LLC: Low Level Contêiner no Linux

Caio Gomes Flausino[1], Diego César Florêncio de Queiroz[1], Daniel Sundfeld[2]

[1]Instituto Federal de Educação, Ciência e Tecnologia de Brasília (IFB)
Campus Brasília, Asa Norte - 70.830-450 - Brasília - DF - Brasil

[2]Faculdade UnB Gama (FGA) - Universidade de Brasília (UnB)
Campus Gama, Setor Leste - 72.444-240 - Gama - DF - Brasil

caio.mensagens@gmail.com, diego.queiroz@ifb.edu.br, daniel.sundfeld@unb.br

Abstract. *Since the 1960s, the remote execution of procedures has been studied in several applications of distributed systems. More recently, with web technologies and security concerns, new technologies have changed the way processes can be executed. WebAssembly allows executing code through a browser and containers isolate the running application. In this work, we compare the performance of basic applications with WebAssembly and the LLC (Low Level Container), an application execution sandbox that we implemented. The results show that WebAssembly is faster than containers, with only SECCOMP security instructions, but LLC has shown to be a promising platform for execution.*

Resumo. *Desde a década de 1960, a execução remota de procedimentos é estudada em diversas aplicações dos sistemas distribuídos . Mais recentemente, com as tecnologias web e preocupações de segurança, novas tecnologias mudaram a forma que os processos podem ser executados. O WebAssembly, permite executar código através de um navegador e os contêiners isolam a aplicação em execução. Neste trabalho, comparamos a performance de aplicações básicas com WebAssembly e o LLC (Low Level Container), um sandbox de execução de aplicações que implementamos. Os resultados mostram que o WebAssembly é mais rápido que contêiners, apenas com instruções de segurança SECCOMP, mas o LLC se mostrou uma promissora plataforma para a execução.*

1. Introdução

Atualmente, é crescente o interesse em executar cargas de trabalho no lado do cliente e para alcançar níveis de desempenho cada vez maiores, mantendo ainda todos os requisitos de segurança, fazendo-se necessário estudar alternativas que vão além do paradigma estabelecido. Sendo assim, cabe discutir se um sistema operacional como o Linux fornece as ferramentas necessárias para construir esta plataforma.

Logo este trabalho visa verificar a possibilidade de utilizar primitivas de contêineres no Linux para a criação de um ambiente *web* capaz de competir com WASM (*WebAssembly*) no lado do cliente, considerando aspectos tanto de segurança e quanto de desempenho.

Foi desenvolvido, então, um *software* de conteinerização apelidado de *Low Level Container* (LLC) para isolar as aplicações nativas. Em seguida, foi criada uma *Application Programming Interface* (API) para permitir que aplicações no navegador pudessem

enviar executáveis ao ambiente isolado. Finalmente, procedeu-se uma análise comparitiva de desempenho e de recursos de segurança entre WebAssembly (WASM) e o LLC, conforme a abordagem quantitativa proposta por [Creswell 2016].

2. Referencial teórico

Conforme Neto et al. (2018) e Aichatou et al. (2016), uma prova de conceito é uma prática de pesquisa que busca aprimorar o entendimento ou conhecimento sobre o objeto em estudo, auxiliando na adoção ou desenvolvimentos de novas tecnologias ou produtos. Dessa forma, é possível observar os principais problemas antes de se adotar uma tecnologia.

Portanto, essa pesquisa procurou trazer uma prova de conceito, tentando avaliar os principais problemas e benefícios de se usar tecnologias de contêineres Linux como alternativa ao WASM, atual ambiente padrão da *web* para a execução de código nativo. Ademais, vale destacar as tecnologias presentes no *kernel* Linux para isolamento dos processos, que foram utilizados de base para o LLC: *namespaces*, *LSMs* (Linux Security Modules), *Capabilities* e o SECCOMP (*SECure COMPuting*) .

Os navegadores, no entanto, procuram implementar a segurança partir de um *bytecode*, deixando com que o WASM verifique e garanta que aquele código não possa escapar do *sandbox*. Ou seja, ao invés de fazer verificações de segurança dinâmicamente, elas são feitas durante a compilação do *bytecode*, gerando ao final um binário seguro.

3. LLC: Low Level Contêiner

A prova de conceito proposta envolve dois elementos principais: um servidor NodeJS, o qual permite a comunicação entre os *sites* e o *host*; e um *runtime* customizado, que inicializa contêineres com diversas as configurações de segurança e foi apelidada de LLC.

O funcionamento e fluxo podem ser descritos resumidamente da seguinte forma: os *websites* se comunicam com o servidor NodeJS no *host* por meio de mensagens HTTP para enviarem código; baixado o código fonte, o servidor com ajuda do *runtime* inicializa um contêiner de compilação; finalizada a compilação, o código é executado no cliente e seu resultado é retornado ao *website* como uma resposta HTTP (todo esse processo sendo realizado no mesmo *host*).

Além disso, o LLC foi configurado com o máximo de restrições possíveis por padrão: *whitelisting* de chamadas de sistema; perfil AppArmor que bloqueia a escrita, leitura e execução; bit *no_new_privs* habilitado; e isolamento the *namespacing* e da raiz do programa.

4. Análise e resultados

Para coletar resultados de desempenho, foram desenvolvidos vários softwares em C++ que representassem alguns dos algoritmos padrões na área da computação como bubble sort e a sequência 'fibonnaci'. Em seguida, os mesmos códigos foram compilados com o GNU GCC e com o *emscripten*. Embora exista o *overhead* por causa da comunicação por rede, esse problema não foi analisado nos resultados de desempenho, visto que a proposta era de avaliar os ambientes em si.

Inicialmente, foi realizada uma comparação entre o ambiente conteinerizado desenvolvido, LLC, e o binário nativo para avaliar se existia algum *overhead* e mensurá-lo. Observamos que as configurações de isolamento com *unshare* (*namespacing*), a configuração padrão; e com o isolamento de raiz, *pivot_root*, possuem uma performance praticamente idêntica ao nativo, aproximadamente da 99% performance.

Essa situação foi alterada com a adição do SECCOMP para a filtragem de *system calls*, causando uma grande perda de performance em quase todos os casos. Esse fenômeno ocorre pelo fato do SECCOMP precisar validar cada chamada de sistema feita pelo programa durante sua execução.

Em seguida, foi comparado o LLC e o WASM do Chrome, sendo possível constatar que o ambiente conteinerizado sem SECCOMP conseguiu ser mais rápido na maior parte dos testes, porém, foi mais lento nos testes com tempo de execução mais curtos devido ao tempo de inicialização do WASM ser menor, já que ele não precisa criar um processo do zero para re-executar o teste. A situação transforma-se novamente com o SECCOMP, tornando praticamente todos os testes mais lentos que os programas em WASM.

5. Conclusão e trabalhos futuros

A partir da prova de conceito desenvolvida, foi possível, então, perceber que contêineres Linux não apresentam grandes benefícios em desempenho que justifiquem a substituição do WASM em navegadores, deixando assim de ser uma alternativa viável à *web*.

Dessa forma, o WASM consegue entregar tanto a segurança quanto o desempenho adequado, embora apenas se comparado aos contêineres com SECCOMP. Sua maior desvantagem no momento em que foi realizada a pesquisa é apenas a falta de suporte para recursos como *threads* nativas e espaço de memória de 64*bits*, ambas atualmente próximas de serem adiconadas aos motores de execução dos navegadores como o V8 do Chrome.

Referências

Aichatou, B., Seck, C., Anne, T. S. B., Deguenovo, G. C., Ntabona, A., and Simmons, R. (2016). Strengthening government leadership in family planning programming in senegal: from proof of concept to proof of implementation in 2 districts. *Global Health: Science and Practice*, 4(4):568–581.

Creswell, J. W. (2016). *Projeto de pesquisa: métodos qualitativo, quantitativo e misto*. Artmed, Porto Alegre.

Neto, A. J. R., Borges, M. M., and Roque, L. (2018). A preliminary study of proof of concept practices and their connection with information systems and information science. In *Proceedings of the Sixth International Conference on Technological Ecosystems for Enhancing Multiculturality*, pages 270–275.

THE KERNEL DEVELOPMENT COMMUNITY (2022). Seccomp bpf (secure computing with filters) — the linux kernel documentation. Disponível em: `https://www.kernel.org/doc/html/latest/userspace-api/seccomp_filter.html`. Acesso em: 15 jan. 2022.