Análise de segurança dos orquestradores Kubernetes, Docker Swarm e Apache Mesos baseada no CVE / MITRE

Nikolas Jensen[1], **Charles Christian 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

Abstract. *The container usage has growing in the last years. Some containerized applications consists of a considerable number of containers which need to be managed. In this context, container orchestration is essential to conteinerized applications being managed efficiently. However, the orchestrators bring a new element which can introduce new vulnerabilities. Our article presents a security analysis of three key orchestrators: Docker Swarm, Kubernetes, and Apache Mesos. Our analysis is based in data from Common Vulnerabilities Exposure (CVEs) repository, being correlated to the major threats related by MITRE. Thus, we presented a matrix correlating the CVEs and MITRE threats in order to indicate the major impacts which one container manager needs to take into account related to risk mitigation.*

Resumo. *O uso de contêineres está crescendo expressivamente nos últimos anos. Aplicações conteinerizadas consistem em uma quantidade expressiva de contêineres que necessita ser gerenciada. Neste contexto, a orquestração de contêineres é essencial para que aplicações possam ter seus contêineres administrados eficientemente. Entretanto, os orquestradores trazem novos elementos que podem introduzir novas vulnerabilidades. Assim, uma análise de aspectos de segurança dos orquestradores é essencial. Este artigo faz uma análise de segurança nos três principais orquestradores: Docker Swarm, Kubernetes e Apache Mesos. A análise é baseada em dados dos repositório das CVEs que são correlacionadas com as principais ameaças relacionadas pelo MITRE. Como resultado é apresentada uma matriz que correlaciona as CVEs com as ameaças MITRE, indicando os principais impactos que devem ser considerados na mitigação dos riscos.*

1. Introdução

Os orquestradores oferecem diversas possibilidades aos utilizadores, podendo tanto realizar o serviço de *frontend* quanto de *backend* [Marathe et al., 2019], como 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. Existem diversos orquestradores, sendo que neste trabalho são abordados: Kubernetes, Docker Swarm e Apache Mesos [Meyerson, 2016,Research, 2020]. O Kubernetes é conhecido por ser mais popular e robusto, mas o Docker Swarm possui uma configuração simplificada. O Apache Mesos é mais generalista, criado para ser portável a qualquer tipo de *cluster* [Hindman et al., 2011]. Os orquestradores também possuem diversas falhas de segurança, as quais são descobertas gradativamente.

O CVE Editorial Board é um grupo de trabalho que vem desde 1999 organizando e catalogando vulnerabilidades em uma lista estruturada chamada de lista CVE, sendo mantida pelo MITRE. Neste artigo, o objetivo é identificar as principais vulnerabilidades conhecidas em cada orquestrador através de uma busca extensiva na base CVE, relacionado-as com base em uma base de ameaças do MITRE para produzir uma matriz de impactos e riscos de cada orquestrador. A análise é necessária para que haja o entendimento de quais são os principais pontos vulneráveis nos orquestradores atualmente, com base em repositórios públicos. No momento em que se utiliza um orquestrador em uma versão que possui vulnerabilidades conhecidas, os envolvidos já estão em risco.

O artigo está organizado como segue. Na Seção 2 são abordados os principais aspectos dos orquestradores. Na Seção 3 é feita uma introdução a CVEs,com uma matriz contendo informações acerca das vulnerabilidades e CVEs nos orquestradores. Na Seção 4 são elencados ataques que exploram as vulnerabilidades listadas no CVE. Na Seção 5 é feita a identificação da superfície de ataque comum aos três orquestradores. A Seção 6 contém a análise dos ataques *vs.* vulnerabilidades *vs.* superfície de ataque, e alguns exemplos de CVEs em orquestradores contendo os três objetos de análise.

2. Docker Swarm, Kubernetes e Apache Mesos

Os orquestradores possuem diversos elementos, os quais alguns são compartilhados, por serem comuns aos orquestradores de contêineres. Porém, cada arquitetura é única e têm suas funcionalidades, tanto o Kubernetes, Docker Swarm e Apache Mesos possuem seus benefícios e problemas.

2.1. Docker Swarm

O Docker Swarm consiste em uma ou mais máquinas que contenham o Docker e executam no modo *swarm*, o qual pode ser configurado por meio de um arquivo YAML [Docker, 2021a], ou seja, os contêineres executam em um *cluster*, e então é delegado um gerenciador e trabalhadores. É possível configurar o *swarm* da maneira que o utilizador desejar, i.e., o número de réplicas de um contêiner, a forma de comunicação de rede, os recursos de armazenamento disponíveis, portas a serem expostas e entre outras modificações a critério do desenvolvedor [Docker, 2021c]. A Figura 1 apresenta a arquitetura do Docker Swarm em alto nível, uma vez que esta não aborda a comunicação, políticas de rede, *tokens*, *Application Programming Interface* (API), etc.

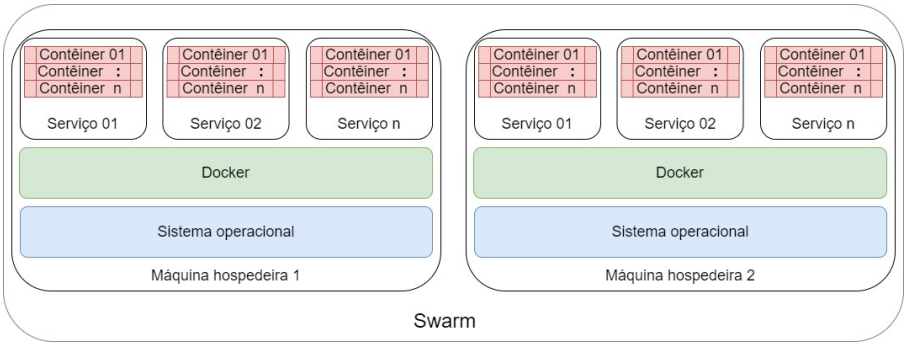

Figura 1. Arquitetura do Docker Swarm

A máquina hospedeira pode ser tanto física quanto virtual, e para uma nova máquina ser inserida no *cluster* é necessário que esta possua o *token* de acesso [Docker, 2021c]. No caso da Figura 1, uma ou mais das máquinas hospedeiras podem ser as gerenciadoras do *cluster* e as restantes são trabalhadoras.

O Docker possui uma API, a qual é utilizada para realizar ações remotas nos contêineres do hospedeiro, e também, realizar o manejo do *swarm*. O acesso a API é crítica, com isto é possível verificar os *tokens* de acesso do *swarm*. Além disso, as informações do Docker que está sendo executando, listar os contêineres e as imagens do sistema [Docker, 2021b].

O Docker Swarm, por ser uma ferramenta utilizada no próprio *daemon*, somente funciona com contêineres Docker [Docker, 2021c], com isso, um invasor já possui informações sobre a tecnologia utilizada e isso pode ser usado para explorar o serviço. Além disso, quando o Docker possui alguma vulnerabilidade é possível que impacte significativamente no *cluster*.

2.2. Apache Mesos

O Apache Mesos, não possui suporte nativo à orquestração de contêineres, necessitando um *framework* para realizar esta tarefa. Este orquestrador possui o objetivo de ser escalável a diferentes *clusters*, utilizando diferentes tecnologias, não utilizando uma tecnologia específica para o desenvolvimento. O Mesos minimiza a necessidade de mudanças no sistema, desta maneira tornando-o simples e robusto, oferecendo uma ferramenta eficiente para aplicações com alto nível de abstração [Hindman et al., 2011].

É possível utilizar uma variedade de *frameworks* com diferentes fins, e o orquestrador irá oferecer recursos para todos estes *frameworks* dentro do *cluster* e agendar suas tarefas. A Figura 2 ilustra a arquitetura do Apache Mesos orquestrando contêineres, o qual utiliza dois *frameworks* para fins de exemplificação.

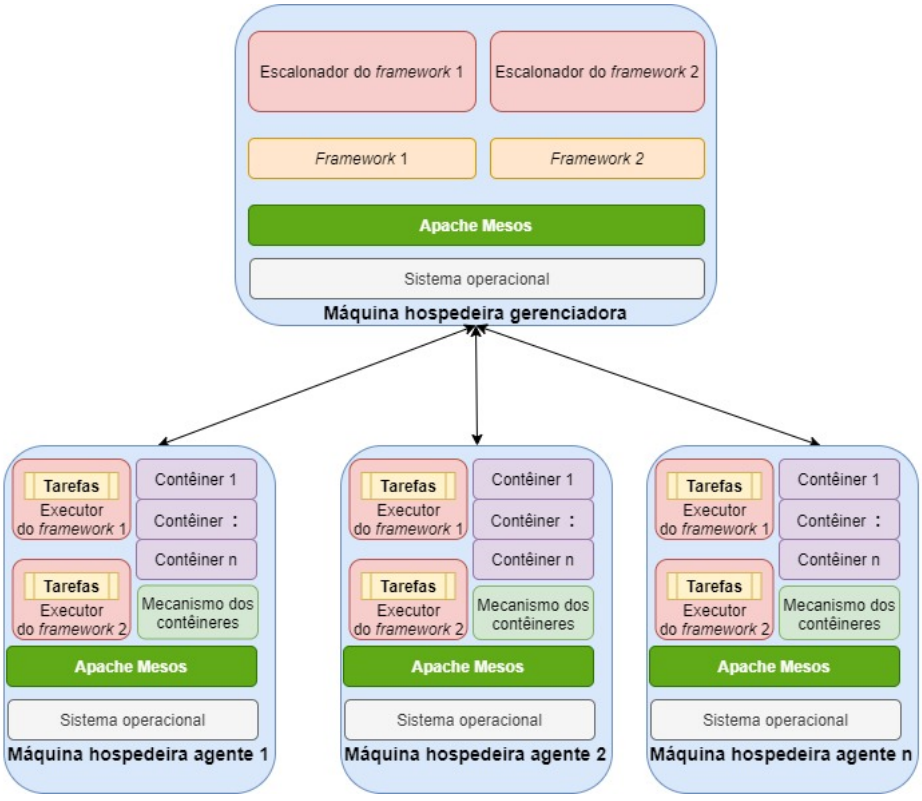

Figura 2. Arquitetura do Apache Mesos.

O *framework* oferece um escalonador no nó gerenciador para alocação de recursos e tarefas, e um executor em cada nó trabalhador para realizar as tarefas. O Apache Mesos possui uma API de acesso, com a qual é possível requisitar diversas informações

do *cluster*. No Mesos existem chamadas de API que somente um gerenciador aceita e outros que trabalhadores também aceitam, estas requisições aos nós revelam informações sensíveis ao *cluster* assim como de arquivos.

2.3. Kubernetes

O Kubernetes possui um sistema declarativo de configuração, ou seja, utiliza-se arquivo do tipo YAML para configurá-lo da maneira que o desenvolvedor desejar. A Figura 3 mostra a arquitetura do Kubernetes em uma visão superficial, a qual não mostra os componentes do gerenciador, como o *load balancer* e a API.

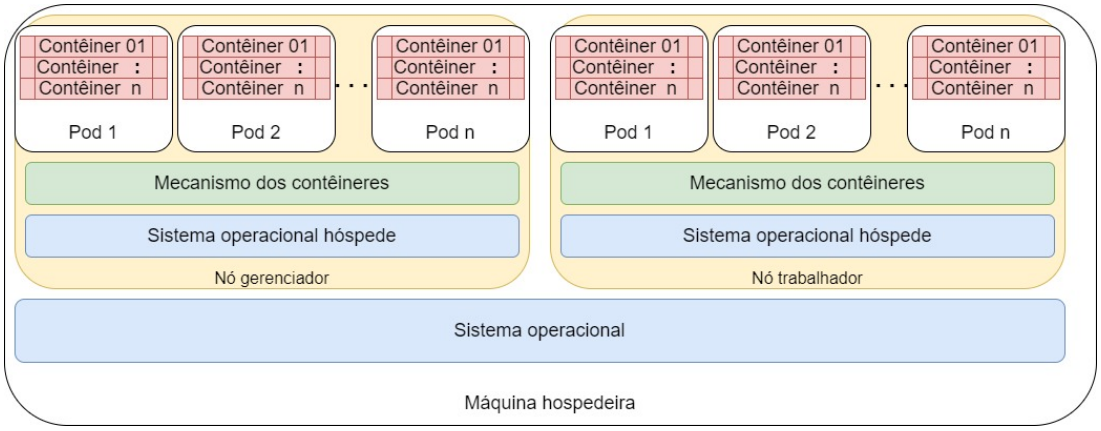

Figura 3. Arquitetura do Kubernetes.

É possível visualizar a arquitetura do *cluster*, no qual utiliza dois nós, um gerenciador e outro trabalhador, é possível incluir quantos trabalhadores forem necessários. Além disso, dentro de cada nó existem os *pods*, os quais irão possuir os contêineres de execução da aplicação e que serão gerenciados pelo orquestrador [Kubernetes, 2021].

O Kubernetes possui um painel de controle, que mantém todos os dados do *cluster* no qual é possível realizar modificações no sistema. O principal componente do painel de controle é a API do Kubernetes [Kubernetes, 2021], logo a segurança desta é essencial ao *cluster*, quem obtém acesso a API possui o controle do *cluster*.

3. Vulnerabilidades conhecidas

O processo de descobrimento de vulnerabilidades está envolvido tanto no meio corporativo privado quanto em organização públicas ou sem fins lucrativos. Diversas empresas se segurança da informação se encontram no mercado na atualidade, e oferecem dos mais variados serviços, desde testes de penetração até implementação de protocolos ou serviços. Como é o caso da Trend Micro [Trend Micro, 2021] que oferece produtos e serviços e da subdivisão da Google, [Google Zero Project, 2021] a qual busca por vulnerabilidades nos produtos utilizados e desenvolvidos na empresa.

Nos últimos cinco anos os programas de recompensa por *bug* vêm crescendo seu número de utilizadores [Malladi and Subramanian, 2020]. Tanto órgãos públicos (e.g., departamento de defesa dos EUA), quanto empresas (e.g., Apple e Facebook) estão aderindo a estes programas de recompensa. Pesquisadores de todo o mundo podem procurar por vulnerabilidades e, caso encontrem, serão recompensados por isso sem estarem necessariamente contratados pela instituição.

Além da preocupação das empresas, que tipicamente estão relacionadas aos produtos e serviços que oferecem, há também a preocupação por parte de organizações em possuir uma base de vulnerabilidades de um modo estruturado e com informações verificadas criteriosamente. Neste contexto, entram as CVEs que podem ter origem de uma empresa ou de algum pesquisador voluntário, e estas possuem impacto em algum produto ou sistema. Segundo o [MITRE (CVE Terminology), 2021], vulnerabilidade e CVEs são diferentes. Neste contexto, uma vulnerabilidade é definida como uma falha no *software, hardware, firmware* ou em um componente do serviço, resultando em uma fraqueza que seja possível explorar e causando um impacto negativo. A CVE é definida como um erro de configuração em um software que habilita o acesso de informações e recursos do sistema [Kronser, 2020].

Atrelada às CVEs, existem as *Common Weakness Enumeration* (CWE), as quais classificam as CVEs em determinadas categorias também criadas pelo MITRE, sendo de ordem maior que a CVE. Quando é descoberta uma vulnerabilidade, é dado um prazo ao fabricante para resolvê-la, após esse tempo esta é exposta ao público [McQueen et al., 2011]. Sendo assim, cada CVE de cada orquestrador possui suas vulnerabilidades relacionadas. A Tabela 1 foi elaborada usando o banco de dados da [MITRE (CVE), 2021], a qual reúne as vulnerabilidades conhecidas e relatadas a CVE.

Tabela 1. CVEs de cada orquestrador, identificadas até 26/07/2021.

	Kubernetes	Docker Swarm	Apache Mesos
Negação de serviço	CVE-2019-11248, CVE-2019-11253, CVE-2020-8557, CVE-2020-8552, CVE-2020-8551, CVE-2019-11254,	CVE-2016-6595 CVE-2017-14992 CVE-2019-16884 CVE-2021-21285	CVE-2017-7687 CVE-2017-9790 CVE-2018-1330
Acesso a dados sensíveis	CVE-2019-11248, CVE-2015-7528, CVE-2018-1002100, CVE-2019-11250, CVE-2019-11243, CVE-2020-8566, CVE-2020-8564, CVE-2019-11252	CVE-2014-5277 CVE-2015-3630 CVE-2019-5736 CVE-2018-15664	CVE-2018-8023
Acesso não autorizado	CVE-2016-7075, CVE-2019-11248, CVE-2016-1905, CVE-2017-1002102, CVE-2019-11245, CVE-2020-8559, CVE-2020-8558, CVE-2020-8555, CVE-2019-11251, CVE-2020-8554	CVE-2014-5277 CVE-2016-9962 CVE-2018-15514 CVE-2014-5282 CVE-2019-5736	CVE-2018-1000420, CVE-2019-0204
Execução de código arbitrário	CVE-2016-1906, CVE-2018-1002101, CVE-2019-1002101	CVE-2014-5282 CVE-2019-5736 CVE-2014-0048	CVE-2019-0204
Executar ações não autorizadas	CVE-2019-11247	CVE-2014-5278	não identificado

Na Tabela 1 foram elencadas diversas vulnerabilidades conhecidas e então categorizadas em cada tipo de ataque. Com base nas vulnerabilidade elencada em cada CVE pela [MITRE (CWE), 2021], a qual lista as vulnerabilidades em sistemas computacionais. O critério utilizado foi o potencial de ataque a essas vulnerabilidades, i.e., dos ataques (Seção 4) os quais podem ser aplicados nestas vulnerabilidades.

O Kubernetes possui mais CVEs relacionadas, uma vez que é o orquestrador mais utilizado em 2020, segundo pesquisa realizada com 540 profissionais de tecnologia [Research, 2020]. No caso do Docker Swarm as CVEs estão categorizadas juntamente com o Docker, para a elaboração da Tabela 1 foram analisadas quais CVEs têm impacto no funcionamento ou nos serviços de um *swarm*.

Até o momento de produção deste artigo, o único orquestrador que possui apenas uma CVE no ano de 2021 é o Docker Swarm, o qual coloca os contêineres do *cluster* em ameaça de negação de serviço explorando esta CVE. O impacto de cada CVE é variado, e.g., a CVE-2019-5736 do Docker afeta a máquina hospedeira, com isso, impactando diretamente a segurança e integridade do *cluster*.

4. Ataques

Um ataque é uma tentativa de destruir, expor, alterar, desabilitar, roubar ou ganhar acesso não autorizado ou fazer uso não autorizado de qualquer coisa que possua valor para a organização [ISO, 2009]. Todo ataque possui: uma motivação, atores e um alvo [Kadivar, 2014], a motivação do porque quer utilizar o ataque, no mínimo dois atores envolvidos, atacante e vítima, e uma vulnerabilidade a ser explorada.

Ataques são utilizados para danificar um sistema ou modificar suas rotinas de operação, explorando uma vulnerabilidade utilizando ferramentas e técnicas [Humayun et al., 2020]. Existem diversas maneiras de comprometer a segurança de um *cluster*, dessa forma, foram elencados os principais tipos de ataques descritos na literatura. É possível comprometer a integridade de um *cluster* com acesso ao terminal de um contêiner, com isso, acessando o sistema de arquivos e possivelmente dados sensíveis.

Uma vez dentro de um contêiner, é possível realizar um ataque horizontal, ou seja, caso a aplicação esteja vulnerável e um invasor consiga acesso ao contêiner, é possível invadir outros contêineres e serviços na mesma rede do *cluster*. Além disso, uma vez que um invasor tenha acesso a um serviço *web* provido por um orquestrador, este pode utilizar de ataques para conseguir informações sensíveis do *cluster*. Logo, analisando as vulnerabilidades da Tabela 1, foram identificados que os principais tipos de ataques são:

- *Server Side Request Forgery* (SSRF): ocorre quando uma URL requisita um serviço localizado internamente ou externamente, e.g., uma imagem. Com isso, um atacante pode redirecionar esta requisição para o servidor interno da aplicação, o qual não está disponível para acesso externo, porém, como o servidor irá receber a requisição da aplicação esta é aceita. É possível então requisitar arquivos internos e confidenciais, ou até, executar comandos arbitrários para conseguir acesso remoto ao terminal contêiner.
- *Open redirect*: Ocorre quando uma página web realiza um redirecionamento de forma insegura, se alguém puder manipula-lá de maneira que redirecione para um site forjado por um atacante.
- *Man-in-the-middle*: Um ataque a nível de rede, o qual ocorre quando um invasor consiga interceptar os pacotes da rede ou os dados que transitam entre aplicações.
- Desserialização insegura: Ocorre quando é recebido um objeto não confiável, é convertido em dados a serem processados, mas devido à falta de verificação pode executar uma ação indevida.
- Travessia por diretórios: Quando um arquivo é buscado pelo seu nome e a entrada não é devidamente validada, o nome pode ser alterado de maneira que busque, sem autorização, outro arquivo qualquer em outro diretório da máquina.

Os ataques podem variar de acordo com a superfície de ataque (Seção 5), a qual distingue as diferentes partes de um *cluster*. Os ataques que ocorrem na aplicação que está sendo executada no orquestrador, e.g., um sistema *web*, possivelmente serão diferentes do quê os ocorridos na infraestrutura.

O impacto de um ataque ao *cluster* é variado, dependendo do que foi atacado e de que maneira. Dentre os impactos, pode ocorrer a destruição de dados, sequestro de recursos ou negação de serviços. Em 2020 tornou-se conhecido o caso de *cryptojacking* na Azure Cloud, na qual invasores sequestraram os recursos de *clusters* Kubernetes voltados à aprendizagem de máquina para mineração de *bitcoin* [Ramel, 2020]. Neste caso, os invasores buscam explorar o máximo possível do ambiente para atingir o seu objetivo, e garantir a persistência do ataque.

5. Identificação de superfície de ataque

A superfície de ataque de uma aplicação ou sistema tem impacto significativo em como será classificada a sua segurança, uma vez que quanto menor há uma tendência de ser mais seguro. Isto ocorre devido à definição de superfície de ataque, que pode variar dependendo de cada autor. [Theisen et al., 2018] cita três definições de diferentes autores, das quais é possível retirar que a superfície de ataque é a união de diversos serviços, protocolos, interfaces entre outros, que sejam visíveis aos usuários e aos recursos do sistema e podem proporcionar meios de ataque. Neste trabalho, uma superfície de ataque é definida como um conjunto de maneiras nas quais um invasor pode causar danos ao sistema [Manadhata and Wing, 2011]. Normalmente, a superfície de ataque é aumentada por serviços extras, i.e., *add-ons*. Contudo, cada vez mais a superfície de ataque envolve a infraestrutura e ferramentas de desenvolvimento, e.g., contêineres e orquestradores [Woody and Ellison, 2020]. A [MITRE (CWSS), 2021] elenca as vulnerabilidades contidas nas CVEs, com isso facilitando o entendimento de qual ataque é possível utilizar. Esta é organizada em três grupos de métricas, i.e., *base findings*, superfície de ataque e características do ambiente, cada uma com seus fatores de risco. A métrica utilizada neste trabalho é a superfície de ataque, devido à sua correlação com a exploração de vulnerabilidades em sistemas, esta contém os seguintes fatores:

- Privilégio necessário: o tipo de privilégios que o atacante já possui acesso ao código/funcionalidade com a vulnerabilidade a ser explorada.
- Camada de privilégio requerido: a camada operacional na qual o atacante deve ter privilégios para continuar a exploração.
- Vetor de acesso: é o canal pelo qual o atacante chega ao ponto com vulnerabilidades.
- Autenticação forte: a força da rotina de autenticação que protege a funcionalidade que possui vulnerabilidades.
- Nível de interação: as ações necessárias a serem feitas por vítimas humanas para realizar um ataque bem sucedido.
- Escopo de desenvolvimento: caso a vulnerabilidade se encontre inteiramente no desenvolvimento do software ou somente em uma parte.

As métricas utilizadas pela *Common Weakness Scoring System* (CWSS), indicam quais os principais pontos a serem analisados e observados quando estuda-se sobre superfície de ataque. A aplicação que o orquestrador está executando pode comprometer sua segurança, uma vez que uma vulnerabilidade encontrada neste pode conceder acesso ao sistema. Além disso, com a cultura de desenvolvimento contínuo e integrado, um atacante pode ter acesso a rede ou aos computadores dos desenvolvedores.

Os três orquestradores deste trabalho possuem API, a qual serve para o controle do *cluster* por inteiro. Com acesso a API o atacante pode executar contêineres ou acessar dados sensíveis. Expor um serviço interno da aplicação como a API do orquestrador, coloca a segurança do *cluster* em risco [Weizman, 2021].

Quando a aplicação possui um contêiner exposto, diversos riscos ocorrem. Mesmo que o contêiner não possua acesso de administrador, é possível realizar um movimento lateral (Figura 4) e com isso podendo acessar / invadir outros contêineres, ou até mesmo outros nós do *cluster*.

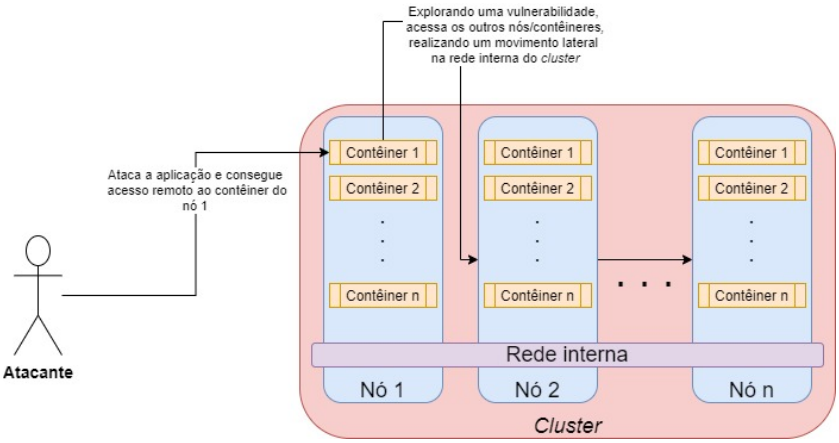

Figura 4. Movimento lateral por um atacante com acesso a um contêiner.

Na Figura 4 é ilustrado a possibilidade de um atacante possuir acesso a um contêiner qualquer de uma aplicação, i.e., explorando uma vulnerabilidade na aplicação, exposição, etc. Com este acesso este pode explorar vulnerabilidades ou CVEs para realizar o movimento lateral no *cluster* e assim acessar outros nós ou contêineres.

O contêiner passa a ser uma superfície da ataque caso se utilize uma ou mais imagens comprometidas, as quais ofereçam acesso não autorizado ao contêiner. Além disso, caso o orquestrador possua alguma vulnerabilidade que possibilite um ataque de *man-in-the-middle* é possível que o atacante consiga informações confidenciais do sistema. Os três orquestradores, analisados neste trabalho, possuem, no mínimo, três similaridades, i.e., a API, contêineres e uma aplicação (Figura 5).

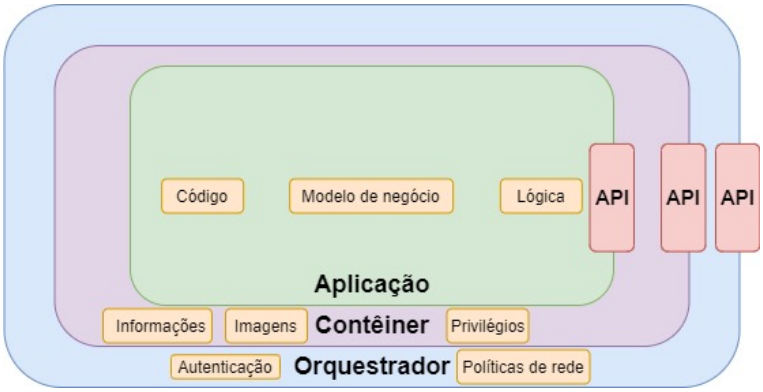

Figura 5. Superfície de ataque com alguns dos seus componentes que são passíveis a sofrerem um ataque.

Além disso, estas fazem parte da superfície de ataque do orquestrador, possuindo caminhos que possibilitem a invasão deste. Na Figura 5 percebe-se que há uma conexão entre estes, o orquestrador engloba todos os elementos do *cluster*, e os contêineres se comunicam por meio da API.

6. Análise: Ataques vs. Vulnerabilidades vs. Superfície

Alguns ataques que podem ser utilizados contra orquestradores foram listados na Seção 4, ressalta-se que não é uma lista exaustiva, uma vez que foi baseada exclusivamente nas CVEs. A superfície de ataque, detalhada na Seção 5, está ligada a estes outros dois conceitos, uma vez que esta irá relacionar quais e quantas camadas estão vulneráveis aos

ataques. Usualmente, aplicações que empregam diversos recursos e APIs geralmente possuem uma maior superfície de ataque, e.g., uma página web estática, i.e., utilizando somente HTML, possuirá uma superfície de ataque menor do que outra aplicação com diversas funcionalidades utilizando Javascript, PHP e MariaDB. Todos os orquestradores estudados neste trabalho possuem vulnerabilidades descobertas e, consequentemente, podem ser explorados por ataques quando a superfície de ataque for acessível. Exemplificando, a exploração em cada um dos orquestradores:

- Na CVE-2020-8554 do Kubernetes é possível realizar o ataque *man-in-the-middle* o qual ocorre, devido a uma autorização incorreta, uma vulnerabilidade que autoriza um atacante a criar um novo serviço com *Internet Protocol* (IP) possibilitando espionar os serviços vizinhos. É relatada a exploração a falta de controle de privilégios do *cluster*, um usuário que tenha acesso a um contêiner ou a API, como visto na Seção 2.3 com permissão de criar serviços e podem controlar *pods*.
- O Apache Mesos possui a CVE-2018-11793 na qual é possível ocorrer o ataque de negação de serviço, explorando uma vulnerabilidade de *overflow* da pilha do sistema operacional. O *overflow* ocorre devido a uma falha na restrição de uso do *buffer* da memória, permitindo que um atacante explore a vulnerabilidade. A CVE mostra que o *cluster* é vulnerável por uma falta de verificação dos dados recebidos, uma vez que é possível enviar um dado e o sistema processá-lo infinitamente de maneira que cause negação de serviço. Pode ser explorado pela aplicação que está no orquestrador, atingindo o sistema.
- O Docker possui a CVE-2018-15664, a qual descreve um ataque a uma vulnerabilidade conhecida como *race condition* a qual ocorre numa execução concorrente com recursos compartilhados sem a correta sincronização. Sendo assim, um processo pode utilizar os dados de outro processo, no caso desta CVE utiliza-se o ataque *symlink-exchange* no qual é possível montar o sistema de arquivos hospedeiro dentro do contêiner acessado. Esta CVE apresenta uma falha de segurança no orquestrador Docker Swarm, uma vez que é possível escrever e ler dados do sistema de arquivos do hospedeiro e assim comprometer a integridade do *cluster*.

Devido a extensão da superfície de ataque, não existe um método definitivo de defesa para um orquestrador. Porém, existem diversos princípios e práticas de defesa para melhorar a segurança do *cluster*, mitigando tanto vulnerabilidades como ataques. Cada componente de um orquestrador deve ter sua segurança averiguada, uma vez que apresenta superfície de ataque, i.e., segurança da rede, imagens e dos contêineres [Mytilinakis, 2020]. Como todos os orquestradores abordados no artigo possuem uma API para controle central, uma boa prática recomendada é não deixar a API exposta à Internet.

Os contêineres utilizados apresentam risco significativo para o *cluster*, por isso, é necessário utilizar imagens de fontes confiáveis, uma vez que podem conter *backdoor* ou um *malware* [Weizman, 2021]. Utilizar políticas de rede atualizadas e efetivas, para que dificulte a espionagem do tráfego de dados, e caso um contêiner seja comprometido, não ocorra movimento lateral, afetando outras partes do *cluster*.

Nas Tabelas 2, 3 e 4 nota-se a relação entre ameaças específicas à um *cluster* que impactam este de diversas maneiras. Cada CVE explora determinada vulnerabilidade e ameaça determinados componentes e situações, e.g., uma CVE que causa como impacto uma negação de serviço ao *cluster* por meio de requisições HTTP a API, coloca uma ameaça ao acesso livre e exposição da API e possui uma severidade. A severidade é baseada no *Common Vulnerability Scoring System* (CVSS), o qual possui um grupo de três

métricas: *base, temporal* e *environment*, a métrica *base* classifica as CVEs em um escore que varia de 0 a 10, podendo sofrer alterações pelas outras duas métricas. A severidade pode ser classificada de acordo com a nota, como especificado [National Vulnerability Database, 2021]: Nenhuma: 0.0; Baixa: 0.1-3.9; Média: 4.0-6.9; Alta: 7.0-8.9; e Crítica: 9.0-10.0.

Tabela 2. Relação entre ameaças, CVEs e o impacto no Kubernetes.

Ameaça	CVE	Severidade	Superfície de ataque	Impacto
Acesso a rede interna do *cluster*	CVE-2020-8554	6.0	API	Destruição de dados
	CVE-2020-8558	5.8		
	CVE-2020-8559	6.0	API / Aplicação	Destruição de dados / Roubo de recursos
	CVE-2018-1002105	7.5		
Listar os segredos do *cluster*	CVE-2020-8566	2.1	API	Destruição de dados
	CVE-2020-8565	2.1		
	CVE-2020-8563	2.1		
Credenciais das aplicações em arquivos de configuração	CVE-2020-8564	2.1		
Instance Metadata API	CVE-2020-8555	3.5	API / Aplicação	
Acesso ao servidor da API	CVE-2020-8552	4.0	API	Negação de serviço
	CVE-2019-1002100	4.0		
	CVE-2019-11253	5.0		
	CVE-2019-11250	3.5		Destruição de dados
Acessar credenciais	CVE-2019-11250	3.5		
	CVE-2019-11252	5.0	API / Aplicação	
	CVE-2019-11243	4.3		
Escrever arquivos no hospedeiro	CVE-2019-1002101	5.8	Contêiner	Destruição de dados / Roubo de recursos
	CVE-2019-11249	5.8	Contêiner / Aplicação	
	CVE-2019-11246	5.8	Contêiner / Aplicação	
Injeção de comandos no contêiner	CVE-2019-1002101	5.8	Contêiner	
Modificar arquivos do cluster	CVE-2018-1002100	3.6	Contêiner / API	Destruição de dados
	CVE-2017-1002102	6.3	Contêiner	
	CVE-2016-1906	10.0	Aplicação	Destruição de dados / Roubo de recursos
Acesso privilegiado a recursos	CVE-2019-1002101	5.8	Contêiner	
	CVE-2016-1905	4.0	API	
Burlar a autenticação	CVE-2016-7075	6.8	Aplicação	Destruição de dados
	CVE-2017-1002100	4.0		
Interface sensível exposta	CVE-2017-1002100	4.0		
	CVE-2015-7528	5.0	API	
	CVE-2016-1906	10.0	Aplicação	Destruição de dados / Roubo de recursos
	CVE-2016-1905	4.0	API	

Uma CVE pode estar relacionada a uma ou mais ameaças, dependendo de sua definição, e.g., a CVE-2017-1002100 (Tabela 2) ameaça burlar a autenticação de um volume persistente por meio de uma interface sensível exposta a Internet na qual este possa realizar requisições. O impacto pode ser variado, no qual a destruição de dados se refere a todo e qualquer impacto nos dados, desde sua integridade até sua confidencialidade. Nesse contexto, os impactos podem ser descritos da seguinte maneira para este trabalho [Weizman, 2021]:

- Destruição de dados: Toda e qualquer ação nos dados, que coloque em risco sua integridade, confidencialidade ou ambos.
- Roubo de recursos: Um atacante utiliza dos recursos da máquina em seu favor, e.g., utilizar a CPU para minerar *bitcoins*.
- Negação de serviço: Atacantes são capazes de deixar um ou mais serviços indisponíveis para os usuários, e.g., bloquear acesso à API, nós do *cluster* ou até deixar contêineres sem funcionamento.

Tabela 3. Relação entre ameaças, CVEs e o impacto no Docker Swarm.

Ameaça	CVE	Severidade	Superfície de ataque	Impacto
Similaridade no nome de um contêiner	CVE-2014-5278	4.3	Contêiner	Destruição de dados / Roubo de recursos
Acesso ao servidor da API	CVE-2021-21285	4.3	API	Negação de serviço
	CVE-2018-15664	6.2	API	Destruição de dados
	CVE-2014-5277	5.0	API	
	CVE-2014-0048	7.5	API	Destruição de dados / Roubo de recursos / Negação de serviço
Acesso privilegiado a recursos	CVE-2019-16884	5.0	Contêiner	Destruição de dados
Escrever arquivos no hospedeiro	CVE-2019-5736	9.3	Contêiner	Destruição de dados / Roubo de recursos / Negação de serviço
	CVE-2016-9962	4.4	Contêiner	
	CVE-2018-15664	6.2	API	Destruição de dados
Imagens comprometidas	CVE-2019-5736	9.3	Contêiner	Destruição de dados / Roubo de recursos / Negação de serviço
	CVE-2014-5278	4.3	Contêiner	Destruição de dados / Roubo de recursos
	CVE-2014-5282	5.5	Contêiner / API	
	CVE-2017-14992	4.3	Contêiner	Negação de serviço
	CVE-2015-3630	7.2	Contêiner	Destruição de dados / Roubo de recursos / Negação de Serviço
Execução de comandos num contêiner	CVE-2014-0048	9.8	Contêiner	Destruição de dados / Roubo de recursos / Negação de serviço
	CVE-2016-9962	4.4	Contêiner	Destruição de dados / Roubo de recursos / Negação de serviço
Contêiner privilegiado	CVE-2016-9962	4.4	Contêiner	Destruição de dados / Roubo de recursos / Negação de serviço
Listar os segredos do *cluster*	CVE-2016-6595	4.0	API	Negação de serviço
Acesso a rede interna do *cluster*	CVE-2014-5277	5.0	Aplicação / API	Destruição de dados

Tabela 4. Relação entre ameaças, CVEs e o impacto no Apache Mesos.

Ameaça	CVE	Severidade	Superfície de ataque	Impacto
Vulnerabilidade na aplicação	CVE-2017-9790	5.0	Aplicação	Negação de serviço
	CVE-2017-7687	5.0	Aplicação	
	CVE-2018-1330	5.0	Aplicação	
	CVE-2018-11793	5.0	Aplicação	
Acesso ao servidor da API	CVE-2018-8023	4.3	API	Destruição de dados
Listar os segredos do *cluster*	CVE-2018-8023	4.3	API	
Imagens comprometidas	CVE-2019-0204	9.3	Contêiner	Destruição de dados / Roubo de recursos
Escrever arquivos no hospedeiro	CVE-2019-0204	9.3	Contêiner	
Acessar credenciais	CVE-2018-1000421	4.0	API / Aplicação	Destruição de dados
	CVE-2018-1000420	4.0	Aplicação	

É de suma importância manter o orquestrador com a última versão estável, na qual é possível que as vulnerabilidades tenham sido corrigidas. Garantir que as configurações estejam seguindo as políticas de segurança do orquestrador, e.g., nunca executar um contêiner como administrador. Toda CVE possui um impacto negativo no *cluster*, algumas afetam seu funcionamento, outras roubam dados confidenciais da aplicação, mas toda e qualquer vulnerabilidade deve ser mitigada para não haver risco aos utilizadores.

7. Considerações & Trabalhos futuros

Destaca-se que o Apache Mesos possui uma arquitetura de orquestração bem distinta das empregadas Docker Swarm e Kubernetes. O Apache Mesos, se seguir o princípio de possuir mais componentes significa possuir uma maior superfície de ataques e deveria ter mais vulnerabilidades. Contudo, com base na análise realizada percebe-se que o Apache Mesos é o que possui menos CVE entre os analisados. Um aspecto que pode justificar é o fato do Apache Mesos ser o orquestrador menos utilizado [Belagatti, 2019]: Kubernetes 42%, Docker Swarm 35% e Apache Mesos 3%.

Pode-se observar, que apesar de cada orquestrador possuir suas particularidades, tratando-se de vulnerabilidades e superfície de ataque há similaridades. Como estes possuem componentes em comum, e.g., nós, controlador, API, etc. possuem vulnerabilidades que podem ter ataques em comum, ou então formas de explorar a superfície de ataque comuns entre estes. Um ataque que pode utilizado em todos os orquestradores é o SSRF (Seção 4), o qual explora a rede interna do *cluster*, sendo possível conseguir credenciais, dados sensíveis, etc. Isto ocorre, pois, como descrito na Seção 2 todos os orquestradores possuem uma rede interna para gerência de recursos, dados e contêineres.

Os trabalhos futuros visam implementar e testar o impacto da exploração de uma CVE em *cluster* implementados sobre máquinas virtuais em uma nuvem OpenStack. Os experimentos terão por finalidade garantir aprimoramentos nas políticas de segurança e falhas de configuração, abrangendo diversos componentes, ambientes e arquiteturas.

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

Referências

Belagatti, P. (2019). "Docker Swarm or Kubernetes?": Is It the Right Question to Ask? https://dzone.com/articles/quotdocker-swarm-or-kubernetesquot-is-it-the-right.

Docker (2021a). Deploy to swarm. https://docs.docker.com/get-started/swarm-deploy/.

Docker (2021b). Engine api v1.24. https://docs.docker.com/engine/api/v1.24/.

Docker (2021c). Swarm mode key concepts. https://docs.docker.com/engine/swarm/key-concepts/.

Google Zero Project (2021). News and updates from the project zero team at google. https://googleprojectzero.blogspot.com/.

Hindman, B., Konwinski, A., Zaharia, M., Ghodsi, A., Joseph, A. D., Katz, R., Shenker, S., and Stoica, I. (2011). Mesos: A platform for fine-grained resource sharing in the data center. In *8th USENIX Symposium on Networked Systems Design and Implementation (NSDI 11)*, Boston, MA. USENIX Association.

Humayun, M., Niazi, M., Jhanjhi, N., Alshayeb, M., and Mahmood, S. (2020). Cyber security threats and vulnerabilities: A systematic mapping study. *Arabian Journal for Science and Engineering*, 45(4):3171–3189.

ISO (2009). Information technology – Security techniques – Information security management systems – Overview and vocabulary. Standard, International Organization for Standardization, Geneva, CH.

Kadivar, M. (2014). Cyber-attack attributes. *Technology Innovation Management Review*, 4:22–27.

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.

Malladi, S. S. and Subramanian, H. C. (2020). Bug bounty programs for cybersecurity: Practices, issues, and recommendations. *IEEE Software*, 37(1):31–39.

Manadhata, P. K. and Wing, J. M. (2011). An attack surface metric. *IEEE Transactions on Software Engineering*, 37(3):371–386.

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.

McQueen, M., Wright, J. L., and Wellman, L. (2011). Are vulnerability disclosure deadlines justified? In *2011 Third International Workshop on Security Measurements and Metrics*, pages 96–101.

Meyerson, J. (2016). Ben hindman on apache mesos. *IEEE Software*, 33(1):117–120.

MITRE (CVE) (2021). Cve details. https://www.cvedetails.com/.

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

MITRE (CWE) (2021). Common weakness enumeration (cwe). https://cwe.mitre.org/.

MITRE (CWSS) (2021). Common weakness scoring system (cwss). https://cwe.mitre.org/cwss/cwss_v1.0.1.html.

Mytilinakis, P. (2020). Attack methods and defenses on kubernetes. Master's thesis, Departamento de sistemas digitais.

National Vulnerability Database (2021). The common vulnerability scoring system (cvss). https://nvd.nist.gov/vuln-metrics/cvss.

Ramel, D. (2020). Hackers turn kubernetes machine learning to crypto mining in azure cloud. https://virtualizationreview.com/articles/2020/06/24/azure-cloud-exploit.aspx.

Research, . (2020). The state of container and kubernetes security.

Theisen, C., Munaiah, N., Al-Zyoud, M., Carver, J. C., Meneely, A., and Williams, L. (2018). Attack surface definitions: A systematic literature review. *Information and Software Technology*, 104:94–103.

Trend Micro (2021). A empresa. https://www.trendmicro.com/.

Weizman, Y. (2021). Secure containerized environments with updated threat matrix for kubernetes.

Woody, C. and Ellison, R. J. (2020). Attack surface analysis - reduce system and organizational risk. Technical report, Software Engineering Institute, Carnegie Mellon University.