

LINUX **Control Groups** (CGroups) Documentation. Disponível em: https://www.kernel.org/doc/html/latest/admin-guide/cgroup-v1/. Acesso em: 10 ago. 2024.

LINUX **Namespaces Documentation**. Disponível em: https://man7.org/linux/man-pages/man7/namespaces.7.html. Acesso em: 15 ago. 2024.

LOPES, R. R. F. *et al*. Accelerating NATO Transformation with SnTEE: Experiments with Network Security Function Virtualization in Coalition Networks. **2023 International Conference on Military Communications and Information Systems (ICMCIS)**, Skopje, North Macedonia. 2023.

LYNDON, A. Compilation for Intrusion Detection Systems. Master's thesis. **College of Engineering and Technology of Ohio University**, Ohio, 2004.

MANDUJANO, S. A Multiagent Approach to Outbound Intrusion Detection. PhD thesis. **Instituto Tecnológico y de Estudios Superiores de Monterrey**, Doctoral Program in Artificial Intelligence. 2004.

MEHDI, A.; WALIA, R. Terraform: Streamlining Infrastructure Deployment and Management Through Infrastructure as Code. **2023 International Conference on Computing Communication and Intelligent Systems (ICCCIS)**, pp. 851-856. 2023.

MICROSOFT. **Azure Resouce Manager**. Disponível em: https://azure. microsoft.com/get-started/azure-portal/resource-manager. Acesso em: 01 out. 2024.

MOLINA-CORONADO, B. *et al*. Survey of Network Intrusion Detection Methods From the Perspective of the Knowledge Discovery in Databases Process. **IEEE Transactions on Network and Service Management**, v. 17, n. 4, p. 2451–2479. 2020.

MORRIS, K. Infrastructure as Code: **Dynamic Systems for the Cloud**. O'Reilly. 2021.

NISIOTI, A.; MYLONAS, A.; YOO, P. D. *et al*. From Intrusion Detection to Attacker Attribution: A Comprehensive Survey of Unsupervised Methods. **IEEE Communications Surveys & Tutorials**, v. 20, n. 4, p. 3369–3388. 2018.

NIST. **Special Publications**. Fonte: nvlpubs.nist.gov: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-94.pdf. 2007.

NIST. **Special Publications**. Fonte: nvlpubs.nist.gov: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-145.pdf. 2011.

NIST. **Special Publications**. Fonte: nvlpubs.nist.gov: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-172.pdf. 2021.

OLÍMPIO, G. *et al*. Model update for intrusion detection: Analyzing the performance of delayed labeling and active learning strategies. **Computers and Security**, vol. 134, p.

103451, Nov. 2023. Disponível em: http://dx.doi.org/10.1016/j.cose.2023.103451. Acesso em: 30 nov. 2023.

OPDEBEECK, R.; ZEROUALI, A.; ROOVER, C. de Control and data flow in security smell detection for infrastructure as code: Is it worth the effort. **in 2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)**. IEEE, May 2023. Disponível em: http://dx.doi.org/10.1109/MSR59073.2023.00079. Acesso em: 30 mai. 2023.

PATHAK, P.; SHRIVAS, A. K. Intrusion Detection System Based Ensemble Model for Classification of Attacks. **2024 OPJU International Technology Conference (OTCON) on Smart Computing for Innovation and Advancement in Industry 4.0**, Raigarh, India, 2024, pp. 1-6. 2024.

PERFORCE, PUPPET; **Infrastructure and IT Automation at Scale.** October 2024. Disponível em: https://www.puppet.com/. Acesso em: 30 out. 2024.

PORTNOY, L.; ESKIN, E., STOLFO, S. Intrusion Detection with Unlabeled Data Using Clustering. **In Proceedings of ACM Workshop on Data Mining Applied to Security**. 2001.

PROGRESSCHEF, CHEF. **Extend DevOps Value**, October 2024. Disponível em: https://www.chef.io/. Acesso em: 31 out. 2024.

PUTRA, W. R. *et al*. Infrastructure as Code for Security Automation and Network Infrastructure Monitoring. **MATRIK: Jurnal Manajemen**, Teknik Informatika dan Rekayasa Komputer, 22(1), pp. 201-214. 2022.

RAHMAN, A.; R. MAHDAVI-HEZAVEH; WILLIAMS L. A systematic mapping study of infrastructure as code research. **Information and Software Technology**, vol. 108, p. 65–77, Apr. 2019. Disponível em: http://dx.doi.org/10.1016/j.infsof.2018.12.004. Acesso em: 03 out. 2019.

RAHMAN, A.; PARNIN, C. Detecting and characterizing propagation of security weaknesses in puppet-based infrastructure management. **IEEE Transactions on Software Engineering**, p. 1–18. Disponível em: http://dx.doi.org/10.1109/TSE.2023.3265962. Acesso em: 30 nov. 2023.

RED HAT. **O que é IaaS?** Disponível em: https://www.redhat.com/pt-br/topics/cloud-computing/what-is-iaas. Acesso em: 08 ago. 2024.

RONG, C. *et al*. Openiac: open infrastructure as code - the network is my computer. **Journal of Cloud Computing**, vol. 11, no. 1, May 2022. Disponível em: http://dx.doi.org/10.1186/s13677-022-00285-7. Acesso em: 03 mai. 2022.

SAAVEDRA, N.; FERREIRA, J. F. Glitch: Automated polyglot security smell detection in infrastructure as code. **in Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering**, ser. ASE '22. ACM, Oct. 2022. Disponível em: http://dx.doi.org/10.1145/3551349.3556945. Acesso em: 30 out. 2022.

SANTOS, K. C.; MIANI, R. S.; SILVA, F. de O. Evaluating the impact of data preprocessing techniques on the performance of intrusion detection systems, **Journal of Network and**

Systems Management. vol. 32, no. 2. Disponível em: http://dx.doi.org/10.1007/s10922-024-09813-z. 2024. Acesso em: 29 mar. 2024.

SERVICES, A. W. **AWS CloudFormation Speed up Cloud Provisioning with Infrastructure as Code**, October 2024. Disponível em: https://aws.amazon.com/cloudformation/. Acesso em: 30 out. 2024.

THAKKAR, P. *et al*. Dynamic microservice provisioning in 5g networks using edge–cloud continuum. **Journal of Network and Systems Management**, vol. 32, no. 4, Sep. 2024. Disponível em: http://dx.doi.org/10.1007/s10922-024-09859-z. Acesso em: 20 set. 2024.

TUFAIL, S. *et al*. Advancements and Challenges in Machine Learning: A Comprehensive Review of Models, Libraries, Applications, and Algorithms. **Electronics** 2023, 12, 1789. 2023.

VIEGAS, K. E. *et al*. BigFlow: Real-time and reliable anomaly-based intrusion detection for high-speed networks. **Future Generation Computer Systems**. 93. 10.1016/j.future.2018.09.051. 2018.

VIEGAS, E.; SANTIN, A. O.; ABREU, V. Jr, Machine learning intrusion detection in big data era: A multi-objective approach for longer model lifespans. **IEEE Transactions on Network Science and Engineering**, vol. 8, no. 1, p. 366–376, Jan. 2021. Disponível em: http://dx.doi.org/10.1109/TNSE.2020.3038618. Acesso em: 8 ago. 2024.

WANG, R. **Infrastructure as Code, Patterns and Practices**: With examples in Python and Terraform. Manning. 2022.

WANG, Y.; LIU, D. H.; CHAN, S. Intrusion detection methods based on integrated deep learning model. **Computers and Security**, vol. 103, p. 102177, Apr. 2021. Disponível em: http://dx.doi.org/10.1016/j.cose.2021.102177. Acesso em: 11 ago. 2024.

WOOD, M.; Erlinger, M. Intrusion Detection Message Exchange Requirements. **RFC 4766**, DOI 10.17487/RFC4766, March 2007.