IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Recebido a 18 de Fevereiro de 2019, aceite a 10 de Abril de 2019, data de publicação 17 de Abril de 2019, data da versão actual 1 de Maio de 2019.

Identificador de Objecto Digital 10.1109/ACCESS.2019.2911732

Segurança dos contentores: Questões, Desafios, e a Estrada em Frente

SARI SULTAN, (Membro estudante, IEEE), IMTIAZ AHMAD, E TASSOS DIMITRIOU, (Membro sénior, IEEE)
Departamento de Engenharia Informática, Universidade do Kuwait, Safat 13060, Kuwait
Autor correspondente: Imtiaz Ahmad (imtiaz.ahmad@ku.edu.kw)

ABSTRACT Os contentores surgiram como uma alternativa leve às máquinas virtuais (VMs) que oferecem um melhor apoio à arquitectura dos microserviços. Prevê-se que o valor do mercado de contentores atinja 2,7 mil milhões de dólares em 2020, em comparação com 762 milhões em 2016. Embora sejam considerados o método padronizado para a implementação de micro-serviços, desempenhando um papel importante nos campos emergentes da computação em nuvem, tais como malhas de serviços, os estudos de mercado mostram que a segurança dos contentores é a principal preocupação e barreira à adopção para muitas empresas. Neste artigo, pesquisamos a literatura sobre segurança e soluções em contentores. Concluímos quatro casos de uso generalizado que devem cobrir os requisitos de segurança dentro do cenário de ameaça dos contentores de acolhimento. Os casos de utilização incluem: (I) protecção de um contentor contra aplicações no seu interior, (II) protecção entre contentores,
(III) proteger o hospedeiro de contentores, e (IV) proteger os contentores de um hospedeiro malicioso ou semi-selvagem. Verificámos que os três primeiros casos de utilização utilizam uma solução baseada em software que se baseia principalmente em funcionalidades de kernel Linux (por exemplo, namespaces, CGroups, capacidades, e seccomp) e módulos de segurança Linux (por exemplo, AppArmor). O último caso de utilização baseia-se em soluções baseadas em hardware, tais como módulos de plataforma confiáveis (TPMs) e suporte de plataforma confiável (p. ex., Intel SGX). Esperamos que a nossa análise ajude os investigadores a compreender os requisitos de segurança dos contentores e a obter uma imagem mais clara de possíveis vulnerabilidades e ataques. Finalmente, destacamos problemas de investigação abertos e futuras direcções de investigação que possam gerar mais investigação nesta área.

INDEX TERMS Containers, Docker, contentores Linux, virtualização ao nível do SO, virtualização leve, segurança, levantamento.

I. INTRODUÇÃO

As máquinas virtuais (VMs) proporcionam uma excelente segurança. No entanto, o seu isolamento de segurança cria um estrangulamento para o total de VMs que podem funcionar num servidor, porque cada VM deve ter a sua própria cópia do sistema operativo (SO), bibliotecas, recursos dedicados, e aplicações. Isto tem um efeito prejudicial sobre o desempenho (por exemplo, longo tempo de arranque) e tamanho de armazenamento. O advento da prática de desenvolvimento de software DevOps [1], [2] e microserviços sublinhou a necessidade de uma solução mais rápida do que as VMs, uma vez que não é eficiente executar cada microser- vício numa VM separada devido ao seu longo tempo de arranque e ao aumento da utilização de recursos.

A virtualização baseada em contentores emergiu como uma alternativa leve aos VMs. Muitos contentores podem partilhar o mesmo OS kernel em vez de terem uma cópia dedicada para cada um deles como nos VMs. Isto reduz grandemente o tempo de arranque e os recursos necessários para cada imagem. Por exemplo, um recipiente pode arrancar em 50 milissegundos, enquanto uma VM pode demorar até 30-40 segundos.

O editor associado que coordenou a revisão deste manuscrito e o aprovou para publicação foi Kashif Saleem.

onds para começar [3]. Muitas tecnologias de contentores estão disponíveis tais como LXC, OpenVZ, Linux-Vserver, sendo o Docker o predominante. A Figura 1a mostra uma arquitectura de contentor simples enquanto que a Figura 1b mostra uma arquitectura simples de arquitecto- VM. Os contentores são uma opção mais plausível para os micro-vices do que os VMs, devido aos numerosos benefícios tais como serem leves, rápidos, mais fáceis de implementar, e permitirem uma melhor utilização de recursos e controlo de versões. Os contentores estão a ser utilizados para diferentes aplicações tais como serviços de Internet das Coisas (IoT), carros inteligentes, computação de nevoeiro, malhas de serviço, e assim por diante [3]-[8].

A introdução de arquitecturas de microserviço ajudou a aumentar a agilidade do software, em que as peças de software se tornaram unidades independentes de desenvolvimento, versionamento, implementação e escalonamento [8]. Os microserviços são utilizados por numerosas organizações orgânicas como a Amazon, Spotify, Netflix, e Twitter para a entrega do seu software [9]. Os contentores são considerados o padrão para implantar micro-serviços e aplicações na nuvem [10]. Os contentores são também importantes para o futuro da computação em nuvem e espera-se que o seu valor de mercado atinja
2,7 mil milhões de dólares até 2020 (foi de 762 milhões de dólares em 2016) [11], [12], [12].

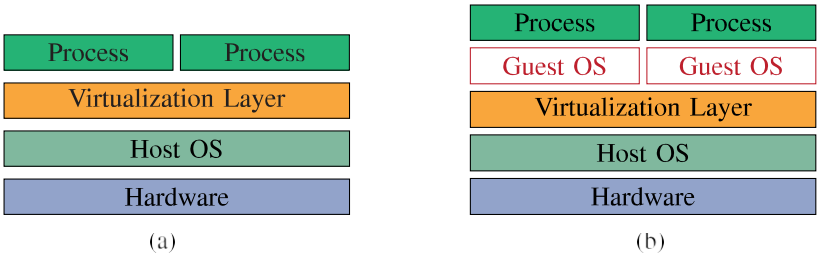


FIGURA 1. Comparação entre contentor e hipervisor. (a) Recipiente. (b) Máquina virtual.

Contudo, os contentores são menos seguros do que os VM [13]-[15]. Assim, a segurança é o principal obstáculo à adopção generalizada de contentores [16].

Embora existam vários inquéritos que abordam as VMs, não se concentram em questões de segurança de contentores [17]-[19]. Uma vez que os VMs se baseiam na partilha do kernel do SO entre eles, enquanto cada VM tem o seu próprio kernel, a segurança do contentor é diferente da do seu homólogo VM, uma vez que se baseia em diferentes suportes arquitectónicos e turais. Assim, a compreensão das ameaças e soluções de segurança dos contentores é muito importante devido à falta de revisões sistemáticas sobre elas na literatura. Isto é problemático porque cada uma das soluções apresentadas diz respeito a um caso de uso muito específico. Por exemplo, o suporte de plataforma de confiança (por exemplo, Intel Software Guard Extensions (SGX)) é utilizado principalmente para permitir o funcionamento de contentores num anfitrião não confiável, assim o seguimento desses diferentes casos de utilização pode ser frustrante para o leitor.

Neste trabalho, fornecemos quatro casos de utilização geral que devem abranger a maioria dos casos de utilização a nível do contentor de acolhimento. Isto deverá ajudar os leitores a compreender melhor as questões de segurança sobre os contentores e os mecanismos disponíveis para os proteger. Os casos de utilização são: (I) protecção de um recipiente contra aplicações no seu interior, (II) protecção entre recipientes, (III) protecção do hospedeiro contra recipientes, e (IV) protecção de recipientes contra um hospedeiro malicioso ou semi-selvagem. Discutimos as soluções baseadas em software disponíveis que são tipicamente utilizadas para os três primeiros casos de utilização e as soluções baseadas em hardware que são utilizadas para o último. O nosso modelo de ameaça para os quatro casos de utilização poderia ser utilizado pelos investigadores para melhorar a sua compreensão de possíveis nerabilidades e ataques vulcânicos e para ilustrar claramente o que as suas soluções proporcionam. Finalmente, destacamos os problemas abertos e as futuras direcções de investigação, a fim de motivar o trabalho futuro nesta excitante área.

O resto do presente documento está organizado da seguinte forma. A Secção II discute material de apoio, recursos relevantes, e critérios de selecção. Na Secção III, apresentamos o nosso modelo de ameaça e os casos de utilização propostos. Na Secção IV, apresentamos os mecanismos de protecção de software e hardware utilizados para garantir a segurança dos con- tinadores. Em particular, a Secção IV-A apresenta os mecanismos baseados em software, enquanto a Secção IV-B apresenta os baseados em hardware. Na Secção V é apresentada uma discussão sobre vulnerabilidades de contentores, explorações, ferramentas de dis- cobertura, e normas relevantes. Na Secção VI, discutimos futuras direcções de investigação e questões em aberto. Finalmente, a Secção VII conclui este documento.

II. ANTECEDENTES

Nesta secção, apresentamos material de fundo sobre arquitecturas de contenção, bem como arquitecturas monolíticas e de microserviços. A Secção II-A fornece detalhes de fundo sobre recipientes, enquanto a Secção II-B considera as arquitecturas monolíticas e de micro-serviços. A Secção II-C apresenta os nossos recursos seleccionados e critérios de selecção.

A. CONTAINERS

Diferentes nomes são usados para se referir a contentores na ture literária- incluindo virtualização de nível de SO e virtualização leve- alização. Docker, LXC, e RKT são exemplos de gestores de contentores. Muitos estudos concentram-se no Docker porque é o ambiente predominante de tempo de execução de contentores. A vir- tualização do hardware refere-se a VMs tradicionais e hipervisores.

Os contentores melhoram duas desvantagens principais das VMs [14]: primeiro, partilham o mesmo núcleo de SO e podem partilhar recursos enquanto cada VM precisa da sua própria cópia. Em segundo lugar, os contentores podem ser iniciados e parados quase instantaneamente enquanto os VMs precisam de tempo considerável para iniciar [3]. Os contentores também provaram ser mais eficientes do que as VMs para algumas aplicações tais como microserviços, porque são leves e não requerem uma cópia completa do SO para cada imagem. No entanto, os contentores ainda necessitam de um núcleo totalmente funcional que é partilhado entre diferentes contentores. Além disso, a concepção dos microserviços sublinha a importância dos contentores de estado efémero, em que qualquer persistência de dados vai para outro armazenamento ou serviço de dados. Os contentores são considerados a forma padrão de implementar os micro-serviços na nuvem [10].

Container as a Service (CaaS ou CoaaS) cria um novo modelo de entrega para cloud computing [3]. Muitas compa- nies oferecem serviços de contentores que permitem uma grande variedade de aplicações contentorizadas para vários mercados [20]. Embora a virtualização a nível de OS seja uma tecnologia promissora com muitos benefícios, ela enfrenta um grande número de desafios. Por exemplo, a partilha do núcleo do SO anfitrião introduz muitas questões de segurança, o que as torna menos seguras do que as VM [21].

A figura 1a mostra a arquitectura de um simples contentor. A vista da ave de uma pilha de contentores é necessária porque a implantação depende de várias partes. A figura 2 mostra componentes de pilha de con- tainer e tecnologias de realização; baseia-se na arquitectura de [22] que foi modificada para fundir as tecnologias de realização e os componentes de pilha juntos. Esta figura mostra que um contentor é um bloco de construção

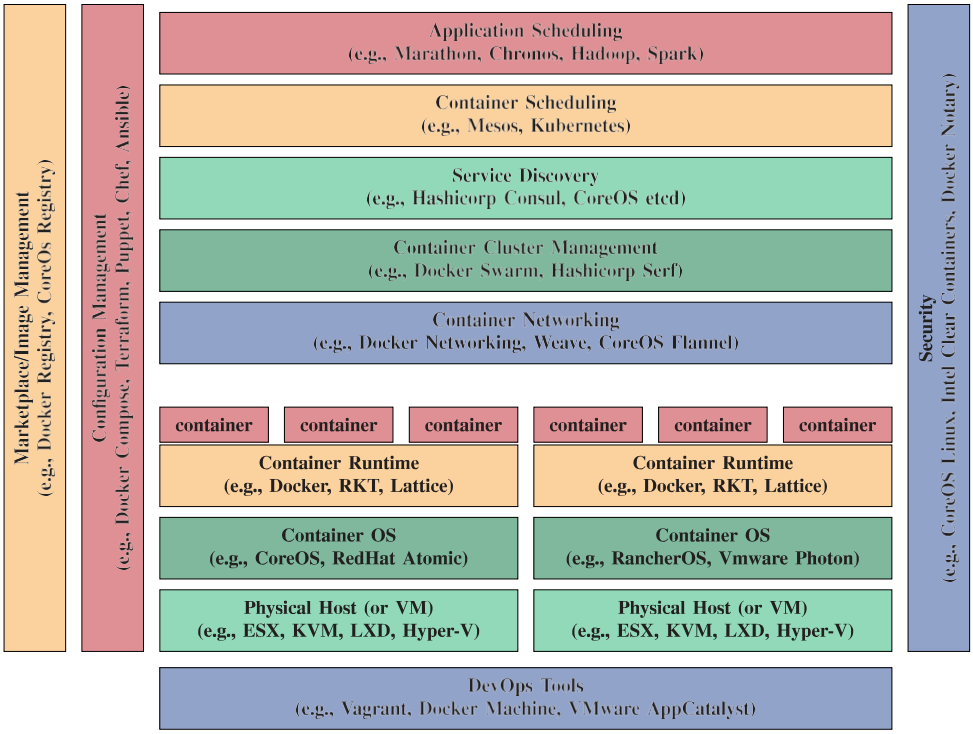


FIGURA 2. Tecnologias de empilhamento e realização de contentores.

para uma pilha tecnológica maior que pode ser utilizada para facilitar a implantação de microserviços. Peinl *et al*. [23] apresentaram um sur- vey sobre as ferramentas disponíveis para a gestão de contentores. Classificaram diferentes soluções, tanto na literatura académica como na indústria, bem como cartografaram-nas de acordo com os requisitos baseados num estudo de caso que forneceram. Além disso, identificaram lacunas nestas ferramentas e requisitos de integração e propuseram as suas próprias ferramentas a fim de ultrapassar estas deficiências.

B. ARQUITECTURAS MONOLÍTICAS E MICRO-SERVIÇOS

Uma aplicação monolítica refere-se a software cujas partes estão fortemente acopladas e não podem ser executadas independentemente [24]. Embora as aplicações monolíticas possam correr dentro de um recipiente, é altamente recomendada a utilização de arquitectura de microserviço quando se utilizam recipientes [25]. Antes do advento do Service Ori- ented Architectures (SOAs), e especificamente microservices, a maioria das aplicações costumava ser monolítica. Por outro lado, os microserviços ajudam a construir aplicações que consistem em peças soltas que podem ser operadas de forma independente.

As arquitecturas Microservice revolucionaram a forma como os cátions são construídos hoje em dia. Permitem que os criadores sejam mais inovadores e abertos às novas tecnologias. Por exemplo, se uma empresa quiser experimentar uma nova linguagem de programa - ming, pode construir um único microserviço com essa linguagem, o que terá um efeito mínimo em toda a aplicação, ao contrário da utilização de uma arquitectura monolítica, que requer a reescrita de todas as partes. Os microserviços e recipientes são assuntos estreitamente relacionados onde os recipientes são considerados a opção de implementação padronizada para microserviços.

[10]. Executar cada micro-serviço num VM separado não é eficiente porque os VMs são pesados em comparação com os contentores [25]. Os contentores são alternativas importantes aos VM e têm uma série de benefícios sobre eles, especialmente no perfor- mance e no tamanho. O advento dos contentores realçou a importância das arquitecturas de microserviço em relação às arquitecturas mono-litíticas mais antigas. Contudo, os contentores são afectados por numerosas questões de segurança que constituem os principais obstáculos à sua adopção pelas empresas. Dragoni *et al.* [24] apresentaram um estudo recente sobre a emergência de micro-serviços e como o seu desenvolvimento melhorou os inconvenientes da arquitectura monolítica.

C. REVISÃO BIBLIOGRÁFICA SOBRE SEGURANÇA DE CONTENTORES

Neste trabalho, incluímos artigos dos anais dos principais locais de investigação académica, revistas e livros. Ocasionalmente, baseámo-nos em pesquisas não publicadas ou publicadas sob formas não comerciais, tais como relatórios, declarações políticas, etc. Tais artigos foram incluídos porque a investigação sobre recipientes é um campo intrinsecamente prático que é dominado pela indústria e é publicado em diferentes fontes em linha. Os nossos critérios de selecção centraram-se nas seguintes áreas relacionadas com contentores: características de segurança, soluções, ameaças, vulnerabilidades, explorações, ferramentas, normas, metodologias de avaliação, aplicações, e alteradores de contentores - nativos. Confiámos principalmente no Google Scholar e utilizámos principalmente as seguintes palavras-chave de pesquisa: "Segurança de contentores", "Segurança de docas", "Contentores Linux". Depois removemos fontes que eram genéricas para os contentores e não discutimos

QUADRO 1. Especificações do modelo de ameaça para aplicações, recipientes e hospedeiro para os casos de utilização estudados. *"Semi"* refere-se ao semi-onestado. As aplicações em recipientes semi-onestos/maliciosos podem ser semi-onestas ou maliciosas também.

Use Case	Apps can be			Containers can be			Host can be		
	honest	semi	malicious	honest	semi	malicious	honest	semi	malicious
(I) Protect container from applications	✓	✓	✓	✓	-	-	✓	-	-
(II) Inter-container protection	✓	-	-	✓	✓	✓	✓	-	-
(III) Protect host from containers	✓	✓	✓	✓	✓	✓	✓	-	-
(IV) Protect containers from host	✓	-	-	✓	-	-	✓	✓	✓

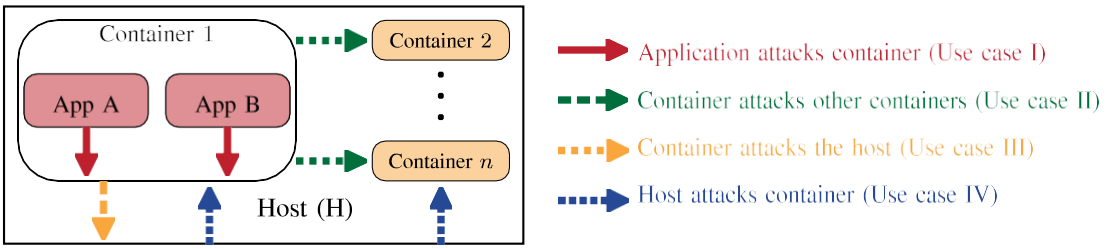


FIGURA 3. Visão geral dos requisitos de protecção de segurança em contentores.

questões de segurança. Finalmente, utilizámos citações retrospectivas dos recursos seleccionados para expandir a nossa pesquisa. A maioria das referências foi publicada entre 2014 e 2018, incluindo 21 artigos em 2016, 45 artigos em 2017, e 25 artigos em 2018. A pesquisa de segurança de contentores começou a ganhar ímpeto em 2015.

III. MODELO DE TRANSPORTE

Muitos casos de utilização podem ser derivados para ilustrar questões e soluções de segurança de contentores Linux. Acreditamos que seria inefectivo criar uma lista exaustiva de todos esses casos de utilização, uma vez que seria frustrante para o leitor mantê-los a par. Assim, fornecemos uma nova taxonomia para casos de utilização ao nível do contentor de acolhimento para melhor identificar os requisitos de protecção e possíveis soluções. O registo e a orquestração não são o foco principal da nossa investigação, cujo objectivo é lançar luz sobre os níveis do hospedeiro e do contentor que representam o foco principal para as tecnologias de contentores.

Consideramos um anfitrião H que tem $|C|$ ($|.|$ denota a cardinalidade de um conjunto) número de contentores $C = \{c_1, c_2, \ldots, c_n\}$. Cada contentor $c_i \in C$ pode executar pelo menos uma aplicação de um conjunto de aplicações A^c i = $\{a_1, a_2, ----, a_m\}$. Além disso, assumimos que o anfitrião H e cada contentor ($c_i \in C$) têm recursos limitados, o que os torna vulneráveis a ataques de disponibilidade-direccionados. Abordamos quatro casos de utilização, cada um dos quais com o seu próprio modelo adversário e objectivos de segurança. A especificação do modelo de ameaça para aplicações, contentores, e hospedeiros é apresentada no Quadro 1. Um adversário *semi-onestado* é um adversário passivo que poderia cooperar na recolha de informação mas que não se desviará da execução do protocolo. Um adversário *malicioso* é um adversário activo que pode desviar-se da especificação do protocolo a fim de recolher informação, enganar, ou perturbar o sistema e visar outros componentes do sistema. Os quatro casos são mostrados na Figura 3 e são descritos nas secções seguintes.

1) CASO I DE UTILIZAÇÃO PROTECÇÃO DE UM RECIPIENTE CONTRA APLICAÇÕES NO SEU INTERIOR

Neste caso de utilização, cada aplicação dentro de um recipiente em funcionamento $c_i \in C$ pode ser honesta, semidonesta, ou maliciosa. Assumimos que as aplicações não podem quebrar as políticas de controlo de acesso, se definidas. Além disso, assumimos que algumas aplicações podem exigir privilégios de raiz (ou partes do acesso total de raiz). Acreditamos que este é um caso muito importante porque se uma aplicação pudesse
ganhar controlo sobre o gestor de contentores, poderá ser capaz de visar o sistema anfitrião e outros contentores dentro do sistema. No entanto, uma aplicação de ataque pode assumir o controlo de um contentor vulnerável específico. O nosso objectivo é minimizar os ataques intra contentor. A tabela 2 mostra uma lista de possíveis ataques e soluções para este caso de utilização.

2) CASO DE UTILIZAÇÃO II: PROTECÇÃO ENTRE CONTENTORES

Neste caso de utilização, assumimos que um ou mais dos recipientes são semi-onestos ou maliciosos, ou seja, $\exists CM \subseteq C, |CM| \geq 1$ onde CM denota um conjunto de con- tiners semi-onestos ou maliciosos. Estes contentores podem estar dentro do hospedeiro H ou em diferentes hospedeiros. Estar em diferentes anfitriões é importante para o emergente campo de malhas de serviço [8]. Assumimos que as aplicações dentro de $c_i \in CM$ também podem ser maliciosas ou semi-honestas. Embora uma aplicação atacante possa assumir o controlo de outras aplicações dentro de um contentor, assumimos que as aplicações em contentores honestos permanecem honestas para este caso de utilização porque proteger as aplicações uma da outra não é um problema específico do contentor.

Seguem-se alguns ataques que poderiam ser executados. Um contentor semi-honesto poderia ser capaz de aceder a dados confidenciais de outros contentores, aprender padrões de utilização de recursos e visar a integridade da informação da aplicação. Além disso, um contentor mal-intencionado pode realizar ataques semelhantes aos de um contentor semi-honestado e pode visar a disponibilidade de outro contentor. Por exemplo, um contentor malicioso pode consumir mais para a

QUADRO 2. Possíveis cenários de ataque para caso de utilização (I) (proteger um recipiente das aplicações nele contidas) e soluções sugeridas. (CVE significa Common Vulnerabilities and Exposure (Vulnerabilidades e Exposição Comuns).

Threat	Possible Attack	Possible Scenario	Possible Solution
Image vulnerabilities	Remote code execution	Remote code execution using ShellShock vulnerability that affected most Linux systems (CVE-2014-6271).	Periodic vulnerability scanning for images and applications.
Image configuration defects	Unauthorized access	Running an application with unnecessary root privileges will allow the application to have full control over the container.	Running application on the least privilege needed. Linux Capabilities are also used to provide a specific functionality of the root privileges (not all functionalities).
	Network-based intrusions	Enabling remote access (e.g., SSH) would give attackers an opportunity to gain access over the container if not configured properly.	Management on containers should be done through container runtime APIs. SSH and other remote access tools should not be enabled [31].
Embedded malware	Virus, Worm, Trojan, Ransomeware, etc.	Ransomeware gaining access to a container.	Monitoring the processes with anti-malware software. Keep this software up-to-date. Running trusted apps only.
Embedded clear-text secrets	Information disclosure, tampering, privacy issues	Database connection parameters that are stored in clear text in the container can be used by a compromised application to access/edit the database.	Secrets should be stored outside the image. Orchestrator (e.g., Kubernetes) has native management of secrets.
Use of untrusted images	Many attacks since the image is untrusted and can contain various threats such as backdoors	Untrusted images might have a preinstalled backdoor inside them.	Use only trusted images/registries, and verify the images.
Vulnerabilities within container runtime	Privilege escalation and container runtime escape attack	CVE-2017-5123 mentions a vulnerability in Docker runtime that allows applications to modify the capabilities.	Container runtime should be monitored for vulnerabilities and updated periodically.
Application vulnerabilities	Denial-of-Service (DoS)	A vulnerable application (e.g., Apache server with Slowloris vulnerability) can be targeted with different types of attacks one of which is DoS which would target the availability of the container.	Running applications with least privilege possible to reduce the possible damage when compromised. The root filesystem should be read-only. Scanning applications periodically for vulnerabilities.

recursos de acolhimento dedicados a contentores, caso em que torna inúteis outros contentores. O nosso objectivo aqui é proteger os contentores uns dos outros, um caso perfeito seria que os contentores não soubessem nada sobre outros contentores (como VMs), a menos que fosse necessário (por exemplo, comunicação em rede). As exigências de protecção não são específicas dos contentores.

Um dos ataques de alto risco que afecta os contentores é o Melt- down. Em [26], os autores montaram com sucesso um ataque a Docker, LXC, e OpenVZ. Este ataque permitiu que o adversário - sary divulgasse informações sobre o núcleo do sistema operativo e todos os outros contentores que funcionam no mesmo sistema. Isto teve um efeito detrimen- tal nos fornecedores de serviços de nuvem, onde um utilizador malicioso podia aceder a informação de todos os outros contentores alojados no sistema. Spectre [27] é outra ameaça séria de conter- ers, onde engana outras aplicações para aceder a locais arbi- trary na sua memória. Tanto os ataques Spectre como Meltdown representam uma séria ameaça aos contentores [26], [27]. A tabela 3 mostra uma lista de possíveis ataques e soluções para este caso de utilização.

3) CASO DE UTILIZAÇÃO III: PROTEGER O HOSPEDEIRO (E AS APLICAÇÕES NO SEU INTERIOR) DOS CONTENTORES

Neste caso de utilização, assumimos que pelo menos um contentor é semi-honesto ou malicioso dentro do hospedeiro H ($CM \subseteq C$, $|CM| \geq 1$, onde CM denota um conjunto de contentores semi-honestos ou maliciosos). Assumimos que as aplicações que estão dentro de $c_i \in CM$, bem como as aplicações em contentores honestos, podem ser semi-honestas ou maliciosas. Um contentor semi-honestado pode ter acesso a informação confidencial do hospedeiro ou mesmo visar a sua integridade. Um contentor malicioso pode ter como alvo a disponibilidade do hospedeiro, consumindo os seus recursos. O nosso objectivo aqui é eliminar a capacidade de qualquer contentor de visar a confidencialidade, integridade e disponibilidade dos componentes do hospedeiro. Um cenário perfeito seria para fazer os contentores agirem como VMs. (Recentemente, a Intel fundiu o seu projecto de contentores transparentes com os contentores kata e afirma que os contentores kata têm um isolamento semelhante ao dos VM [28]). A tabela 4 mostra uma lista de possíveis ataques e soluções para este caso de utilização. Uma das aplicações promissoras dos contentores é a sua utilização para limitar os ataques de drenagem de recursos de software.

QUADRO 3. Possíveis cenários de ataque para caso de utilização (II) (protecção entre contentores) e soluções sugeridas.

Threat	Possible Attack	Possible Scenario	Possible Solution
Untrusted images	Attack all containers within the host	Untrusted images pose a serious threat to containers in the host. An untrusted image might have pre-installed vulnerability scanner which can scan all images in the network to find vulnerabilities and later exploit them.	Use only trusted images/registries, and verify the images using signatures. Vulnerability scanning.
Application vulnerabilities	DoS on other containers	A vulnerable application might cause DoS attack on neighboring containers (e.g., syn flood, ICMP flood) or using large amount of resources from the host, in which it affects other containers' operation.	Periodic vulnerability scanning for images and applications.
Poorly separated inter-container traffic	ARP spoofing (man-in-the-middle attack) and MAC flooding	If traffic among containers is poorly separated this might allow a compromised container to perform man-in-the-middle attack on other containers [32].	Containers should not be able to communicate unless necessary. Network should be configured as least-necessary privilege.
	DoS	A compromised container might flood the network, which renders other containers useless or highly degrade the performance.	Containers should not be able to communicate unless necessary. Chelladhurai et al. [33] proposed an algorithm to prevent DoS attacks in Docker systems via limiting container resources.
Unbounded network access from containers	Port/vulnerability scanning	If a compromised container has access to a network of other containers from various sensitivity levels it can easily perform port/vulnerability scanning and discover vulnerable containers to be targeted.	Containers should be separated into virtual networks based on sensitivity level [31]. Network monitoring for anomalies (port scanning) should be implemented too.
Insecure container runtime	Remote code execution, DoS	CVE-2014-9357, CVE-2014-6407 are some examples of remote code execution vulnerabilities in Docker. An attacker could use a vulnerability or configuration error in the runtime to target other containers (e.g., stop other containers).	Container runtime should be monitored for vulnerabilities and updated periodically.

serviços. Como uma extensão do seu trabalho anterior em [29], Catuogno *et al.* [30] propuseram uma metodologia para medir a eficácia dos contentores contra ataques de drenagem de recursos em que os recursos de um hospedeiro poderiam ser drenados por um serviço suspeito. Assim, propuseram a execução de cada serviço num con- tiner, em que as restrições utilizadas nos contentores ajudam a moldar os seus recursos permitidos. Os autores utilizaram o Docker para proteger contra ataques de drenagem de energia que provaram ser uma técnica simples e eficaz.

4) CASO DE UTILIZAÇÃO IV: PROTECÇÃO DOS RECIPIENTES CONTRA O HOSPEDEIRO

A exploração de contentores em hospedeiros não confiáveis deve ser evitada, especialmente com o advento do CaaS, onde os contentores podem ser alugados a um fornecedor de CaaS. Neste caso de utilização, assumimos que os contentores são honestos, mas o hospedeiro é ou semi-honesto ou malicioso. Um hospedeiro semi- honesto pode aprender sobre informação confidencial sobre contentores porque controla dispositivos de trabalho em rede, memória, armazenamento, e processadores. Um hospedeiro malicioso pode também visar a integridade do contentor e a(s) sua(s) aplicação(ões). Numerosos ataques passivos e activos podem ser lançados a partir de hospedeiros semi-selvagens e maliciosos contra os con- tainers neles contidos. Exemplos de ataques passivos incluem a definição de perfis

actividades de aplicação em contentores e acesso não autorizado para dados de contentores. Os ataques activos podem ser mais prejudiciais, uma vez que um hospedeiro malicioso pode alterar o comportamento da aplicação.

IV. MECANISMOS DE PROTECÇÃO DE SOFTWARE E HARDWARE

A figura 4 mostra os casos de utilização com base na nossa taxonomia e
soluções possíveis. Utilizámos a categorização baseada em soluções, uma vez que os três primeiros casos de utilização (discutidos na Secção III) dependem de soluções baseadas em software e o último pode ser resolvido com soluções baseadas em hardware. As soluções baseadas em software são discutidas na Secção IV-A e as soluções baseadas em hardware são discutidas na Secção IV-B.

A. MECANISMOS DE PROTECÇÃO COM BASE EM SOFTWARE

Nesta secção, discutimos as soluções de software disponíveis que são utilizadas para a segurança de contentores (listadas na Figura 4). A tecnologia de contentores depende fortemente de soluções baseadas em software que são ou Características de Segurança Linux (LSFs) ou Módulos de Segurança Linux (LSMs). Na Secção IV-A.1, discutimos os LSFs porque a maioria dos contentores são baseados em Linux e o nosso âmbito está centrado nos contentores Linux. Primeiro, apresentamos *namespaces* na Secção IV-A.1.a e fornecemos alguns

QUADRO 4. Possíveis cenários de ataque para caso de utilização (III) (proteger o hospedeiro dos contentores) e soluções sugeridas.

Threat	Possible Attack	Possible Scenario	Possible Solution
Host OS attack surface	Attacks on unnecessary services	Print spooler service and other services are unnecessary on almost all servers. Hence, keeping them running will make the host inherit their vulnerabilities. Print spooler for example can be exploited (example vulnerabilities are MS16-087 and CVE-2010-2729) which enables remote code execution and privilege escalation.	Use container specific OS (e.g., CoreOS, RancherOS). If this is not possible, then follow NIST's guidelines for securing servers [34]. Stop all unnecessary services.
Containers share host OS kernel	Container escape attack	CVE-2017-5123, shocker exploit, CVE-2016-5195 (Dirty Cow) are examples of recent vulnerabilities that allow containers to escape the container runtime and target the host OS. This could allow remote code execution, which eventually leads to compromised host.	Periodic vulnerability scanning for images, applications, and container runtime. Namespaces, capabilities, and LSMs are important to control such issues.
Resources accountability	DOS	A container might consume most of the host resources causing DoS on the host and other containers in it.	Capabilities are used to control how many resources are allowed to be used by each container.
Host filesystems	Tampering	Mounting a local file system on the host allows a container with sufficient permission to cause data tampering, this is especially dangerous for sensitive directories with configuration data [31].	Containers should run with minimal set of permissions, and file changes of containers should persist to specific storage volumes for this purpose [31].

exemplos da sua utilização. Em segundo lugar, *os CGroups* são investigados na Secção IV-A.1.b. Em terceiro lugar, *as capacidades* são abordadas na Secção IV-A.1.c. Finalmente, o *modo de cálculo seguro (sec- comp)* é abordado na Secção IV-A.1.d. Na Secção IV-A.2, discutimos LSMs e fornecemos uma lista dos LSMs mais comuns.

1) CARACTERÍSTICAS DO NÚCLEO DO LINUX

a: NAMESPAÇOS

Namespaces realizam o trabalho de isolamento e virtualização dos recursos do sistema para uma colecção de processos. Os Namespaces funcionam como um divisor das tabelas identificadoras e outras estruturas ligadas aos recursos globais do kernel em instâncias isoladas. Partilham os sistemas de ficheiros, processos, utilizadores, nomes de hosts, e outros componentes. Assim, cada espaço de nomes de sistemas de ficheiros terá a sua tabela de montagem privada e directório de raiz. Para cada contentor, pode ser vista uma visão única dos recursos. A visão limitada dos recursos para um processo dentro de um contentor também pode ser alargada a um processo infantil [35]. Os espaços de nomes são componentes cruciais do edifício para controlar quais os recursos que o contentor pode ver.

Namespaces asseguram o isolamento dos processos que são executados - ning num contentor para os cegar de ver outros processos a correr num contentor diferente [36]. No entanto, uma questão com namespaces é que alguns recursos ainda não têm consciência de namespace, tais como dispositivos [35]. Existem inúmeros namespaces disponíveis, cada um dos quais é responsável pelo isolamento de recursos específicos. A tabela 5 mostra uma lista de namespaces disponíveis e que recurso é utilizado para isolar [37].

Exemplo de Namespaces: Espaço de nomes PID

O espaço de nomes PID assegura que um processo só verá processos que estão dentro do seu próprio espaço de nomes PID (por exemplo, um con- tainer só verá os seus processos). A Figura 5 mostra um exemplo

de isolamento PID para processos de pais e filhos. Na máquina hospedeira, notamos que o processo de *init* tem PID 1. No entanto, quando o processo de criança para um recipiente começa, o espaço de nomes PID permite-lhe começar a numerar os PID a partir de 1 dentro do contentor. Este número será mapeado para um PID diferente na máquina anfitriã. Neste exemplo, o PID 6 no hospedeiro será mapeado para o PID 1 na máquina. Da mesma forma, os PID 7, 8, e 9 serão mapeados para 2, 3, e 4 respectivamente.

Protecção entre contentores utilizando Namespaces (Usar caso II) Tal como discutido anteriormente, os namespaces são poderosos recursos Linux para isolar recursos entre diferentes contentores. Isto ajuda a evitar que os contentores acedam aos recursos uns dos outros, o que aumenta a sua segurança. Por exemplo, com... espaços de nomes, um contentor pode facilmente ver os processos de outro contentor e interagir com eles (assumindo que tem privilégios suficientes). No entanto, ao utilizar os namespaces do PID, um con- tainer só verá os seus próprios processos. Outros namespaces podem ser aplicado a contentores para isolar diferentes recursos.

Proteger o anfitrião dos contentores que utilizam Namespaces (Caso III de utilização)

À semelhança do que discutimos na secção anterior, se um contentor consegue ver os processos do anfitrião, então isto pode representar um sério risco de segurança. O mesmo exemplo dos nomes PID - o ritmo pode ser aplicado aqui também. Um contentor comprometido que tenha conseguido chegar a um privilégio de raiz pode perturbar o funcionamento dos processos do hospedeiro. Contudo, com os nomes PID - ritmos, isolar a visão do contentor aos seus próprios processos ajuda a mitigar tais riscos.

Soluções que utilizam namespaces

Jian e Chen [38] estudaram o ataque de fuga do contentor para Docker. Apresentaram uma técnica de defesa que tenta resolver o ataque de fuga com base no estado do namespace

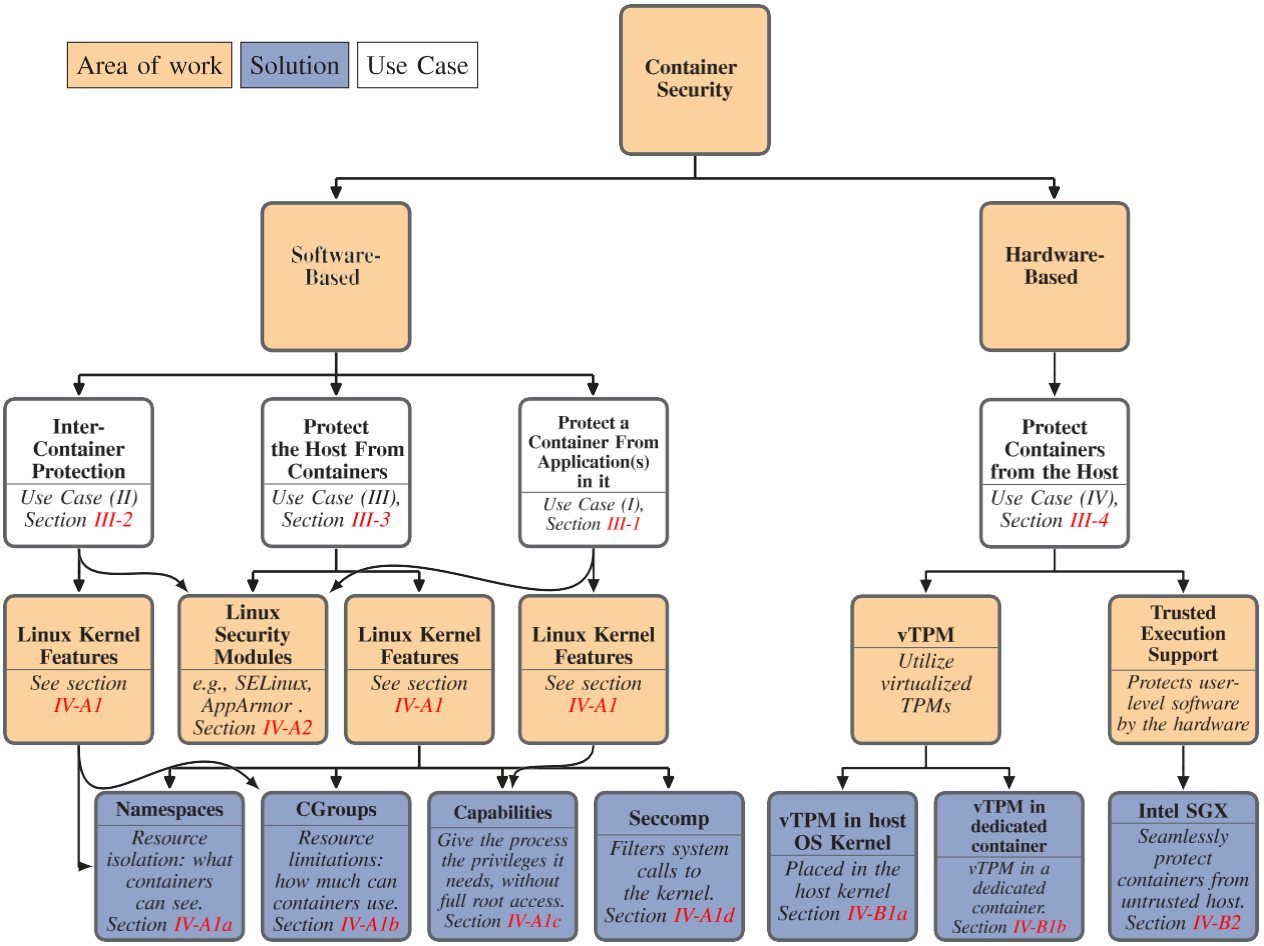
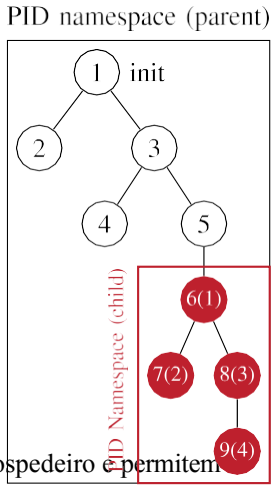


FIGURA 4. Requisitos de protecção, casos de utilização, e soluções.

QUADRO 5. Linux forneceu namespaces.

Namespace	Constant	Isolates
IPC	CLONE_NEWIPC	System V IPC, POSIX message queues
Network	CLONE_NEWNET	Network devices, stacks, ports, etc.
Mount	CLONE_NEWNS	Mount points
PID	CLONE_NEWPID	Process IDs
User	CLONE_NEWUSER	User and group IDs
UTS	CLONE_NEWUTS	Hostname and NIS domain name



inspecção durante a execução do processo. Isto deverá ajudar na detecção de anomalias e prevenir ainda mais ataques de fuga. A principal motivação do seu trabalho é que se um adversário pró-grama ganhou acesso à raiz, tentará alterar os espaços de nomes. A solução proposta detecta o estado do namespace e assinala quaisquer alterações que possam indicar um contentor comprometido.

Gao *et al.* [39] estudaram os canais de fuga de informação dentro de recipientes (este estudo foi expandido mais tarde em [40]). Abordam os canais que poderiam vazar informação do hospedeiro e permitem

FIGURA 5. Isolamento do espaço de nomes do PID.

adversários para lançarem ataques avançados contra o fornecedor de ser-vício da nuvem. Os autores utilizaram recipientes Docker e LXC como banco de ensaio e verificaram os canais de fuga de informação descobertos em cinco grandes fornecedores de serviços de nuvem. Eles mostraram que esses canais podem aumentar o efeito do ataque e reduzir

o seu custo. Um dos ataques importantes que analisaram é o ataque de energia, em que um adversário pode lançar cargas de trabalho intensivas quando o sistema já está em alta utilização com base na informação vazada, o que poderia afectar negativamente o desempenho do sistema. Para além de discutirem as causas de raiz da fuga de informação em contentores, os autores propuseram um mecanismo de defesa em duas fases. Implementaram um espaço de nomes baseado em energia para proporcionar um consumo de grãos finos ao nível do contentor. A sua avaliação mostrou que este mecanismo é eficaz na neutralização de ataques de poder. No entanto, Yu *et al.* [41] mostram que a solução apresentada em
[39] não é muito eficiente, uma vez que torna os contentores pesados, semelhantes aos VMs.

b: GRUPOS DE CONTROLO (CGROUPS)

Os CGroups são funcionalidades Linux que controlam a responsabilidade e limitação da utilização de recursos, tais como tempo de execução da unidade central de processamento (CPU), memória do sistema, entrada/saída (E/S), e largura de banda da rede. Em contraste com os namespaces, os CGroups limitam quantos recursos podem ser utilizados enquanto os namespaces controlam que recursos um contentor pode ver (ou seja, isolamento). Além disso, os CGroups impedem os contentores de utilizar todos os recursos disponíveis e de passar fome noutros processos. Um CGroup é arranjado como uma fatia para cada recurso. Depois, um conjunto de tarefas pode ser anexado a cada grupo CG específico. Assim, os grupos de tarefas são forçados a utilizar a sua própria parte dos recursos [35], [42].

Protecção entre contentores utilizando CGroups (Caso de utilização II)

Os CGroups são vitais para a protecção de recipientes. Conforme discutido, os CGroups circunscrevem a utilização de recursos permitida, pelo que um contentor não pode utilizar mais recursos do que o que lhe é designado. Isto ajuda a proteger outros recipientes de numerosos ataques, tais como ataques de Negação de Serviço (DoS). Por exemplo, um contentor pode utilizar tanto da RAM disponível do hospedeiro que outros contentores não podem funcionar correctamente.

Proteger o hospedeiro de contentores utilizando CGroups (Caso de utilização III)

Semelhante à protecção entre contentores, os CGroups forçam os limites de utilização de recursos nos contentores. Isto ajuda a evitar que o contentor efectue um ataque DoS ao próprio hospedeiro. Além disso, os CGroups dão ao hospedeiro o poder não só de limitar a utilização de recursos, mas também de contabilizar a quantidade do recurso utilizado. Isto pode ajudar a implementar quotas de utilização que podem ser um factor muito importante no modelo emergente de entrega de CaaS de computação em nuvem.

Soluções que utilizam CGroups

Chen *et al.* [43] apresentaram um mecha- nismo de protecção em tempo real contra ataques DoS chamado ContainerDrone. Primeiro, a protecção DoS da CPU utiliza grupos CG para atribuir um conjunto de núcleos a cada tarefa. Também utiliza características Docker para restringir um processo de elevar a sua prioridade. Depois, para protecção de DoS de memória, os autores demonstraram experimentalmente que os CGroups não são eficientes porque embora possam restringir a largura de banda de memória atribuída, as aplicações maliciosas poderiam ainda assim utilizar um acesso intensivo para essa quantidade de memória. Assim, para proteger a memória contra ataques de DoS, utilizaram um kernel MemGuard

para evitar que cada núcleo de CPU exceda o acesso à memória.

c: CAPACIDADES

Os sistemas Linux implementam a opção binária de dicotomia radicular e não radicular. No contexto dos contentores, essas opções binárias podem ser problemáticas. Por exemplo, um servidor web (por exemplo, Apache) precisa de se ligar a uma porta específica (por exemplo, porta TCP 80). Sem utilizar capacidades, o processo do servidor web deve ter acesso à raiz para executar a sua tarefa. Isto representa um grande perigo, porque se for comprometido, um atacante será capaz de controlar todo o sistema. As capacidades transformam a dicotomia raiz e não raiz em controlo de acesso de grão fino [36]. Assim, os contentores (normalmente não o daemon ou o gestor de contentores) não precisarão de ter o privilégio total de raiz (assumindo que existe uma capacidade disponível para as tarefas necessárias). Existem trinta e oito capacidades que cobrem uma grande variedade de tarefas [44].

As capacidades são muito importantes para proteger um recipiente de correr aplicações no seu interior (Utilize o caso I). Para o nosso exemplo anterior sobre servidores web, um contentor pode atribuir ao processo a capacidade CAP_NET_BIND_SERVICE. Isto permite que o contentor execute uma versão desse servidor web sem requerer acesso total à raiz. Assumindo que o servidor web contém uma vul- nerabilidade que tenha sido explorada por um adversário, a existência desta capacidade irá circunscrever o adversário a uma única operação de raiz (ou seja, portos de ligação). Por outro lado, se a capacidade não tiver sido estabelecida, a aplicação comprometida ou maliciosa pode realizar operações de raiz completas no contentor. Além disso, as capacidades são importantes para proteger o hospedeiro dos contentores (caso de utilização III). No exemplo anterior, vimos que um contentor pode limitar as suas aplicações mas o que acontece se o próprio contentor for malicioso? Isto causa um risco directo para o hospedeiro. Assim, um conjunto de capacidades pode ser atribuído ao contentor, o que poderia reduzir o risco de ameaças operacionais de raiz.

d: MODO DE CÁLCULO SEGURO (SECCOMP)

Seccomp é uma característica do kernel Linux que filtra as chamadas do sistema para o kernel. O Secomp é mais fino do que as capacidades [45], uma vez que diferentes perfis seccomp podem ser aplicados a diferentes filtros. Isto ajuda a diminuir o número de chamadas de sistema provenientes de contentores, o que poderia reduzir ainda mais as possíveis ameaças, uma vez que a maioria dos ataques aproveita as explorações do kernel através de chamadas de sistema.

Soluções que utilizam o Seccomp

Lei *et al.* [46] apresentaram Split-Phase Execution of Applica- tion Containers (SPEAKER) com o objectivo de diferenciar entre as chamadas de sistema necessárias e desnecessárias feitas pelos contentores. SPEAKER é baseado no seccomp Linux. Os autores observaram que as chamadas de sistema são utilizadas na fase de arranque a curto prazo dos contentores, pelo que podem ser retiradas com segurança da operação a longo prazo dos contentores. A SPEAKER reduz o número de chamadas de sistema feitas pelos contentores através da diferenciação entre essas fases de curto e longo prazo. Alargaram o seccomp Linux para actualizar automática e dinamicamente as chamadas de sistema disponíveis para aplicações quando transitam de

a fase de arranque a curto prazo para a fase de arranque a longo prazo. Avaliaram também o SPEAKER on Docker hub images para servidores web populares e contentores de datastore e encontraram uma redução de mais de 35% nas chamadas do sistema com uma sobrecarga de desempenho unimportante.

Wan *et al.* [47] apresentaram uma solução para caixas de areia de minas para contentores baseada em testes automáticos. Durante a fase de testes, a solução proposta extrai as chamadas do sistema utilizado pelo contentor. Utilizando Seccomp, a solução cria um perfil para cada aplicação com base nas chamadas de sistema vistas durante a fase de teste e nega todas as outras chamadas. Há duas questões principais com esta solução. Primeiro, a criação do perfil leva um tempo relativamente longo (ou seja, cerca de 11 minutos). Em segundo lugar, um contentor vul- nerável ou comprometido poderia aceder às chamadas de sistema exploráveis necessárias durante a fase de teste. No entanto, assumindo que o recipiente era seguro durante os testes, então o perfil com cremalheira deverá reduzir consideravelmente a superfície de ataque.

e: NAMESPACES, CGROUPS, SECCOMP, E UTILIZAÇÃO DE CAPACIDADES NO ESTIVADOR

Docker gera automaticamente um conjunto de espaços de nomes de contentores e grupos de controlo quando o contentor é iniciado com Docker run [36]. O Docker utiliza namespaces e grupos CG para criar um ambiente virtual seguro para os seus contentores. Por exemplo, para oferecer uma rede separada para cada contentor, o espaço de nomes de rede isola alguns recursos de rede como os endereços Internet pro- tocol (IP) [48].

O Docker depende dos CGroups para agrupar os processos em curso no contentor. Os CGroups revelam métricas sobre a utilização de CPU e blocos I/O juntamente com a gestão dos recursos do Docker, tais como CPU e memória. As configurações dos recursos do Docker ou são com limites rígidos ou suaves. Os limites rígidos são utilizados para especificar uma quantidade específica de recursos para o contentor. Limites suaves dão ao contentor os recursos necessários na máquina [49].

A questão com as capacidades expressas pela equipa Docker em [36] é que o conjunto padrão de capacidades pode não proporcionar um isolamento de segurança completo. As configurações por defeito do Docker utilizam as capacidades listadas na Tabela 6 [36], [50]. O Docker adiciona suporte para adição e remoção de capacidades. Além disso, os utilizadores podem definir o seu próprio perfil. Devemos notar aqui que à medida que o número de capacidades adicionadas ao contentor aumenta, haverá um maior risco de segurança. Isto porque o contentor será capaz de executar mais tarefas privilegiadas de raiz.

O perfil seccomp por defeito do Docker bloqueia 44 das 300 chamadas de sistema disponíveis [45]. Contudo, Lei *et al.* [46] afirmam que Docker não pode personalizar as chamadas de sistema para uma aplicação específica.

2) MÓDULOS DE SEGURANÇA LINUX (LSMs)

Morris *et al.* [51] alegaram que os anismos de controlo de acesso melhorado não são amplamente aceites para manter os sistemas operativos. Isto deve-se ao facto de não haver consenso sobre a solução certa dentro da comunidade de segurança. Os LSMs permitem a implementação de uma grande variedade de modelos de segurança no kernel Linux como módulos carregáveis [51]. Isto significa que um utilizador pode seleccionar o

QUADRO 6. Capacidades por defeito do Docker.

Capability	Description
CHOWN	Changes file owner and group.
DAC_ OVERRIDE	Discretionary Access Control (DAC), allows bypassing read, write, and execute permission checks.
FSETID	Does not clear the set-user-ID and set-group-ID bits on file modification.
FOWNER	Bypass permission checks on some operations, set ACL on files.
MKNOD	Creates a file system node.
NET_RAW	Use PACKET and RAW sockets, bind to any address.
SETGID	Arbitrarily manipulate process GIDs.
SETUID	Arbitrarily manipulate process UIDs.
SETFCAP	Setting file capabilities. Introduced in Linux 2.6.24.
SETPCAP	Has different behaviors based on CAP_SETF-CAP.
NET_BIND_-SERVICE	Allows binding a socket to privileged port (i.e., < 1024).
SYS_CHROOT	Changing the root directory for a caller process.
KILL	Send any signal to any process or process group.
AUDIT_ WRITE	Enables writing records to kernel auditing log. Available since Linux 2.6.11.

em vez de ser forçado a utilizar a que veio com o SO. Os LSMs concentram-se em fornecer as necessidades de implementação do Controlo de Acesso Obrigatório (MAC) [52] com alterações mínimas ao próprio Kernel.

Os LSMs datam da Cimeira do Kernel Linux 2001, quando a Agência Nacional de Segurança dos EUA (NSA) propôs a inclusão do Security-Enhanced Linux (SELinux) no Kernel Linux 2.5 [53]. Depois disso, o projecto LSM começou e muitos módulos foram desenvolvidos para suportar vários modelos de segurança. No caso de não se seleccionar especificamente um LSM, o LSM por defeito será o sistema de capacidades Linux [54]. Actualmente, há numerosos LSMs disponíveis, como se pode ver na Tabela 7.

Normalmente, os LSMs são utilizados para os três primeiros casos de utilização, tal como definido na Secção III. Mattetti *et al.* [55] apresentaram a estrutura do LiCShield que visa assegurar os con- tainers e cargas de trabalho do Linux utilizando construções automáticas de regras. O LiCShield traça a execução da imagem e gera perfis para LSMs (principalmente AppArmor), o que reflecte sobre as capacidades da imagem para aumentar a protecção do anfitrião, através de operações de contentores estreitos. A estrutura introduz uma sobrecarga de desempenho negligenciável. Uma desvantagem do LiCShield é que não efectua o scan de vulnerabilidade para as imagens, o que pode permitir a persistência de ameaças latentes. O LiCShield pode ajudar nos casos de utilização I e III (protegendo um recipiente das aplicações nele contidas e protegendo o hospedeiro dos recipientes).

Loukidis-Andreou *et al.* [56] apresentaram um sistema automatizado de segurança de contentores Docker que se baseia no AppArmor LSM chamado Docker-sec. O Docker-sec inclui

QUADRO 7. Módulos de segurança Linux disponíveis (LSMs), com SELinux e AppArmor sendo os LSMs mais utilizados.

Name	Description
AppArmor	Task centered policies. The profiles are created/loaded from user space.
LoadPin	Ensures that all kernel-loaded files originate from the same file system, which allows the systems with verified unchangeable file system to override restrictions on loading modules and firmware without signing the files individually.
SELinux	Fine-grained access control features. For example, instead of having read, write, and execute permissions you can specify unlink, append only, move a file and other permissions.
Smack	Simplified Mandatory Access Control Kernel (SMACK) focuses on simplicity as a primary design goal. It provides similar functionalities to SELinux.
TOMOYO	Name based MAC. It allows processes to declare the resources and behaviors required to achieve its goals and restricts the process to the declared parameters.
YAMA	Collects the un-handled security protections that are not handled by the kernel itself.

dois mecanismos primários. Primeiro, quando uma imagem é criada, Docker-sec cria um conjunto estático de regras de acesso de acordo com os parâmetros de criação da imagem. Segundo, durante o tempo de execução, o conjunto inicial é melhorado para restringir ainda mais as ameaças da imagem. De acordo com os autores, Docker-sec pode automatizar- icamente proteger contra vulnerabilidades de dia zero, ao mesmo tempo que tem um desempenho mínimo de sobrecarga. MP *et al.* [57] estudaram o efeito de diferentes LSMs e LSFs em contentores Docker. Consideraram SELinux, AppArmor, e TOMOYO e forneceram uma prova de conceito de que estes são eficazes em mit- ignição de vários riscos, tais como código malicioso e bugs em código de namespaces.

Bacis *et al.* [58] propuseram uma solução para ligar as poli- cies SELinux com imagens de contentores Docker para aumentar a sua segurança. Alegam que uma grande ameaça aos contentores provém de contentores maliciosos (ou comprometidos) que visam outros con- tiners no mesmo hospedeiro. Uma desvantagem do seu trabalho é que só podem prevenir vulnerabilidades essenciais do núcleo. Contudo, isto ainda pode ser útil nos casos de utilização I, II, e III (protecção de um recipiente contra aplicações nele contidas, protecção entre recipientes, e protecção do hospedeiro contra recipientes).

Um dos principais problemas com os LSMs é que são partilhados por todos os contentores, em que um único contentor não pode utilizar um LSM específico [59]. Para melhorar esta questão, Sun *et al.* [60] apresentaram uma nova abordagem para permitir LSMs para contentores. Apresentaram *namespaces de segurança* para permitir que os contentores tenham controlo autónomo sobre a sua segurança e permitir que cada contentor tenha o seu próprio perfil de segurança. Apresentaram também um mecanismo de encaminhamento para garantir que as decisões tomadas por um contentor específico não possam afectar o hospedeiro ou outros contentores. Para testar o seu método, desenvolveram uma abstracção de namespace para AppArmor LSM. Os seus resultados mostram que os nomes de segurança - passos protegem contra vários problemas de segurança dentro de contentores com $< 0{,}7\%$ de latência aumentada.

B. MECANISMOS DE PROTECÇÃO COM BASE EM HARDWARE

Esta secção aborda soluções para proteger contentores de um hospedeiro semi-onestado ou malicioso, bem como outros contentores. Estas soluções visam o caso de utilização IV, protegendo os contentores do hospedeiro (tal como definido na Secção III). Abordamos dois mecanismos disponíveis: A utilização de Módulos de Plataforma Virtual Fidedigna (vTPMs) e a utilização do Intel SGX como um mecanismo de suporte de plataforma fiável.

1) MÓDULOS DE PLATAFORMA VIRTUAL DE CONFIANÇA (vTPM)

Uma técnica comummente utilizada na computação de confiança é o Trusted Platform Modules (TPMs) que é hardware que se destina a ser utilizado como um processador criptográfico. Os TPMs fornecem suporte de hardware para atestar, selar dados, arranque seguro, e aceleração de algoritmos [61], [62]. Martin [63] informação detalhada pré-alojada sobre computação e TPMs de confiança. Com o advento da computação em nuvem e da virtualização, os investigadores começaram a procurar alternativas aos TPMs de hardware a fim de serem mais adequados para o hipervisor. Isto porque o hipervisor precisa de tornar o TPM disponível, ao mesmo tempo, a uma pletora de VMs [64]. Os TPMs de software, também conhecidos como TPMs Virtuais (vTPMs) foram criados para corresponder a estas necessidades [64], [65].

vTPMs têm sido estudados por muitos investigadores na virtualização do hardware [65]-[68]. Wan *et al.* [69] examinaram a computação em nuvem de confiança utilizando vTPMs e analisaram soluções existentes que utilizam vTPMs para melhor compreender a segurança de diferentes implementações. Devemos notar aqui que os vTPMs (ou TPMs de software) não são tão seguros como os TPMs de hardware, pois sofrem de numerosas vulnerabilidades e não podem suportar níveis de protecção como os TPMs de hardware [70], [71]. Hosseinzadeh *et al.* [64] apresentaram o primeiro estudo que implementa os vTPMs para contentores. As duas propostas de implementos são discutidas nas secções seguintes (Secções IV-B.1.a e IV-B.1.b). Souppaya *et al.* [31] recomitiram o seguinte padrão para atestação: começar com boot seguro/medido; fornecer uma plataforma de sistema verificada; construir uma cadeia de confiança no hardware enraizado; estender a cadeia de confiança ao carregador de boot, kernel do SO, imagens do sistema, tempo de execução do contentor, e finalmente imagens do contentor.

a: vTPM IN HOST OS KERNEL

A primeira implementação proposta por Hosseinzadeh *et al.* [64] foi a de colocar o vTPM no Kernel do SO anfitrião. Isto permite que o TPM esteja disponível para diferentes recipientes. Para atribuir um vTPM a um novo contentor, o gestor do contentor pede ao kernel do SO anfitrião que crie um novo vTPM e depois atribui-o ao novo contentor. A figura 6 mostra a arquitectura deste tipo. Contudo, o nível de protecção é ainda inferior ao da abordagem hypervisor completa [64]. Dois cenários possíveis estão disponíveis para requisitos de garantia de segurança [35]:

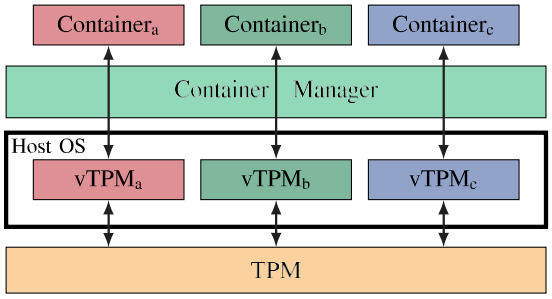


FIGURA 6. vTPM implementação no kernel.

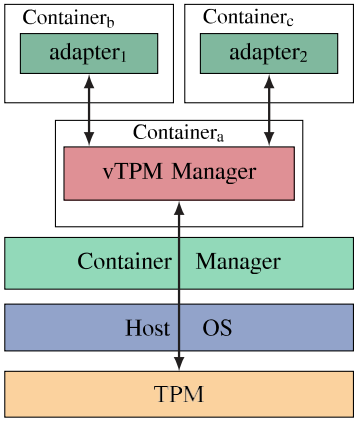


FIGURA 7. vTPM implementação num contentor dedicado.

1) Sistema Anfitrião Totalmente Fidedigno: A confiança pode ser estabelecida através da extensão da raiz da confiança usando o TPM. Assim, o SO anfitrião será considerado de confiança e qualquer vTPM gerado também pode ser confiado. Assim, os contentores podem atestar o seu estado utilizando o recurso de extensão de hash implementado no vTPM.
2) Sistema Anfitrião semi-confiado: Isto requer a confiança no vTPM que é gerado pelo anfitrião. Neste caso, o TPM fornece uma chave de endosso que será alargada dando aos vTPMs a sua própria instância e implementando protocolos para a assinatura das chaves de endosso dos vTPMs usando o TPM.

b: vTPM EM UM CONTEÚDO DEDICADO

A figura 7 mostra a arquitectura desta implementação. O vTPM permite o acesso ao TPM físico de forma diferente. Uma interface sin- gle é utilizada para a gestão do vTPM, onde cada contentor desta interface necessitará de um *adaptador de* software. O contentor que aloja o vTPM processará os pedidos recebidos de outros contentores através de um canal de comunicação específico (por exemplo, comunicação inter-processo). Isto alivia a carga de múltiplos módulos de kernel como na implementação anterior (Secção IV-B.1.a).

2) INTEL SGX

Em 2015, a Intel criou o SGX que é compatível com as suas CPUs. O Intel SGX é um conjunto de extensões que permite ao hardware de arquitectura Intel fornecer garantias de confidencialidade e integridade quando o software privilegiado subjacente

(p. ex., hipervisor, núcleo) é potencialmente malicioso [72]. O Intel SGX protege sem problemas os contentores das camadas subjacentes (por exemplo, fornecedor de nuvens, ou máquina anfitriã). Suporta enclaves seguros [73] que ajudam a proteger dados de aplicações de outras aplicações, incluindo software mais privilegiado.

Arnautov *et al.* [74] apresentaram Contentores Linux Seguros com Intel SGX (SCONE) para proteger os contentores de ataques. O design do SCONE proporciona uma base de confiança mais pequena, baixo desempenho de overhead, e suporta chamadas de sistema assíncronas e threading a nível de utilizador. Como discutimos anteriormente, tipicamente, as protecções de contentores empregam mecanismos de software para proteger os contentores contra o acesso não autorizado. Isto pode proteger os contentores de outros contentores e ataques externos. Contudo, os contentores também precisam de ser protegidos de um sistema com pernas privilegiadas, tal como o kernel ou hipervisor do SO [74]. Isto porque os adversários normalmente visam vulnerabilidades não no próprio contentor mas no sistema/administração privilegiada virtualizada [74], [75].

Hunt *et al.* [76] apresentaram Ryoan que utiliza a Intel SGX para fornecer uma caixa de areia distribuída. Utiliza enclaves para proteger instâncias de sandbox de formas maliciosas de placa de computação e permite-lhe processar dados secretos ao mesmo tempo que evita o vazamento de dados secretos. Os autores avaliaram Ryoan sobre alguns problemas, tais como filtragem de correio electrónico, análise de saúde, pro- cessing de imagem, e tradução automática. As limitações de hardware do SGX devem ser tidas em consideração, tais como não ser capaz de executar aplicações não modificadas. Recentemente, os investigadores começaram a explorar soluções para permitir que o SGX corra aplicações não modificadas [77]. A execução de aplicações não modificadas não é eficiente e introduz uma sobrecarga considerável. A modificação de aplicações, por outro lado, pode não ser uma tarefa fácil na maioria dos casos, o que limitará a utilização global do SGX entre os programadores.

Vaucher *et al.* [10] apresentaram uma outra solução para ajudar os programadores a gerir os seus contentores num fornecedor de serviços de nuvem não confiável. Propuseram a integração do SGX no interior do orquestrador Kubernetes. Isto é muito importante para os fornecedores de CaaS. Testaram a sua solução proposta num agrupamento privado utilizando os vestígios do Google Borg. Os autores descobriram que o desafio está principalmente na programação de contentores para as máquinas SGX em prioridade. Observaram que o desempenho se degrada quando a capacidade de memória dos enclaves SGX é esgotada. Os autores afirmam que nenhum orquestrador oferece actualmente apoio nativo para estatísticas de utilização de informação sobre recursos utilizados por con- tainers que utilizam SGX. Alargaram o driver SGX do Linux para recolher informações sobre o tempo de execução do SGX e direccionam-nas para o orquestrador. São necessários mais estudos sobre a praticidade do SGX e/ou tecnologias semelhantes (por exemplo, ARM Trust Zone, AMD SEV, IBM SecureBlue++).

O SGX é amplamente utilizado para proteger recipientes e outras aplicações que funcionam em hospedeiros não confiáveis. No entanto, há vários ataques que visam o SGX que podem afectar as soluções de con- tiner utilizando esta tecnologia. Exemplos dos ataques mais proeminentes contra o SGX são: Ataques de canais controlados [78], ataques furtivos baseados em páginas [79], sombreamento de ramos [80], BranchScope [81], cache de último nível

QUADRO 8. Resumo dos estudos sobre as vulnerabilidades da imagem do centro de Docker. Estes estudos centram-se nos riscos de imagem e de registo, ver Quadro 9 para tipos de risco.

Ref	Findings	Strengths	Weaknesses
[88], 2015	More than 30% of official Docker Hub repositories contain images that are highly vulnerable.	• Built open source tools to automate the process Banyan Collector and Banyan Insights, published on GitHub. • Used standardized metrics such as CVE and CVSS. • Tested 960 unique images. • Performed detailed analysis to address main packages that caused these vulnerabilities.	• CVSS vulnerability score is referred as NVD which is the National Vulnerability Database and not the score standard itself, this might create confusion for the reader. • Failed to mention mitigation to these vulnerabilities. We believe most of the them could be eliminated by image's update, which might be a standardized procedure when deploying containers. • Image scanning process is not clear. False positives for example can highly alter the results.
[89], 2017	70% of Docker Hub images have high severity issues and 54% have critical severity issues based on 1000 images scanned using an in-house tool and a commercial scanner.	• Used standardized metrics such as CVE and CVSS. • Scanned images without running them (i.e., static scanning).	• Scanning is performed using commercial scanner (Outpost). The reliability and method of this scanner was not studied. • The method for selecting the 1000 images was not clear. Are they unique images as in [88]?
[90], 2017	Studied 356,218 Docker images and found that a Docker image has on average 180 vulnerabilities, many images are outdated, and vulnerabilities usually propagate from parent to child images. Also showed that 90% of the images suffer from high severity vulnerabilities.	• Covered both community and official images. • Analyzed very large number of images. • Used open source vulnerability analyzer called CoreOS Clair. • Used standardized metrics such as CVE, CVSS. • Detailed analysis of vulnerabilities types and severity.	The study used CoreOS Clair vulnerability scanner for Docker Hub images. Clair was mainly designed to scan CoreOS images deployed Quay.io registry. Another tool was available for Docker Hub which is Docker Security Scanner. The scope of the study was limited to Docker Hub registry, and it is not justified why to use CoreOS scanner instead of Docker Scanner at least for official images. We believe this would reduce false positives since Docker Scanner was designed for Docker Hub.

ataques [82], ataques de cache L1 sem hyperthreading [83], ataques de cache L1 com hyperthreading [84], e ataques baseados em tradução lookaside buffer (TLB) [85]. Intel Transac- tion Synchronization Extensions SGX (T-SGX) [86] pro- vide melhor defesa contra o tradicional SGX [87]. No entanto, a eficácia de tais abordagens ainda não foi verificada para aplicações específicas de contentores.

V. OUTROS ASPECTOS DA SEGURANÇA DOS CONTENTORES

Anteriormente discutimos solu- ções baseadas em software e hardware. Há dois outros aspectos importantes para manter a segurança dos contentores: gestão de vulnerabilidades e directrizes padrão. Na Secção V-A, discutimos estudos sobre as vulnerabilidades de con- tiner, exploits, e ferramentas relacionadas. Não utilizámos os casos de uso propostos para este aspecto porque as vulnerabilidades não visam, em geral, um caso de uso específico. Por exemplo, algumas vulnerabilidades poderiam permitir ataques entre contentores (por exemplo, derretimento), outras poderiam ser utilizadas para permitir que os anfitriões ataquem contentores (por exemplo, ataques de canal controlado). Depois, na Secção V-B, consideramos o próximo aspecto importante na avaliação da segurança dos contentores, que trata dos esforços no sentido de metodologias de avaliação e métodos de padronização para a implantação de contentores seguros.

A. VULNERABILIDADES, EXPLORAÇÕES, E FERRAMENTAS

Alguns investigadores mostram que um grande número de imagens de contentores sofrem de vulnerabilidades de segurança. O número de vulnerabilidades está a aumentar com o tempo, o que realça um problema nos processos de remediação das vulnerabilidades dos contentores.

Por exemplo, em 2015, Gummaraju *et al.* [88] mostraram que 30+% das imagens oficiais do Docker continham vulnerabilidades de alto impacto secu- ridade. Henriksson e Falk [89] digitalizaram as imagens top 1000 Docker, em 2017, e mostraram que 70% das imagens sofriam de problemas de elevada severidade e 54% tinham problemas críticos de severidade. Isto mostra um aumento nas vulnerabilidades...
capacidades para imagens Docker em comparação com o estudo anterior em 2015 [88]. Além disso, Shu *et al.* [90] estudaram as capacidades vulneráveis que existem nas imagens Docker hub e propuseram a estrutura de Análise de Vulnerabilidade de Imagens Docker (DIVA) para automatizar o processo de análise de imagens Docker. Analisaram 356.218 imagens e concluíram que as imagens comunitárias e oficiais contêm 180 vulnerabilidades em média, e mostraram que 90% das imagens sofriam de vulnerabilidades de alta gravidade. A tabela 8 resume esses estudos juntamente com os seus pontos fortes e fracos. Acreditamos que a DIVA pode desempenhar um papel importante na descoberta automática de vulnerabilidades de imagem, caso em que poderia ajudar os utilizadores a certificarem-se de que a imagem descarregada não é vulnerável antes de a utilizarem.

Enquanto os trabalhos acima referidos se concentraram na descoberta de vulnerabilidades vulcânicas, outros investigadores mostraram o efeito real das explorações de vulnerabilidade. Martin *et al.* [91] apresentaram uma análise de vulnerabilidade para o ecossistema Docker. Os autores detalharam alguns cenários do mundo real para as vulnerabilidades e como estas poderiam ser exploradas e propuseram possíveis correcções. Lin *et al.* [92] criaram um conjunto de dados de 223 explorações que são eficazes em plataformas de contentores. Depois, avaliaram a segurança de diferentes contentores Linux utilizando um subconjunto dessas explorações. Os autores descobriram que 56,8% destes exploits

são bem sucedidos contra as configurações padrão de contentores. Da mesma forma, Kabbe [93] apresentou uma análise de segurança dos contentores Docker. O autor concentrou-se em mostrar como as capacidades vulneráveis podem ser exploradas num ambiente de produção contra explorações como DirtyCow (CVE-2016-5195), Shell-shock (CVE-2014-6271), Heartbleed (CVE-2014-0160), e Fork-bomb.

Luo *et al.* [94] identificaram possíveis canais encobertos que o alcatrão - obter Docker. Estes canais laterais podem vazar informação entre recipientes ou entre um recipiente e o seu anfitrião, atingindo uma capacidade superior a 700b/s. Mohallel *et al.* [95] avaliaram a diferença de superfície de ataque entre um servidor web instalado n u m contentor e outro instalado num SO de base. Instalaram Apache, Nginx, e MySQL em três contentores diferentes fornecidos no hub Docker, utilizando um servidor Debian para o SO de base. Avaliaram a segurança de cada servidor web usando um scanner de vulnerabilidade de rede local. Os resultados concluíram que os contentores Docker aumentaram a superfície de ataque devido a vulnerabilidades de imagem Docker. Lu e Chen [96] destacaram a importância dos testes de penetração para garantir a segurança do Docker. Analisaram primeiro um dos principais gestores de contentores (ou seja, Docker) e compararam-no com a virtualização tradicional. Depois os testes de penetração foram ainda avaliados para ataques específicos tais como DoS, fuga de contentores, e canais laterais.

Todos estes estudos mostram que as vulnerabilidades constituem uma ameaça seri ous para a segurança dos contentores. Para mitigar esta ameaça, Barlev *et al.* [97] apresentaram o Starlight, um instrumento centralizado de protecção do sistema que intercepta e analisa eventos para determinar se o administrador do sistema precisa de ser alertado. Os lados negativos do Starlight podem incluir a falta da capacidade de descobrir ameaças latentes porque não efectua escaneamento nem tem recurso de quarentena para contentores comprometidos. Estas duas questões foram abordadas por Bila *et al.* [98] que propuseram uma técnica de atenuação automática de ameaças (semelhante a [55]) utilizando uma arquitectura sem servidor baseada em OpenWhisk e Kubernetes. A principal vantagem desta ferramenta é que efectua um scan de vulnerabilidade de imagem e fornece quarentena para contentores comprometidos.

B. SOBRE NORMAS, METODOLOGIAS DE AVALIAÇÃO, E RECOMENDAÇÕES

Nesta secção, discutimos os esforços no sentido da normalização da segurança dos contentores, metodologias de avaliação e implementação de directrizes de documentação. Vários investigadores ([99], [100]) concordam que não existem estratégias de avaliação claras nem formas sistemáticas de definir, estudar, e verificar a segurança dos contentores. As conclusões de Abbott [14] são semelhantes aos dois trabalhos anteriores, uma vez que é afirmado que não existem métodos de avaliação - gies ou normas para a segurança de contentores.

Para este fim, Abbott [14] propôs uma nova metodologia para avaliar a segurança das imagens de contentores, concentrando-se nas acções a realizar pelo utilizador para avaliar a segurança da imagem de um contentor. O autor testou quatro imagens populares de contentores: nginx, redis, google/cadvisor, mbabineau/ cfn-bootstrap. Os resultados mostraram problemas de segurança em cada

das imagens testadas e forneceu formas de resolver cada questão. A referência [101] discutiu, comparou e avaliou diferentes características de segurança dos contentores. Além disso, o autor discutiu as ameaças, superfícies de ataque, e dicas de endurecimento para aumentar a segurança dos contentores. Goyal [102] apresentou uma referência com- prehensive que fornece directrizes para a segurança de Docker (utilizando Docker CE 17.06 ou posterior) através de uma lista de verificação passo a passo.

Os esforços no sentido da normalização da segurança dos contentores têm sido realizados desde finais de 2017. Em Setembro de 2017, o Instituto Nacional de Normas e Tecnologia (NIST) apresentou a primeira publicação especial sobre directrizes de segurança de aplicação de contentores [31]. Discutiu ameaças à segurança, recomendações, e contramedidas. Para implementar estas recomendações generalizadas e contramedidas - são necessárias uma (ou mais) soluções. Resumimos a norma NIST no Quadro 9, considerámos algumas outras ameaças que não foram abrangidas pela norma e acrescentámos estudos que abordaram cada tipo de ameaça. Foram realizados esforços no sentido da normalização pela Fundação Linux, o que levou ao nascimento da OCI em Junho de 2015, com o seu principal objectivo de criar padrões industriais para o formato de contentores e tempo de execução. Em Setembro de 2017, a OCI publicou a v.1.0.0 para especificações de tempo de execução do formato de imagem do contentor. Em Outubro de 2017, a NIST apresentou outra publicação especial ([35]) focando os requisitos de garantia para a implantação de contentores e forneceu soluções de implantação para as recomendações feitas na publicação especial anterior ([31]).

O NIST não abordou a segurança dos contentores a partir de um ponto de vista de investigação, uma vez que está mais orientado para utilizações industriais. Além disso, não abordaram o corpo de investigação por detrás da segurança dos contentores, o que poderia limitar os benefícios para os investigadores. Além disso, não forneceram questões de investigação abertas e orientações de investigação futura. O nosso trabalho responde às preocupações acima mencionadas e destaca outros riscos que não são abordados, tais como o ataque de Meltdown, que coloca todos os contentores em risco. Acreditamos que as normas são importantes para fornecer orientações que permitam ultrapassar as questões de segurança nos contentores. No entanto, como vimos neste inquérito, várias necessidades e questões de segurança não são abordadas ou bem compreendidas (por exemplo, namespaces específicos de contentores, LSM específicos de contentores, e Melt-down). Por conseguinte, acreditamos que trabalhos anteriores sobre normalização requerem mais investigação.

O quadro 10 mostra um resumo de estudos seleccionados aqui abordados e os casos de utilização que lhes podem ser aplicados.

VI. QUESTÕES EM ABERTO E FUTURAS ORIENTAÇÕES DE INVESTIGAÇÃO

A investigação sobre tecnologias de contentores ainda se encontra numa fase formativa e necessita de mais avaliação experimental [113]. Esta secção fornece várias direcções de investigação futuras baseadas na nossa revisão e nas recomendações de outros investigadores.

A. DERRETIMENTO E ATAQUES DE ESPECTROS

O derretimento explora a execução fora de ordem nos processadores mod- ern para extrair informação sobre o SO e outros recipientes. Os contentores são baseados no conceito de

S. Sultan *et al*: Segurança de Contentores: Questões, Desafios, e a Estrada em Frente
IEEE Access

QUADRO 9. Resumo das ameaças de contentores e estudos abordados em cada categoria.

	Risco	Contra-medida(s)	Estudos
Riscos de	Vulnerabilidades da imagem	Utilizar ferramentas e processos de gestão de vulnerabilidades.	
	Defeitos de configuração da imagem	Utilizar ferramentas e processos para cumprir as melhores práticas de configuração segura. Desactivar SSH e ferramentas de administração remota para prevenir ataques baseados em rede.	
	Malware embutido utilizam [55, 59, S8],	Monitorizar imagens para malware incorporado utilizando processos de monitorização que assinaturas de malware e detecção comportamental.	90, 93, 96, 103]
	Segredos de Cleartext incorporados	Os segredos não devem ser guardados em imagens. A maioria dos orquestradores (por exemplo, Kubernetes) inclui apoio para segredos. Os segredos só devem ser fornecidos a imagens quando necessário.	
	Utilização de imagens não	Deve manter um grupo de registos e imagens de confiança e não utilizar quaisquer imagens não fidedignas.	
Riscos do Registo	fiáveis Ligação insegura aos registos	Configurar ferramentas de desenvolvimento, tempo de execução de contentores, e orquestradores para se ligarem a registos através de canais encriptados.	
	Imagens antigas em registos [89, 88, 90,	Remover quaisquer imagens inseguras e vulneráveis do registo. Usar o controlo de versão em imagens para aceder especificamente a uma versão específica.	93, 103, 104, 105]
	Autenticação Insuficiente Acesso	O acesso aos recursos de registo deve exigir autenticação segura e autorização.	
Riscos da	Administrativo Ilimitado	As equipas de teste só têm acesso a recipientes de teste com acesso limitado a recipientes em ambiente de produção. Além disso, os orquestradores devem utilizar o modelo de acesso menos privilegiado.	
	Acesso não autorizado	O acesso às contas administrativas deve ser controlado (p. ex., a auto-contribuição multi-factor).	[106, 55, 107, 104, 105]
	Tráfego Intercontainer mal separado	Separar o tráfego da rede em rede virtual, com base no nível de sensibilidade da rede.	
	Mistura de níveis de sensibilidade da carga de trabalho	Isolar os destacamentos para grupos específicos de hospedeiros pelos seus níveis de sensibilidade. Evitar colocar uma carga de trabalho de alta sensibilidade com níveis de sensibilidade mais baixos.	
	Nó Orquestrador Trust	A plataforma de orquestração deve fornecer características para criar um ambiente seguro para os applicatic'ns que executa. Assegurar que todas as aplicações são introduzidas em segurança no agrupamento.	
Riscos do	Vulnerabilidades em tempo de execução	Monitorizar o tempo de execução do contentor por vulnerabilidades, porque o tempo de execução vulnerável expõe todas as imagens no seu interior.	
	Acesso à rede sem restrições a partir de contentores	Controlar o tráfego de saída da rede a partir de contentores.	[55, 94, 105, 3S, 93, 93, 39, 103, q]
	Configurações inseguras de tempo de funcionamento do contentor	Cumprir as normas de configuração de tempo de funcionamento do contentor (por exemplo, Center for Internet Security Docker Benchmark [102]).	
	Vulnerabilidades de Aplicação	Ferramentas de detecção de vulnerabilidade de contentores e de instruções.	
	Contentores Rogue	Dicotomia rigorosa entre ambientes de desenvolvimento, teste e produção. Cada um deve vir com os controlos necessários para o registo e controlo de acesso.	
Riscos do SO	Grande superfície de ataque	Os sistemas operativos específicos de contentores têm menos ameaças que os outros sistemas operativos porque foram especificamente concebidos para acolher contentores. As organizações que não podem utilizar sistemas operativos específicos para contentores podem seguir os Guias de Segurança Geral do Servidor [34].	
	Kernel Partilhado	Cargas de trabalho de contentores de grupo em anfitriões com base nos seus níveis de sensibilidade. Não misturar	[55, 94, 94,
	Vulnerabilidades dos Componentes	Direitos de Acesso Impróprios dos Utilizadores	carga de trabalho contentorizada e não contentorizada no mesmo
	do Sistema Operativo Anfitrião	Sistema de Arquivo Anfitrião Alteração do sistema	

	hospedeiro. Implementar ferramentas de gestão para validar a versão dos componentes fornecidos para OS. As autenticações para o sistema operativo devem ser auditadas. As anomalias de login devem ser monitorizadas. Os contentores devem funcionar com permissões mínimas do sistema de ficheiros.	108, 39, 109]	
Outro	Canal lateral e ataques de fuga de informação	Eliminar a relação entre a informação vazada e os dados secretos. [26, 27, 110, Meltdown [26] e Spectre [27] são vulnerabilidades de alto risco que afectaram Docker, 40, 94, 39, LXC, e OpenVZ. O derretimento supera os mecanismos de isolamento da memória em qualquer tecnologia de virtualização e sistemas operativos. Baseia-se na execução fora de ordem nos pré-censores de hoje. Uma contra-medida é a conversão para a plena virtualização. A Spectre tenta enganar outras aplicações para aceder a uma localização arbitrária da memória na sua memória.	96, 52, 111, 78, 79, 80, 81, 83, 84, S5, S6, S7]
	Ataques da IAGO [112]	Estreitar a interface entre os componentes de confiança e o SO não confiável.	[74]

QUADRO 10. Resumo de estudos seleccionados e casos de utilização aplicáveis.

Ano de Ref.	Resumo	Caso de uso (I)	Caso de uso (I1)	Caso	Mecanismos
[99]	2014 Estudou diferentes mecanismos de segurança e protecção de contentores. Mostrou que todos os contentores dependem de características de separação semelhantes para evitar vários ataques.				
[16]	2015 Estudou diferentes mecanismos de protecção dos contentores Docker.				
[55]	2015 Propôs um quadro para automatizar o processo de construção de regras para os LSMs. [2015 Proposta uma solução para aumentar a segurança da imagem do contentor Docker através da ligação SELinux LSM políticas para a imagem.				
[103]	2015 Estudou a segurança dos contentores Docker e os mecanismos de protecção disponíveis para aumentar a sua segurança.				
[33]	2016 Mecanismos de segurança abordados para proteger os contentores Docker dos ataques DoS.				
[57]	2016 Estudou diferentes técnicas de endurecimento para aumentar a segurança dos contentores Docker.				
[64]	2016 Estudou vTPMs para aumentar a segurança dos contentores para computação em nuvem.				
[74]	2016 Utilizou a Intel SGX para aumentar a segurança dos contentores em anfitriões não confiáveis.				
[94]	2016 Identificou canais encobertos para Docker, e estudou o efeito de configuração de capacidades na segurança de contentores Docker.				
[95]	201 6 Realizou experiências para estudar o efeito dos contentores Docker sobre a superfície de ataque do hospedeiro.				
[1]	2016 Apresentou um quadro de gestão de contentores chamado MIGRATE que ofusca o comportamento de execução de um contentor para se defender contra os ataques do canal lateral dos inquilinos vizinhos.				
[15]	2016 Estudou questões de segurança e soluções para contentores Docker utilizando casos de uso prático.				
[97]	2016 Propôs uma ferramenta chamada Starlight que analisa as chamadas do sistema de um contentor para detectar operações maliciosas.				
[101]	2016 Discutiram-se os mecanismos de segurança e protecção de contentores disponíveis.				
[2017	Apresentou uma solução para ataques de fuga de Docker com base na inspecção do estado do namespaces.				
[46]	2017 Apresentou uma solução chamada SPEAKER que se baseia em chamadas de sistema seccomp e analisadas. As chamadas de sistema desnecessárias são b1c'cked.				
[47]	2017 Apresentou uma solução para automatizar a criação de perfis seccomp para c'n um período de formação baseado na aplicação. As chamadas de sistema não vistas durante esse período serão b1c'c'cked.				
[102]	2017 Apresentou directrizes para a segurança de contentores Docker.				
[31]	2017 NIST guidelines for securing containers. [35] fornece soluções de implantação para essas directrizes.				
[27]	2018 Apresentou ataques Spectre que violam vários mecanismos de isolamento utilizados pelos contentores.				
[30]	2018 Prorrogação para [29]. Mediu a eficácia dos recipientes contra ataques de drenagem de recursos (por exemplo, drenagem de energia).				
[40]	2018 Prorrogação para [39]. Estudo de ataques de fuga de informação em cinco grandes prestadores de serviços de nuvem. Propuseram uma solução para o namespace baseado em energia para proteger contra ataques de energia.				
[43]	201 S Apresentou uma técnica chamada ContainerDrone para proteger contra ataques DoS usando CGroups para proteger o uso de CPU e módulo de kernel MemGuard para proteger o acesso à memória.				
[2015	Apresentou uma solução que incorpora o SGX com o orquestrador Kubernetes para facilitar aos programadores a colocação de contentores em fornecedores de serviços de nuvem não confiáveis.				
[26]	2015 Apresentou um ataque de derretimento que pode allc'w um recipiente malicioso para vazar informações sobre outros recipientes no mesmo hospedeiro. O ataque funcionou em Docker, LXC, e OpenVZ sem quaisquer restrições.				
[56]	2018 Apresentou uma solução chamada Docker-sec para automatizar a criação e actualização de regras AppArmor para imagens Docker.				
[60]	2018 Uma das questões com namespaces é que eles são partilhados entre diferentes contentores no mesmo anfitrião. Este trabalho apresenta uma solução para fornecer namespaces de grão fino para permitir que cada contentor utilize o seu próprio perfil de segurança.				
[57]	201 S SGX é uma ferramenta importante utilizada na protecção de recipientes contra o hospedeiro. Este estudo mostra que o SGX sofre de numerosos ataques de canal lateral e mostra que o T-SGX [86] proporciona uma melhor defesa contra tais ataques.				

LSFs	✗	✓	✓	✗ LSFs, Ferramentas de cercelos (por
LSFs	✓	✓	✓	✗ exemplo,
LSFs e LSMs LSMs LSMs	✓	✗	✓	✗ Wondershaper) LSFs,
	✓	✓	✓	✗ LSMs
LSFs e LSMs	✓	✓	✓	✗
LSFs e LSMs LSMs e LSFs vTMPs	✓	✓	✓	✗ LSFs, MemGuard
Intel SGX	✗	✗	✗	✓
LSFs e LSMs Vulnerabilidade Scanning LSFs	✗	✗	✗	✓ SGX
LSFs e LSMs Análise das chamadas ao sistema	✗	✓	✗	✗ Exploração da
	✓	✓	✓	✗ execução fora de
	✓		✓	✗ ordem
LSFs e LSMs LSFs	✓	✓	✓	✗
LSFs, análise de chamadas de sistema	✓	✓	✓	✗ LSMs
LSFs, análise de chamadas de sistema	✓	✗	✓	✗
LSFs e LSMs LSFs, LSMs, TPMs,	✗	✓	✓	✗ LSFs
varrimento da vulnerabilidade,	✓	✓	✓	✓
e SGX				
Ataques especulativos de execução	✗	✗	✗	✓ SGX

partilhando o mesmo núcleo. Recentemente, Lipp *et al*. [26] mostraram que um recipiente malicioso pode vazar informação sobre outros recipientes no mesmo hospedeiro. Isto representa uma séria ameaça para todos os fornecedores de serviços de nuvem que fornecem CaaS. Spectre [27] é outra séria ameaça aos contentores, uma vez que engana outras aplicações - para aceder a locais arbitrários na sua memória. Ambos os ataques visam o caso II (tal como definido na Secção III). Estas questões precisam de ser abordadas porque representam uma séria ameaça a todos os sistemas de contentores e podem causar grandes danos aos fornecedores de nuvens.

B. NORMAS DE SEGURANÇA DE CONTENTORES

É necessária uma investigação mais aprofundada para a padronização da implantação de contentores - ment, protocolos de comunicação, e método de avaliação - ologias. Recomendações semelhantes foram propostas por muitos investigadores [14], [100], [108]. Os modelos e normas unificados pró-posição devem ter em conta os diferentes requisitos e plataformas de aplicação, aspectos de usabilidade, praticabilidade, automatização, simplicidade, e facilidade de adopção.

C. FORENSE DIGITAL PARA CONTENTORES

A perícia forense digital é utilizada para analisar incidentes de segurança. À medida que o mundo se dirige para os microserviços que utilizam contentores, as técnicas forenses digitais devem ser capazes de analisar os incidentes de segurança relacionados com os mesmos. De acordo com Dewald *et al.* [114], a perícia forense de contentores de Docker ainda não foi abordada (a partir de 2018). O seu estudo pode motivar mais trabalho nesta área, uma vez que forneceu detalhes sobre as novas provas introduzidas pela utilização de contentores e como os actuais métodos de investigação poderiam ser alterados para contentores proporcionais.

A investigação forense digital deve responder se os métodos de investigação do aluguer de contentores são suficientes e que novos métodos são possivelmente necessários neste domínio.

D. USABILIDADE DOS INSTRUMENTOS DE AVALIAÇÃO DE VULNERABILIDADES

Vários investigadores mostraram que as imagens Docker contêm muitas vulnerabilidades de alto risco que variam entre 30% a 90% [88]-[90], indicando um problema real com tais imagens. Como muitas ferramentas de avaliação de vulnerabilidade estão disponíveis para as imagens Docker, a questão que levantamos aqui é sobre a usabilidade de tais ferramentas no ambiente de produção e se a sua utilização poderia dificultar ou não o processo de implantação. Acreditamos que é necessário mais trabalho para estudar e con- trast disponíveis scanners de vulnerabilidade de contentores no que respeita ao desempenho, usabilidade, automatização, e integração com as actuais ferramentas de implantação e orquestração. Isto seria uma tremenda ajuda pró-vídeo para os programadores e empresas que precisam de conhecer os riscos que enfrentam antes, durante e depois da implantação de uma imagem específica. Também recomendamos um exame mais aprofundado da viabilidade de mitigar automaticamente as vulnerabilidades dos contentores descobertos, aplicando automaticamente as soluções necessárias.

De Lucia [115] apresentou uma revisão, em Maio de 2017, sobre o isolamento secu- ritário de VMs, contentores, e unikernels. O estudo mostra que as VMs e os unikernels fornecem um bom isolamento com- pared aos contentores. Contudo, os VMs são ineficientes porque são grandes, os unikernels não são óptimos porque carecem de níveis de privilégios (que ajudam a separar o código de kernel do código de aplicação), e os contentores não fornecem um isolamento tão bom como os VMs ou unikernels. O estudo conclui que não existe uma solução óptima que tenha sido desenvolvida até à data. Afirma ainda que uma solução óptima deve combinar as boas características das VMs, contentores e unikernels.

Em Dezembro de 2017, os contentores transparentes fundiram-se com os contentores Kata que a Intel afirma terem o mesmo nível de segurança que os VM [28]. É necessária uma análise mais aprofundada sobre VMs, contentores, unikernels, e híbridos. Como sugerido por De Lucia [115], são necessários estudos sobre a viabilidade, benefícios, e segurança da combinação híbrida. Isto ajudaria a identificar a melhor tecnologia para diferentes cenários.

F. SEGURANÇA E PRIVACIDADE DOS CONTENTORES PARA CANDIDATURAS DE LOTAS

O recente relatório da Symantec (publicado em Março de 2018) mostra um aumento global de 600% nos ataques que visam a Internet sem fios para 2017. Recentemente, os contentores são utilizados em aplicações de LPC porque são mais leves que as VM [3]-[7]. Celesti *et al.* [5] sublinharam a importância dos contentores para melhorar o serviço de Internet sem fios (IoT) provi- sioning. Os autores estudaram questões de desempenho e possíveis vantagens quando os contentores são utilizados para serviços de nuvens de IOT. Os resultados mostraram que os contentores introduziam uma sobrecarga aceitável de perfor- mance em cenários reais. Morabito *et al.* [6] mostraram ainda que as questões de segurança surgem quando se utilizam contentores para a Internet sem fios devido à partilha de recursos.

Morabito *et al.* [116] analisaram contentores e unikernels escalabilidade, privacidade, segurança, e desempenho para aplicações de IoT numéricas. Os autores sublinharam algumas questões abertas para a integração de contentores e unikernels com aplicações IoT, tais como orquestrações e monitorização, segurança e privacidade, normas e regulamentos, enquadramento de gestão - obras e aplicações, portabilidade, armazenamento de dados, e prontidão e perspectivas da indústria das telecomunicações. Para segurança e privacidade, os autores destacaram o desafio da certificação de aplicações e a necessidade de mecanismos de validação para identificar imagens adulteradas. Morabito [117] encorajou o desenvolvimento de mecanismos de segurança que estão relacionados com aplicações de IoT, tais como [118]. Haritha e Lavanya [119] apresentaram um inquérito sobre questões de segurança da IdC e forneceram um conjunto de questões e desafios em aberto. Assim, voltamos a salientar a importância de estudar a segurança dos contentores, privacidade, e técnicas de normalização - nologias para diferentes aplicações da IdC, tais como redes inteligentes, veículos inteligentes, realidade aumentada, redes de sensores inteligentes, E-Health, e virtualização da função de rede (NFV).

G. CADEIA DE BLOQUEIO PARA VERIFICAÇÃO DE CONTENTORES

Como vimos anteriormente, muitas das questões de segurança em contentores surgem da utilização de imagens não verificadas. Por exemplo, a instalação por defeito do Docker não verifica a autenticidade da imagem [104]. Para este fim, o Notário pode ser utilizado para verificar a autenticidade das imagens Docker, no entanto, é um solução centralizada. Uma solução melhor é utilizar uma verificação descentralizada que poderia utilizar uma cadeia de bloqueios. Xu *et al.* [104] propuseram uma solu- ção para verificação de imagens Docker utilizando uma cadeia de bloqueio chamada Docker Trust (DDT) descentralizada. Acreditamos que é necessário um investimento adicional para a aplicabilidade e eficácia da certificação descentralizada para imagens de contentores.

H. CONTENTORES ESPECÍFICOS LSMs

Uma questão com os LSMs é que são partilhados por todos os con- tiners, nos quais um único contentor não pode utilizar um LSM específico [59]. Como discutimos anteriormente na Secção IV-A.2, os LSM desempenham um papel fulcral no fornecimento de segurança para os sistemas Linux em geral. No entanto, a partilha de LSMs entre diferentes contentores pode ser paralisante, uma vez que diferentes contentores podem ter requisitos de protecção diferentes. Por conseguinte, recomendamos mais investigação no sentido de LSMs específicos de contentores para aumentar e facilitar a segurança dos contentores. O trabalho de Johansen e Schaufer [120] centra-se na disponibilização de LSMs a contentores e como mencionam "Até à data, o acesso dos contentores ao LSM tem sido limitado, mas tem havido trabalho para alterar a situação". A forma de o conseguir, segundo os autores, é através da virtualização dos LSM. Recentemente, Sun *et al.* [60] apresentaram um estudo onde forneceram LSMs específicos para contentores (para AppArmor). É necessária mais investigação sobre este tópico, abordando especialmente a criação de perfis para imagens de contentores e a sua aplicabilidade utilizando as actuais ferramentas de orquestração.

VII. CONCLUSÕES

Os contentores são importantes para o futuro da computação em nuvem. Os microsserviços e contentores estão intimamente relacionados, onde os con- tiners são considerados a forma padronizada para a implementação de microsserviços. Os contentores são importantes para o campo emergente das malhas de serviços que dependem também dos micro-serviços. No entanto, uma das principais barreiras de adopção da implantação generalizada de contentores são as questões de segurança que enfrentam. Tanto quanto sabemos, não existem inquéritos exaustivos sobre segurança de contentores; o nosso trabalho tentou preencher esta lacuna, analisando a literatura e identificando as principais ameaças que se devem à imagem, registo, orquestração, contentor, canais laterais, e riscos do sistema operativo anfitrião. Assim, propusemos quatro casos de utilização a nível de contentor-hospedeiro para elucidar como as soluções actuais podem ser utilizadas para aumentar a segurança do contentor. Os casos de utilização são:

(I) protecção de um recipiente contra aplicações no seu interior, (II) protecção entre recipientes, (III) protecção contra recipientes, e

(IV) protegendo os recipientes de um hospedeiro malicioso ou semi-selvagem. As soluções disponíveis para os quatro casos de utilização podem ser

(i) soluções de software tais como namespaces Linux, CGroups, capacidades, seccomp, e LSMs, ou (ii) soluções de hardware tais como a utilização de vTPMs e a utilização de suporte de plataforma de confiança (por exemplo, Intel SGX).

Identificámos ainda alguns desafios abertos e orientações de investigação para contentores. As direcções estão centradas na importância de uma melhor gestão da vulnerabilidade, investigação digital, alternativas de contentores, e LSMs específicos para contentores. Esperamos que o nosso trabalho possa lançar luz sobre o poder e limitações das tecnologias de contentores, estimular a interacção entre os profissionais, e gerar mais investigação na área.

AGRADECIMENTOS

Os autores gostariam de agradecer aos revisores anónimos pelos seus comentários úteis e construtivos que muito contribuíram para melhorar a versão final do artigo.

REFERÊNCIAS

[1] M. Httermann, *DevOps for Developers*. Nova Iorque, NY, EUA: Apress, 2012.

[2] C. Tozzi. (Jan. 2017). *O que é que os contentores têm a ver com DevOps, afinal?* [Online]. Disponível: https://containerjournal.com/2017/ 01/11/containers-devops-de qualquer forma/

[3] K. Kaur, T. Dhand, N. Kumar, e S. Zeadally, "Container-as-a-service at the edge": Trade-off between energy efficiency and service availability at fog nano data centers", *IEEE Wireless Commun.*, vol. 24, no. 3,
pp. 48-56, Jun. 2017.

[4] H. Khazaei, H. Bannazadeh, e A. Leon-Garcia, "SAVI-IoT": Uma plataforma IoT auto-gerida de contentores',' em *Proc. IEEE 5th Int. Conf. Future Internet Things Things Cloud (FiCloud)*, Ago. 2017, pp. 227-234.

[5] A. Celesti, D. Mulfari, M. Fazio, M. Villari, e A. Puliafito, "Exploring container virtualization in IoT clouds," em *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, Maio de 2016, pp. 1-6.

[6] R. Morabito, I. Farris, A. Iera, e T. Taleb, "Evaluating performance of containerized IoT services for clustered devices at the network edge", *IEEE Internet Things J.*, vol. 4, no. 4, pp. 1019-1030, Ago. 2017.

[7] R. Morabito, R. Petrolo, V. Loscrì, N. Mitton, G. Ruggeri, e
A. Molinaro, "Lightweight virtualization as enabling technology for future smart cars," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Man- age. (IM)*, Maio 2017, pp. 1238-1245.

[8] P. Jamshidi, C. Pahl, N. C. Mendonça, J. Lewis, e S. Tilkov, "Microser-vices": The journey so far and challenges ahead", *IEEE Softw.*, vol. 35, no. 3, pp. 24-35, Maio/Junho. 2018.

[9] J. Soldani, D. A. Tamburri, e W.-J. Van Den Heuvel, "As dores e os ganhos dos microserviços": A systematic grey literature review," *J. Syst. Softw.*, vol. 146, pp. 215-232, Dez. 2018.

[10] S. Vaucher, R. Pires, P. Felber, M. Pasin, V. Schiavoni, e C. Fet- zer, "SGX-Aware orquestração de contentores para aglomerados heterogéneos", em *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2018,
pp. 730–741.

[11] R. Buyya *et al.* (2017). "'Um manifesto para a futura geração de nuvens com- posta: Orientações de investigação para a próxima década". [Online]. Disponível: https://arxiv.org/abs/1711.09123

[12] 451Pesquisa. (2017). *451 Pesquisa Diz Recipiente de Aplicação Market para atingir 2,7 mil milhões até 2020*. Acedido: 22 de Outubro de 2017 [Online]. Disponível: https://www.enterpriseai.news/2017/01/10/451-research-says-application-container-market-reach-2-7-billion-2020/

[13] D. Walsh. (2014). *Os Contentores de Docker são realmente seguros?* Acedido: 27 de Dezembro de 2017. [Online]. Disponível: https://opensource.com/business/ 14/7/docker-security-selinux

[14] B. M. Abbott, "Uma metodologia de avaliação de segurança para imagens de contentores".
Tese de Mestrado, Brigham Young Univ., Provo, UT, EUA, 2017.

[15] T. Combe, A. Martin, e R. Di Pietro, "Para atracar ou não para atracar": A security perspective", *IEEE Cloud Comput.*, vol. 3, não. 5, pp. 54-62, Set./Out. 2016.

[16] A. Bettini. (2015). Exploração de vulnerabilidades em envi- ramentos de contentores Docker. FlawCheck, Black Hat Europe, Países Baixos. [Online]. Disponível em: https://www.blackhat.com/docs/eu-15/materials/eu-15-Bettini-Vulnerabilidade-Exploração-Em-Docker-Container-Enverments.pdf

[17] M. Pearce, S. Zeadally, e R. Hunt, "Virtualização": Issues, security threats, and solutions," *ACM Computing. Surv.*, vol. 45, no. 2, Fev. 2013, Art. no. 17.

[18] G. Pék, L. Buttyán, e B. Bencsáth, "Um levantamento das questões de segurança na virtualização do hardware", *ACM Informática. Surv.*, vol. 45, no. 3, Jun. 2013, Art. no. 40.
[19] F. Zhang, G. Liu, X. Fu, e R. Yahyapour, "Um inquérito sobre migração de máquinas virtuais": Desafios, técnicas, e questões abertas", *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 1206-1243, 2nd Quart., 2018.
[20] W. Gentzsch e B. Yenier, "Novos recipientes de software para engenharia e simulações científicas na nuvem", *Int. J. Grid High Perform. Informática. (IJGHPC)*, vol. 8, no. 1, pp. 38-49, Jan. 2016.
[21] M. G. Xavier, M. V. Neves, F. D. Rossi, T. C. Ferreto, T. Lange, e C. A. De Rose, "Avaliação do desempenho da tualização de virús baseada em contentores para ambientes informáticos de alto desempenho", em *Proc. 21 Euromicro Int. Conf. Parallel, Distrib., Netw.-Based Process.*, Fev./Mar. 2013, pp. 233-240.
[22] M. Stuart. (2015). *Arquitecturas de Contentores em Evolução*. Acedido: 13 de Novembro de 2017. [Online]. Disponível: https://wikibon.com/evolving- contentor-arquitecturas/
[23] R. Peinl, F. Holzschuher, e F. Pfitzer, "Docker cluster management for the cloud-survey results and own solution", *J. Grid Comput.*, vol. 14, no. 2, pp. 265-282, Jun. 2016.
[24] N. Dragoni *et al.*, "Microservices": Yesterday, today, and tomorrow," em *Present and Ulterior Software Engineering*. Cham, Suíça: Springer, 2017, pp. 195-216.
[25] B. Dourado. (2016). *3 Razões Porque Deve Sempre Executar Aplicações Microservices em Contentores*. Acedido: 11 de Novembro, 2017. [Online]. Disponível: https:// techbeacon.com/app-dev-testing/3-reasons-why-you-should-alway-runs-run
-microservices-apps-containers
[26] M. Lipp *et al.*, "Meltdown": Reading Kernel memory from user space," in *Proc. 27th USENIX Secur. Symp. USENIX Secur.*, 2018, pp. 973-990.
[27] P. Kocher *et al.* (2018). "Ataques de espectro": Exploração de execução especulativa". [Online]. Disponível em: https://arxiv.org/abs/1801.01203
[28] Informações. (2018). *Contentores Intel Clear Containers: Agora parte dos contentores Kata*. Acedido: Mar. 20, 2018. [Online]. Disponível: https://clearlinux.org/ contentores
[29] L. Catuogno, C. Galdi, e N. Pasquino, "Measuring the effectiveness of containerization to prevent power drining attacks", em *Proc. IEEE Int. Workshop de Medição. Netw. (MN)*, Set. 2017, pp. 1-6.
[30] L. Catuogno, C. Galdi, e N. Pasquino, "Uma metodologia eficaz para medir a utilização de recursos de software", *IEEE Trans. Instrum. Meas.*, vol. 67, no. 10, pp. 2487-2494, Out. 2018.
[31] M. Souppaya, J. Morello, e K. Scarfone, *Application Container Security Guide*, vol. 800. Gaithersburg, MD, USA: NIST, 2017, p. 190.
[32] P. Bogaerts. (Jan. 2017). *Arp Spoofing Docker Containers*. [Online]. Disponível: https://dockersec.blogspot.com/2017/01/arp-spoofing-docker- contentores_26.html
[33] J. Chelladhurai, P. R. Chelliah, e S. A. Kumar, "Securing docker containers from denial of service (DOS) attacks," em *Proc. IEEE Int. Conf. Serviços de Informática. (SCC)*, Jun./Jul. 2016, pp. 856-859.
[34] K. Scarfone, W. Jansen, e M. Tracy, *Guide to General Server Security*, vol. 800. Gaithersburg, MD, USA: NIST, 2008, p. 123.
[35] R. Chandramouli, "Requisitos de garantia de segurança para a instalação de contentores de aplicação de linux", Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. 8176, 2017.
[36] D. S. Equipa. (2017). *Segurança do Docker*. Acedido: 13 de Novembro de 2017. [Online].
Disponível: https://docs.docker.com/engine/security/security/
[37] Man7.org. (2017). *Namespaces-Overview of Linux Namespaces*. Acedido: 13 de Novembro de 2017. [Online]. Disponível: http://man7.org/linux/ man-pages/man7/namespaces.7.html
[38] Z. Jian e L. Chen, "Um método de defesa contra ataque de fuga de estivadores", em *Proc. Int. Conf. Cryptogr., Secur. Privacy*, Mar. 2017, pp. 142–146.
[39] X. Gao, Z. Gu, M. Kayaalp, D. Pendarakis, e H. Wang, "Container-Vazamentos": Ameaças de segurança emergentes de fugas de informação em nuvens de contentores', em *Proc. 47th Annu. IEEE/IFIP Int. Conf. Syst. de confiança. Netw. (DSN)*, Jun. 2017, pp. 237-248.
[40] X. Gao, B. Steenkamer, Z. Gu, M. Kayaalp, D. Pendarakis, e H. Wang, "A study on the security implications of information leak- ages in container clouds", *IEEE Trans. Dependable Secure Comput.*, a ser publicado.
[41] D. Yu, Y. Jin, Y. Zhang, e X. Zheng, "Um inquérito sobre questões de segurança na comunicação de serviços de aplicações de nevoeiro com recurso a Microservices", em *Concurrency and Computation: Prática e Experiência*. Hoboken, NJ, EUA: Wiley, 2018, p. e4436. doi: 10.1002/cpe.4436.

implementing multi-resource server on multi-core linux plat- form," in *Proc. IEEE 18ª Conf. Emerg. Technol. Factory Automat. (ETFA)*, Set. 2013, pp. 1-4.

[43] J. Chen, Z. Feng, J.-Y. Wen, B. Liu, e L. Sha. (2018). "Uma estrutura de controlo DoS baseada em contentores resistentes ao ataque para sistemas de UAV em tempo real". [Online]. Disponível: https://arxiv.org/abs/1812.02834

[44] Man7.org. (2017). *Capacidades Linux*. Acedido: 5 de Dezembro, 2017. [Online].
Disponível: http://man7.org/linux/man-pages/man7/capabilities.7.html

[45] RedHat. (Maio de 2018). *Capítulo 8. Capacidades Linux e Seccomp*. Acedido: 16 de Maio de 2018. [Online]. Disponível: https://access.redhat.com/ documentation/en-us/red_hat_enterprise_linux_atomic_host/7/html /container_guide_security_guide/linux_capabilities_and_seccomp

[46] L. Lei *et al.*, "SPEAKER: Split-phase execution of application containers," in *Proc. Int. Conf. Detection Intrusions Malware, Vulnerability Assessment (Avaliação da vulnerabilidade)*. Cham, Suíça: Springer, Jun. 2017, pp. 230-251.

[47] Z. Wan, D. Lo, X. Xia, L. Cai, e S. Li, "Mining sandboxes for linux containers," em *Proc. IEEE Int. Conf. Softw. Test., Verification Validation (ICST)*, Mar. 2017, pp. 92-102.

[48] D. Merkel, "Docker": Contentores de linux leves para um desenvolvimento e implantação consistentes',' *Linux J.*, vol. 2014, no. 239, Mar. 2014, Art. no. 2. [Online]. Disponível em: https://dl.acm.org/citation.cfm?id=2600241

[49] Y. Al-Dhuraibi, F. Paraiso, N. Djarallah, e P. Merle, "Autonomic vertical elasticidade dos contentores portuários com ELASTICDOCKER," in *Proc. IEEE 10ª Int.Conf. Cloud Comput. (CLOUD)*, Jun. 2017, pp. 472-479.

[50] Docker. (2017). *Docker Default Capabilities*. Acedido: 1 de Dezembro, 2017. [Online]. Disponível: https://github.com/moby/moby/blob/master/oci/ defaults.go

[51] J. Morris, S. Smalley, e G. Kroah-Hartman, "módulos de segurança Linux": Suporte de segurança geral para o núcleo do linux," em *Proc. USENIX Secur. Symp.*, Ago. 2002, pp. 17-31.

[52] H. Lindqvist, "Controlo de acesso obrigatório", tese de mestrado, Dept. Informática. Sci., Umea Univ., Umeå, Suécia, 2006, vol. 87.

[53] S. Smalley, T. Fraser, e C. Vance, "módulos de segurança Linux": Ganchos de segurança geral para Linux," 2001. [Online]. Disponível: https://www.kernel.org/doc/htmldocs/lsm/index.html

[54] Kernel.org. (2017). *Utilização do Módulo de Segurança Linux*. Acedido: 7 de Dezembro, 2017. [Online]. Disponível: https://www.kernel.org/doc/html/ v4.13/admin-guide/LSM/index.html

[55] M. Mattetti, A. Shulman-Peleg, Y. Allouche, A. Corradi, S. Dolev, e L. Foschini, "Securing the infrastructure and the workloads of linux con- tainers", em *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)*, Set. 2015,
pp. 559–567.

[56] F. Loukidis-Andreou, I. Giannakopoulos, K. Doka, e N. Koziris, "Docker-Sec: A full automated container security enhancement mechanism," em *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2018, pp. 1561-1564.

[57] A. R. MP, A. Kumar, S. J. Pai, e A. Gopal, "Reforçar a segurança do estivador utilizando técnicas de endurecimento do linux", em *Proc. 2ª Int. Conf. Theor. Informática. Comun. Technol. (iCATccT)*, Jul. 2016, pp. 94-99.

[58] E. Bacis, S. Mutti, S. Capelli, e S. Paraboschi, "Dockerpolicymodules": Controlo de acesso obrigatório para contentores portuários," em *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)*, Set. 2015, pp. 749-750.

[59] J. Johansen. (2018). *Tornar os Módulos de Segurança Linux Disponíveis para os utilizadores da Con- tainers*. Acedido: Mar. 20, 2018. [Online]. Disponível: https://archive. fosdem.org/2018/schedule/event/containers_lsm/

[60] Y. Sun, D. Safford, M. Zohar, D. Pendarakis, Z. Gu, e T. Jaeger, "Security namespace: making linux security frameworks available to containers," in *Proc. 27th USENIX Secur. Symp. USENIX Secur.*, 2018,
pp. 1423–1439.

[61] T. Morris, "módulo de plataforma de confiança", em *Enciclopédia de Criptografia e Segurança*. Boston, MA, USA: Springer, 2011, pp. 1332-1335. doi: 10.1007/978-1-4419-5906-5_796.

[62] E. W. Felten, "Compreender a computação de confiança": *IEEE Security Privacy*, vol. 1, não. 3, pp. 60-62, Maio/Junho. 2003.

[63] A. Martin, "A introdução de dez páginas à computação de confiança", Informática. Lab., Oxford Univ., 2008. [Online]. Disponível em: https://www.cs.ox.ac.uk/files/1873/RR-08-11.PDF

[64] S. Hosseinzadeh e S. Laurén, e V. Leppänen, "Security in container-based virtualization through vTPM," in *Proc. IEEE/ACM 9th Int. Conf. Utility Cloud Comput. (UCC)*, Dez. 2016, pp. 214-219.

[65] R. Perez *et al.*, "vTPM: Virtualizando o módulo de plataforma de confiança," em *Proc. 15ª Conf. USENIX Secur. Symp.*, 2006, pp. 305-320.

[66] P. England e J. Loeser, "Para-virtualized TPM sharing," in *Proc. 1st Int. Conf. Trusted Comput. Conf. Inf. Technol., Trusted Comput.- Challenges Appl.*, Mar. 2008, pp. 119-132.

[67] B. Danev, R. J. Masti, G. O. Karame, e S. Capkun, "Enabling secure VM-vTPM migration in private clouds," in *Proc. 27th Annu. Comput. Secur. Appl. Conf.*, Dez. 2011, pp. 187-196.

[68] P. Fan, B. Zhao, Y. Shi, Z. Chen, e M. Ni, "An improved vTPM-VM live migration protocol", *Wuhan Univ. J. Natural Sci.*, vol. 20, no. 6, pp. 512-520, Dez. 2015.

[69] X. Wan, Z. Xiao, e Y. Ren, "Building trust into cloud computing using virtualization of TPM," in *Proc. 4th Int. Conf. Multimedia Inf. Netw. Secur.*, Nov. 2012, pp. 59-63.

[70] F. Stumpf e C. Eckert, "Enhancing trusted platform modules with hardware-based virtualization techniques," in *Proc. 2nd Int. Conf. Emerg. Secur. Inf., Syst.Technol.*, Ago. 2008, pp. 1-9.

[71] TrustedComputingGroup. (2016). *Trusted Platform Module (TPM) 2.0: Uma breve introdução*. Acedido: 9 de Dezembro, 2017. [Online]. Disponível: https://www.trustedcomputinggroup.org/wp-content/uploads/ TPM-2.0-A-Brief-Introduction.pdf

[72] V. Costan e S. Devadas, "Intel SGX explained", Cryptol. ePrint Arch., Tech. Rep. 2016/086, 2016. [Online]. Disponível em: https://eprint.iacr.org/2016/086

[73] M. Hoekstra, R. Lal, P. Pappachan, V. Phegade, e J. Del Cuvillo, "Usando instruções inovadoras para criar soluções de software de confiança", em *Proc. 2ª Int. Workshop Hardw. Architectural Support Secur. Privacy*, Jun. 2013, Art. no. 11.

[74] S. Arnautov *et al.*, "SCONE: Contentores de linux seguros com informação SGX," em *Proc. OSDI*, 2016, pp. 689-703.

[75] K. Zetter, "NSA hacker chief explains how to keep him out of your system", Wired, 2016. [Online]. Disponível: https:/ / w w w . w i r e d . c o m / 2 0 1 6 / 0 1 / n s a - h a c k e r - c h i e f - e x p l a i n s - h o w - t o - keep-him-out-your-system/

[76] T. Hunt, Z. Zhu, Y. Xu, S. Peter, e E. Witchel, "Ryoan": A distributed sandbox for untrusted computation on secret data," em *Proc. OSDI*, 2016, pp. 533–549.

[77] C.-C. Tsai, D. E. Porter, e M. Vij, "Graphene-SGX: Um prático SO de biblioteca para aplicações não modificadas em SGX," em *Proc. USENIX Annu. Tech. Conf. USENIX ATC*, 2017, pp. 645-658.

[78] Y. Xu, W. Cui, e M. Peinado, "ataques de canal controlado": Determinístic side channels for untrusted operating systems',' em *Proc. IEEE Symp. Secur. Privacy*, Maio de 2015, pp. 640-656.

[79] J. Van Bulck, N. Weichbrodt, R. Kapitza, F. Piessens, e R. Strackx, "Contando os seus segredos sem falhas de página": Ataques furtivos baseados em tabelas de páginas sobre execução enclave', em *Proc. 26th USENIX Secur. Symp. USENIX Secur.*, 2017, pp. 1041-1056.

[80] S. Lee, M.-W. Shih, P. Gera, T. Kim, H. Kim, e M. Peinado, "Infer- ring fine-grained control flow inside SGX enclaves with branch shad- due," in *Proc. 26th USENIX Secur. Symp., USENIX Secur.*, 2017, pp. 557–574.

[81] D. Evtyushkin *et al.*, "BranchScope": Um novo ataque de canal lateral sobre o preditor de ramos direccionais',' in *Proc. 23rd Int. Conf. Programa de Apoio Arquitectónico. Lang. Operating Syst.*, Mar. 2018, pp. 693–707.

[82] M. Schwarz, S. Weiser, D. Gruss, C. Maurice, e S. Mangard, "extensão da guarda Malware": Using sgx to hide cache attacks," em *Proc. Int. Conf. Detection Intrusions Malware, Vulnerability Assessment (Avaliação da vulnerabilidade)*. Cham, Suíça: Springer, Jun. 2017, pp. 3-24.

[83] M. Hähnel, W. Cui, e M. Peinado, "Canais laterais de alta resolução para sistemas operacionais não confiáveis", em *Proc. USENIX Annu. Techn. Conf. (USENIX ATC)*, 2017, pp. 299-312.

[84] F. Brasser, U. Müller, A. Dmitrienko, K. Kostiainen, S. Capkun, e A.-R. Sadeghi. (2017). "Software grande exposição: os ataques de cache SGX são práticos," [Online]. Disponível: https://arxiv.org/abs/1702.07521

[85] W. Wang *et al.*, "Caldeirão com fugas na terra escura": Understanding mem- ory side-channel hazards in SGX',' em *Proc. ACM SIGSAC Conf. Comput. Comun. Secur.*, Oct./Nov. 2017, pp. 2421-2434.

[86] M.-W. Shih, S. Lee, T. Kim, e M. Peinado, "T-SGX: Erradicando ataques de canais controlados contra programas de enclave," em *Proc. 2017 Annu. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, San Diego, CA, EUA, 2017, pp. 1-16.

[87] M.-W. Shih. (Jun. 2018).*TSX-Based Defenses Against SGX Side-Channel Attacks*. Acedido: 16 de Janeiro de 2019. [Online]. Disponível: https://gts3.org/2018/tsgx-defense.html

oficiais no Docker Hub contêm vulnerabilidades de segurança de alta prioridade", BanyanOps, 2015. [Online]. Disponível: https://web.archive.org/web/20150921104849/http://www.banyanops.com/pdf/BanyanOps-AnalyzingDockerHub-WhitePaper.pdf

[89] O. Henriksson e M. Falk, "Análise da vulnerabilidade estática das imagens das docas", tese de mestrado, Blekinge Inst. Technol., Karlskrona, Suécia, 2017.

[90] R. Shu, X. Gu, e W. Enck, "A study of security vulnerabilities on docker hub," in *Proc. 7th ACM Conf. Data Appl. Secur. Privacy*, Mar. 2017, pp. 269-280.

[91] A. Martin, S. Raponi, T. Combe, e R. Di Pietro, "Docker ecosystem-Vulnerability analysis," *Computador. Commun.*, vol. 122, pp. 30-43, Jun. 2018. [Online]. Disponível: http://www.sciencedirect.com/science/article/pii/S0140366417300956

[92] X. Lin, L. Lei, Y. Wang, J. Jing, K. Sun, e Q. Zhou, "Um estudo de medição sobre a segurança de contentores de linux": Ataques e contramedidas", em *Proc. 34th Annu. Informática. Secur. Conf. Aplic.*, Dez. 2018, pp. 418-429.

[93] J.-A. Kabbe, "Análise de segurança de contentores Docker num ambiente de produção", tese M.S., NTNU, Noruega, 2017.

[94] Y. Luo, W. Luo, X. Sun, Q. Shen, A. Ruan, e Z. Wu, "Sussurros entre os recipientes: High-capacity covert channel attacks in docker,' em *Proc. IEEE Trustcom/BigDataSE/ISPA*, Ago. 2016, pp. 630-637.

[95] A. A. Mohallel, J. M. Bass, e A. Dehghantaha, "Experimentando com o estivador": Linux container and base os attack surfaces," em *Proc. Int. Conf. Inf. Soc. (i-Society)*, Oct. 2016, pp. 17-21.

[96] T. Lu e J. Chen, "Investigação de tecnologia de testes de penetração em ambiente portuário", em *Proc. 5ª Int. Conf. Mechatronics, Mater., Chem. Com- put. Eng. (ICMMCCE)*, Set. 2017, pp. 1354-1359.

[97] S. Barlev, Z. Basil, S. Kohanim, R. Peleg, S. Regev, e A. Shulman-Peleg, "Seguro mas utilizável": Protecção de servidores e contentores de linux," *IBM J. Res. Develop.*, vol. 60, no. 4, pp. 1-12, Jul./Aug. 2016.

[98] N. Bila, P. Dettori, A. Kanso, Y. Watanabe, e A. Youssef, "Leveraging the serverless architecture for securing linux containers," em *Proc. IEEE 37th Int. Conf. Distrib. Informática. Syst. Workshops (ICDCSW)*, Jun. 2017, pp. 401–404.

[99] E. Reshetova, J. Karhunen, T. Nyman, e N. Asokan, "Segurança de tecnologias de virtualização a nível de SO," em *Proc. Conf. Nórdica. Secure IT Syst.* Cham, Suíça: Springer, 2014, pp. 77-93.

[100] L. Catuogno e C. Galdi, "On the evaluation of security properties of containerized systems," in *Proc. 15 Int. Conf. Ubiquitous Com- put. Commun. Int. Symp. Cyberspace Secur. (IUCC-CSS)*, Dez. 2016, pp. 69–76.

[101] A. Grattafiori, "Understanding and hardening linux containers," NCC Group, Manchester, U.K., Livro Branco Versão 1.1, 2016. [Online]. Disponível: https://www.nccgroup.trust/globalassets/our-research/us/whitepapers/2016/april/ncc_group_understanding_hardening_linux_containers-1-1.pdf

[102] P. Goyal. (2017). *CIS Docker Community Edition Benchmark v1.1.0*. Acedido: 10 de Dezembro de 2017. [Online]. Disponível: https://www.cisecurity.org/ benchmark/docker/

[103] T. Bui. (2015). "'Análise da segurança dos estivadores". [Online]. Disponível: https://arxiv.org/abs/1501.02967

[104] Q. Xu, C. Jin, M. F. B. M. Rasid, B. Veeravalli, e K. M. M. M. Aung, "Blockchain-based decentralized content trust for docker images", *Mul- timedia Tools Appl.*, vol. 77, no. 14, pp. 18223-18248, Jun. 2018.

[105] A. Brinckman *et al.*, "A comparative evaluation of blockchain systems for application sharing using containers," in *Proc. IEEE 13th Int. Conf. e-Science (e-Science)*, Out. 2017, pp. 490-497.

[106] D. Sæther, "Segurança no enxame de Docker": Serviço de orquestração para sistemas de software des- tributed", tese de mestrado, Dept. Inform., Univ. Bergen, Bergen, Noruega, 2018.

[107] N. Paladi, A. Michalas, e H.-V. Dang, "Towards secure cloud orchestration for multicloud deployments," in *Proc. 5th Workshop CrossCloud Infrastruct. Plataformas*, Abr. 2018, Art. nº. 4.

[108] M. A. Babar e B. Ramsey, "Understanding container isolation mechanisms for building security-sensitive private cloud", CREST, Univ. Ade- laide, Adelaide, SA, Austrália, Tech. Rep., 2017.

[109] Y. Li, B. Dolan-Gavitt, S. Weber, e J. Cappos, "Lock-in-Pop: Proteger os núcleos privilegiados do sistema operacional mantendo-se no caminho batido", em *Proc. USENIX Annu. Tech. Conf. (USENIXATC)*, 2017, pp. 1-13.

[110] R. Sprabery, K. Evchenko, A. Raj, R. B. Bobba, S. Mohan, e R. Campbell, "Programação, isolamento e atribuição de cache": A side-channel defense," em *Proc. IEEE Int. Conf. Cloud Eng. (IC2E)*, Abr. 2018, pp. 34–40.

[111] M. Azab e M. Eltoweissy, "Migrate": Towards a lightweight moving target defense against cloud side-channels," em *Proc. IEEE Secur. Workshops sobre Privacidade (SPW)*, Maio de 2016, pp. 96-103.
[112] S. Checkoway e H. Shacham, "ataques de Iago": Why the system call API is a bad untrusted rpc interface," *ACM SIGPLAN Notices*, vol. 48, no. 4,
pp. 253-264, Abr. 2013.
[113] C. Pahl, A. Brogi, J. Soldani, e P. Jamshidi, "Cloud container technologies": A state-of-the-art review," *IEEE Trans. Cloud Comput.,* a ser publicado.
[114] A. Dewald, M. Luft, e J. Suleder. (2018). *Análise de Incidentes e Sics Foren em Ambientes Docker*. [Online]. Disponível: https://static.ernw.de/whitepaper/ERNW_Whitepaper64_IncidentForensicDocker_signed. pdf
[115] M. J. De Lucia, "A survey on security isolation of virtualization, containers, and unikernels," Res. do Exército dos EUA. Laboratório. Aberdeen Proving Ground, MD, EUA, Tech. Rep. AD1035194, 2017.
[116] R. Morabito, V. Cozzolino, A. Y. Ding, N. Beijar, e J. Ott, "Consolidate IoT edge computing with lightweight virtualization", *IEEE Netw.*, vol. 32, no. 1, pp. 102-111, Jan./Fev. 2018.
[117] R. Morabito, "Virtualização na Internet de dispositivos de ponta com tecnologias de contentores: A performance evaluation", "*IEEE Access"*, vol. 5,
pp. 8835–8850, 2017.
[118] E. Castanho. (2016). *O Futuro da IOT: Contentores AIM para Resolver a Crise da Secu- ridade*. Acedido: 28 de Dezembro de 2017. [Online]. Disponível: https://www. linux.com/news/future-iot-containers-aim-solve-solve-security-crisis
[119] A. Haritha e A. Lavanya, "Internet das coisas": Questões de segurança',' *Inter-nacional J. Eng. Sci. Invention*, vol. 6, no. 11, pp. 45-52, 2017.
[120] J. Johansen e C. Schaufler, "Namespacing and stacking the LSM," em *Proc. Linux Plumbers Conf.*, 2017, pp. 1-5.

SARI SULTAN recebeu a licenciatura em engenharia de com- puter da Universidade Hashemite. Está actualmente a tirar o M.Sc. em engenharia de com- puter com a Universidade do Kuwait, através de
uma Bolsa de Estudo de Excelência. Os seus interesses de investigação incluem segurança, privacidade, forense digital, e inteligência artificial. É um Hacker Ético Certificado (CEH), um Computer Hacking Forensics Investigator (CHFI), e um Auditor Principal do Sistema de Gestão de Segurança da Informação ISO 27001.

IMTIAZ AHMAD recebeu o Bacharelato em Engenharia Eléctrica pela Universidade de Engenharia e Tecnologia de Lahore, Paquistão, o M.Sc. em Engenharia Eléctrica de
a King Fahd University of Petroleum e Miner- als, Dhahran, Arábia Saudita, e o doutoramento em engenharia informática pela Syracuse University, Syracuse, NY, EUA, em 1984, 1988, e 1992, respectivamente. Desde 1992, está no Departamento de Engenharia Informática, Kuwait
Universidade, Kuwait, onde é actualmente Professor. As suas pesquisas inter-ests incluem a automatização da concepção de sistemas digitais, síntese de alto nível, computação distribuída, e redes definidas por software.

A TASSOS DIMITRIOU está actualmente no Departamento de Engenharia Informática, Universidade do Kuwait, e no Departamento de Investigação e Académica...
puter Technology Institute, Grécia. Antes disso, foi Professor Associado do Athens Information Technology, Grécia, onde dirigia o Algorithms and Security Group, e Professor Adjunto da Carnegie Mellon Uni- versity, EUA, e da Aalborg University, Dinamarca.
Conduz investigação em áreas que vão desde os fundamentos teóricos da criptografia até à concepção e implementação de protocolos de comunicação eficientes e seguros de vanguarda. É dada ênfase à autenticação e privacidade de vários tipos de redes (adhoc, redes de sensores, RFID e redes inteligentes), arquitecturas de segurança para redes com e sem fios e redes de telecomunicações, e o desenvolvimento de aplicações seguras para redes e comércio electrónico. É Membro Sénior do IEEE, Conferencista Distinguido da ACM, e Fulbright Fellow. Mais informações sobre ele podem ser encontradas em http://tassosdimitriou.com/