X-Containers: Breaking Down Barriers to Improve Performance and Isolation of Cloud-Native Containers

Zhiming Shen
Cornell University

Joint work with Zhen Sun, Gur-Eyal Sela, Eugene Bagdasaryan, Christina Delimitrou, Robbert Van Renesse, Hakim Weatherspoon
Software Containers

20% of companies have containers deployed

Docker containerization debuted

50% of companies have containers deployed

SOURCE: GARTNER
Cloud-Native Container Platforms
Cloud-Native Container Platforms

- Single Concern Principle: Every container should address a single concern and do it well.

- Making containers easier to
  - Replace, reuse, and upgrade transparently
  - Scale horizontally
  - Debug and troubleshoot

*Image src: https://pivotal.io/cloud-native*
The Problem

Shared kernel attack surface and TCB

Not allowed to install kernel modules

Hard to tune or optimize for a specific container
Existing Solutions

- **Linux Container Process Process**
- **VM Linux Process Process**
- **KVM Linux**
- **gVisor Linux Process Process**
- **Container Clear Container**
- **Isolation Customization Optimization Portability Performance**

**Require nested hardware virtualization support in the cloud**

**Ptrace mode: high overhead**

**KVM mode: require nested virtualization**
X-Containers achieve

- VM-level **Isolation**
- Support of Kernel **Customization**
- Support of Kernel **Optimization**
- Good **Portability** (without the need of hardware-assisted virtualization)
- High **Performance**

AND

- Backward **Compatibility**
X-Containers

User mode

Kernel mode

OS Kernel
X-Containers

![Diagram of X-Containers]

- User mode
- Kernel mode
- Exokernel

Diagram showing the concept of X-Containers, highlighting the difference between user mode and kernel mode, and the role of the Exokernel.
X-Containers

User mode

Kernel mode

Exokernel
X-Containers

User mode

Kernel mode

X-Container

Process

Process

X-LibOS

X-Container

Process

Process

X-LibOS

X-Kernel
X-Containers

• A new security paradigm for cloud-native containers

• X-Kernel: an exokernel with a small attack surface and TCB
• X-LibOS: a LibOS that decouples security isolation from the process model
Threat Model and Design Trade-offs

• Threat model

• Trade-offs
  • Reduced intra-container isolation
  • Improved inter-container isolation and performance
  • Process isolation and kernel-supported security features are not effective
Implementation

• X-LibOS from Linux kernel
  • Binary compatibility
  • Highly customizable

• X-Kernel from Xen
  • Para-virtualization interface
  • Concurrent multi-processing

• Limitations
  • Memory management
  • Spawning time
Optimizing System Calls

- Existing solutions
  - Patch source code
  - Link to another library

- Our solution
  - Automatic Binary Optimization Module (ABOM)
  - Binary level equivalence
  - Position-independence

For many applications, more than 90% of syscalls are turned into function calls
Evaluation Setup

• Testbed
  • Amazon EC2
  • Google Compute Engine

• Compared container runtimes
  • Docker
  • gVisor (Ptrace in Amazon, and KVM in Google)
  • Clear-Container (only in Google)
  • Xen-Container
  • X-Container

• Configurations
  • Patched for Meltdown
System Call Performance

Up to 27X of Docker (patched) and 1.6X of Clear-Container
Real Application Performance

Normalized Throughput

NGINX

Amazon

Google

1.21x~1.27x

Memcached

Amazon

Google

2.64x~3.08x

Redis

Amazon

Google

1x~1.2x

Apache

Amazon

Google

0.64x~0.72x
Spawning Time and Memory Footprint

Reduced to 460ms. Can be further reduced to <10ms.
More Evaluations in the Paper

• More micro/macro benchmarks
• Patched and unpatched for Meltdown
• Comparing to Unikernel and Graphene
• Scalability (up to 400 containers on a single host)
Conclusion

• X-Containers: a new security paradigm for isolating single-concerned cloud-native containers
  • X-Kernel: an exokernel with a small attack surface and TCB
  • X-LibOS: A LibOS that decouples security isolation from the process model
  • Trade-off: intra-container isolation vs. inter-container isolation

• Implemented with Xen and Linux
  • Binary compatibility
  • Concurrent multi-processing

• More at http://x-containers.org

Thank You. Questions?
Backup Slides
<table>
<thead>
<tr>
<th></th>
<th>Container</th>
<th>gVisor</th>
<th>Clear-Container</th>
<th>LightVM</th>
<th>X-Container</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inter-container isolation</td>
<td>Poor</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>System call performance</td>
<td>Limited</td>
<td>Poor</td>
<td>Limited</td>
<td>Poor</td>
<td>Good</td>
</tr>
<tr>
<td>Portability</td>
<td>Good</td>
<td>Good</td>
<td>Limited</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>Compatibility</td>
<td>Good</td>
<td>Limited</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>Intra-container isolation</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Reduced</td>
</tr>
<tr>
<td>Memory efficiency</td>
<td>Good</td>
<td>Good</td>
<td>Limited</td>
<td>Limited</td>
<td>Limited</td>
</tr>
<tr>
<td>Spawning time</td>
<td>Short</td>
<td>Short</td>
<td>Moderate</td>
<td>Moderate</td>
<td>Moderate</td>
</tr>
<tr>
<td>Software licensing</td>
<td>Clean</td>
<td>Clean</td>
<td>Clean</td>
<td>Clean</td>
<td>Need discussion</td>
</tr>
</tbody>
</table>

Pros and Cons of the X-Container Architecture
Comparing Isolation Boundaries

- Kernel
- Container
- Hypervisor
- Virtual Machine
- VM
- Process
- LibOS
- L4Linux
- Process
- Microkernel
- Exokernel
- X-Container
- Process
- X-LibOS
- Unikernel, Dune, EbbRT, OSv
- Library OS (Exokernel)
- X-Container
- Library OS (LibOS)
- Process
Automatic Binary Optimization Module (ABOM)

00000000000eb6a0 <__read>:
eb6a9:   b8 00 00 00 00   mov $0x0,%eax  
eb6ae:   0f 05          syscall

7-Byte Replacement (Case 1)

00000000000eb6a0 <__read>:
eb6a9:   ff 14 25 08 00 60 ff   callq *0xffffffffffffff600008

7-Byte Replacement (Case 2)

000000000007f400 < syscall.Syscall>:
7f41d:   48 8b 44 24 08   mov 0x8(%rsp),%eax  
7f422:   0f 05          syscall

9-Byte Replacement (Phase-1)

0000000000010330 <__restore_rt>:
10330:   48 c7 c0 0f 00 00 00 mov $0xf,%rax  
10337:   0f 05          syscall

9-Byte Replacement (Phase-2)
The Exokernel Approach

• Separating protection and management