Uma análise das vulnerabilidades de segurança do Kubernetes

Nikolas Jensen[1], **Charles C. Miers**[1]

[1] Departamento de Ciência da Computação (DCC)
Universidade do Estado de Santa Catarina (UDESC)

nikolas.j@edu.udesc.br, charles.miers@udesc.br

Resumo. *Os orquestradores de contêineres vêm ganhando mais utilizadores a cada ano, sendo utilizado nas infraestrutura de pequenas e grandes empresas. Atualmente, o Kubernetes é o orquestrador mais utilizado e apresenta uma arquitetura de gerenciador e trabalhadores. O uso de diversos recursos normalmente aumenta a superfície de ataque, sendo assim, é possível explorar vulnerabilidades para atacar o cluster. Este artigo tem por objetivo fazer uma análise das principais vulnerabilidades do Kubernetes, relacionando as vulnerabilidades e os componentes de um sistema de orquestração.*

1. Introdução

Os orquestradores de contêineres possuem diversos componentes, podem tanto realizar o serviço de *front-end* quanto de *back-end* [Marathe et al., 2019], balanceando o volume de trabalho entre diversos nós. É possível oferecer serviços de diversas maneiras, e.g., *Infrastructure as a Service* (IaaS), *Load Balance as a Service* (LBaaS), *Software as a Service* (SaaS), etc. A orquestração, conjuntamente aos contêineres, oferece uma considerável gama de serviços a serem ofertados.

Neste trabalho o orquestrador a ser analisado é o Kubernetes, o qual é o que apresenta melhores resultados com relação a eficiência, utilização de *hardware*, etc. [Mercl and Pavlik, 2018]. Além disso, o Kubernetes tem sido utilizado em diversas pesquisas na academia, não somente em relação a segurança, mas como também desempenho. Este orquestrador possui diversos recursos para facilitar o desenvolvimento de aplicações, aprimorar a segurança e eficiência, escalabilidade do sistema, replicabilidade, etc. [Kubernetes, 2021]. Porém, com estes diversos recursos a superfície de ataque aumenta, com uma superfície de ataque expressiva, a dificuldade em garantir a segurança do sistema é maior.

Este artigo está organizado em cinco seções, na Seção 2 é explicado o funcionamento do Kubernetes com uma visão geral dos seus principais componentes. Na Seção 3 busca dar entendimento ao que é uma vulnerabilidade no sistema e como poder ser classificada. Aplicando os conceitos visto na Seção 2 e na Seção 3, é possível entender como são exploradas as vulnerabilidades num *cluster*, na Seção 4.

2. Kubernetes

O Kubernetes é um dos orquestradores mais utilizados atualmente em diversas empresas, em entrevista com 500 profissionais de tecnologia, 88% destes afirma que utiliza o Kubernetes em sua empresa [Red Hat, 2021]. Possui diversos componentes, os quais, cada

um realiza sua função com o objetivo final de oferecer eficiência e boa usabilidade de recursos para algum sistema.

Os recursos disponíveis oferecem facilidades para administrar as aplicações de forma que um ser humano não precise dedicar seu tempo a algo que foi automatizado. Os nós podem ser tanto máquinas físicas como virtuais, e existem dois tipos de nó em um *cluster* Kubernetes, o controlador e o trabalhador. Além disso, o Kubernetes introduz o conceito de *pod* o qual é a menor unidade de desenvolvimento em um *cluster* que é possível criar e gerenciar [Kubernetes, 2021], este isola os contêineres e cada um contém um *namespace* próprio. O nó controlador possui acesso ao painel de controle do *cluster*, e este possui diversos componentes dentro, sendo o servidor de *Application Programming Interface* (API), etcd para armazenamento dos dados do *cluster*, escalonador e um controlador para nós, processos, *endpoints* e dos dados de contas [Kubernetes, 2021].

O servidor da API irá expor o *cluster* para alguma rede, podendo ser interna ou externa. Esta será responsável por orquestrar todas as operações do *cluster* [Shamim et al., 2020], e.g., inserção de um novo *pod* em um nó.

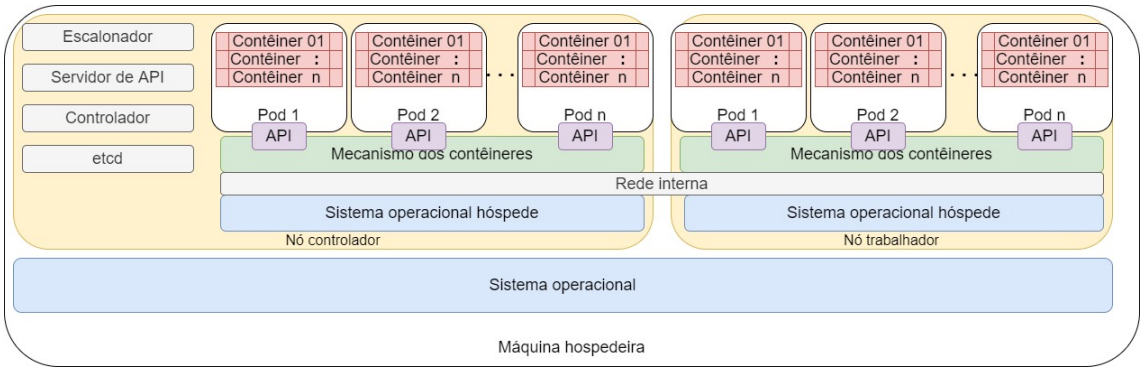

Figura 1. Diagrama da arquitetura de um *cluster* Kubernetes

Na Figura 1 exemplifica-se a arquitetura de um *cluster* Kubernetes, possuindo uma única máquina física e duas máquinas virtuais servindo como nó controlador e trabalhador. É possível notar que os nós são unidos por uma rede interna do *cluster*, isto serve para que cada nó se conheça mutuamente e possam trocar informações.

O trabalho dos balanceadores de carga nesta rede interna é direcionar o tráfego de trabalho para os diversos *pods* e contêineres. O tráfego é recebido pelo balanceador de carga no nó controlador e este é repassado para os demais nós, e os pacotes enviados realizam a mesma rota que os recebidos [Takahashi et al., 2018]. Assim, a rota de comunicação sempre é a mesma e sempre passa por alguns componentes específicos.

As políticas do Kubernetes servem para garantir mais segurança ao *cluster* impondo regras. Para aprimorar a segurança da rede do *cluster* é possível implementar políticas de segurança de rede, como restringir os *Internet Protocol*s (IPs) com os quais determinado nó, *pod* ou contêiner pode se comunicar [Shamim et al., 2020]. Uma política de segurança é importante para que limite-se a superfície de ataque do contêiner, prevenindo que ações indesejadas sejam realizas por atacantes.

3. Vulnerabilidades

Segundo [MITRE (CVE Terminology), 2021] uma vulnerabilidade pode ser definida como uma falha no *software, hardware, firmware* ou um componente do serviço, resultando em uma fraqueza que seja possível explorar causando um impacto negativo. Sendo assim, a vulnerabilidade pode ser um erro do desenvolvedor da aplicação, ou até uma falha na linguagem utilizada.

As vulnerabilidades podem surgir em qualquer tipo de equipamento, desde dispositivos físicos, virtuais, etc. Vulnerabilidades apresentam riscos aos sistemas, como podem ser exploradas por ameaças ao sistema, as ameaças normalmente utilizam as vulnerabilidades para causar dano ou perda [Nyanchama, 2005]. Sendo assim, as vulnerabilidades não devem ser ignoradas, e sua busca deve ser constante, visto que o risco existe e terá impacto.

A *Common Vulnerabilities Exposure* (CVE) é definida como um erro de configuração num software que habilita o acesso de informações e recursos do sistema [Kronser, 2020]. Logo, a CVE será atribuída a uma ou mais tecnologias, tanto empresariais como de código aberto. Além disso, as *Common Weakness Enumeration* (CWE) classificam as CVEs em determinadas categorias também criadas pelo MITRE, as quais são de ordem maior que a CVE.

4. Vulnerabilidades exploradas em *cluster* Kubernetes

O Kubernetes possui diversos recursos oferecidos a serem utilizados. Porém, com o amplo uso destes recursos a superfície de ataque do *cluster* tende a aumentar. Exemplificando, [David and Barr, 2021] explora o recurso de balanceamento de carga e auto replicação do Kubernetes para realizar um ataque de negação de serviço.

O Kubernetes adota um sistema de IP-*per-pod*, isto significa que cada *pod* possuirá um IP único, e os contêineres dentro de cada *pod* compartilham o mesmo *namespace* [Kubernetes, 2021]. Com esta característica, o risco de interceptação de mensagens se torna maior, uma vez que se encontram no mesmo *namespace*, e como abordado na Seção 2, o caminho dos pacotes na rede é o mesmo.

A exposição da API do Kubernetes para a Internet acaba gerando uma ameaça, uma vez que o serviço pode estar comprometido, e caso não possua políticas de segurança rígidas, a integridade do *cluster* pode ser comprometida. Além disso, utilizando uma política de segurança ineficaz de um *pod*, na qual esta permita execução com privilégio de administrador, o invasor pode ter mais facilidade para garantir a persistência do ataque.

Os contêineres podem estar executando uma aplicação vulnerável, ou até um *malware* [Shamim et al., 2020], logo, uma política de segurança consistente evita que afete todo o *cluster*. O *cluster* possui uma rede interna, com isso, o ataque *Server Side Request Forgery* (SSRF) acaba sendo efetivo. Este ataque possui como característica principal acessar máquinas dentro de uma rede interna. A CVE-2020-8555 descreve um ataque SSRF no controlador do gerenciador, possibilitando revelar informações sensíveis.

As vulnerabilidades de um *cluster* podem ser exploradas nos seus diversos componentes. Na Tabela 1 são apresentadas vulnerabilidades listadas pelo MITRE e também não listadas, relacionando a exploração delas em alguns componentes de um *cluster* Kubernetes. Sendo na Tabela 1: SR: Segurança de rede; SH: Segurança do hospedeiro; SM:

Segurança do mecanismo do contêiner; SI: Segurança da imagem do contêiner; e NA: Não se aplica.

Tabela 1. Tabela relacionando algumas vulnerabilidades listadas no repositório CVE e também outras não listadas.

	Pod	Controlador	Load balancer	API
Autenticação incorreta	SR	SR	NA	SR
Permissão indevida	SM, SR	SR	NA	SI, SM
Revelação de informações sensíveis	SH, SM	SH, SM, SI	NA	SM, SR
Manejo incorreto de erros	NA	SM	NA	NA
Validação imprópria de entrada	NA	SI, SR	SH, SR	NA
Alocação sem limites de recursos	SH, SI	NA	SH	NA
Erro de configuração	SR, SH, SM	NA	SR, SM	NA
Movimento lateral	SR, SM	SR, SH	NA	NA

O impacto é em relação a segurança da rede, hospedeiro, mecanismo do contêineres e da imagem dos contêineres [Jensen and Miers, 2020]. A relação entre a vulnerabilidade em determinado componente impacta na segurança de forma distinta.

4.1. Erros de configuração

Um *cluster* possui diversos componentes que podem ou não serem adicionados para diversos fins. Além de mais componentes normalmente aumentarem a superfície de ataque, a correta configuração destes se torna mais trabalhosa, visto que, são muitos detalhes a atentar-se. Dos incidentes de segurança do Kubernetes, 59% foram causados devido a erros de configuração [Red Hat, 2021]. Nota-se, que os erros de configuração são comuns, e também danosos ao *cluster*, uma vez que é tratado como uma vulnerabilidade no sistema.

Configurar políticas de segurança é um meio de aprimorar a segurança do *cluster*, mas podem ocorrer diversos erros durante a configuração que geram ameaças ao *cluster* [Budigiri et al., 2021]. Durante a configuração de uma política de rede pode ocorrer erros de especificação de nomes de *pods*, tornando uma comunicação insegura e ameaçando a segurança do *cluster*. Uma pequena falta de atenção como esta pode acabar por comprometer toda as informações de uma organização.

Um caso em que atacantes exploraram erros de configuração, foi em ambientes Argo, que utiliza o Kubernetes. Alguns ambientes Argo permitiam que qualquer um executasse código arbitrário no ambiente de execução [Robinson and Fishbein, 2021]. Sendo assim, os atacantes utilizaram os recursos computacionais destes ambientes para mineração de criptomoedas.

4.2. Movimento lateral

Depois que um atacante que comprometa uma rede de computadores, ele pode gradualmente expandir o ataque por mais máquinas, por meio do movimento lateral [Wilkens et al., 2019]. O movimento lateral normalmente possui como alvo uma rede interna de alguma corporação, na qual é possível invadir computadores de funcionários, banco de dados, servidores, etc.

A Figura 2 exemplifica um movimento lateral num *cluster* Kubernetes de forma superficial. O atacante possui acesso a um contêiner, e.g., por meio de uma vulnerabilidade numa aplicação web, e então expande seu ataque para os outros nós na rede.

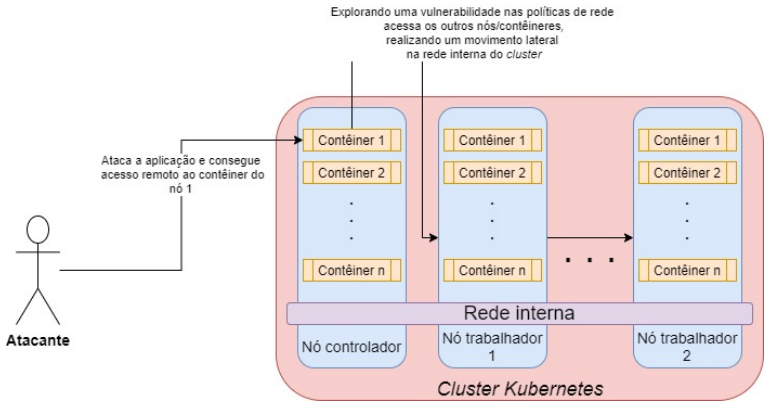

Figura 2. Exemplo de movimento lateral dentro de um *cluster* Kubernetes

A comunicação do Kubernetes é realizada na arquitetura IP-*per-pod*. Os contêineres dentro de cada *pod* compartilham a mesma pilha de rede, comunicando-se via *localhost*. Sendo assim, um contêiner comprometido e conectado a rede possui acesso aos outros contêineres no mesmo namespace [Minna et al., 2021]. Um contêiner comprometido pode não ter muita informação, porém, ao acessar outro, este pode conter informações sensíveis ou privilégios que possibilitem ao atacante invadir outros *pods* ou até mesmo outros nós.

Um ataque que ganhou visibilidade foi o ”Azurescape”, o qual um atacante executava um contêiner com uma imagem maliciosa, nesse caso, explorando a CVE-2019-5736. A partir disto monitorava o tráfego de rede no controlador dos nós, para capturar uma chave de autorização, que então era possível conseguir acesso remoto com permissão de administrador ao servidor da API do Kubernetes [Seals, 2021].

5. Considerações & Trabalhos futuros

O Kubernetes é considerado um dos orquestradores com mais recursos atualmente. Este número expressivo de recursos aumenta sua superfície de ataque. As vulnerabilidades podem ser exploradas nos mais diversos locais do *cluster*, afetando a integridade da rede, ou até comprometendo o nó gerenciador.

As políticas de segurança podem evitar diversos ataques, e.g., SSRF, mas quando mal configuradas podem gerar vulnerabilidades ao *cluster*. Ou até roubo de recursos computacionais, como descrito na Subseção 4.1. O impacto do ataque pode ser maior caso o atacante consiga realizar o movimento lateral pela rede afetando os outros contêineres. Como no caso visto na Subseção 4.2, onde um atacante conseguia acesso de administrador ao servidor da API do Kubernetes. Sendo assim, focou-se mais nestas duas vulnerabilidades, pois, são as mais comuns de ocorrerem em ambientes reais, e o movimento lateral é muito característico do *cluster* Kubernetes devido a sua comunicação interna.

Os trabalhos futuros incluem a implementação de um *cluster* Kubernetes para testar a superfície de ataque das vulnerabilidades descritas neste trabalho e outras. Os resultados devem ser analisados para identificar o impacto destas no funcionamento do *cluster*.

Agradecimentos: Os autores agradecem o apoio do LabP2D/UDESC e da FAPESC.

Referências

Budigiri, G., Baumann, C., Muhlberg, J., Truyen, E., and Joosen, W. (2021). Network policies in kubernetes: Performance evaluation and security analysis. pages 407–412.

David, R. B. and Barr, A. (2021). Kubernetes autoscaling: Yoyo attack vulnerability and mitigation. In *CLOSER*.

Jensen, N. and Miers, C. C. (2020). Segurança de contêineres: taxonomia baseada na arquitetura. In *Anais da XVIII Escola Regional de Redes de Computadores*, pages 128–134, Porto Alegre, RS, Brasil. SBC.

Kronser, A. (2020). Common vulnerabilities and exposures : Analyzing the development of computer security threats. Master's thesis, Helsingin yliopisto.

Kubernetes (2021). Kubernetes documentation. https://kubernetes.io/docs.

Marathe, N., Gandhi, A., and Shah, J. M. (2019). Docker swarm and kubernetes in cloud computing environment. In *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, pages 179–184.

Mercl, L. and Pavlik, J. (2018). The comparison of container orchestrators.

Minna, F., Blaise, A., Rebecchi, F., Chandrasekaran, B., and Massacci, F. (2021). Understanding the security implications of kubernetes networking. *IEEE Security Privacy*, pages 2–12.

MITRE (CVE Terminology) (2021). Cve terminology. https://cve.mitre.org/about/terminology.html.

Nyanchama, M. (2005). Enterprise vulnerability management and its role in information security management. *Information Systems Security*, 14:29–56.

Red Hat (2021). Kubernetes adoption, security, and market trends report 2021s. https://www.redhat.com/en/resources/kubernetes-adoption-security-market-trends-2021-overview.

Robinson, R. and Fishbein, N. (2021). New attacks on kubernetes via misconfigured argo workflows. https://www.intezer.com/blog/container-security/new-attacks-on-kubernetes-via-misconfigured-argo-workflows/.

Seals, T. (2021). 'Azurescape' Kubernetes Attack Allows Cross-Container Cloud Compromise. https://threatpost.com/azurescape-kubernetes-attack-container-cloud-compromise/169319/.

Shamim, M. S. I., Bhuiyan, F. A., and Rahman, A. (2020). Xi commandments of kubernetes security: A systematization of knowledge related to kubernetes security practices. *2020 IEEE Secure Development (SecDev)*, pages 58–64.

Takahashi, K., Aida, K., Tanjo, T., and Sun, J. (2018). A portable load balancer for kubernetes cluster. In *Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region*, HPC Asia 2018, page 222–231, New York, NY, USA. Association for Computing Machinery.

Wilkens, F., Haas, S., Kaaser, D., Kling, P., and Fischer, M. (2019). Towards efficient reconstruction of attacker lateral movement. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, ARES '19, New York, NY, USA. Association for Computing Machinery.