Hardware Assisted OS Virtualization

Bhushan Jain, and Donald E. Porter
Stony Brook University
{*bpjain, porter*}*@cs.stonybrook.edu*

1. Extended Abstract

Operating System-level virtualization, also known as a *container*, is an increasingly popular approach to isolating applications that use the same underlying OS kernel [2, 5–7]. Containers have recently gained popularity as the default back-end for Docker, an application packaging and distribution system used by companies including Google [3].

The purported reason to use containers over a hardware virtual machine, such as VMware or Xen, is reduced overheads. Containers forego the ability to run different OSes—an essential feature of VMs, but can be appropriate for scenarios where all guest applications are programmed to the same OS API. Containers are implemented by copying a subset of OS data structures, which one would expect to be lighter-weight than running another complete OS instance. Similarly, data structure initialization can be faster than booting a legacy OS kernel.

This difference in implementation techniques raises concerns about security. Unlike VMs, containers expose the host system call table to each guest, and rely on pointer hooks to redirect system calls to isolated data structure instances, called namespaces in Linux. One security concern for containers is that there may be exploitable vulnerabilities in the pointer indirection code, leading to information leakage or privilege escalation. System calls servicing one guest operate in the same kernel address space as the data structures for other guests. For this reason containers also disallow functionality such as loading kernel extensions.

A second security concern for containers is that any vulnerabilities in the system call API of the host kernel are shared, unlike VMs. Specifically, a kernel bug that is exploited through a system call argument is a shared vulnerability with a co-resident container, but not on a co-resident VM. As a point of reference, the national vulnerability database [4] lists 147 such exploits out of 291 total Linux vulnerabilities for the period 2011–2013. In short, containers inherit the same security problems as monolithic operating systems written in unsafe languages, which caused people to turn to hypervisors for security isolation. In contrast, the interface exported by a shared hypervisor is narrower, and less functionality executes in an address space shared among guests. Moreover, the hypervisor isolates each VM's memory for kernel data structures using a second set of page tables like Extended Page Tables (EPT) [1].

Memory isolation like VMs for just the container specific kernel objects can solve the problem of security isolation for containers. We leverage hardware virtualization features like EPT to create a hardware isolated memory namespace, and prevent access of container specific kernel objects outside the namespace. We redesign the OS to be EPT page protection friendly for containers. We change the allocators for container specific objects to be at page granularity so that the protection boundary can naturally map to page translation and page protection provided by EPT. We maintain the light-weightness of the containers by sharing the OS between containers and the host while providing the memory isolation using a second set of page tables like EPT.

The contributions of this work are as follows:

- A redesign of the OS to be EPT page protection friendly.
- Provide VM-like EPT based isolation to containers without sacrificing the container's efficiency with very low overheads.

References

[1] N. Bhatia. Performance Evaluation of Intel EPT Hardware Assist. *VMware ESX white paper)*, Oct. 2012.

[2] S. Bhattiprolu, E. W. Biederman, S. Hallyn, and D. Lezcano. Virtual servers and checkpoint/restart in mainstream Linux. *OSR*, 42:104–113, July 2008.

[3] C. Metz. Google Embraces Docker, the Next Big Thing in Cloud Computing. *WIRED*, June 2014. `http://www.wired.com/2014/06/eric-brewer-google-docker/`.

[4] NIST. National Vulnerability Database. `http://nvd.nist.gov/`, 2008.

[5] D. Price and A. Tucker. Solaris Zones: Operating system support for consolidating commercial workloads. In *LISA*, pages 241–254, 2004.

[6] S. Soltesz, H. Pötzl, M. E. Fiuczynski, A. Bavier, and L. Peterson. Container-based Operating System Virtualization: A Scalable, High-performance Alternative to Hypervisors. In *EuroSys '07*, pages 275–287, New York, NY, USA, 2007. ACM.

[7] M. Stokely and C. Lee. The FreeBSD Handbook, 3rd Edition, Vol 1: Users's Guide, 2003.