Blending Containers and Virtual Machines: A Study of Firecracker and gVisor

Anjali Tyler Caraza-Harter Michael M. Swift

Department of Computer Sciences University of Wisconsin-Madison
Existing (legacy) solutions

### Hypervisor based virtualization

- Different kernel
- High boot time
- High memory footprint
- Strong isolation

### Container virtualization

- Same kernel
- Fast boot time
- Low memory footprint
- Weak isolation
## Existing (legacy) solutions

<table>
<thead>
<tr>
<th>App</th>
<th>App</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bins/Libs</td>
<td>Bins/Libs</td>
</tr>
<tr>
<td>Guest OS</td>
<td>Guest OS</td>
</tr>
</tbody>
</table>

- **Hypervisor**
  - Different kernel
  - High boot time
  - High memory footprint
  - Strong isolation

- **Container Engine**
  - Same kernel
  - Fast boot time
  - Low memory footprint
  - Weak isolation
The dilemma

Application Developer

- Which has the best performance?
- How vulnerable are they to attacks from other containers?

Isolation Platform Developer

- How to design the architecture to minimize dependency on the host?

Kernel Developer

- How to streamline kernel support for isolation platforms?
Our work

- Initial study
  - Compare properties of three secure isolation platforms
    - Linux containers
    - gVisor
    - Firecracker
  - Evaluate fundamental performance via microbenchmarks
  - Assess dependence on OS services via code tracing
Outline

- **Platform architecture**
  - Firecracker
  - gVisor
- **Comparisons**
  - Syscalls
  - CPU
  - Network
  - Overall
Platform Architecture: Firecracker

- Exports only 3 devices
- Limited system calls using SECure COMputing (seccomp) filters
- Written in Rust - type safe, memory safe, no unsafe C code etc.
Platform Architecture: gVisor

- Handles syscalls in sentry
- User space kernel written in Golang
- Sentry is heavily sandboxed
- Gofer for file access
How are they different?

<table>
<thead>
<tr>
<th>Firecracker</th>
<th>gVisor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relies on guest and host kernel</td>
<td>Relies on Sentry and host kernel</td>
</tr>
<tr>
<td>Narrow syscall interface</td>
<td>Wider syscall interface</td>
</tr>
<tr>
<td>Low memory footprint</td>
<td>Low memory footprint</td>
</tr>
<tr>
<td>Type safe language</td>
<td>Type safe language</td>
</tr>
</tbody>
</table>

Both minimize interaction to the host kernel to enhance secure isolation by limiting syscalls interface.
Attack surface of syscall
Attack surface of syscall
Attack surface of syscall

LXC  gVisor
Attack surface of syscall
Attack surface of syscall

- LXC
- gVisor
- Firecracker
Methodology

- Measure performance for microbenchmarks
  - CPU
  - Network
  - Memory
  - File I/O
- lcov to capture lines of source code executed for microbenchmarks
- Hardware - Cloudlab xl170 machine, a ten-core Intel E5-2640v4 running at 2.4 GHz, 64GBECC Memory (4x 16 GB DDR4-2400 DIMMs), Intel DC S3520 480 GB 6G SATA SSD and 10Gbps NIC.
# LCOV - code coverage report

**Current view:** top level  
**Test:** host_mem_10min_new.info  
**Date:** 2020-02-05 17:52:14  
**Lines:** 56945  
**Functions:** 6385  
**Branches:** 28340  
**Coverage:** 71.4%  

<table>
<thead>
<tr>
<th>Directory</th>
<th>Hit</th>
<th>Total</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>arch/x86/crypto</td>
<td>56945</td>
<td>806323</td>
<td>71.4%</td>
</tr>
<tr>
<td>arch/x86/entry</td>
<td>6385</td>
<td>75512</td>
<td>84.4%</td>
</tr>
<tr>
<td>arch/x86/svycall</td>
<td>28340</td>
<td>618467</td>
<td>46.0%</td>
</tr>
</tbody>
</table>

**Directory**  
**Line Coverage (%)**  
**Branches (%)**
if (exporter && exporter->anon_vma && !importer->anon_vma) {
    int error;
    importer->anon_vma = exporter->anon_vma;
    error = anon_vma_clone(importer, exporter);
    if (error)
        return error;
}

vma_adjust_trans_huge(orig_vma, start, end, adjust_next);

if (file) {
    mapping = file->f_mapping;
    root = &mapping->i_mmap;
    uprobe_munmap(vma, vma->vm_start, vma->vm_end);
}

if (adjust_next)
    uprobe_munmap(next, next->vm_start, next->vm_end);

i_mmap_lock_write(mapping);

if (insert) {
    /*
    * Put into interval tree now, so instantiated pages
    * are visible to arm/parisc __flush_dcache_page
    * throughout; but we cannot insert into address
    * space until vma start or end is updated.
    */
    __vma_link_file(insert);
}
General Findings
CPU: Sysbench Performance

![Graph showing CPU speed vs. number of instances]
CPU: Sysbench Performance

- **CPU speed (per instance)**
- **No. of instances**

Lines represent:
- Orange: Host
- Black: Firecracker
- Green: LXC
- Blue: gVisor

The graph shows the decrease in CPU speed per instance as the number of instances increases.
CPU: Coverage

Overall

<table>
<thead>
<tr>
<th></th>
<th>Firecracker</th>
<th>LXC</th>
<th>gVisor</th>
</tr>
</thead>
<tbody>
<tr>
<td>7403</td>
<td>35692</td>
<td>33557</td>
<td>2079</td>
</tr>
<tr>
<td>6026</td>
<td>864</td>
<td>2725</td>
<td>649</td>
</tr>
</tbody>
</table>

/virt

<table>
<thead>
<tr>
<th></th>
<th>Firecracker</th>
<th>LXC</th>
<th>gVisor</th>
</tr>
</thead>
<tbody>
<tr>
<td>541</td>
<td>0</td>
<td>0</td>
<td>32</td>
</tr>
<tr>
<td>649</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
CPU: Coverage

/\arch

Observations

- High overlap between gVisor and LXC.
- Firecracker and gVisor executes architect-specific code more than LXC.
- KVM specific code only executed by Firecracker and gVisor.
Network: Performance

- Aggr. bandwidth (Gbps)
- No. of instances

- Host
- Firecracker
- LXC
- gVisor
- gVisor+host
Network: Coverage

Overall

<table>
<thead>
<tr>
<th></th>
<th>Firecracker</th>
<th>LXC</th>
<th>gVisor</th>
</tr>
</thead>
<tbody>
<tr>
<td>7280</td>
<td>2280</td>
<td>2453</td>
<td></td>
</tr>
<tr>
<td>5777</td>
<td>30200</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3378</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

/net

<table>
<thead>
<tr>
<th></th>
<th>Firecracker</th>
<th>LXC</th>
<th>gVisor</th>
</tr>
</thead>
<tbody>
<tr>
<td>2449</td>
<td>1923</td>
<td>808</td>
<td></td>
</tr>
<tr>
<td>357</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1003</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Network: Coverage

Overall

<table>
<thead>
<tr>
<th></th>
<th>Firecracker</th>
<th>LXC</th>
<th>gVisor</th>
</tr>
</thead>
<tbody>
<tr>
<td>7280</td>
<td>2280</td>
<td>2453</td>
<td></td>
</tr>
<tr>
<td>5777</td>
<td>30200</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3378</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

/net

<table>
<thead>
<tr>
<th></th>
<th>Firecracker</th>
<th>LXC</th>
<th>gVisor</th>
</tr>
</thead>
<tbody>
<tr>
<td>2449</td>
<td>1923</td>
<td>808</td>
<td></td>
</tr>
<tr>
<td>357</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1003</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
gVisor has high coverage despite having a separate user space network stack.

- gVisor and LXC have high overlap.
- Firecracker also has substantial coverage.
Invocation Frequency

```c
bool is_skb_forwardable(const struct net_device *dev,
                       const struct sk_buff *skb){
    unsigned int len;
    if (!(dev->flags & IFF_UP))
        return false;
    len = dev->mtu + dev->hard_header_len +
        VLAN_HLEN;
    if (skb->len <= len)
        return true;
    if (skb_is_gso(skb))
        return true;
    return false;
}
```

<table>
<thead>
<tr>
<th>Lines</th>
<th>LXC</th>
<th>gVisor</th>
<th>Firecracker</th>
</tr>
</thead>
<tbody>
<tr>
<td>1825</td>
<td>0.2 billion</td>
<td>5 million</td>
<td>0</td>
</tr>
<tr>
<td>1827</td>
<td>0.2 billion</td>
<td>5 million</td>
<td>0</td>
</tr>
<tr>
<td>1828</td>
<td>0.2 billion</td>
<td>5 million</td>
<td>0</td>
</tr>
<tr>
<td>1830</td>
<td>8 million</td>
<td>0.9 million</td>
<td>0</td>
</tr>
</tbody>
</table>

`net/core/dev.c`
Overall Coverage

Union of line coverage

<table>
<thead>
<tr>
<th>Lines</th>
<th>Host</th>
<th>LXC</th>
<th>gVisor</th>
<th>Firecracker</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>50000</td>
<td>75000</td>
<td>100000</td>
<td>75000</td>
</tr>
</tbody>
</table>

Venn diagram showing coverage:
- Firecracker: 21184
- LXC: 1470
- gVisor: 6006
- Overlapping regions:
  - Firecracker and LXC: 4631
  - Firecracker and gVisor: 2798
  - LXC and gVisor: 65285
Conclusion

- Neither gVisor nor Firecracker are best for all workloads
- Firecracker – High host kernel code footprint
- gVisor – High dependence on kernel functionality
- Optimize code paths

Next Steps
● Expand to other isolation platforms
Questions?

anjali@wisc.edu

slack