Análise de segurança sobre a comunicação entre contêineres Docker implementados em Linux e Windows

Raphael M. Pennacchi[1], **Charles C. Miers**[1]

[1]Departamento de Ciência da Computação
Universidade do Estado de Santa Catarina

raphaelpennacchi1997@gmail.com,

charles.miers@udesc.br

Resumo. *O uso de contêineres tornou-se popular em vários cenários pelo seu alinhamento com recentes tecnologias de desenvolvimento, assim como baixo uso de recursos. Neste cenário, a solução de contêiner Docker superou o desempenho de outras, sendo essencialmente baseada em sistemas operacionais utilizando o núcleo Linux. A Microsoft recentemente disponibilizou formas de executar contêineres em sua linha de sistemas operacionais para servidores. Existem comprovações que algumas operações em contêineres Linux apresentam problemas de segurança, e o surgimento de Contêineres Windows aumenta esta preocupação. Neste sentido, esse artigo apresenta os resultados iniciais de uma análise de alguns aspectos de controle de recursos, segurança de imagem de contêineres e segurança na comunicação entre contêineres.*

1. Introdução

A virtualização como tecnologia cria uma base para aprimorar a utilização de recursos computacionais, de comunicação e de armazenamento [Kabbe 2017]. Virtualização baseada em hipervisor, tipicamente máquinas virtuais (MVs), fazem a abstração do hardware hospedeiro. Contudo, exigem a instalação de sistema operacional (SO) dedicado, consequentemente, a duplicação de serviços operacionais que consomem recursos, *e.g.,* núcleo, escalonador de tarefas. Embora o compartilhamento de recursos físicos seja desejável, o consumo de recursos por serviços operacionais duplicados é visto como desperdício. A virtualização em nível de SO surgiu como uma alternativa mais eficiente no que se refere ao consumo de recursos virtualizados. Contêineres, assim como são chamadas as virtualizações em nível de SO, passaram a ser empregados para desenvolver aplicações flexíveis, e base para arquitetura de microsserviços [Zhang et al. 2018]. O princípio da virtualização baseada em contêineres é utilizar os recursos do núcleo para criar um ambiente isolado para processos. É possível executar contêineres sob MVs, garantindo um nível a mais de isolamento entre componentes de uma aplicação.

Contêineres são considerado o método padrão para alocação de microsserviços, mas uma pesquisa de mercado [Sultan et al. 2019] revelou que a segurança de contêineres é uma das maiores preocupações e barreira à adoção de contêineres para diversas organizações. Dentre as diversas soluções que implementam o conceito de contêineres umas das mais clássicas é a solução Docker, que iniciou disponibilizando contêineres em ambientes GNU/Linux. Com o surgimento do Docker Windows surgem questões relativas a aspectos de segurança que podem ser sanados por esta versão, ao mesmo tempo que novos

problemas de segurança podem surgir. Em contrapartida, já há estudos sobre segurança de contêineres Docker para plataforma GNU/Linux. Porém, não foram identificados estudos para análise de ameaças de segurança no Docker Windows até o momento da submissão deste artigo. O presente trabalho tem como objetivo delimitar o escopo de análise de segurança de contêineres Docker, tanto GNU/Linux como Windows, usando como base os critérios de [Pannizon 2019]: *(i)* Controle e limitação de recursos; *(ii)* Segurança das imagens de contêineres e *(iii)* Segurança dos dados na comunicação entre contêineres.

2. Arquitetura Docker: Linux vs Windows

A Microsoft, ao incorporar suporte a contêineres Docker na sua linha de SOs para servidores, visa buscar que diversas aplicações Docker possam usufruir a mesma plataforma na qual já possui os seus serviços .Net [Docker 2020]. Neste Sentido, tão importante quanto fornecer o serviço de contêineres baseados em Docker é a necessidade de compartilhar um base comum que possibilite a fácil migração entre contêineres baseados em GNU/Linux para MS-Windows Server 2016 (ou superior) e vice-versa.

As virtualizações com contêineres são realizadas com encapsulamento de processos dentro de um SO. Para serem efetuadas, estas necessitam de acesso de recursos quem atuam diretamente no núcleo e nas funções básicas do SO [Panizzon et al. 2019]. Contêineres Docker devem funcionar de formar similar em duas plataformas distintas. O Docker atua como *middleware* sobre as arquiteturas (Figura 1).

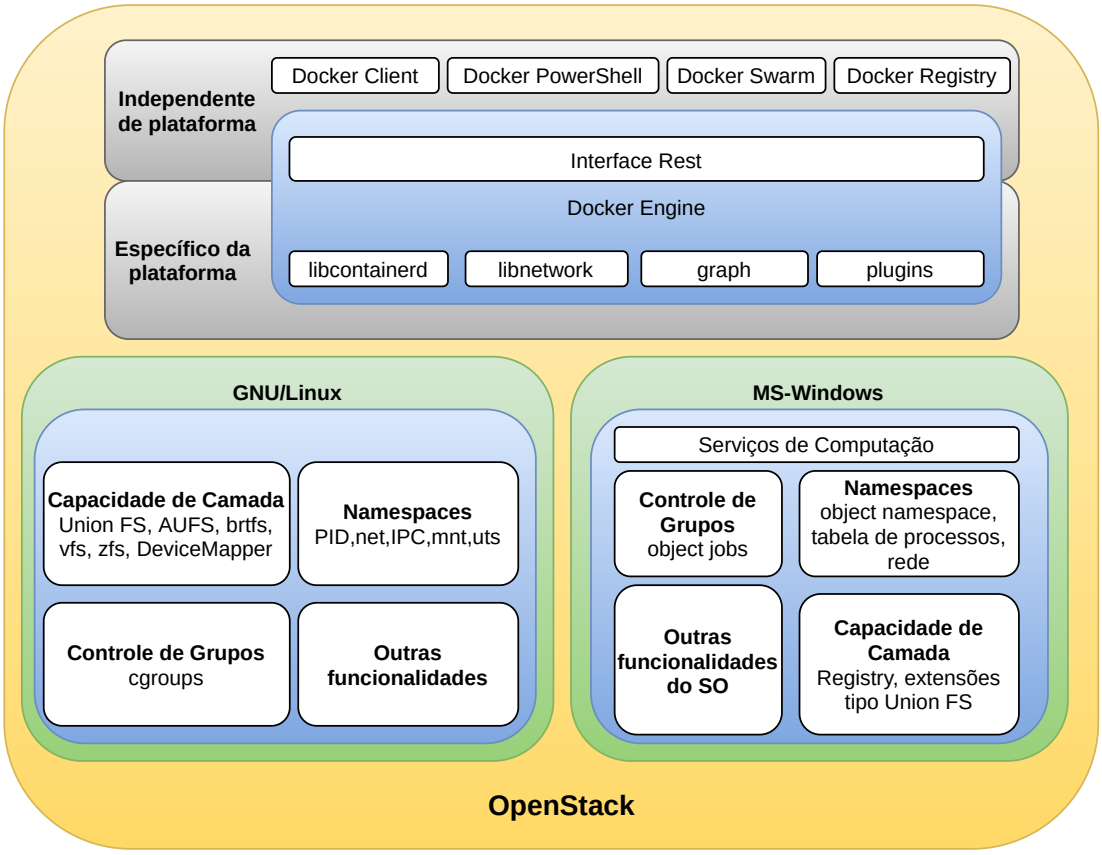

Figura 1. Arquitetura Docker: GNU/Linux e Windows.

A Figura 1 revela como os SOs GNU/Linux e MS-Windows Server fornecem a base para os componentes Docker que são independentes de plataforma. Assim, esse

artigo revela a primeira parte da pesquisa na qual ficam evidenciados os principais componentes do MS-Windows Server para fornecer o serviço de conteinerização.

Neste contexto, surge a questão relativa a problemas de segurança já detectados na arquitetura do GNU/Linux possuírem ou não a suscetibilidade de ocorrer na arquitetura MS-Windows. Já foram detectados alguns problemas referentes ao chroot [Details 2017] [NIST 2019]. No artigo foi testado alguns cenários de ataques possíveis, foi testado um cenário de cada um dos casos de uso do modelo de ameaças III-1, III-2, III-3 [Sultan et al. 2019].

3. Proposta

A proposta consiste em implementar contêineres Docker Linux e Windows em uma MV com as mesmas configurações de *flavor* a fim de verificar: *(i)* Controle e limitação de recursos; *(ii)* Segurança das imagens de contêineres e *(iii)* Segurança dos dados na comunicação entre contêineres. Os experimentos tem por finalidade não apenas verificar o isolamento mas também a degradação ou não do desempenho do host e do contêiner envolvido no experimento. Os testes estão sendo realizados na nuvem privada IaaS OpenStack do LabP2D/UDESC, a Nuvem Tche. Foram criadas duas MVs: uma com GNU/Linux Ubuntu 18.04 Bionic Beaver e outra com Microsoft Windows Server 2016.

Em ambas MVs estão sendo aplicados testes específicos para a análise de segurança referente a cada critério (Seção 4). Para o teste de controle e limitação de recursos, estão sendo utilizados cinco contêineres ativos simultaneamente, em um dos contêineres ativos é utilizado um ataque de negação de serviço (Dos) baseado no princípio de *forkbomb*. Com esse teste pode ser analisado se a limitação padrão de segurança dos contêineres está ativada e se essa limita o uso de memória ou permite um ataque de negação de serviço interno, assim afetando os outros contêineres.

Com base no critério de segurança de imagem é realizada uma análise de falhas em imagens disponíveis em repositórios utilizados pela comunidade (Docker Hub). Utilizando ferramentas de verificação de imagens de contêineres, pode-se identificar possíveis falhas, e analisar se o grau de vulnerabilidade pode ser comprometedor. A partir dessa análise, com base em CVE (Common Vulnerabilities and Exposures) e no modelo de falhas, podem ser propostas formas de mitigar as vulnerabilidades.

Para testar a segurança na comunicação entre contêineres que utilizam o mesmo núcleo, são utilizados três contêineres executados simultaneamente, sendo um destes contêineres empregado para realizar *sniffing*. Com esse teste pode-se verificar o isolamento no canal de comunicação entre contêineres.

4. Critérios de Análise

Os critérios utilizados foram baseados no guia de segurança de contêineres [NIST 2017] e no modelo de falhas [Sultan et al. 2019]. Os critérios utilizados foram:

1. Controle e limitação de recursos: Realizar a análise de mecanismo de controle de acesso e escalação de recursos.
2. Segurança das imagens de contêineres: Verificar a existência de vulnerabilidades em imagens de contêineres hospedadas em repositórios da comunidade a partir de ferramentas de busca de vulnerabilidades, e recomendar soluções baseadas no modelo de falhas [Sultan et al. 2019].

3. Segurança dos dados na comunicação entre contêineres: Analisar a comunicação entre contêineres que utilizam o mesmo núcleo. Verificar o isolamento do canal de comunicação entre os contêineres pares.

Os testes inciais revelaram alguns problemas mas com inconsistências na repetições. Assim, ajustes nos procedimentos de experimento estão em execução. Um dos pontos problemáticos foi a atualização do sistema operacional base, do software da nuvem e do contêiner entre os experimentos. A Nuvem Tche agora está com uma nova versão instável do software e todos os experimentos estão sendo executados com a mesma versão de SO e Docker.

5. Considerações & Trabalhos Futuros

Os experimentos iniciais realizados já revelaram que algumas opções padrão de contêineres Docker GNU/Linux que apresentavam falhas de segurança foram corrigidas no Docker Windows. Por outro lado, formas identificadas falhas no Docker Windows de natureza bem distinta das existentes no GNU/Linux que estão sendo analisadas para compor um artigo completo.

Para uma maior abrangência da pesquisa, um próximo passo é cobrir uma maior área de casos do modelo de falha [Sultan et al. 2019]. Realizando um maior número de testes sobre políticas de segurança nas plataformas distintas, a precisão da analise sobre as diferenças entre as plataformas Docker GNU/Linux e Docker Windows será maior. Permitindo analisar erros específicos, se houver, de cada plataforma.

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

Referências

Details, C. (2017). Cve-2015-6240. "https://www.cvedetails.com/cve/CVE-2015-6240/".

Docker (2020). Get started with Docker for Windows. https://docs.docker.com/docker-for-windows/.

Kabbe, J.-A. (2017). Security analysis of docker containers in a production environment. Master's thesis, NTNU.

NIST (2017). Application container security guide.

NIST (2019). Cve-2019-3811 detail. "https://nvd.nist.gov/vuln/detail/CVE-2019-3811".

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

Pannizon, G. (2019). Uma análise de segurança no uso de contêineres docker em nuvens iaas openstack.

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

Zhang, Q., Liu, L., Pu, C., Dou, Q., Wu, L., and Zhou, W. (2018). A comparative study of containers and virtual machines in big data environment. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 178–185.