SEGURANÇA DE REDE NAS OPERAÇÕES DE CONTÊINERES EM NUVENS OPENSTACK[1]

Henrique Zanela Cochak[2], Charles Christian Miers[3]

[1] Vinculado ao projeto "Análise de segurança de recursos virtualizados em nuvens computacionais IaaS baseadas em OpenStack usando honeypots de baixa interatividade"
[2] Acadêmico do Curso de Ciências da Computação – CCT – Bolsista PROBITI
[3] Orientador, Departamento de Ciências da Computação – CCT – charles.miers@udesc.br

A computação em nuvem se tornou o modelo de entrega de serviços de computação oferecendo inovação mais rápida com recursos flexíveis que implicam em economias de escala. Utiliza-se de conceitos de virtualização para abstrair recursos das máquinas que são alocados em *pools* centralizados e são coordenados por uma camada de gerenciamento. Um dos conceitos de virtualização são os contêineres, uma unidade executável de software, na qual o código do aplicativo, bibliotecas, dependências e arquivos de configuração são empacotados juntos para executá-lo em qualquer lugar, sendo considerados como virtualização de sistema operacional pelo fato de que existe um compartilhamento destes objetos pelo mesmo núcleo do Linux. A conteinerização surge como uma opção da virtualização clássica de máquinas. Relacionado a plataformas de gerenciamento de contêineres, existe a plataforma de serviço Docker a qual foi construída modularmente. Aproveitando desta modularidade, surgiu o Kata Containers, visando aumentar a segurança ao criar contêineres dentro de máquinas virtuais QEMU/KVM. Para fazer isso, é feito a troca do *runtime* (Figura 1), no qual esta funcionalidade como propósito de existência é de conter todo o ciclo de vida de um contêiner, incluindo arquivos de configuração e o ambiente de execução. É possível fazer a troca do *runtime* padrão do Docker, chamado de runC, para o kata, criado pela comunidade Kata Containers devido a modularidade da plataforma Docker. O propósito da pesquisa foi utilizar do *runtime* kata dentro da plataforma Docker e compará-lo com o *runtime* runC, buscando diferenças de desempenho ao analisar por uma perspectiva de redes. Observando as redes padrões já providenciadas pelo Docker: bridge, host, macvlan e overlay, em ambos *runtimes, c*om o auxílio da infraestrutura existente no LabP2D que disponibilizou um servidor em nuvem utilizando OpenStack diretamente em um hipervisor *bare metal.* O experimento é feito dentro de um ambiente mais controlado e uma configuração mais estável. Analisando os pares de cliente-servidor: (bridge, bridge), (host, bridge), (macvlan, macvlan), (overlay, overlay), observa-se que o *runtime* runC possui um desempenho maior (Figura 2 e 3) que o *runtime* kata, devido a forma que ocorre a implementação dos contêineres. O *runtime* runC utiliza do conceito básico de contêiner, implementando o mesmo com o compartilhamento de núcleo do Linux, no qual o *runtime* kata utiliza de um núcleo que encapsula um contêiner, o que gera um nível de abstração maior, resultando em um nível extra de segurança. Este tipo de implementação sofre maiores perdas quando um contêiner servidor fica sobrecarregado de pedidos de processamento na CPU (Figura 4 e 5), resultando em uma diferença ampla na troca de desempenho pela segurança que o Kata Containers busca atingir efetivamente.

Palavras-chave: Contêineres. Docker runC. Kata Containers.

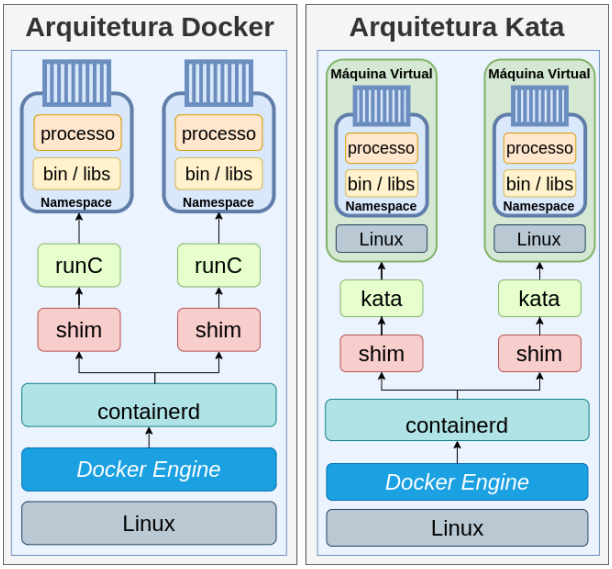

Figura 1. Diferença entre arquiteturas

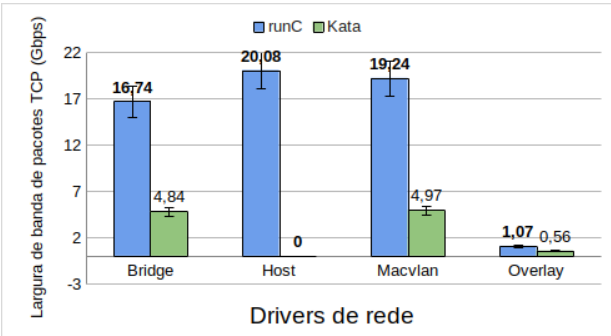

Figura 2. Largura de banda

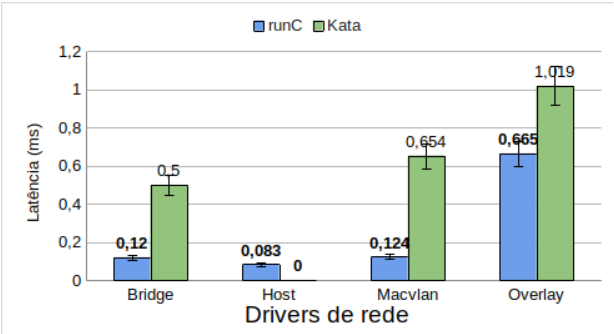

Figura 3. Latência

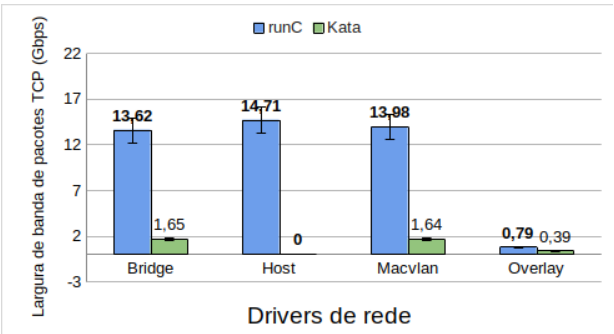

Figura 4. Largura de banda com o stress-ng

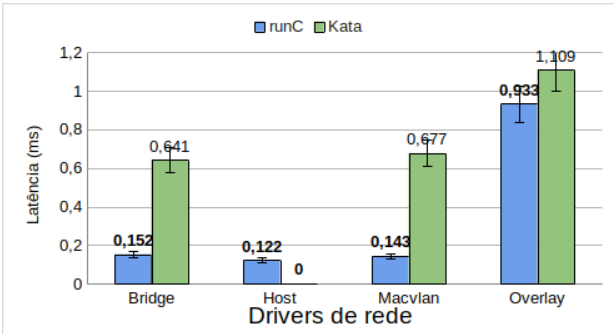

Figura 5. Latência com o stress-ng