# Segurança de contêineres: taxonomia baseada na arquitetura

**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 contêineres visam uma melhor eficiência no uso dos recursos computacionais ao mesmo tempo que fornecem um ambiente virtualizado para aplicações, mas crescem as preocupações sobre a segurança destes. Contudo, devido ao desfalque na segurança de sua utilização, as empresas e usuários se mantêm receosos ao implementá-lo nas suas aplicações. Este trabalho é propõe uma taxonomia mapeando problemas de segurança conhecidos dos contêineres. Este trabalho é propõe uma taxonomia mapeando problemas de segurança conhecidos dos contêineres. Esta taxonomia aborda os pontos principais desta tecnologia, sem foco num tipo de contêiner específico e também classificando os problemas em suas determinadas categorias.*

## 1. Introdução e Motivação

Devido à sua facilidade de implementação, desempenho que oferece à aplicação e sua reusabilidade [Hertz, 2016], a conteinerização está em constante crescimento. Sendo que em 2020 o mercado dos contêineres está previsto para alcançar $2.7 bilhões, $1.938 bilhão de crescimento em 4 anos [Sultan et al., 2019]. Os contêineres se tornam mais atrativos devido a sua leveza quando comparados a máquinas virtuais (MVs), e.g., menos *overhead*, rapidez na inicialização, desligamento e reinicialização [Casalicchio and Iannucci, 2018].

Inicialmente, os usuários dos contêineres, naturalmente, se preocuparam em entender como utilizá-lo e suas possíveis aplicações, e, após isso, surge a preocupação com a segurança das aplicações conteinerizadas. Uma das maiores barreiras para ampliação da adoção dos contêineres pelas empresas está relacionado às questões de segurança [Sultan et al., 2019]. Uma dificuldade ao lidar com contêineres é a sua ligação com o núcleo do sistema operacional (SO) hospedeiro, que diferentemente de uma MV que virtualiza todos os seus componentes, utiliza dos recursos do sistema hospedeiro para sua execução. Um contêiner inseguro pode comprometer a máquina hospedeira [ENISA, 2017], e assim, o dano pode abranger todas as aplicações em execução no hospedeiro.

Já existem taxonomias e documentos relacionados a segurança de contêineres, e.g., [Panizzon et al., 2019], [Lin et al., 2018], [Souppaya et al., 2017] e [Tomar et al., 2020]. Contudo, estas taxonomias usam uma visão mais genérica de agrupamento que tornam a identificação de qual aspecto de segurança está relacionado aos componentes de uso mais comum, tratando problemas de segurança sem especificar em que área se encontra, i.e., rede, hospedeiro, imagem ou mecanismo. Como em [Tomar et al., 2020], o qual trata do ataque *Man-in-the-Middle* como um ataque ao mecanismo do contêiner, onde neste trabalho entra na classificação da segurança de rede.

Este trabalho apresenta uma proposta de classificação dos principais aspectos de seguranças em contêineres orientada ao ecossistema no qual estão inseridos, i.e., segurança de rede, segurança do hospedeiro, segurança do mecanismo de contêiner e segurança da imagem. Esta classificação visa auxiliar na rápida identificação dos principais aspectos relacionados ao ecossistema. As demais taxonomias continuam válidas, mas o propósito desta proposta é organizar as questões de segurança em grupos mais relacionados a prática.

O artigo está organizado como segue. A Seção 2 identifica as principais taxonomias, bem como suas limitações para rápida associação entre categoria e aspectos práticos. A Seção 3 lista a proposta de organização da nova taxonomia e suas classificações. Por fim, a Seção 4 relata as principais considerações e trabalhos futuros.

## 2. Trabalhos relacionados

A taxonomia de [Panizzon et al., 2019] trata de maneira mais generalizada os contêineres, sendo que, o primeiro nível da taxonomia é constituído por três dimensões: conformidade, privacidade e arquitetura. Este trabalho traz mais abrangência tanto para o tema de contêineres, quanto para o ecossistema, saindo da aplicação direta dos contêineres para implicações no mundo real, e.g., questões legais. O trabalho de [Lin et al., 2018] utiliza exemplos de ataques específicos e de maneira direta, com isso, mostra como falhas de segurança implicam em sistemas reais, i.e., vazamento de informações sensíveis, controle remoto, *Denial of Service* (DoS) e o ganho de privilégios. Em [Souppaya et al., 2017] é realizada uma pesquisa acerca de todos os riscos envolvendo contêineres e seu ecossistema abrangendo diversos tópicos de suas tecnologias de maneira generalizada, e com nível alto. [Tomar et al., 2020] realiza uma taxonomia de ataque com três camadas, com foco na segurança da tecnologia *Docker*, utilizada para manejo de contêineres, este utiliza quatro dimensões na segunda camada, sendo: contêiner, aplicação, plataforma *Docker* e hospedeiro. Estes trabalhos foram escolhidos tendo em vista um alinhamento com a proposta deste, ou seja, uma taxonomia elencando problemas envolvendo a conteinerização. Realizar generalização nos trabalhos ocorrem tratando dos problemas de segurança em baixo nível, com aplicações práticas e, algumas vezes, mostrando cenários de ataque, sem necessariamente mostrar qual a base ou como ocorrem os problemas. Todas estas voltadas para sua segurança, tratando de maneira ampla dentro da tecnologia *Docker* a qual foi abordada pelos autores (Tabela 1).

**Tabela 1. Comparação das taxonomias estudadas.**

| **Critério** | [Panizzon et al., 2019] | [Lin et al., 2018] | [Souppaya et al., 2017] | [Tomar et al., 2020] |
|---|---|---|---|---|
| Realiza generalização? | Sim | Não | Sim | Não |
| Aborda segurança de rede? | Sim | Sim | Sim | Sim |
| Aborda segurança do hospedeiro? | Sim | Sim | Sim | Sim |
| Aborda segurança da aplicação? | Não | Sim | Sim | Sim |
| Aborda segurança da imagem? | Sim | Não | Sim | Sim |

A Figura 1 ilustra o ecossistema de contêineres, ressaltando os critérios da Tabela 1.

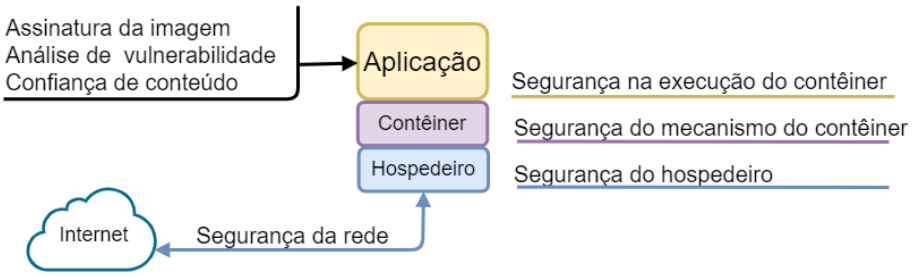


**Figura 1. Ecossistema de segurança dos contêineres.**

## 3. Proposta de classificação

Este trabalho apresenta uma proposta de classificação baseando-se no ecossistema que o contêiner insere-se, sendo que aborda como principais tópicos: segurança do hospedeiro, do mecanismo do contêiner, da rede e da imagem. A Figura 2 ilustra a taxonomia abordada neste trabalho. Uma breve explicação das categorias:

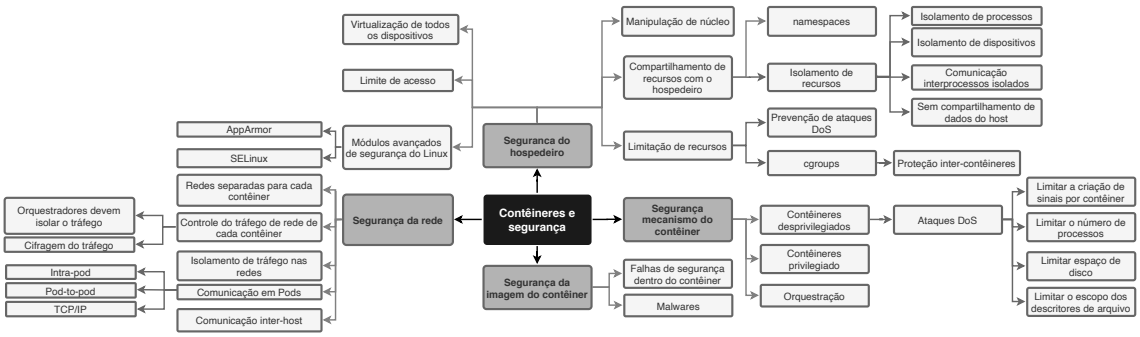


**Figura 2. Taxonomia de segurança dos contêineres.**

- Segurança do hospedeiro: O contêiner é considerado menos seguro que uma máquina virtual, devido ao seu isolamento em relação ao computador hospedeiro ser menor e, isso é uma das barreiras para a maior adesão [Panizzon et al., 2019]. Um contêiner malicioso pode tirar a disponibilidade de recursos do hospedeiro consumindo-os [Sultan et al., 2019], i.e., é importante manter a conexão entre contêiner e hospedeiro segura e limitada. Com isto, para assegurar uma maior segurança, são utilizadas algumas funcionalidades providas pelo núcleo do Linux, e.g., *namespaces*, *CGroups*, *AppArmor*, etc. os quais podem isolar um processo o qual não poderá ser visto por outras partes do sistema [Hayden, 2015].
  - Controle de acesso: um contêiner pode utilizar todos os recursos disponíveis do hospedeiro, tornando-o inutilizável [ENISA, 2017]. Os recursos a serem utilizados pelo contêiner devem ser limitados.
  - Virtualização de todos os dispositivos: devido ao compartilhamento do núcleo com o hospedeiro [Lin et al., 2018], sendo assim, quando mais recursos virtualizados o sistema possuir, mais camadas de abstração são adicionadas, aumentando a segurança do sistema.
  - Manipulação do núcleo: Uma chamada de sistema pode ser sensível uma vez que possibilita afetar a integridade do sistema operacional do hospedeiro. Existem mecanismos para garantir a filtragem das chamadas de sistema, e.g., *Seccomp*, as quais não irão afetar o hospedeiro [Hertz, 2016].
  - Módulos de segurança do Linux: Para garantir maior segurança os acessos deve ser restritos para cada usuário e o núcleo. Para isso existem diversos modelos de segurança para o Linux, configuráveis para um melhor uso do sistema, e.g., *Mandatory Access Control* (MAC) e *Discretionary Access Control* (DAC).

    - AppArmor é implementado utilizando *Linux Security Models* (LSM), e este funciona de maneira que depende de arquivos os quais irão definir os caminhos de arquivos, os quais deverão ser protegidos. Este pode limitar os comandos do núcleo ou as permissões para determinado local de arquivo [Casalicchio and Iannucci, 2018].
    - *Security-Enhanced Linux* (SELinux) é um sistema de rótulos, no qual tudo irá receber um rótulo incluindo processos, arquivos e outros garantindo assim o controle de acesso [Hayden, 2015].
  - Isolamento de recursos: o isolamento de recursos é essencial para garantir segurança é integridade do hospedeiro. O isolamento de recursos ajuda a prevenir que contêineres acessem outras regiões além do seu escopo [Sultan et al., 2019], separando tanto contêineres quando o hospedeiro. Sendo assim, faz-se necessário realizar o devido isolamento de recursos de maneira eficiente para que o hospedeiro não seja afetado por um contêiner infectado.
    - *Namespaces*: O *namespace* criará um espaço virtual o qual irá oferecer diversos recursos individuais como: sistema de arquivos, rede, identificador de processo e um identificador de usuário [Chen, 2019]. Com isso, a garantia de um ambiente separado do principal e seu isolamento tornam mais difícil afetar o sistema hospedeiro.
      - Isolamento de processos: Este irá garantir um ambiente virtualizado para um processo trabalhar, onde ele pode pensar que é o único que está executando no sistema.
      - Comunicação inter-processos: Os processos somente se comunicarão se estiverem dentro do mesmo escopo do *namespace*.
      - Compartilhamento de dados com o hospedeiro: Os contêineres devem ter o mínimo de compartilhamento de dados com o hospedeiro, vendo que caso esteja infectado, corrompê-los se torna uma tarefa fácil. O *namespace* irá oferecer a possibilidade de tratar com arquivos do sistema de maneira isolada, ou seja, sem alterar os arquivos do hospedeiro [Evans, 2016].
      - Isolamento de dispositivos: Os dispositivos devem trabalhar de maneira conjunta, porém, separados de maneira virtualizada, para que caso um seja corrompido não afete os outros.
  - Limitação de recursos: Os recursos do sistema são essenciais para manter o funcionamento correto, sendo assim, manter o controle sobre eles é necessário. Um contêiner não deve ter a possibilidade de exaurir todos os recursos do sistema, podendo afetar o hospedeiro.
    - *CGroups*: Limitará quantos recursos um contêiner poderá utilizar [Hertz, 2016].
    - Ataques de DoS: O ataque consiste em negar recursos para outros processos do sistema, no caso, um contêiner utilizar os recursos e negar para os outros.

- Segurança da rede: Utilizar recursos de rede dentro de um contêiner é uma tarefa que exige cuidado, pois, caso um esteja comprometido, poderá afetar os outros. O isolamento do tráfego de rede, e a cifragem do mesmo também deve ocorrer, pode evitar uma possível intrusão no meio da conexão permitindo ao intruso coletar informações que trafegam na rede. Para um melhor gerenciamento da comunicação entre contêineres foi constituída a ideia de *Pods*, o qual pode conter um ou mais contêineres dentro dele, diversas aplicações os utilizam, e.g., *Kubernetes*.
  - Comunicação entre *Pods*: Os *Pods* interagem sobre a camada dos contêineres, sendo assim, adicionando mais uma camada de abstração [Chakin, 2017]. A comunicação envolvendo *Pods* pode ocorrer de diversas formas, dentre estas é possível compartilhar informações de todo tipo, e.g., arquivos. Os *Pods* compartilham o mesmo *namespace* de rede e *Interprocess Communication* (IPC) [Linchpiner, 2020], tornando a segurança sensível.
  - *Intra-pod*: É realizada sobre um mesmo IP, pois, todos os contêineres de um *Pod* utilizarão o mesmo IP e mesmo *namespace* de rede, se conectando a portas [Chakin, 2017].
  - *Pod-to-pod*: Cada *Pod* deverá saber o IP do outro que realizará a conexão, ou seja, cada um terá o seu.
  - *Inter-host*: A comunicação *Inter-host* ocorre de maneira que caso alguém deseje se comunicar com um contêiner, deverá antes se comunicar com o seu hospedeiro.
  - *Intra-pod*: A comunicação dentro do mesmo *Pod*, entre seus respectivos contêineres é mais simples, visto que o *namespace* de rede é compartilhado.

  - *Inter-pod*: Visto que cada *Pod* terá sua própria rede virtualizada, a comunicação entre estes ocorrerá com uma ponte de rede [Zand, 2019]. Com isso, é possível ver que mesmo estando no mesmo hospedeiro a comunicação acaba se tornando mais dificultosa e exigindo mais recursos.
  - *Pod-to-pod*: A comunicação entre *Pods* separados também utiliza ponte [Zand, 2019], a qual conecta ao *Internet Protocol* (IP) do outro, e então a comunicação entre nós é proporcionada.
  - Transporte: A comunicação entre contêineres suporta tanto TCP quanto UDP.
  - Comunicação *inter-host*: A comunicação entre hospedeiros se dará de maneira indireta, ou seja, o contêiner irá se comunicar com o seu hospedeiro que então este se comunicará com outra máquina.
  - Um contêiner não deve ter informação sobre outro: Os *namespaces* proporcionam isolamento de recurso, para assim garantir a integridade das outras aplicações. Um contêiner conhecer somente suas capacidades irá dar mais segurança para as aplicações e o hospedeiro.
  - Controle de tráfego de cada contêiner: O tráfego de rede de cada contêiner deve ser cuidadosamente realizado, pois, no momento em que ele se comunica com algo fora do seu escopo se tornará vulnerável. O tráfego de rede tende a criar um canal de ataque ao *Hypervisor*, ou até ao hospedeiro [ENISA, 2017].
  - Cifragem do tráfego: O tráfego é algo sensível, pois, nele correm informações que podem possuir risco potencial ao sistema hospedeiro, ou até, ao usuário. Uma cifragem ineficiente torna a rede mais vulnerável aos ataques, e.g., *Man-in-the-Middle* [Tomar et al., 2020].
  - Cada contêiner deverá ter sua rede separada dos demais: Cada contêiner terá seu IP, com isso seu endereçamento na rede, sendo que, sua rede é virtualizada [Hayden, 2015]. A virtualização da rede irá fazer com que tenha uma camada a mais de segurança.
- Segurança do mecanismo do contêiner: O isolamento do contêiner é essencial para garantir maior segurança, tanto para o hospedeiro quanto para a aplicação. Um atacante não deve ter a possibilidade de executar comandos sensíveis ao hospedeiro. Para garantir mais segurança em relação a isso, o contêiner não deve possuir permissões de administrador, pois, sem estas as possibilidades de ataque são mais limitadas [Tomar et al., 2020].
  - Contêineres desprivilegiados: Manter o contêiner sem privilégios de administrador é uma das principais atitudes as quais irão garantir maior segurança do sistema. Fazer um contêiner sem privilégios de administrados adiciona uma camada a mais de segurança [Godoy, 2018]. Porém, não garante inexistência de possibilidade de ataque.
    * Limitar os sinais que os contêineres têm na fila de processos, pois, caso a fila esteja cheia, outros contêineres não poderão realizar processos.
    * Um atacante pode realizar laços infinitos com cálculos que ocupem a CPU [Lin et al., 2018], sendo assim, impossibilitando a utilização da mesma. Caso tenha mais processos que o computador pode processar, irá causar negação de serviço. Ou também, é possível que fique alocando memória constantemente, o que irá deixar a memória inutilizável.
    * Um número considerável de problemas envolvendo DoS podem ser resolvidos com o *CGroups* [Sultan et al., 2019], limitando o número de processos, espaço de memória e de disco e quanto utilizar de recursos do computador.
    * Orquestração: Proporciona facilidade na hora de lidar com os contêineres criados,mas não garantem segurança. A introdução do conceito de *Pods* trouxe uma flexibilidade maior para a comunicação entre estes, assim dificulta ainda mais a tarefa de mantê-los seguros. Os orquestradores realizam ações que se assemelham com as do *CGroups*, como limitação de CPU e memória, além de auxiliar no escalonamento [Casalicchio and Iannucci, 2018], sendo possível identificar que estes também têm a função de auxiliar na segurança.
  - Contêineres privilegiados: Este é o menos recomendado caso deseja-se segurança, pois, seus ataques serão mais danosos tanto ao sistema convidado, quanto ao hospedeiro. No *LXC 1.0*, caso um contêiner estiver privilegiado um atacante pode escapar do *namespace* separado para o contêiner e atuar no sistema hospedeiro como administrador [Graber, 2014]
    * *SYS_RAWIO Abuse*: é um ataque o qual garantirá ao atacante acessar regiões de controle dos periféricos do sistema [Hertz, 2016]. Também é possível burlar o *Seccomp* utilizando a chamada de sistema *ptrace(2)* a qual permite controlar a execução de outros processos.

- Segurança da imagem: Um considerável número de imagens de contêineres disponibilizadas na Internet apresentam vulnerabilidades, as quais, podem ser exploradas por pessoas mal intencionadas. [Sultan et al., 2019] *apud* [Gummaraju et al., 2015] exemplifica uma pesquisa realizada na qual mostrou que mais de 30% das imagens oficiais do Docker continham vulnerabilidades. Recentemente [Chitwadgi and Rajewar, 2020], foi revelado que imagens infectadas mantidas no Docker Hub, utilizavam o contêiner para minerar moedas no computador da vítima. Além disso, é possível que a imagem contenha *backdoor* e passe sem ser detectada, a qual pode causar comprometimento dos dados do hospedeiro ou até mesmo dando espaço para realizar um ataque DoS [Hayden, 2015]. [Shu et al., 2017] avalia diversas imagens Docker que estão disponíveis, tanto oficiais quanto da comunidade. Para isso ele propõe um *framework* que automatiza a varredura de imagens em busca de falhas, e dentre estas, são encontradas imagens vulneráveis desde *Buffer Overflow* até Man-in-the-middle.

## 4. Considerações e trabalhos futuros

Nota-se que em todas as áreas que envolvem a arquitetura dos contêineres, possuem problemas de segurança a serem corrigidos e considerados no desenvolvimento de uma aplicação conteinerizada. A relação entre os problemas é visível, e.g., uma invasão na rede pode comprometer o hospedeiro, e quanto mais tecnologias são adicionadas aos contêineres, maior será a superfície de ataque. Para o próximo trabalho será realizado um estudo acerca do tópico de orquestradores e seus problemas de segurança e, como sua utilização aumenta a superfície de ataque ao contêiner.

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

## Referências

Casalicchio, E. and Iannucci, S. (2018). The state-of-the-art in container technologies.

Chakin, P. (2017). Multi-container pods and container communication in kubernetes. https://www.mirantis.com/blog/multi-container-pods-and-container-communication-in-kubernetes/.

Chen, J. (2019). Making containers more isolated: An overview of sandboxed container technologies. https://unit42.paloaltonetworks.com/making-containers-more-isolated-an-overview-of-sandboxed-container-technologies/.

Chitwadgi, A. and Rajewar, R. (2020). Attackers cryptojacking docker images to mine for monero. https://unit42.paloaltonetworks.com/cryptojacking-docker-images-for-mining-monero/.

ENISA (2017). Security aspects of virtualization. Technical report.

Evans, J. (2016). What even is a container: namespaces and cgroups. https://jvns.ca/blog/2016/10/10/what-even-is-a-container/.

Godoy, R. (2018). Why non-root containers are important for security. https://engineering.bitnami.com/articles/why-non-root-containers-are-important-for-security.html.

Graber, S. (2014). Lxc 1.0: Security features [6/10]. https://stgraber.org/2014/01/01/lxc-1-0-security-features/.

Gummaraju, J., Desikan, T., and Turner, Y. (2015). Over 30% of official images in docker hub contain high priority security vulnerabilities. https://blog.banyansecurity.io/blog/over-30-of-official-images-in-docker-hub-contain-high-priority-security-vulnerabilities.

Hayden, M. (2015). Securing linux containers. *SANS Institute*, page 25.

Hertz, J. (2016). Abusing privileged and unprivileged linux containers.

Lin, X., Lei, L., Wang, Y., Jing, J., Sun, K., and Zhou, Q. (2018). A measurement study on linux container security: Attacks and countermeasures. In *Proceedings of ACSAC '18*, ACSAC '18, page 418–429. ACM.

Linchpiner, P. (2020). Multi-container pods in kubernetes. https://linchpiner.github.io/k8s-multi-container-pods.html.

Panizzon, G., Battisti, J. H. F., Koslovski, G. P., Pillon, M. A., and Miers, C. C. (2019). A Taxonomy of container security on computational clouds: concerns and solutions. *Revista de Informática Teórica e Aplicada*, 26(1):47–59.

Shu, R., Gu, X., and Enck, W. (2017). A study of security vulnerabilities on docker hub. CODASPY '17, page 269–280, New York, NY, USA. Association for Computing Machinery.

Souppaya, M., Morello, J., and Scarfone, K. (2017). Application container security guide. *NIST Special Publication*, 800:190.

Sultan, S., Ahmad, I., and Dimitriou, T. (2019). Container security: Issues, challenges, and the road ahead. *IEEE Access*, 7:52976–52996.

Tomar, A., Jeena, D., Mishra, P., and Bisht, R. (2020). Docker security: A threat model, attack taxonomy and real-time attack scenario of dos. *IEEE Access*, pages 150–155.

Zand, M. (2019). Review of pod-to-pod communications in kubernetes. https://superuser.openstack.org/articles/review-of-pod-to-pod-communications-in-kubernetes/.