Segurança Proativa em Arquitetura de Microsserviços e Container: um estudo de caso

Jhony Maiki Maseto[1], Diego Fabio Schuh[1], Daniel Paulo dos Santos[1], Rodrigo Miguel Tomazi[1], Ariel Gustavo Zuquello[1]

[1]Universidade Comunitária da Região de Chapecó - UNOCHAPECÓ

{jhony,df_schuh,daniel_paulo,rodrigo.tomazi,ariel.zuquello}@unochapeco.edu.br

Abstract. *Security in containerized environments is essential for a consistent and stable application. Vulnerabilities, unauthorized access, code injection, and information theft can lead to complete service unavailability. Therefore, this study proposes a proactive approach based on action research and a practical case study. The methodology was applied to the implementation of a Docker Swarm solution at a Higher Education Institution (HEI), focusing on resource optimization and security. The solution included network segmentation, privilege limitation with access only allowed via SSH with RSA key registration, and the integration of Wazuh as an Host-based Intrusion Detection System (HIDS). In addition, the adoption of continuous container update practices based on CVE queries. The results demonstrated that the solution was effective in monitoring traffic and automatically responding to attacks, as observed in a shell shock test. Therefore, the combination of Docker Swarm with security tools such as Wazuh creates a robust and scalable environment, efficiently protecting applications and data.*

Resumo. *A segurança em ambientes conteinerizados é essencial para se ter uma aplicação consistente e estável. Vulnerabilidades, acessos não autorizados, injeção de código e roubo de informações podem levar à indisponibilidade completa do serviço. Diante disso, este estudo propõe uma abordagem proativa baseada em pesquisa-ação e um estudo de caso prático. A metodologia foi aplicada na implementação de uma solução com Docker Swarm em uma Instituição de Ensino Superior (IES), com foco na otimização de recursos e segurança. A solução incluiu a segmentação de redes, a limitação de privilégios com acesso permitido apenas via SSH com cadastro de chaves RSA e a integração do Wazuh como Sistema de Detecção de Intrusão baseado em Host (HIDS), além da adoção de práticas de atualização contínua dos containers, com base em consultas de CVEs. Os resultados demonstraram que a solução foi eficaz em monitorar o tráfego e responder automaticamente a ataques, como observado em um teste de "shell shock". E, portanto, a combinação do Docker Swarm com ferramentas de segurança como o Wazuh cria um ambiente robusto e escalável, protegendo aplicações e dados de forma eficiente.*

1. Introdução

O período de convergência tecnológica da atual sociedade tem forçado organizações a estarem mais dependentes da informação. Diante deste cenário, é possível afirmar que a informação pode ser considerada como um dos maiores bens das organizações [Aramuni e Maia 2020].

A computação em nuvem tem ganhado cada vez mais destaque. A indústria de tecnologia desenvolveu diversas tecnologias de virtualização que permitem que múltiplas aplicações rodem no mesmo *hardware* físico, organizando aplicações usando

Máquinas Virtuais (VMs). Entretanto, as VMs apresentam algumas desvantagens: são grandes, podem ter desempenho instável ao executar várias instâncias, levam tempo para inicializar e não resolvem problemas como gerenciamento, atualizações de *software* e integração/entrega contínuas. A partir disto, surgiu a conteinerização, uma nova abordagem que oferece virtualização no nível do Sistema Operacional (SO) [Potdar *et al.* 2020].

Diferentemente da virtualização tradicional, que atua no nível do *hardware*, a conteinerização utiliza o SO do hospedeiro, compartilhando bibliotecas e recursos relevantes. Isso a torna mais eficiente, pois elimina a necessidade de um sistema operacional para cada aplicação. As aplicações e suas dependências são empacotadas em *container*, que são executados rapidamente no *kernel* do SO do hospedeiro, em um espaço isolado [Potdar et al. 2020]. A plataforma Docker é fundamental para a criação e execução desses *container*, garantindo que o ambiente suporte qualquer aplicação relacionada [Docker 2025].

Os *container* facilitam a implantação rápida de aplicativos, mas também facilitam a exploração de vulnerabilidades por invasores. Junto a isso, problemas de segurança surgem e é crucial que as organizações compreendam e lidem com as ameaças de segurança. Aplicativos de *software* são suscetíveis a vários tipos de vulnerabilidades, incluindo *bugs* de *software* conhecidos e desconhecidos, injeção de código malicioso e acesso não autorizado. Para mitigar o risco de vulnerabilidades em aplicativos, as organizações devem seguir práticas de codificação seguras, aplicar *patches* de *software* prontamente e usar ferramentas de segurança para verificar vulnerabilidades em aplicativos [Arasu 2023].

Este trabalho tem como objetivo apresentar as principais técnicas e tecnologias utilizadas para implementação de *container* em uma organização visando a segurança da informação sobre as aplicações neste hospedadas, pensando principalmente em confidencialidade, integridade e disponibilidade. Implementação de soluções de segurança proativas, controle de acesso dos desenvolvedores e escalabilidade das aplicações são os principais tópicos abordados.

2. Trabalhos relacionados

Uma visão geral abrangente dos desafios e soluções únicas inerentes a um ambiente de microsserviços distribuídos são oriundas de revisões sistemáticas da literatura e alguns estudos pesquisaram trabalhos relacionados a aspectos de segurança de sistemas baseados em microsserviços.

Proteção de microsserviços é um tópico que vem tomando forma e importância, se espalhando entre as organizações, com isso é possível classificar e identificar sinais que podem ser um problema no âmbito de segurança da informação [Ponce et al. 2022]. Pensando principalmente nos desenvolvedores, é possível observar ameaças conhecidas em microsserviços, com isso, esses profissionais podem detectar, mitigar e prevenir as ameaças de segurança oriundas de microsserviços, seguindo alguns procedimentos definidos pela criação de um guia [Hannousse e Yahiouche 2021].

A segurança da infraestrutura subjacente é uma grande preocupação, levantando ameaças e estratégias de mitigação relacionadas a ambientes com *container* e

plataformas de orquestração [Pahl 2018]. Além das preocupações mencionadas, é importante observar também que, em caso de algum ataque ou falha de segurança, é como se recuperar deles [Pereira-Vale *et al.* 2019].

A segurança de APIs (*Application Programming Interface*), sendo um dos principais canais de comunicação nas aplicações, também é amplamente abordada na literatura, com pesquisas gerais fornecendo uma visão geral das estratégias para proteger essas interfaces [Kothapalli 2021]. O campo evoluiu para adotar padrões arquitetônicos e melhores práticas que integram a segurança ao ciclo de vida do desenvolvimento. O modelo DevSecOps (*Development*, *Security*, *Operations*), que incorpora segurança em *pipelines* de integração e entrega contínuas, é um pilar da segurança moderna de microsserviços. Fornecer diretrizes cruciais para proteger esses processos contínuos e adotar modelos como *zero trust* trazem grandes benefícios à segurança [Chandramouli 2019].

Os trabalhos relacionados mostram que a segurança da informação em microsserviços vem evoluindo à medida que as aplicações tornam-se cada vez mais complexas e à medida que as organizações vêm adotando essa tecnologia. Sendo assim, este trabalho visa trazer tópicos, procedimentos, políticas e ferramentas utilizadas para uma maior segurança, tanto em acesso aos *container* quanto nas aplicações em si. Estes foram aplicados em uma organização real para utilização de aplicações de sistema de gestão educacional.

3. Metodologia

Esta pesquisa é verificada por meio de um estudo de caso, de natureza aplicada. Dessa forma, a questão de pesquisa é abordada por meio da metodologia de pesquisa-ação. A Figura 1 ilustra as etapas para aplicação da metodologia deste estudo.

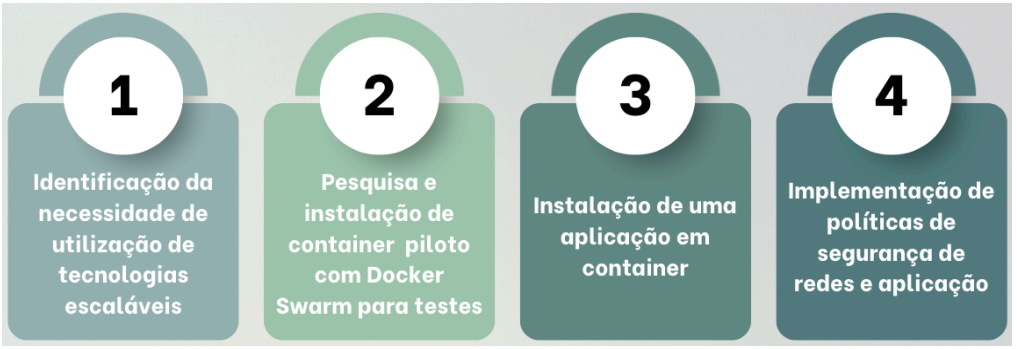

Figura 1. Etapas da metodologia.

Inicialmente, na etapa 1 a organização identificou a necessidade de ampliação na estrutura das aplicações, porém com a necessidade de otimizar recursos de hardware e *software*, buscando alternativas para novos serviços e soluções para os clientes. Na sequência, a etapa 2 trata da pesquisa de soluções de implementação de microsserviços, sendo possível otimizar recursos de bibliotecas, por exemplo, para implementação de várias aplicações no mesmo servidor, com a possibilidade de escalabilidade.

A etapa 3 já elucida a implementação de uma aplicação dentro da stack do Docker Swarm, a fim de identificar lacunas e testar a solução nesta nova arquitetura. E na etapa 4, serão abordadas as políticas de segurança da informação e infraestrutura.

4. Identificação da necessidade e estudos de *container*

Soluções e tecnologias que utilizam arquitetura de *software* monolítica, pensando principalmente em escalabilidade, tornam-se alternativas engessadas com necessidade de *upgrades* de *hardware* para acompanhar o crescimento e expansão na utilização das ferramentas.

Nesse cenário, a Instituição de Ensino Superior (IES) passou a investigar alternativas mais flexíveis e escaláveis que pudessem servir as soluções, principalmente *web*, com grande volume de conexões simultâneas, assim como permitir a portabilidade para ambientes de nuvem, aumentando a resiliência e independência sobre a infraestrutura física atual.

De fato a IES possuía poucos microsserviços em instâncias não clusterizadas do Docker mas sem efetividade na adoção da tecnologia, então, a partir disso o próximo passo foi estudar mais sobre a solução com Docker Swarm, criando um *cluster* específico adicionando a ele o *proxy* reverso.

4.1. *Container* e Docker Compose

Diferentes paradigmas arquitetônicos vêm sendo investigados visando acelerar os processos e reduzir os custos relacionados ao desenvolvimento e à implantação de aplicações em ambientes de computação em nuvem. A emergente virtualização baseada em *container*, em particular o Docker, tem se consolidado como uma alternativa promissora para essas soluções. Essa abordagem permite que os *container* compartilhem não apenas os recursos físicos, mas também o sistema operacional e suas bibliotecas de suporte [Wan *et al.* 2018].

O Docker Compose é uma ferramenta que permite orquestrar múltiplos *container* de forma declarativa, por meio da definição de suas configurações e propriedades de execução em um arquivo. Cada *container* é tratado como um "serviço", entendido como uma unidade que interage com outros *containers* e que possui propriedades de execução. Essa ferramenta simplifica a implantação das aplicações, possibilitando, com um único comando, criar e iniciar todos os serviços de sua configuração. [Turnbull 2016].

Dentro da arquitetura de microsserviços, cada componente é executado de forma independente e pode ser substituído ou atualizado sem impactar diretamente os demais. O advento da arquitetura de *container* e microsserviços oferece a possibilidade de melhorar a escalabilidade e a elasticidade do desenvolvimento de aplicações [Wan *et al.* 2018].

5. Instalação de aplicações em *containers*

A partir de pesquisas e do conhecimento prévio da equipe, a IES decidiu adotar o Docker Swarm para os novos projetos, devido a sua escalabilidade, facilidade de aprendizado e a simplicidade de instalação e manutenção.

Foram migrados alguns projetos, que operavam como monolitos ou em *container* avulsos para este novo ambiente. Além disso, um projeto com escopo mais amplo foi implementado em microsserviços, com foco na resolução de um problema em específico de uma aplicação *web*. A implementação desta solução trouxe diversos benefícios, sendo o principal, a resolução do problema enfrentado, permitindo assim, trabalhar de forma escalável com várias réplicas, sem a necessidade de modificações significativas.

6. Segurança em *containers*

Após a implementação da nova solução, levantou-se a seguinte questão. Como inspecionar o tráfego passante aos *containers*? E neste ponto, surge a possibilidade de utilizar a ferramenta Wazuh, que já era utilizada para *logs* de eventos de segurança.

E assim, foram adaptados alguns procedimentos que eram utilizados no modelo monolítico para o contexto dos *container*, garantindo maior segurança neste ambiente.

6.1. Segmentação, isolamento da rede e controle de acesso

Para prevenir a expansão lateral de ataques, cada *container* opera em sua própria rede isolada. Portas internas de serviços, como bancos de dados, não são expostas, reduzindo assim drasticamente o risco de acesso não autorizado ao *host*. Além disso, todo o tráfego externo destinado aos *container* em produção é inspecionado por um *proxy* reverso com Wazuh e um *firewall* de camada de aplicação, garantindo uma barreira de proteção robusta.

A segurança do container é garantida pelo seu isolamento. É fundamental que um container seja isolado do sistema hospedeiro e das outras aplicações para evitar ataques. Por exemplo, um atacante não deve conseguir executar comandos que afetem o sistema principal. Para evitar isso, os containers não devem ter permissões de administrador, o que restringe significativamente as oportunidades de ataque. Cada contêiner funciona com sua própria rede isolada, o que significa que cada um tem seu próprio endereço IP (*Internet Protocol)* [Jensen e Miers 2020].

Para criação de novos *containers*, apenas imagens criadas pela equipe ou de projetos amplamente reconhecidos são implementadas e configuradas com privilégios mínimos (*rootless*). Ao gerar um novo *container*, a imagem "*minimal*" é utilizada, na sequência são instalados apenas os pacotes necessários para utilização da aplicação. Essa prática reduz a superfície de ataque e garante que os *container* rodem apenas o essencial para a aplicação. Todas as imagens são gravadas em registro privado.

Para acesso restrito e controlado, estes foram limitados apenas aos *hosts* de produção, permitindo acesso apenas via SSH (*Secure Shell*) com chaves RSA (*Rivest-Shamir-Adleman*) dos desenvolvedores. Além disso, o *deploy* de projetos em ambientes de produção foi feito exclusivamente por meio da solução GitLab, garantindo que as mesmas políticas de segurança e acesso "sudo" sejam aplicadas.

A implementação do *proxy* reverso em um *container* com Nginx dentro do *cluster* de Swarm, que recebe requisições de clientes e repassa para os serviços internos que não tem comunicação direta com a internet, traz mais segurança para o aplicativo hospedado.

6.2. Integração com ferramentas de HIDS

Pensando na otimização e automatização do monitoramento do ambiente, a integração de uma ferramenta de HIDS (*Host-based Intrusion Detection System*) foi implementada intitulada Wazuh. Com a utilização desta ferramenta é possível exportar os *logs* dos *containers* para o *host*, o que permite que a ferramenta inspecione os *logs* em tempo real e tome ações de resposta automática a possíveis ameaças.

Ao implementar a ferramenta Wazuh nos *hosts* de um *Cluster* Swarm é possível tomar ações sobre os acessos a determinados serviços que trafegam através de um *proxy* reverso, de forma a inspecionar e bloquear conexões maliciosas e/ou suspeitas, gerar alertas e histórico de tentativas de acesso sobre um determinado conteúdo exposto.

A estratégia teve como base um *cluster* de Swarm com *proxy* reverso usando Nginx com a imagem da comunidade e pequenas alterações. O Docker Swarm é uma solução nativa do Docker, com isso, a ferramenta expõe a API padrão da plataforma sobre o *cluster*, possibilitando que as aplicações e ferramentas interajam com o ambiente como se fosse uma única instância Docker, viabilizando o balanceamento de carga e a alta disponibilidade das aplicações conteinerizadas. [Turnbull 2016].

No docker *compose* foram externados os *logs* do Nginx para uma partição do *host* hospedeiro ao qual existe a ferramenta do Wazuh instalada, conforme demonstra o código de configuração YML na Figura 2.

```
services:
  proxy:
    image: marsella.unochapeco.edu.br:5000/reverse-proxy:
    ports:
      - target: 80
        published: 80
        protocol: tcp
        mode: host
      - target: 443
        published: 443
        protocol: tcp
        mode: host
    environment:
      - TZ=America/Sao_Paulo
    volumes:
      - /var/log/nginx/:/var/log/nginx/
```

Figura 2. Compose YML do container no Nginx

Com estas exportação, os *logs* de “access.log” e “error.log” são visíveis no host da aplicação e podem ser inspecionados pela ferramenta. Com isso, é possível ter uma visibilidade de todo tráfego passante pelo *proxy* reverso bem como das demais aplicações que usam o *proxy* como rota para seus *containers*. A Figura 3 apresenta o *dashboard* com os eventos da ferramenta Wazuh.

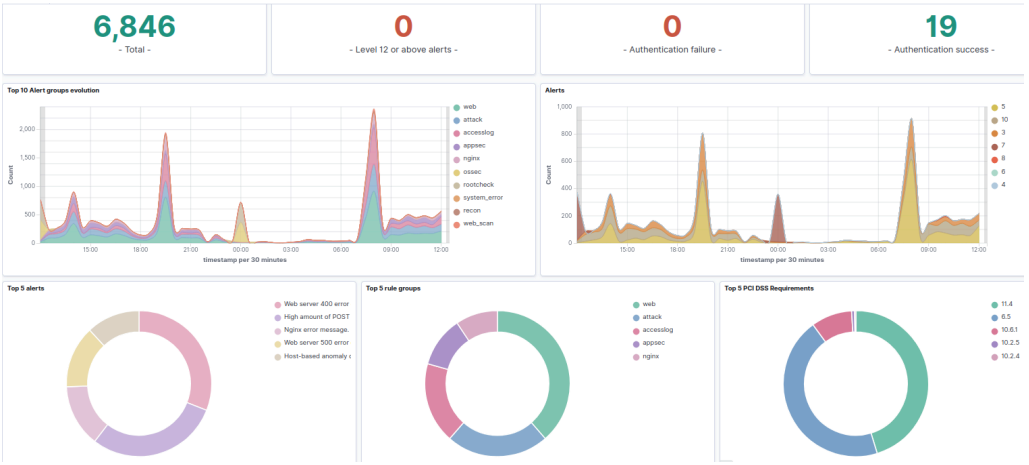

Figura 3. Tela de eventos do Wazuh

A ferramenta Wazuh classifica os eventos em 15 níveis, sendo o nível 1 como nível de notificação baixo de segurança e o 15 como nível severo. Na Figura 4 é apresentado um exemplo do "erro 400", sendo que a ferramenta elencou o evento como nível 10 para um acesso retornando este erro. Neste caso não tomou nenhuma ação, apenas alertando sobre o evento, sendo esse um problema conhecido na aplicação.

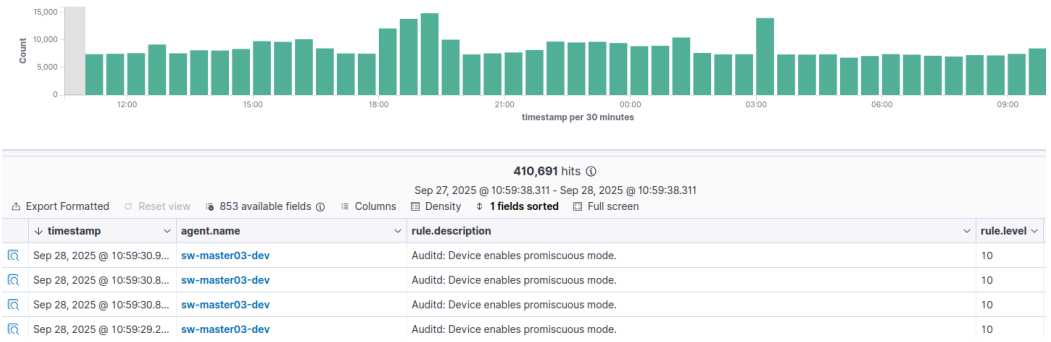

Figura 4. Tela de eventos em detalhes do Wazuh

Forçando um comportamento malicioso para elucidar o tratamento feito pela ferramenta, gerando um ataque de *shell shock* por meio de uma estação de forma remota em com o IP 177.131.124.254. Na Figura 6 é possível verificar que a primeira tentativa chegou ao destino e obteve uma resposta do *host*, não foi efetiva pois o *host* não está vulnerável a este tipo de ataque, entretanto ajuda a visualizar o funcionamento da proteção. Já a segunda tentativa apresentada, também apresentada na Figura 5, resultou em *timeout*, pois o bloqueio já foi executado pela ação do *software*.

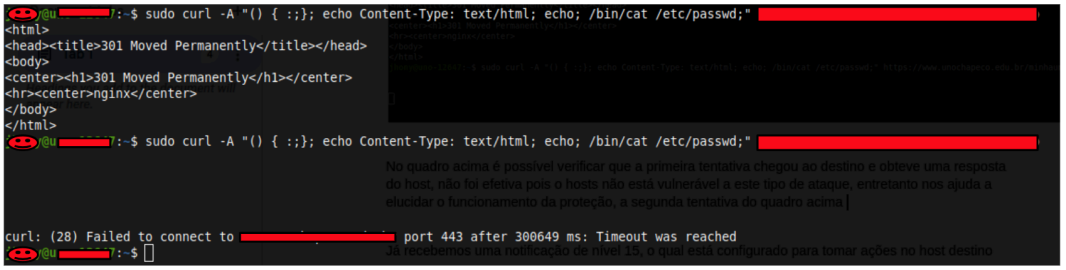

Figura 5. Executando um ataque *Shell Shock* contra o *cluster*

Um ataque de *shell shock* é um ataque que busca obter informações do SO (Sistema Operacional) por meio de um ataque “HTTPS”, caso este esteja vulnerável, neste caso, buscava-se pelo arquivo “passwd” contendo os usuário do sistema onde a aplicação estava hospedada.

Para essa simulação, a equipe recebe *e-mails* de notificações e neste caso em específico, uma notificação de nível 15 foi indicada, ao qual está configurado para tomar ações no *host* de destino. Na ferramenta Wazuh também é possível visualizar o alerta gerado, sendo que a conexão foi interceptada, conforme representa a Figura 6.

t	cluster.node	manager
t	data.id	301
t	data.protocol	GET
t	data.srcip	177.131.124.254
t	data.url	/
t	decoder.name	web-accesslog
t	full_log	177.131.124.254 - - [27/Sep/2025:21:44:50 -0300] "GET / HTTP/1.1" 3 01 0 "-" "() { :; }; /bin/bash -c \x22(wget -qO- http://74.194.19 1.52/rondo.qre.sh||busybox wget -qO- http://74.194.191.52/rondo.qr e.sh||curl -s http://74.194.191.52/rondo.qre.sh)|sh\x22& #Mozill a/5.0 (makenoise@tutanota.de)"
t	id	1759020292.10023017
t	input.type	log
t	location	/var/log/nginx/access.log
t	manager.name	wazuh.master
t	rule.description	Shellshock attack detected

Figura 6. Tela com detalhes do incidente no Wazuh

A Figura 7 apresenta a regra criada para essa ação, bloqueando qualquer acesso por 3600 segundos.

```
<active-response>
  <!-- Firewall Drop response. Block the IP for
       - 3600 seconds on the firewall (iptables,
       - ipfilter, etc).
  -->
  <command>firewall-drop</command>
```

Figura 7. Configuração de ação no agente

Além destas ferramentas, a verificação periódica das falhas de segurança dos *containers* é fundamental. A busca em portais de segurança da informação, a fim de localizar CVEs (*Common Vulnerabilities and Exposures*), é um procedimento importante para encontrar problemas e vulnerabilidades conhecidas na comunidade, para em seguida, aplicar os *patches* de segurança no ambiente.

7. Conclusão

Este trabalho teve como objetivo principal apresentar técnicas e tecnologias para a implementação segura de *container* em uma organização, com foco na confidencialidade, integridade e disponibilidade das aplicações hospedadas. Por meio de um estudo de caso prático em ambiente de produção na IES, demonstrou-se como a adoção de tecnologias de conteinerização, como o Docker Swarm, pode otimizar recursos e aumentar a escalabilidade, ao mesmo tempo em que políticas de segurança proativas são aplicadas para mitigar riscos.

A metodologia de pesquisa-ação permitiu a identificação de lacunas de segurança na migração de aplicações monolíticas para a arquitetura de microsserviços. Em resposta, implementou-se uma solução robusta que integra segmentação de rede, controle de acesso restrito e a ferramenta Wazuh como um HIDS. A segmentação de rede, combinada com a utilização de um *proxy* reverso e a execução de *container* com privilégios mínimos, demonstrou ser eficaz na redução da superfície de ataque e na prevenção de movimentos laterais.

A integração do Wazuh, em particular, provou ser um diferencial para a segurança do ambiente. Ao exportar *logs* de acesso do *proxy* reverso, a ferramenta possibilitou o monitoramento em tempo real do tráfego, a detecção de comportamentos maliciosos e a tomada de ações automáticas de resposta, como o bloqueio de IPs de origem de ataques. O teste de *shell shock* simulado evidenciou a capacidade do sistema em detectar e neutralizar tentativas de intrusão, garantindo a resiliência do ambiente.

Além disso, a definição de políticas de atualização e continuidade da solução é um fator importante para o bom funcionamento das aplicações.

8. Referências

Aramuni, J. P., Maia, L. C. (2020). O impacto da Engenharia Social na Segurança da Informação: uma abordagem orientada à Gestão Corporativa. AtoZ: novas práticas em informação e conhecimento, v. 7, n. 1, p. 31, 10 jan. 2020.

Arasu E. A. (2023). The Importance of Container Security in Preventing Cyber Threats. ISSA Journal. p 20-24.

Chandramouli, R. (2019). Security in Microservices Architecture (NIST Special Publication 800-204). National Institute of Standards and Technology.

Docker (2025) https://docs.docker.com/. Acesso em 03 de agosto de 2025.

Hannousse, A., Yahiouche, S. (2021). Securing Microservices and Microservice Architectures: A Systematic Mapping Study. Computer Science Review, 40, 100378.

Jensen N., Miers C. C. (2020). Segurança de contêineres: taxonomia baseada na arquitetura. In: Escola Regional de Redes de Computadores (ERRC), 18. Evento Online. Porto Alegre: Sociedade Brasileira de Computação, 2020. p. 128-134.

Kothapalli, M. (2021). Securing Microservices Architecture: Best Practices and Challenges. Journal of Scientific and Engineering Research, 2021, 8(10):187-192

Pereira-Vale, A., Márquez, G., Astudillo, H., Fernandez, E. B. (2019). Security Mechanisms Used in Microservices-Based Systems: A Systematic Mapping. XLV Latin American Computing Conference (CLEI), Panama, Panama, 2019, pp. 01-10, doi: 10.1109/CLEI47609.2019.235060.

Ponce, F., Soldani, J., Astudillo, H., Brogi A. (2022). Smells and Refactorings for Microservices Security: A Multivocal Literature Review. Journal of Systems and Software, 187, 111162.

Potdar A. M., Narayan D. G., Kengond S., Mulla M. M. (2020). Performance Evaluation of Docker Container and Virtual Machine. Third International Conference on Computing and Network Communications.

Turnbull, J. (2016). The Docker Book: Containerization is the new virtualization. Disponível em: <https://www.academia.edu/34912717/The_docker_book>.

Wan, X., Guan X., Wang T., Bai G., Choi B. (2018). Application deployment using Microservice and Docker containers: Framework and optimization. Department of Computer Science and Technology, Nanjing Tech University, Nanjing, 211816.