Revista de Informática Teórica e Aplicada - RITA - ISSN 2175-2745
Vol. 26, Num. 1 (2019) 47-59

RESEARCH ARTICLE

A Taxonomy of container security on computational clouds: concerns and solutions

Uma taxonomia para segurança de contêineres em nuvens computacionais: problemas e soluções

Guilherme Panizzon[1], João Henrique Faes Battisti[1], Guilherme Piêgas Koslovski[2], Maurício Aronne Pillon[2], Charles Christian Miers[2*]

Abstract: Virtualization in cloud computing has been used in combination with environments Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) in order to provide performance, isolation, and scalability. However, containers and virtual machines (VMs) are susceptible to the vulnerabilities present in the core of operating system as well as container solutions, which are a risk for information and service operation of all entities sharing a same host. The safety recommendation guides aims to mitigate the security in this scenario, but the selection of containerization solutions taking into account security requirements is a complex task. Thus, we propose a security taxonomy focused on containers to cloud computing in order to assist the classification and evaluation containers security mechanisms and solutions.
Keywords: Cloud computing — Container — Security — Taxonomy

Resumo: A virtualização na computação em nuvem tem sido usada em combinação com os ambientes de Plataforma como Serviço (PaaS) e Infraestrutura como Serviço (IaaS) para provisionar desempenho, isolamento e escalabilidade. No entanto, contêineres e máquina virtuals (MVs) são suscetíveis às vulnerabilidades presentes no núcleo do sistema operacional, bem como às soluções de contêiner, que são um risco para a operação de informações e serviços de todas as entidades que compartilham um mesmo *host*. Os guias de recomendação auxiliam a mitigar segurança neste cenário, no entanto, a seleção de soluções de conteinerização, levando em conta os requisitos de segurança, é uma tarefa complexa. Assim, este trabalho apresenta uma proposta de taxonomia de segurança focada em contêineres para computação em nuvem, com o intuito de auxiliar a classificação e avaliação de mecanismos e soluções de segurança para contêineres.
Palavras-Chave: Computação em nuvem — Contêiner — Segurança — Taxonomia

[1] *Departamento de Ciência da Computação (DCC), Universidade do Estado de Santa Catarina (UDESC), Brasil*
[2] *Programa de Pós-Graduação em Computação Aplicada (PPGCA), DCC, Universidade do Estado de Santa Catarina (UDESC), Brasil*
*__Corresponding author__: charles.miers@udesc.br
DOI: http://dx.doi.org/10.22456/2175-2745.86196 • **Received:** 26/08/2018 • **Accepted:** 29/01/2019

1. Introdução

A computação em nuvem é considerada um paradigma computacional que requer múltiplos aspectos de segurança nas tecnologias adjacentes que a compõe [1]. A segurança, seguida de questões como privacidade, conformidade e aspectos legais, é uma das principais barreiras que nuvens enfrentam para sua adoção em larga escala por corporações, indústrias e comunidades acadêmicas. Dentre as tecnologias que suportam os serviços em nuvens, as relacionadas com virtualização de recursos computacionais possuem destaque [2]. Provedores compartilham o substrato computacional entre múltiplos clientes/serviços utilizando virtualização baseada em hipervisores e/ou baseada em contêineres. Os principais benefícios proporcionados pelo uso combinado de ambas tecnologias compreendem o provisionamento de desempenho computacional, isolamento e escalabilidade para aplicações hospedadas. Enquanto virtualização baseada em hipervisores encontra-se em sua maturidade tecnológica, contêineres ainda são iniciantes neste ramo. A associação de contêineres a modelos de serviços *Platform as a Service* (PaaS) e *Infraestructure as a Service* (IaaS) ocorreu com o surgimento do

Docker[1] em 2013. Recentemente, surgiram outras soluções para conteinerização, tais como *Linux Containers* (LXC)[2], CoreOS rkt[3] e *Linux Container Hypervisor* (LXD)[4], todas carentes de documentação relacionadas a segurança destes componentes no contexto de nuvem computacional. O fato dos contêineres serem uma tecnologia *jovem* é uma preocupação no momento de escolha de uso, o que ainda é agravado pelo fato de contêineres poderem utilizar diretamente os recursos do núcleo do sistema operacional (SO) hospedeiro [4]. Dado o compartilhamento direto de recursos do núcleo, SO hospedeiro e/ou contêineres, hospedados em uma mesma máquina física, estão sujeitos a ataques de escala de privilégios, podendo comprometer a segurança dos recursos [5].

Ciente do uso crescente de contêineres nas organizações, e das possíveis vulnerabilidades da área, este trabalho apresenta uma síntese sobre as principais recomendações publicadas em guias de segurança e propõe uma taxonomia para mecanismos e soluções de contêineres. Esta taxonomia tem por objetivo auxiliar na análise e classificação de riscos e vulnerabilidades, presentes em mecanismos e soluções de contêineres em ambientes PaaS e IaaS. Arquitetura, Conformidade e Privacidade são os três principais pilares desta taxonomia, nomeados de dimensões.

O conteúdo deste trabalho está organizado da seguinte forma: a Seção 2 revisa as definições e possíveis combinações de hipervisores e contêineres. Na Seção 3 são detalhadas as principais preocupações do uso de contêineres e guias de recomendações de segurança, revisando os trabalhos relacionados. A taxonomia de segurança elaborada com base nos guias de recomendação da Seção 3 é apresentada detalhadamente na Seção 4. A Seção 5 apresenta as considerações finais e trabalhos futuros.

2. Virtualização de recursos em nuvens computacionais

A virtualização é uma tecnologia presente nas mais modernas infraestruturas de computação em nuvem. O uso de virtualização na computação em nuvem ganhou aderência por prover três propriedades: isolamento, desempenho e portabilidade (*e.g.*, Amazon Elastic Compute Cloud (EC2), Google App Engine e OpenStack)-[6, 7, 8]. Atualmente, além da virtualização tradicional baseada em hipervisores (Seção 2.1), a virtualização utilizando contêineres (Seção 2.2) tem ganhado destaque.

[1]Docker é uma solução para criação e manipulação de contêineres que utiliza os recursos de segurança do núcleo Linux *e.g.*, *namespaces*, *mandatory access control*, *control groups*, *chroot* e *seccomp policies* [3].

[2]LXC possibilita virtualização em nível do SO. No início, o Docker utilizou LXC para fazer o gerenciamento das imagens dos contêineres e o *deployment* de serviços.

[3]CoreOS rkt é uma solução baseada em nível de aplicação concorrente do *Docker*.

[4]LXD é considerado um hipervisor para contêineres por fornecer as mesmas características de hipervisores de máquinas virtuais (MVs).

2.1 Virtualização baseada em hipervisor

Com a utilização de um hipervisor é possível executar múltiplas instâncias de diferentes MVs sobre uma única máquina física, na qual cada SO, isoladamente, gerencia seus recursos, neste caso virtuais, de processamento, de rede e de armazenamento. Os recursos físicos são gerenciados pelo hipervisor, virtualizados e disponibilizados as MVs. Atualmente, pode-se ter duas categorias de hipervisores: *bare-metal* (Tipo-1), que tem seu isolamento a nível de núcleo, e *hosted* (Tipo-2), o qual oferece somente isolamento entre processos [3, 9]. Ressalta-se que o Tipo-1 garante um nível mais elevado de isolamento e segurança e que o Tipo-2 é totalmente dependente do SO hospedeiro. No caso do Tipo-2, vulnerabilidades do hospedeiro podem comprometer tanto o hipervisor quanto as MVs hospedadas. Neste modelo de virtualização, cada instância, nomeada de MV, é uma cópia de um SO completo. Como qualquer SO, aqueles instalados em MVs requerem recursos para gerenciamento dos processos e para estruturas de operacionalização do próprio SO. Se existir n MVs, o hipervisor alocará n estruturas de operacionalização do SO, por exemplo.

2.2 Virtualização baseada em contêineres

A virtualização baseada em contêineres tem seu foco no isolamento entre processos. Para isso, a conteinerização apoia-se nos recursos do núcleo do sistema e não busca abstrair o *hardware* do hospedeiro [10]. Um contêiner possibilita a virtualização de um aplicação implementando a gestão de tarefas, escalonamento e multiplexação de recursos.

A arquitetura baseada em contêineres permite que sejam executados diversos serviços em um mesmo *host*, cada uma dessas funções são separadas e definidas por diferentes e separados contêineres. O uso de contêineres possibilita a implementação do sistema mais escalável e flexível tornando seu desenvolvimento mais simples e funcional [5]. A grande questão da utilização de contêineres desta maneira, é a necessidade de gerenciamento e proteção dos objetos interligados. Tendo em vista que é mais difícil e mais complexo, em termos de segurança, mesclar questões de SO, redes de computadores e aplicações. O princípio da virtualização baseada em contêineres é utilizar os recursos do núcleo do SO para criar um ambiente isolado para ps processos.

2.3 Hipervisor *vs.* contêineres

O uso da virtualização baseada em hipervisor possui benefícios quanto à segurança, quando comparado com as demais tecnologias para prover vários serviços em uma mesma plataforma. Explorar vulnerabilidades do hospedeiro ou do convidado é uma tarefa complexa, pois, ao longo dos últimos anos, provedores de PaaS e de IaaS desenvolveram gerentes de recursos virtuais aderentes a esta técnica de virtualização e oferecerem um nível elevado de isolamento [11].

No contexto de PaaS, contêineres atuam como uma ferramenta que auxilia no desenvolvimento de software, oferecendo portabilidade e os mesmos princípios da virtualização baseada em MVs [12]. MVs, em PaaS, tem como foco o

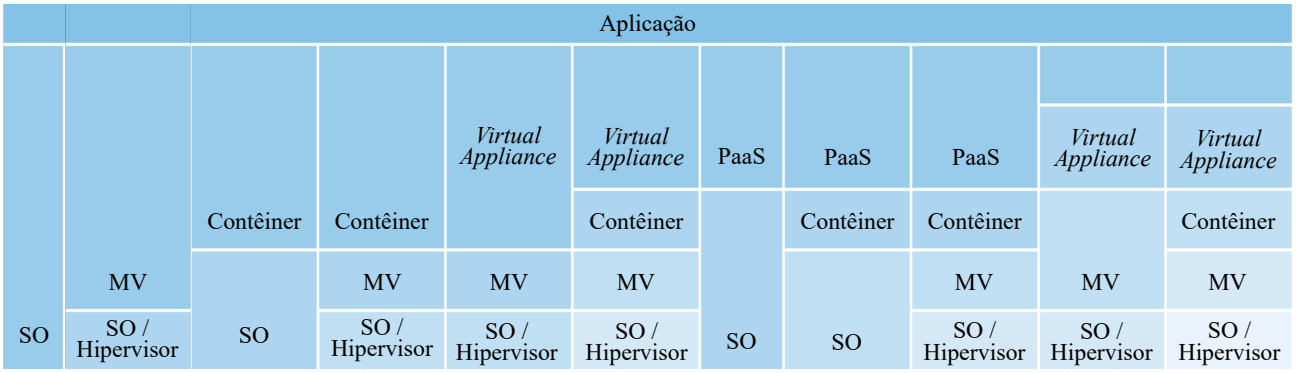

Figura 1. Possíveis combinações do uso de MVs e contêineres em nuvem computacional.

gerenciamento e alocação de recursos do *hardware*. Portanto, as técnicas de virtualização, hipervisor e contêiner, têm aplicabilidades distintas em PaaS. Todavia, no contexto de IaaS, o mesmo não ocorre, contêineres podem substituir o uso de MVs. Este tipo de operação é possível com o uso de soluções como LXD, que atuam como um hipervisor para contêineres. A execução de uma MV implica na inicialização do SO acarretando atrasos na disponibilização da aplicação final, devido ao tempo de inicialização do SO, desempenho, por se tratar de uma estrutura complexa, e de armazenamento, pois estruturas e arquivos são copiados [13]. Este comportamento não é observado em contêineres, pois como ele está associado as estruturas do núcleo do sistema do hospedeiro, não precisa gerenciar estas estruturas e inicializações [14].

Por outro lado, MVs têm um modelo de isolamento mais sólido do que contêineres. Contêineres compartilham o mesmo núcleo e podem ser executados com recursos e privilégios variados em um hospedeiro. Portanto, o grau de segmentação entre os mesmos é inferior do que o fornecido em MVs, feito através do hipervisor. Como contêineres encontram-se no mesmo núcleo, o isolamento é fornecido através de configurações no próprio núcleo. Um contêiner é visto pelo núcleo como um processo ou conjunto de processos e processos são passíveis de iteração e comunicação entre si. Portanto, um descuido na configuração de contêineres pode habilitar a capacidade de interação entre contêineres e o próprio hospedeiro, não necessariamente, desejável [15]. É possível executar contêineres sob MVs garantindo um nível a mais de isolamento entre componentes de uma aplicação [16].

2.4 Virtualização em nuvem computacional

No contexto de nuvem computacional, contêineres e MVs são tecnologias complementares. As diferenças entre as técnicas de virtualização favorecem a implantação de aplicações com necessidades distintas. A Figura 1 ilustra possíveis combinações do uso de contêineres e MVs que variam conforme a capacidade do SO/Hipervisor, portabilidade e pelo seu uso em PaaS.

Existem diversas combinações de contêineres e MVs, tal que a escolha da combinação depende da finalidade e solução de conteinerização. Por exemplo, se o objetivo é realizar a implantação de aplicações com uma estrutura mínima, a abordagem comum é SO - contêiner, a qual possui um desempenho superior em comparação com SO/Hipervisor - MV [16]. Desenvolvedores de software tendem a utilizar ambientes PaaS pela possibilidade de conteinerização em nível de SO, maximizando o desempenho e realizando o gerenciamento agrupado da aplicação [15]. Além do disso, caso o ambiente PaaS não forneça suporte à contêineres, a abordagem utilizada é executar o contêiner em nível de aplicação sob uma MV.

3. Segurança no uso de contêineres

A segurança em contêineres é uma das maiores barreiras levantadas para a adoção da tecnologia no ambiente de produção [17], sendo que há tanto problemas de segurança nos próprios mecanismos empregados para fornecer os contêineres (*e.g.*, isolamento) como nos mecanismos para orquestrar os contêineres e seus ecossistemas [5]. O risco envolvido ao compartilhar recursos de um mesmo núcleo com diversos contêineres não pode ser ignorado. O uso de contêineres e MVs apresentam propriedades semelhantes [18] (*e.g.*, compartilhamento de processamento, rede e armazenamento). Contudo, preocupações de segurança relacionados às MVs são detalhadamente abrangidas pelos principais guias de recomendação de segurança para a computação em nuvem como: CSA [19], NIST [20] e ENISA [21].

Analisando estes guias elencados [19, 20, 21], constata-se que, no que se refere à contêineres, apresentam apenas uma breve descrição conceitual, não fazendo uma clara definição dos aspectos de segurança inerentes a sua abordagem diferenciada. Trabalhos relacionados à computação em nuvem vem sendo desenvolvidos com o objetivo de analisar questões de segurança em plataformas PaaS, IaaS e *Software as a Service* (SaaS) [1]. Estes trabalhos apresentam métodos que auxiliam na avaliação de risco de aspectos de segurança [21, 19] e desafios quanto à privacidade, governança e conformidade [20]. Entretanto, contemplam superficialmente mecanismos para reforçar a política de controle do gerenciamento de contêineres, imagens e acesso aos recursos do núcleo [22, 23, 3, 24]. Assim, os principais guias e trabalhos

específicos, relacionados a segurança para contêineres e que servem de base para este trabalho são: *(i)* [25], contemplando recomendações, mecanismos de segurança, ataques, ameaças e métodos para garantir a segurança do uso de contêineres no ambiente de produção; *(ii)* [24], apresentando informações conceituais e estratégias para garantir a segurança na execução de contêineres; e *(iii)* [26], discutindo boas práticas, vulnerabilidades, riscos relacionados e proposta de soluções.

Dentre os materiais consultados, não foi identificado nenhum padrão ou recomendação para avaliar os aspectos de segurança do uso de contêineres, apenas uma análise quantitativa das principais preocupações e soluções de segurança voltada à computação em nuvem [2, 11]. Em suma, os trabalhos relacionados visam relacionar ameaças, riscos e vulnerabilidades presentes em mecanismos do núcleo do SO e que são utilizados por tecnologias de conteinerização [27, 25].

Com base no trabalho de [2], o qual apresenta uma taxonomia para classificar e organizar informações sobre aspectos de segurança na computação em nuvem, é proposto um modelo aplicável ao contexto de uso de contêineres. Esse modelo é composto de seis categorias: *(i)* Segurança da Rede; *(ii)* Conteinerização; *(iii)* Política de Segurança; *(iv)* Conformidade; *(v)* Segurança de Dados; e *(vi)* Questões Legais.

No ano de 2017, o *National Institute of Standards and Technology* (NIST) elaborou uma documentação relevante sobre contêineres e virtualização [15], que após ser analisada implicou em adicionar a nova categoria de "*(vii)* Risco do sistema operacional". Esta nova categoria leva em consideração os problemas herdados do SO em que é aplicado a virtualização por contêineres. Em suma, neste trabalho são propostas um total de sete categorias (Tabela 1), em cada categoria são apresentados os principais problemas e preocupações de acordo com os guias de recomendação e segurança mencionados.

Em cada categoria são apresentados os principais problemas e preocupações de acordo com os guias de recomendação e segurança mencionados. Seguindo a abordagem empregada por [2], foi realizada uma análise voltada para a segurança em contêineres que resultou na categorização descrita na Tabela 1. Embora a categorização indique os aspectos inerentes e desejáveis relacionados com a segurança em contêineres, uma organização é necessária para auxiliar na análise das soluções existentes.

4. Taxonomia da segurança de contêineres

Com o objetivo de criar um modelo que auxilie na análise da segurança do uso de contêineres, essa seção apresenta os riscos e vulnerabilidades mencionados, organizados através de categorias com uma hierarquia envolvida, criando assim uma taxonomia da segurança de contêineres, ilustrada na Figura 2. A taxonomia foi elaborada visando tanto o generalismo quanto à granularidade, isto é: é possível classificar um determinado aspecto/solução de acordo com o grau de detalhamento desejado.

Figura 2. Primeiro nível da proposta de taxonomia de segurança do uso de contêineres em nuvem computacional.

Conforme a Figura 2, o primeiro nível da taxonomia é constituído de três dimensões:

- Arquitetura;
- Privacidade; e
- Conformidade.

O motivo dessa estruturação é classificar aspectos de segurança de maneira independente das demais dimensões. Contudo, existe a possibilidade de classificar um mesmo aspecto de segurança relacionando-o com mais de uma dimensão. O motivo dessa possibilidade é pelas dimensões Privacidade e Conformidade apresentarem aspectos de segurança que envolvem a responsabilidade do provedor de serviço computacional.

4.1 Dimensão Arquitetura

A dimensão Arquitetura (Figura 3) apresenta questões voltadas a configurações de segurança da rede e da transferência de dados entre contêineres. Como também, soluções e mecanismos responsáveis por controlar o acesso aos recursos do SO e verificar a autenticidade de dependência de imagens de contêineres. Para cada dimensão foram propostas subdimensões, a fim de auxiliar na avaliação do uso de contêineres tanto em PaaS como em IaaS e a possível abordagem de mecanismos de controle de acesso do núcleo:

- Arquitetura de segurança;
- Conteinerização;
- Segurança de rede; e
- Risco do sistema operacional.

Arquitetura de Segurança: a Arquitetura de segurança define os controles e recursos para mitigar a segurança. Essencialmente, tudo é proibido a menos que exista uma regra de acesso permitindo.

i) Controle de acesso obrigatório: No uso de contêineres é comum a falta da criação de regras de acesso e a desabilitação de mecanismos de segurança por parte dos administradores, devido à complexidade da configuração. SELinux [28] e AppArmor [29] são exemplos de mecanismos que atuam como elementos-chave no controle de acesso a recursos do núcleo.

Tabela 1. Categorização de segurança em contêineres.

Categoria	
Segurança da rede: Problemas relacionados à comunicação e transferência de dados entre contêineres de um mesmo hospedeiro ou remotos.	1. Configurações de segurança: Questões voltadas à configurações, tecnologias e protocolos a fim de garantir a segurança da rede sem comprometer o desempenho da transferência de dados. 2. Segurança na transferência de dados: Mecanismos que garantam a proteção da rede contra os principais ataques direcionados à contêineres.
Conteinerização: Isolamento entre contêineres e recursos do SO, vulnerabilidades presentes em imagens e no hipervisor, e problemas associados com o uso da tecnologia.	1. Isolamento: contêineres compartilham recursos de um mesmo SO ficando suscetíveis à ataques e vazamento de dados. O conceito de isolamento também é aplicado em processos, recursos, dispositivo, rede, sistema de arquivos, e limitação de recursos. 2. Negação de serviços e consumo de recursos: Ataques de negação de serviço (*e.g., forkbombs* e exposição de recursos do *hardware*) sob um mesmo contêiner ou entre contêineres de um mesmo hospedeiro. 3. Vulnerabilidades do contêiner: Vulnerabilidades e aspectos de segurança relacionado às soluções de contêineres baseado em SO e aplicação. 4. Vulnerabilidades da solução de conteinerização: Vulnerabilidades e aspectos de segurança em soluções que atuam como hipervisor para contêineres. 5. Orquestração de contêineres: Risco associado ao uso de soluções proprietárias pela não divulgação da avaliação de segurança ou requisitos de segurança não suportados. 6. Ataques internos ao contêiner: Violação da segurança do contêiner através de escalação de privilégios. 7. Vazamento de dados: Exploração de vulnerabilidades das aplicações em execução. 8. Imagens de contêineres: Falta de controle e mecanismos para verificação da autenticidade de bibliotecas e dependências de imagens de contêineres. 9. Ataques *cross-side-channel* no contêiner: Ataques entre contêineres de um mesmo hospedeiro ou da mesma rede.
Política de Segurança: Questões relacionadas ao controle de acesso aos recursos do núcleo do SO.	1. Controle de acesso obrigatório: Preocupação com vulnerabilidades presentes em soluções de controle de acesso obrigatório e configurações das chamadas de serviço do núcleo.
Conformidade: Problemas relacionados a auditoria de vulnerabilidades e ao quanto o serviço atende/adere as políticas, normas e procedimentos.	1. Acordo de nível de serviço: Mecanismos que garantam a disponibilidade, desempenho e procedimentos de segurança do serviço. 2. Auditoria: Avaliação da segurança de imagens e contêineres em execução.
Segurança de dados: Problemas relacionados à identificação e controle de dados armazenados em contêineres. Políticas para minimizar riscos e possíveis danos.	1. Criptografia: Uso da criptografia para garantir a confidencialidade. 2. Redundância: Preocupações com mecanismos que garantem a disponibilidade e integridade dos dados em caso de eventuais problemas no serviço.
Questões legais: Aspectos legais do uso de contêineres na nuvem computacional.	1. Privilégios do provedor de serviço: Atividades maliciosas realizadas por empregados mal-intencionados que possam afetar a disponibilidade, integridade e a disponibilidade de dados e serviços. 2. E-Discovery: Questões relacionadas à confidencialidade de informações quando solicitado a divulgação dos dados por órgãos judiciais.
Risco sistemna opracional do *host*: Problemas relacionados ao sistema operacional local da máquina.	1. Superficie de Ataque: Coleção de todas vulnerabilidades que os invasores podem explorar e acessar as vulnerabilidades do *host*. 2. Compartilhamento do Núcleo: Embora contêineres ofereçam um ótimo isolamento em nível de software, ter um núcleo compartilhado gera uma larga abertura para invasores. 3. Vulnerabilidade de componentes do *host*: Componentes que possuem falhas em nível de software. Falhas estas muitas vezes desconhecidas que podem ser exploradas. 4. Acessos impróprios diretamente no *host*: Acessos aos contêineres através do *host*, sem necessariamente passarem pelas camadas de orquestração.

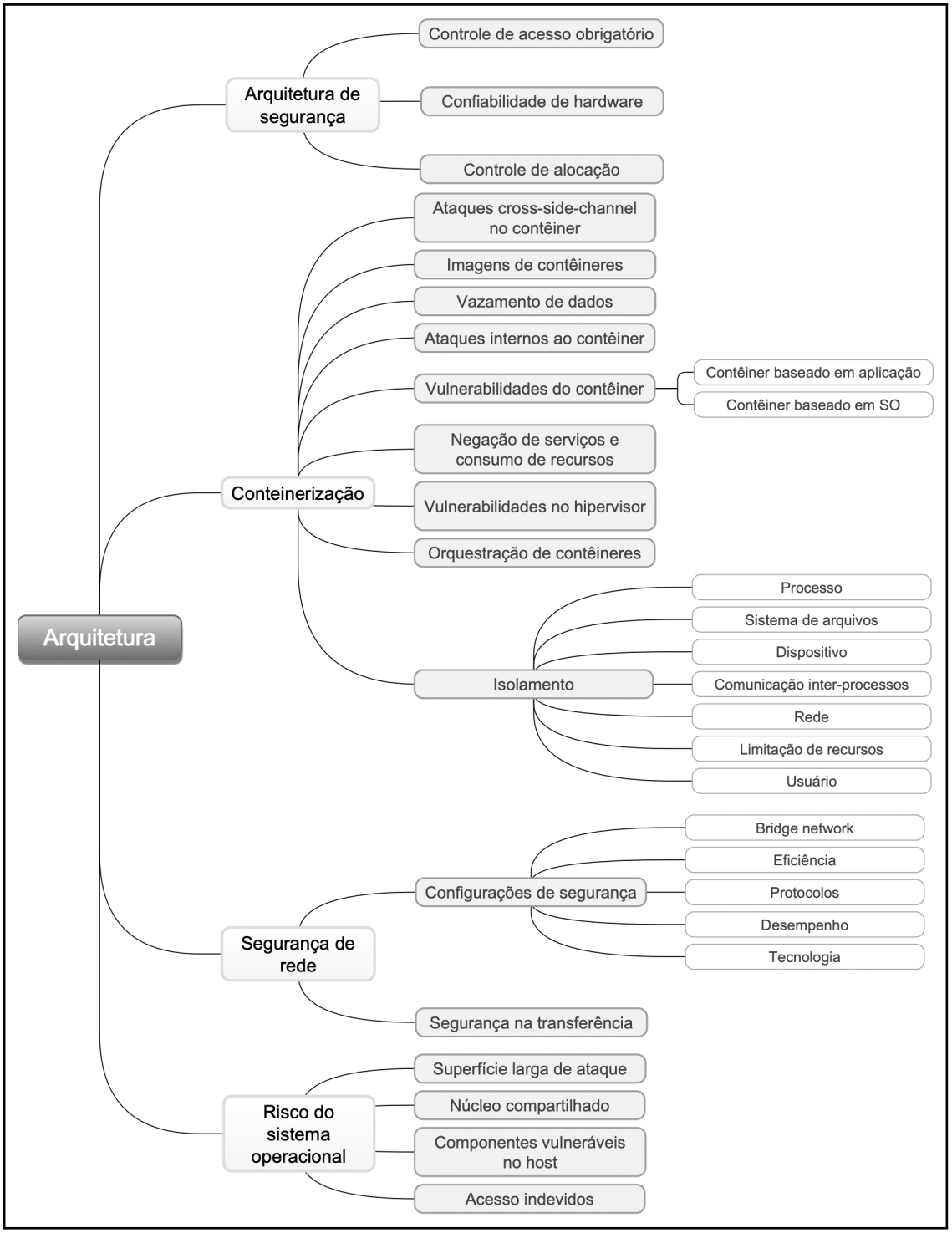

Figura 3. Taxonomia de segurança: dimensão arquitetura.

ii) Confiabilidade de hardware: A facilidade de fazer o *deployment* de contêineres em diversos computadores leva a preocupação se este equipamento é de fato de quem diz ser. Neste sentido, desenvolvedores podem querer assegurar-se da confiabilidade do hardware no que diz respeito se este faz parte de um conjunto de equipamen-

tos seus ou do provedor contratado. Soluções como *Trusted Execution Environments* (TEE) [30] para ambientes confiáveis podem ser empregadas para tratar deste aspecto. Além disso, há soluções a nível de hardware como o Intel *Software Guard Extensions* (SGX) [31] que já podem ser empregadas no contexto de segurança de contêineres através de soluções como o *Secure Linux Containers* (SCONE) [32].

iii) Controle de alocação: O uso de contêineres justamente busca usar o compartilhamento de um hospedeiro (SO e/ou MV) para diminuir sua carga administrativa, este fato implica na alocação de vários contêineres em um mesmo hospedeiro. Atacantes podem buscar manipular a alocação do seu contêiner em um mesmo hospedeiro para facilitar a execução de ataques contra os demais contêineres alocados no hospedeiro.

Conteinerização : Relacionada à segurança da virtualização baseada em contêineres. Preocupações com o isolamento de recursos do núcleo do SO, vulnerabilidades associadas à imagens de contêineres, soluções proprietárias e ataques ao contêiner.

i) Ataques *cross-side-channel* no contêiner: Ataques contra contêineres em execução sob um mesmo SO hospedeiro ou presentes na mesma rede. O objetivo é explorar serviços e aplicações de uso comum entre contêineres (*e.g.*, comprometer o contêiner que executa o banco de dados ou aplicação do ataque ARP *Spoofing* para capturar credenciais de contêineres vizinhos).

ii) Imagens de contêineres: Preocupação com ameaças e mecanismos que analisam a autenticidade das dependências presentes nas imagens de contêineres. Cerca de 30% das imagens do *Docker Hub* [33]) são suscetíveis a uma variedade de ataques (*e.g.*, *shellshock, heartbleed* e *poodle*) [34]. Muitos desses ataques são possíveis devido a vulnerabilidade de software e configurações indevidas.

iii) Vazamento de dados: Preocupação com a confidencialidade da informação voltada à configurações de aplicações executadas em contêineres. Possível vazamento de dados por armazenar artefatos de segurança em imagens de contêineres, exposição de variáveis de ambiente à processos filhos e armazenamento de *logs* de aplicações em locais compartilhado. Nestes *logs* há a possibilidade da existência de informações sensíveis como *tokens* de autenticação.

iv) Ataques internos ao contêiner: Escalamento de privilégios dentro do contêiner através de vulnerabilidades (*e.g.*, bibliotecas desatualizadas e exposição do contêiner em redes não confiáveis) presentes em aplicações web com o uso de técnicas de invasão (*e.g.*, injeção de comandos, violação de memória e escrita arbitrária em arquivos).

v) Vulnerabilidades do contêiner: Questões de segurança voltadas à tecnologias que atuam como solução para a conteinerização.

a) Contêiner baseado em aplicação: Questões de segurança relacionadas à tecnologias de contêineres em nível de aplicação (*e.g.*, Docker e CoreOS rkt [35]).

b) Contêiner baseado em SO: Questões de segurança relacionadas às tecnologias de contêineres em SO (*e.g.*, LXC[5] e OpenVZ[6]).

vi) Negação de serviços e consumo de recursos: Questões relacionadas à ataques de negação de serviço como *forkbombs*, exposição direta e indireta de recursos e de mecanismos de criptografia do núcleo [25]. Neste tipo de ataque, é possível ter como alvo o contêiner em execução, o vizinho e até mesmo o SO hospedeiro.

vii) Vulnerabilidades do hipervisor: Vulnerabilidades em soluções que atuam como hipervisor para contêineres (*e.g.*, LXD), garantindo isolamento de recursos em nível de hardware.

viii) Orquestração de contêineres: Para que o gerenciamento em *cluster* de contêineres torne-se altamente escalável e de alto desempenho, é essencial o uso de *frameworks* (*e.g.*, Kubernetes[7], Docker Swarm, Mesos e OpenStack Magnum[8]). Muitos *frameworks* são soluções proprietárias e a avaliação de segurança de algumas é desconhecida [25].

ix) Isolamento: As soluções para contêineres utilizam *namespaces*[9] do núcleo do SO para prover isolamento entre processos. Um exemplo é o Docker, que fornece suporte aos recursos. Contudo, não são todos os *namespaces* que a tecnologia oferece por padrão.

a) Processo: *Namespace* responsável por prover a identificação de processos do contêiner, permitindo que cada contêiner possua uma árvore de processos interna. Uma vulnerabilidade recentemente encontrada no Docker é o vazamento de dados sobre os processos de um contêiner [27].

b) Sistema de arquivos: Provê isolamento do sistema de arquivos entre contêineres com o objetivo de minimizar o compartilhamento de dados. É utilizado o *namespace mount* para separar arquivos entre contêineres e do próprio SO hospedeiro. Contudo, existem vulnerabilidades e podem ser exploradas por atacantes.

c) Dispositivo: Utilizado para isolar *drivers* de dispositivos compartilhados entre contêineres ou pelo SO

[5] ⟨https://linuxcontainers.org/lxc/introduction/⟩
[6] ⟨https://wiki.debian.org/OpenVz⟩
[7] ⟨http://kubernetes.io/docs/whatisk8s/⟩
[8] ⟨https://wiki.openstack.org/wiki/Magnum⟩
[9] Namespaces é uma funcionalidade do núcleo do SO que fornece isolamento e virtualização de recursos a um conjunto de processos [36].

hospedeiro. Todavia, a exposição desses *drivers* é uma ameaça e pode ser explorada com o intuito de escalar privilégios, ganhar acesso ilegal a dados ou a partições.

d) Comunicação inter-processos: Responsável por isolar os recursos da comunicação entre processos (*e.g.*, compartilhamento de memória, semáforos e fila de mensagens). Ataques de negação de serviço são utilizados para comprometer essa comunicação.

e) Rede: Responsável pelo isolamento de recursos de rede entre processos como: interfaces, roteadores e regras do acesso do *firewall*. Em plataformas PaaS, contêineres de diferentes empresas compartilham o núcleo de um mesmo SO hospedeiro. Consequentemente a comunicação entre contêineres não é restrita na camada de enlace, ficando suscetível a ataques *cross-container*[27, 37].

f) Limitação de recursos: Preocupação com o uso de mecanismos (*e.g.*, cgroups [38]) para limitar recursos que são consumidos por processos de um determinado contêiner.

g) Usuário: Principal recurso utilizado por contêineres. Responsável por prover contêineres não privilegiados. Ou seja, dentro do contêiner o usuário tem privilégio *root*, mas fora não. Contudo, ataques de negação de serviço podem ser utilizados para comprometer os recursos desse *namespace*.

Segurança da Rede: Problemas relacionados à comunicação entre contêineres e com o SO hospedeiro.

i) Configurações de segurança: Questões voltada à configurações internas do contêiner, *e.g.*, regras de acesso do *firewall*, encaminhamento de pacotes, interfaces de rede, modo da rede e versão do protocolo.

a) *Bridge Network*: Quando um contêiner é instanciado, por padrão, o tipo de configuração da rede é a *Bridge*. A rede em modo *bridge* automaticamente encaminha pacotes para todas as interfaces que fazem parte da dela. Também, permite que contêineres comuniquem-se entre si e com o seu SO hospedeiro. Com sua similaridade à rede de *switches*, o atacante possui a possibilidade de realizar ataques ARP *Spoofing*, *Spanning Tree Protocol* (STP) e ataques ao protocolo IPv6 [27].

b) Eficiência: Configurações de rede necessárias para garantir o desempenho sem comprometer a eficiência.

c) Protocolos: Uso, configuração e suporte (*e.g.*, IPv6) aos protocolos necessários para garantir o funcionamento da rede.

d) Desempenho: Questões de segurança de mecanismos que melhoram o desempenho da rede.

e) Tecnologia: Configuração e integração de novas tecnologias na rede, (*e.g.*, *Software Defined Networking* (SDN) [39]).

ii) Segurança na transferência: Com o massivo compartilhamento de informações entre contêineres e o hospedeiro, é necessário a utilização de mecanismos que garantam a segurança da rede na troca de informações. Aqui ressalta-se o uso de protocolos de aplicação que empregam criptografia (*e.g.*, SSH e HTTPS) e *Virtual Private Network*s (VPNs) [40].

Risco do sistema operacional:

i) Superfície larga de ataque: São várias vulnerabilidades que invasores podem explorar ao acessarem o *host*. Como exemplo tem-se um serviço que realiza conexão com a rede ou com a internet e a partir desta conexão ele abre possíveis pontos para realização de ataques.

ii) Núcleo Compartilhado: Sabe-se que as aplicações destinadas para contêineres oferecem um bom isolamento no nível de software. O compartilhamento do núcleo no abre possíveis pontos para ataque de invasores.

iii) Componentes vulneráveis do *host*: Estes componentes podem possuir falhas em nível de software, que muitas vezes são desconhecidas e necessitam de atualização, que podem abrir lacunas para um eventual ataque.

iv) Acessos indevidos: Estes são acessos realizados diretamente na máquina/usuário *host*, que permitem total acesso aos contêineres sem passarem por camadas de orquestração que delimitam o poder do usuário.

4.2 Dimensão Privacidade

A dimensão Privacidade (Figura 4) abrange preocupações com a segurança da informação e o uso da criptografia para proteger informações confidenciais e mecanismos para controlar a redundância dos dados. Ainda, questões voltadas ao controle dos dados pelo provedor do serviço em ambientes PaaS e IaaS.

Conforme a Figura 4, percebe-se a divisão em duas subdimensões principais: Segurança de Dados e Questões Legais. O objetivo é separar questões relacionadas à segurança de questões legais junto ao provedor do serviço. Aspectos legais do provedor do serviço valem tanto para a virtualização baseada em contêineres quanto para MVs.

Segurança de dados: Mecanismos e políticas que garantam a integridade, disponibilidade e a confidencialidade de dados do contêiner.

i) Criptografia: Questões voltadas a cifragem de contêineres em ambientes virtuais e da autenticidade de imagens. Além disso, vulnerabilidades presentes em mecanismos (*e.g.*, *Docker Content Trust*[10]) que auxiliam no processo de verificação da autenticidade.

[10] ⟨https://blog.docker.com/2015/08/content-trust-docker-1-8/⟩

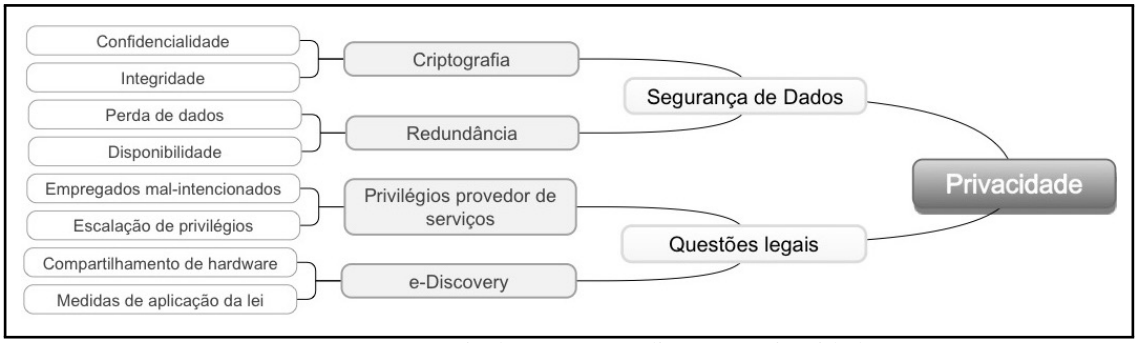

Figura 4. Taxonomia de segurança: dimensão privacidade.

a) Confidencialidade: Preocupações com a confidencialidade dos dados em contêineres e soluções como Docker-squash[11], que auxiliam na eliminação de dados sensíveis (*e.g.*, dados do *cache*, *logs* e chaves de acesso) [41].

b) Integridade: Mecanismos de proteção à integridade dos dados como: sistemas automatizados de *hashes* e outras que possam auxiliar a verificar se o contêiner e os seus dados foram alterados indevidamente ou corrompidos.

ii) Redundância: Preocupação com a perda de dados e disponibilidade de dados sensíveis armazenados em contêineres. É essencial garantir a integridade de dados independente do ciclo de vida do contêiner ou de eventuais problemas da aplicação.

a) Perda de dados: Questões voltadas à persistência de dados em aplicações e serviços em execução no contêiner. Uso de mecanismos como *Convoy*[12] que permitem a criação e gerenciamento de volumes para armazenamento persistente de dados em contêineres que empregam Docker. A replicação e *backup* de dados automático e *snapshot* de dados também são exemplos de abordagens neste sentido.

b) Disponibilidade: Para garantir a disponibilidade e a segurança no gerenciamento de dados, é recomendado o uso de soluções de armazenamento de dados dedicado [42].

Questões Legais: Aspectos de segurança com relação jurídica entre o cliente e o provedor de serviço computacional.

i) Privilégios do provedor do serviço: O provedor de serviço de conteinerização na computação em nuvem pode ter controle e acesso de todos os dados e procedimentos da aplicação. A preocupação é com riscos de segurança por haver esse privilégio.

a) Empregados mal-intencionados: Funcionários de má índole podem ter influência sob a perda, vazamento e a confidencialidade da informação. Dessa forma, há uma relevância considerável em analisar (*e.g.*, investimento em segurança e equipe) o provedor de serviço contratado [43].

b) Escalação de privilégios: Risco associado ao compartilhar o hardware com diversas organizações. A escolha de uma plataforma privada ou de acesso limitado deve ser considerada a fim de amenizar o risco.

ii) *e-Discovery*: Aspectos referentes à aplicação da lei para a divulgação de informações armazenadas em contêineres.

a) Compartilhamento de *hardware*: Uma solução comum em PaaS é o compartilhamento do hardware com diversas organizações. Tornou-se uma prática comum em nuvens públicas. O compartilhamento do hardware é um ponto crítico para a confidencialidade da informação.

b) Medidas de aplicação da lei: Uma medida de aplicação da lei é solicitar a divulgação de dados de um contêiner pelo provedor do serviço. A estratégia de compartilhar o hardware com diversas organizações pode afetar a confidencialidade da informação.

4.3 Dimensão Conformidade

A dimensão Conformidade apresenta preocupações com políticas de segurança, perda do controle da informação, configurações e indisponibilidade do serviço por parte do provedor. Além disso, preocupações com a transparência e avaliação da segurança em imagens e configurações de contêineres.

De acordo com a Figura 5, a dimensão conformidade apresenta duas principais subdimensões: SLA e Auditoria. O motivo dessa separação é tornar aspectos de auditoria independente de políticas do provedor do serviço.

[11] ⟨https://github.com/jwilder/docker-squash⟩

[12] ⟨http://rancher.com/introducing-convoy-a-docker-volume-driver-for-backup\-and-recovery-of-persistent-data/⟩

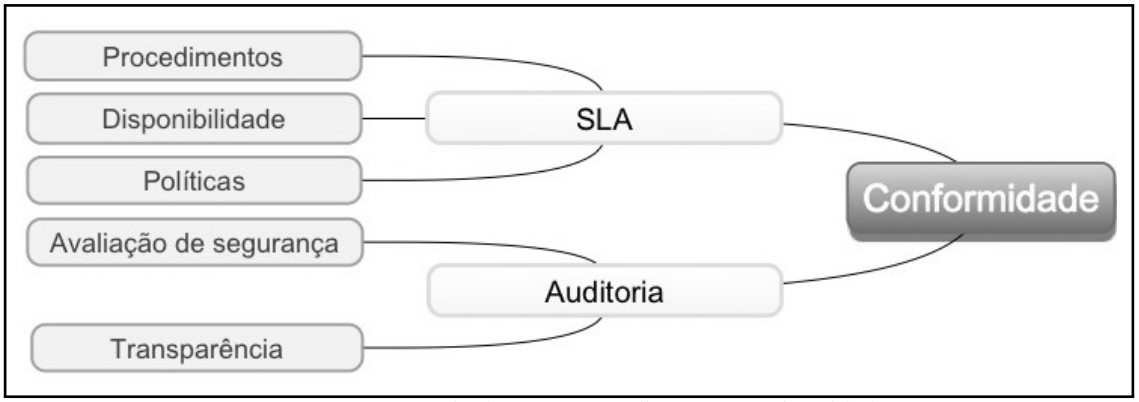

Figura 5. Taxonomia de segurança: dimensão conformidade.

SLA: Questões relacionadas ao provedor de serviço de computação em nuvem e mecanismos que garantam a disponibilidade e procedimentos básicos de segurança.

i) Procedimentos: Os usuários de serviços PaaS e IaaS são tipicamente responsáveis pelas definições de políticas, procedimentos e processos. Enquanto o provedor é responsável pela execução das mesmas. Conhecer e definir procedimentos de segurança é uma preocupação a ser levada em consideração.

ii) Disponibilidade: Questões relacionadas a disponibilidade do serviço de conteinerização na nuvem. Plataformas PaaS e IaaS comumente mencionam em suas políticas de conformidade à disponibilidade do serviço numa taxa entre 99.5% e 100% [44].

iii) Políticas: Questões como cobertura do serviço, uso excessivo de recursos, métodos, penalidades e licença de software. O desconhecimento de políticas de conformidade é uma ameaça ao serviço.

Auditoria: Possibilidade de auditar a segurança de contêineres por clientes, usuários e provedores do serviço com uso de metodologias e mecanismos.

i) Avaliação de segurança: Relevância em identificar contêineres que possuem dependências desatualizadas e imagens com potenciais vulnerabilidades. Para isso, são utilizados mecanismos de avaliação de segurança em contêineres, (*e.g.*, OpenSCAP[13], CoreOS Clair[14] e Docker Bench Security[15]).

ii) Transparência: Transparência e métodos são necessários para analisar continuamente a conteinerização em plataformas PaaS e IaaS. Ao saber quais tecnologias e softwares são empregados, facilita-se em identificar aspectos de segurança que podem ser assegurados e quais ainda podem vir a ser um problema. Aqui também implica em deixar explícito como as operações de transferência de dados e execução dos contêineres são realizadas.

4.4 Classificação de soluções para contêineres

A taxonomia proposta deve auxiliar na classificação dos principais mecanismos e soluções de contêineres empregados em nuvens computacionais. Neste sentido, a Tabela 2 foi elaborada para exemplificar a aplicação da taxonomia usando como base algumas das referências / soluções citdas neste trabalho.

Analisando a Tabela 2, constata-se a ocorrência de um maior número de soluções relacionadas à dimensão Arquitetura. O motivo é que Privacidade e Conformidade possuem questões relacionadas ao provedor do serviço de computação em nuvem de tal forma que não necessariamente envolvem mecanismos e soluções. Neste caso, o aspecto é voltado à acordos, procedimentos ou políticas.

Quanto as ocorrências de Arquitetura, o NIST [15] trouxe contramedidas para uma considerável parte dos problemas identificados nesta taxonomia, desta forma possibilita uma maior mitigação e identificação das soluções apropriadas. Já a Red Hat [50] identificou possíveis soluções para o isolamento da rede compartilhada pelos contêineres, mas esta solução depende da arquitetura e topologia da solução empregada no contêiner.

5. Considerações & Trabalho futuros

As ferramentas para virtualização de *data centers* de nuvens computacionais são fundamentais para o provisionamento de serviços. Atualmente, além da utilização de máquinas virtuais, os contêineres tem sido utilizados para consolidar e organizar hospedeiros e aplicações hospedadas. Entretanto, diferentemente do cenário de máquinas virtuais, os aspectos de segurança de contêineres ainda estão em fase de amadurecimento, constituindo um ponto crítico para provedores e clientes.

Voltado ao critério da segurança das imagens de contêineres foi possível constatar que até mesmo imagens de distribuições

[13] ⟨http://static.open-scap.org/openscap-1.0/oscap_user_manual.html⟩
[14] ⟨https://github.com/coreos/clair⟩
[15] ⟨https://github.com/docker/docker-bench-security/⟩

Tabela 2. Classificação de mecanismos e soluções de segurança de contêineres.

Solução/Referência	Dimensão principal	Subdimensões
Docker Content Trust	Arquitetura	Conteinerização - Imagens de Contêineres - Vulnerabilidades
Cgroups [38]	Arquitetura	Conteinerização - Isolamento - Limitação de Recursos
SDN [39]	Arquitetura	Segurança da Rede - Configurações de Segurança
Docker Bench Security	Conformidade	Auditoria - Avaliação de Segurança
Vault [45]	Privacidade	Segurança de Dados - Criptografia - Confidencialidade
Docker	Arquitetura	Conteinerização - Vulnerabilidades do Contêiner - Contêiner Baseado em Aplicação
LXC	Arquitetura	Conteinerização - Vulnerabilidades do Contêiner - Contêiner Baseado em SO
Flocker [46]	Privacidade	Segurança de Dados - Redundância - Perda de Dados
OpenSCAP	Conformidade	Auditoria - Avaliação de Segurança
LXD	Arquitetura	Conteinerização - Vulnerabilidades do hipervisor
Kubernetes	Arquitetura	Conteinerização - Orquestração de contêineres
Docker-squash	Privacidade	Segurança de Dados - Criptografia - Confidencialidade
CoreOS Clair	Conformidade	Auditoria - Avaliação de Segurança
Docker-Swarm [47]	Arquitetura	Conteinerização - Orquestração de Contêineres
Convoy	Privacidade	Segurança de Dados - Redundância - Perda de Dados
Twistlock Trust [48]	Conformidade	Auditoria - Avaliação de Segurança
Twistlock Runtime [49]	Arquitetura	Arquitetura de Segurança - Controle de Acesso
AppArmor [29]	Arquitetura	Arquitetura de Segurança - Controle de Acesso
SCONE [29]	Arquitetura	Arquitetura de Segurança - Confiabilidade de hardware

oficiais estão vulneráveis à ameaças de dependências como bibliotecas, serviços e códigos arbitrários que compõe a imagem. Além disso, foi possível notar o amadurecimento de soluções para a auditoria das imagens, especificamente o Docker Security Scanning. Todavia, a solução não é de código fonte aberta. Também, as ferramentas são apenas informativas e não bloqueiam o uso de imagens vulneráveis. Para amenizar a superfície de ataque recomenda-se uso do repositório oficial Docker Hub. De forma geral, a maior parte das soluções para os problemas encontrados podem ser segmentado estão em ações como: auditar políticas de segurança, caso não seja necessário, não alterar as regras impostas por padrão pelo Docker para a filtragem de chamadas do sistema e configurar corretamente a comunicação de contêineres através de redes definidas pelo usuário.

Para auxiliar na compreensão e organização dos problemas e ferramentas relacionados com a segurança de contêineres, o presente trabalho apresentou uma taxonomia para avaliação de mecanismos e soluções. A taxonomia proposta foi utilizada frente as soluções existentes atualmente para comprovar a sua aplicabilidade.

A taxonomia desenvolvida neste trabalho serve como base para uma segunda etapa na qual será elaborado um método para análise de pontos críticos de segurança em contêineres. Outro trabalho futuro, já em realização consiste e um estudo de caso mais aprofundado sobre as diversas formas de uso da solução Docker em nuvens computacionais IaaS baseada na solução de código aberto OpenStack.

Agradecimentos

Os autores agradecem a Fundação de Amparo a Pesquisa de Santa Catarina (FAPESC) pelo apoio ao Laboratório de Processamento Paralelo Distribuído (LabP2D) do CCT/UDESC, e aos membros do LabP2D pela colaboração na pesquisa.

Contribuição dos Autores

- Guilherme Panizzon: Pesquisa e escrita inicial do artigo;
- João Henrique Faes Battisti: Atualização de guias de segurança e correções de conceitos;
- Charles Christian Miers: Coordenação do trabalho e direcionamento da pesquisa;
- Maurício Aronne Pillon: Direcionamento de pesquisa e correções de conceitos; e
- Guilherme Piêgas Koslovski: Direcionamento de pesquisa e correções de conceitos.

References

[1] HASHIZUME, K. et al. An analysis of security issues for cloud computing. *Journal of Internet Services and Applications*, v. 4, n. 1, p. 5, 2013. ISSN 1869-0238. Disponível em: ⟨http://dx.doi.org/10.1186/1869-0238-4-5⟩.

[2] GONZALEZ, N. et al. A quantitative analysis of current security concerns and solutions for cloud computing. *Journal of Cloud Computing: Advances, Systems and Applications*, v. 1, n. 1, p. 1–18, 2012. ISSN 2192-113X. Disponível em: ⟨http://dx.doi.org/10.1186/2192-113X-1-11⟩.

[3] BUI, T. Analysis of docker security. *CoRR*, abs/1501.02967, 2015. Disponível em: ⟨http://arxiv.org/abs/1501.02967⟩.

[4] COMBE, T.; MARTIN, A.; PIETRO, R. D. To Docker or Not to Docker: A Security Perspective. *IEEE Cloud Computing*, v. 3, n. 5, p. 54–62, set. 2016. ISSN 2325-6095.

[5] TOZZI, C. *3 Container Security Advantages and 3 Security Challenges*. 2018. Disponível em: ⟨https://containerjournal.com/2018/08/16/3-container-security-advantages-and-3-security-challenges/⟩.

[6] TAVANGARIAN, D. Cloud Computing: New Paradigms and Challenges. In: MEESAD, P.; BOONKRONG, S.; UNGER, H. (Ed.). *Recent Advances in Information and Communication Technology 2016*. [S.l.]: Springer International Publishing, 2016. (Advances in Intelligent Systems and Computing), p. 3–3. ISBN 978-3-319-40415-8.

[7] KANG, M. et al. A Comparison of System Performance on a Private OpenStack Cloud and Amazon EC2. In: *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*. [S.l.: s.n.], 2017. p. 310–317.

[8] IZRAILEVSKY, Y.; BELL, C. Cloud Reliability. *IEEE Cloud Computing*, v. 5, n. 3, p. 39–44, maio 2018. ISSN 2325-6095.

[9] KULKARNI, P. *Getting your hands dirty with Containers*. [S.l.], 2016. 1-45 p. Disponível em: ⟨https://www.cse.iitb.ac.in/~prashanth/containers/seminar/manual.pdf⟩. Acesso em: 21 set. 2016.

[10] EDER, M. Hypervisor- vs. container-based virtualization. *Network Architectures and Services*, p. 11–17, 7 2016.

[11] MIERS, C. et al. Análise de Segurança para Soluções de Computação em Nuvem. In: *SBRC 2014 Minicursos*. [S.l.: s.n.], 2014.

[12] PAHL, C. Containerisation and the paas cloud. In: *Complex, Intelligent, and Software Intensive Systems (CISIS), 2015 Ninth International Conference on*. [S.l.: s.n.], 2015. p. 1–6. ISSN 2325-6095.

[13] LI, W.; KANSO, A. Comparing Containers versus Virtual Machines for Achieving High Availability. In: *2015 IEEE International Conference on Cloud Engineering*. [S.l.: s.n.], 2015. p. 353–358.

[14] SALAH, T. et al. Performance comparison between container-based and vm-based services. In: *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*. [S.l.: s.n.], 2017. p. 185–190. ISSN 2472-8144.

[15] SOUPPAYA JOHN MORELLO, K. S. M. *SP 800-190. Application Container Security Guide*. Gaithersburg, MD, United States, 2017. Disponível em: ⟨"https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-190.pdf"⟩. Acesso em: 14 mar. 2018.

[16] BERNSTEIN, D. *Containers and Cloud: From LXC to Docker to Kubernetes*. 2014. 81-84 p.

[17] CLUSTERHQ. *Container Market Adoption*. 2016. ⟨https://clusterhq.com/assets/pdfs/state-of-container-usage-june-2016.pdf⟩. Disponível em: ⟨https://clusterhq.com/assets/pdfs/state-of-container-usage-june-2016.pdf⟩.

[18] OpenStack Security. *Exploring Opportunities: Containers and OpenStack*. [S.l.], 2016. Disponível em: ⟨https://www.openstack.org/assets/pdf-downloads/Containers-and-OpenStack.pdf⟩. Acesso em: 21 set. 2016.

[19] ALLIANCE, C. S. *Security Guidance for Critical Areas of Focus in Cloud Computing V3.0.* [S.l.], 2011. Disponível em: ⟨http://www.cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf⟩. Acesso em: 03 mar. 2018.

[20] JANSEN, W.; GRANCE, T. *SP 800-144. Guidelines on Security and Privacy in Public Cloud Computing*. Gaithersburg, MD, United States, 2011. Disponível em: ⟨"http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-144.pdf"⟩. Acesso em: 21 set. 2016.

[21] ENISA. *Benefits, Risks and Recommendations for Information Security*. [S.l.], 2012.

[22] ZHANG, M.; MARINO, D.; EFSTATHOPOULOS, P. Harbormaster: Policy enforcement for containers. In: *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*. [S.l.: s.n.], 2015. p. 355–362.

[23] BACIS, E. et al. Dockerpolicymodules: Mandatory access control for docker containers. In: *Communications and Network Security (CNS), 2015 IEEE Conference on*. [S.l.: s.n.], 2015. p. 749–750.

[24] CONTAINERS, S. L. *Securing Linux Containers*. 2015. 1–25 p. ⟨https://www.sans.org/reading-room/whitepapers/linux/securing-linux-containers-36142⟩. Disponível em: ⟨https://www.sans.org/reading-room/whitepapers/linux/securing-linux-containers-36142⟩.

[25] NCC Group. technical report, *Understanding and Hardening Linux Containers*. 2016. ⟨https://www.nccgroup.trust/uk/our-research/understanding-and-hardening-linux-containers/⟩.

Disponível em: ⟨https://www.nccgroup.trust/uk/our-research/understanding-and-hardening-linux-containers/⟩.

[26] BARLEV, S. et al. Secure yet usable: protecting servers and linux containers. *IBM Journal of Research and Development*, v. 60, n. 4, p. 12:1–12:10, July 2016. ISSN 0018-8646.

[27] NCC Group. *Abusing Privileged and Unprivileged Linux Containers*. [S.l.], 2016. Disponível em: ⟨https://www.nccgroup.trust/uk/our-research/abusing-privileged-and-unprivileged-linux-containers/⟩.

[28] SELINUX. *SELinux Wiki*. ⟨http://selinuxproject.org/page/Main_Page⟩. Disponível em: ⟨http://selinuxproject.org/page/Main_Page⟩.

[29] APPARMOR. *AppArmor: Mandatory Access Control (MAC) system*. 2016. ⟨http://wiki.apparmor.net/index.php/Main_Page⟩. Disponível em: ⟨http://wiki.apparmor.net/index.php/Main_Page⟩.

[30] GLOBALPLATFORM. *TEE System Architecturev1.2*.

[31] INTEL SGX Homepage.

[32] ARNAUTOV, S. et al. SCONE: Secure linux containers with intel SGX. In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. Savannah, GA: [s.n.], 2016. p. 689–703. ISBN 978-1-931971-33-1.

[33] DOCKER. *Overview of Docker Hub*. 2017. ⟨https://docs.docker.com/docker-hub/⟩. Disponível em: ⟨https://docs.docker.com/docker-hub/⟩.

[34] GUMMARAJU, T. D. J.; TURNER, Y. *Over 30% of Official Images in Docker Hub Contain High Priority Security Vulnerabilities*. 2015. Disponível em https://www.banyanops.com/pdf/BanyanOps-AnalyzingDockerHub-WhitePaper.pdf. Disponível em: ⟨https://www.banyanops.com/blog/analyzing-docker-hub/⟩.

[35] POLVI, A. *CoreOS is building a container runtime, rkt*. 2016. Disponível em: ⟨https://coreos.com/blog/rocket/⟩.

[36] NAMESPACES, L. *namespaces(7) - Linux manual page*. 2017. ⟨http://man7.org/linux/man-pages/man7/namespaces.7.html⟩. Disponível em: ⟨http://man7.org/linux/man-pages/man7/namespaces.7.html⟩.

[37] ALVES, F. C. C. et al. Uma Revisão Sobre as Publicações de Sistemas de Detecção de Intrusão. *Revista de Informática Teórica e Aplicada*, v. 23, n. 2, p. 67–99, dec 2016. ISSN 21752745. Disponível em: ⟨http://www.seer.ufrgs.br/rita/article/view/RITA-VOL23-NR2-67⟩.

[38] ARCHWIKI. *Cgroups - ArchWiki*. 2016. ⟨https://wiki.archlinux.org/index.php/cgroups⟩. Disponível em: ⟨https://wiki.archlinux.org/index.php/cgroups⟩.

[39] KREUTZ, D. et al. Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE*, v. 103, n. 1, p. 63, 2015. Disponível em: ⟨http://arxiv.org/abs/1406.0440⟩.

[40] ALEDHARI, M. et al. *Protecting Internet Traffic: Security challengesand solutions*. IEEE Internet Technology Policy Community, 2017. Disponível em: ⟨https://internetinitiative.ieee.org/images/files/resources/white_papers/internet_traffic_feb2017.pdf⟩.

[41] CLOSE, M. *Protecting Sensitive Information in Docker Container Images*. 2016. Disponível em: ⟨https://www.ctl.io/developers/blog/post/tutorial-protecting-sensitive-info-docker⟩.

[42] BAL, B. *Automation and Orchestration with Docker and Containers*. 2016. Disponível em: ⟨https://mesosphere.com/wp-content/uploads/2016/06/TheNewStack_Book3_Automation_and_Orchestration_with_Docker_and_Containers-1-1.pdf⟩.

[43] WHEELER, D. A. *Cloud Security: Virtualization, Containers, and Related Issues*. 2016. Disponível em: ⟨http://www.dwheeler.com/essays/cloud-security-virtualization-containers.html⟩.

[44] BADGER, L. et al. *SP 800-146. Cloud Computing Synopsis and Recommendations*. Gaithersburg, MD, United States, 2012. Disponível em: ⟨http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-146.pdf⟩.

[45] HASHICORP. *Introduction to Vault*. 2016. ⟨https://www.vaultproject.io/intro/index.html⟩. Disponível em: ⟨https://www.vaultproject.io/intro/index.html⟩.

[46] CLUSTERHQ. *What is Flocker?* 2016. ⟨https://clusterhq.com/flocker/introduction/⟩. Disponível em: ⟨https://clusterhq.com/flocker/introduction/⟩.

[47] DOCKER. *Docker Swarm*. 2016. ⟨https://docs.docker.com/swarm/overview/⟩. Disponível em: ⟨https://docs.docker.com/swarm/overview/⟩.

[48] TRUST, T. *Twistlock Trust*. 2016. ⟨https://www.twistlock.com/trust/⟩. Disponível em: ⟨https://www.twistlock.com/trust/⟩.

[49] TRUST, T. *Twistlock Runtime*. 2016. ⟨https://www.twistlock.com/runtime/⟩. Disponível em: ⟨https://www.twistlock.com/runtime/⟩.

[50] HAT, R. *Ten Layers of Container Security*. [S.l.], 2017.