Contribution ID: **267** Type: **not specified**

Address Space Isolation for Container Security

Tuesday 10 September 2019 15:30 (15 minutes)

Containers are generally percieved less secure than virtual machines. Without going into a theological argument about the actual state of the affairs, we suggest to explore the possibility of using address space isolation inside the kernel to make containers even more secure.

Assuming that kernel bugs and therefore vulnerabilities are inevitable it is worth isolating parts of the kernel to minimize damage that these vulnerabilities can cause.

One way to create such isolation is to assign an address space to the Linux namespaces, so that tasks running in namespace A have different view of kernel memory mappings than the tasks running in namespace B.

For instance, by keeping all the objects in a network namespace private, we can achieve levels of isolation equivalent to running a separated network stack.

Another possible usecase is isolating address spaces for different user namespaces.

Beside marrying namespaces with address spaces we also considering implementaiton of isolated memory mappings using mmap()/madvise() so that a region of the caller's memory would be hidden from the rest of the system.

We are going to give a short update on current status of our research and we are going to discuss implications of the address space isolation and possible future directions:

- What are the trade-offs between letting user-space to control the isolation or keeping the control completely in-kernel.
- What should be user-visible interface for address space management? Does it need to be on/off switch at kernel command line or do we need runtime knobs for that? Or maybe even "address space namespace" or "address space cgroup"?
- How can we evaluate the security improvements beyond empiric obvservation that when less code and data are mapped, there are less vulnerabilities exposed?

I agree to abide by the anti-harassment policy

Yes

Primary authors: RAPOPORT, Mike; BOTTOMLEY, James (IBM)

Presenters: RAPOPORT, Mike; BOTTOMLEY, James (IBM)

Session Classification: Containers and Checkpoint/Restore MC