Container Security and Sandboxing

Yiying Zhang
Lec6: Security Implications of Containers

TABLE 1. Threat model specifications for apps, containers, and host for the studied use cases. ‘Semi’ refers to semi-honest. Apps in semi-honest/malicious containers can be semi-honest or malicious too.

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Apps can be honest semi malicious</th>
<th>Containers can be honest semi malicious</th>
<th>Host can be honest semi malicious</th>
</tr>
</thead>
<tbody>
<tr>
<td>(I) Protect container from applications</td>
<td>✓ ✓ ✓</td>
<td>✓ - -</td>
<td>✓ - -</td>
</tr>
<tr>
<td>(II) Inter-container protection</td>
<td>✓ ✓ -</td>
<td>✓ ✓ ✓</td>
<td>✓ - -</td>
</tr>
<tr>
<td>(III) Protect host from containers</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
<td>✓ - -</td>
</tr>
<tr>
<td>(IV) Protect containers from host</td>
<td>✓ ✓ -</td>
<td>✓ - -</td>
<td>✓ ✓ ✓</td>
</tr>
</tbody>
</table>

Source: S. Sultan et al.: Container Security: Issues, Challenges, and the Road Ahead
Lec6: Threats of Containers

• Unlike VMs whose interface is hardware instructions, containers’ interface is OS system calls

• More difficult to protect syscalls
  • Involve large amount of code in the OS
  • And there are many syscalls
Lec6: Threats of Container Images

- Difficult to understand the source/provenance of images

Figure 1: Scenario of vulnerability spread

Source: B. Tak et al.: Understanding Security Implications of Using Containers in the Cloud
Secure Pods

Sandboxing workloads in Kubernetes

Tim Allclair  < tallclair@google.com >
Software Engineer, Google
@tallclair
What is a secure pod?
Threats from the outside

Keeping the attackers out:

- Application Security
- Firewall
- Integrity Checks
- Intrusion Detection
- ...

Google
Threats from the *inside*

*Keeping the attackers in.*
How do we protect from internal threats?
Lec7: Sandboxing

- Rule-based sandboxing: reduce the attack surface by restricting what applications can access
  - e.g., AppArmor, SELinux, Seccomp-bpf
- Rules can be fragile (not properly capture threats) and can’t prevent side channel attacks
Sandbox Use Cases

• Sandbox vulnerable code
• Sandbox untrusted code
• Provide max defense in depth
• Sandbox multitenant code
• Sandbox multitenant services
• Mutually untrusted users want to share a cluster
• Sidecar container ha distinct privileges
Sandbox Layer

- Container
  - A subset of containers in a pod
  - Both container and pod isolation
- Pod
  - A set of pots in a sandbox
- Node
- Pros and cons?
Attack Surfaces

➢ Kernel
➢ Storage
➢ Network
➢ Daemons
➢ Logging, monitoring, ...
➢ Hardware
➢ ...

Google
Attack Surfaces

➢ **Kernel**
➢ **Storage**
➢ **Network**
➢ **Daemons**
   *Logging, monitoring, ...*
➢ **Hardware**
➢ **...**
Attacks via the **Kernel**
## Vulnerability Trends Over Time

<table>
<thead>
<tr>
<th>Year</th>
<th># of Vulnerabilities</th>
<th>DoS</th>
<th>Code Execution</th>
<th>Overflow</th>
<th>Memory Corruption</th>
<th>Sql Injection</th>
<th>XSS</th>
<th>Directory Traversal</th>
<th>Http Response Splitting</th>
<th>Bypass something</th>
<th>Gain Information</th>
<th>Gain Privileges</th>
<th>CSRF</th>
<th>File Inclusion</th>
<th># of exploits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1999</td>
<td>19</td>
<td>7</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>5</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>2001</td>
<td>22</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>2002</td>
<td>15</td>
<td>3</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>2003</td>
<td>19</td>
<td>8</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>2004</td>
<td>51</td>
<td>20</td>
<td></td>
<td>5</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>12</td>
</tr>
<tr>
<td>2005</td>
<td>133</td>
<td>90</td>
<td>19</td>
<td>19</td>
<td>1</td>
<td></td>
<td></td>
<td>6</td>
<td>5</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>2006</td>
<td>90</td>
<td>61</td>
<td>5</td>
<td>7</td>
<td>7</td>
<td></td>
<td></td>
<td>2</td>
<td>5</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>2007</td>
<td>62</td>
<td>41</td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>2008</td>
<td>71</td>
<td>43</td>
<td>3</td>
<td>17</td>
<td>4</td>
<td></td>
<td></td>
<td>4</td>
<td>6</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>12</td>
</tr>
<tr>
<td>2009</td>
<td>102</td>
<td>64</td>
<td>2</td>
<td>21</td>
<td>6</td>
<td></td>
<td></td>
<td>7</td>
<td>11</td>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>21</td>
</tr>
<tr>
<td>2010</td>
<td>123</td>
<td>67</td>
<td>3</td>
<td>16</td>
<td>7</td>
<td></td>
<td></td>
<td>7</td>
<td>30</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>14</td>
</tr>
<tr>
<td>2011</td>
<td>83</td>
<td>62</td>
<td>1</td>
<td>21</td>
<td>10</td>
<td></td>
<td></td>
<td>1</td>
<td>21</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>2012</td>
<td>115</td>
<td>83</td>
<td>4</td>
<td>25</td>
<td>10</td>
<td></td>
<td></td>
<td>6</td>
<td>19</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11</td>
</tr>
<tr>
<td>2013</td>
<td>189</td>
<td>101</td>
<td>6</td>
<td>41</td>
<td>13</td>
<td></td>
<td></td>
<td>11</td>
<td>57</td>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>26</td>
</tr>
<tr>
<td>2014</td>
<td>130</td>
<td>88</td>
<td>8</td>
<td>20</td>
<td>9</td>
<td></td>
<td></td>
<td>11</td>
<td>28</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>20</td>
</tr>
<tr>
<td>2015</td>
<td>86</td>
<td>55</td>
<td>6</td>
<td>15</td>
<td>4</td>
<td></td>
<td></td>
<td>11</td>
<td>10</td>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>17</td>
</tr>
<tr>
<td>2016</td>
<td>217</td>
<td>153</td>
<td>5</td>
<td>38</td>
<td>18</td>
<td></td>
<td></td>
<td>12</td>
<td>35</td>
<td>52</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>52</td>
</tr>
<tr>
<td>2017</td>
<td>454</td>
<td>147</td>
<td>169</td>
<td>54</td>
<td>26</td>
<td></td>
<td></td>
<td>1</td>
<td>17</td>
<td>89</td>
<td>36</td>
<td></td>
<td></td>
<td></td>
<td>36</td>
</tr>
<tr>
<td>2018</td>
<td>177</td>
<td>84</td>
<td>3</td>
<td>28</td>
<td>9</td>
<td></td>
<td></td>
<td>4</td>
<td>20</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>2019</td>
<td>170</td>
<td>44</td>
<td>5</td>
<td>25</td>
<td>6</td>
<td></td>
<td></td>
<td>4</td>
<td>18</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>2333</td>
<td>1230</td>
<td>246</td>
<td>373</td>
<td>130</td>
<td></td>
<td></td>
<td>3</td>
<td>116</td>
<td>369</td>
<td>261</td>
<td></td>
<td></td>
<td></td>
<td>29</td>
</tr>
</tbody>
</table>

% Of All

source: https://www.cvedetails.com/product/47/Linux-Linux-Kernel.html?vendor_id=33
### Vulnerabilities Published In 2017

**Execute Code**

<table>
<thead>
<tr>
<th>CVE ID</th>
<th>CVSS Score</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVE-2016-10229</td>
<td>10.0</td>
<td>Exec Code in the Linux kernel before 4.5 allows remote attackers to execute arbitrary code via UDP traffic that triggers an unsafe second checksum calculation during execution of a recv system call with the MSG_PEEK flag.</td>
</tr>
<tr>
<td>CVE-2016-05611</td>
<td>10.0</td>
<td>Exec Code in the Broadcom Wi-Fi firmware could enable a remote attacker to execute arbitrary code within the context of the Wi-Fi SoC. This issue is rated as Critical due to the possibility of remote code execution in the context of the Wi-Fi SoC. Product: Android. Versions: Kernel-3.10, Kernel-3.18. Android ID: A-34195105. References: B-RB-1710814.</td>
</tr>
<tr>
<td>CVE-2017-13715</td>
<td>10.0</td>
<td>DoS Exec Code in the Linux kernel before 4.3 does not ensure that ip_proto, ip_proto, and thoff are initialized, which allows remote attackers to cause a denial of service (system crash) or possibly execute arbitrary code via a single crafted MPLS packet.</td>
</tr>
</tbody>
</table>

**Related Links**

Today: kernel isolation features

```yaml
apiVersion: v1
class: Pod
metadata:
  name: restricted-pod
  annotations:
    seccomp.security.alpha.kubernetes.io/pod: docker/default
    apparmor.security.beta.kubernetes.io/pod: runtime/default
spec:
  securityContext:
    runAsUser: 1234
    runAsNonRoot: true
containers:
  - name: untrusted-container
    image: sketchy:v1
    securityContext:
      allowPrivilegeEscalation: false
```
Today: kernel isolation features

apiVersion: v1
kind: Pod
metadata:
  name: restricted-pod
annotations:
  seccomp.security.alpha.kubernetes.io/pod: docker/default
  apparmor.security.beta.kubernetes.io/pod: runtime/default
spec:
  securityContext:
    runAsUser: 1234
    runAsNonRoot: true
  containers:
    - name: untrusted-container
      image: sketchy:v1
      securityContext:
        allowPrivilegeEscalation: false
Today: kernel isolation features

```yaml
apiVersion: v1
class: Pod
metadata:
  name: restricted-pod
  annotations:
    seccomp.security.alpha.kubernetes.io/pod: docker/default
    apparmor.security.beta.kubernetes.io/pod: runtime/default
spec:
  securityContext:
    runAsUser: 1234
    runAsNonRoot: true
containers:
  - name: untrusted-container
    image: sketchy:v1
    securityContext:
      allowPrivilegeEscalation: false
```
apiVersion: v1
kind: Pod
metadata:
  name: restricted-pod
annotations:
  seccomp.security.alpha.kubernetes.io/pod: docker/default
  apparmor.security.beta.kubernetes.io/pod: runtime/default
spec:
  securityContext:
    runAsUser: 1234
    runAsNonRoot: true
containers:
- name: untrusted-container
  image: sketchy:v1
  securityContext:
    allowPrivilegeEscalation: false
Sandboxes

Node Host

Kubelet

Runtime

Sandboxed Pod

Container

Container

Kernel
Sandboxes - gVisor

Node Host

Kubelet

Runtime

Sandboxed Pod (runsc)

Container

Emulated Kernel

Platform

Kernel

KVM
Future: sandboxed

apiVersion: v1
kind: Pod
metadata:
  name: sandboxed-pod
spec:
  securityContext:
    sandboxed: true
containers:
  - name: untrusted-container
    image: sketchy:v1

API still under discussion
Sandboxes

Node Host

Kubelet

Runtime

Sandboxed Pod

Container

Thwarted!

Kernel

KVM
What's the catch?

1. Cost & Performance
   *But getting better!*

2. Not 100% compatible
   *Close though*

3. Incomplete solution
   *Requires network hardening*
Attack Surfaces

➢ Kernel
➢ Storage
➢ Network
➢ Daemons
➢ Logging, monitoring, ...
➢ Hardware
➢ ...

Google
CVE-2017-1002101: Host-resolved *symlinks*

Node Host

Pod

Container

Volume

Escape!

Kernel
CVE-2017-1002101: Host-resolved symlinks
TODO: Sandboxed storage

1. Readonly storage via **readonly protocols**
2. Ephemeral storage **opaque to host**
3. **Direct access** block volumes
4. Sandboxed persistent filesystems ???
Attack Surfaces

➢ Kernel
➢ Storage
➢ Network
➢ Daemons
  *Logging, monitoring, ...*
➢ Hardware
➢ ...

Image credit: Albarubescens
Attacks over the network

Possible solution: enforce iptables rules at host
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata: { ... }
spec:
  podSelector:
    matchLabels:
      sandboxed: true
  policyTypes:
  - Egress
  egress:
  - to:
    - ipBlock:
      cidr: 0.0.0.0/0
    except:
      - 10.0.0.0/8
      - 172.16.0.0/12
      - 192.168.0.0/16
Attack Surfaces

- Kernel
- Storage
- Network
- Daemons
- **Logging, monitoring, ...**
- Hardware
- ...
Attacks via system logs

Node
Pod (Native)
Container

Pod (Sandboxed)
Container
stdio/stderr
Agent
stdio/stderr
vssock
Runtime

fluentd
Kubelet
log file
Kubelet
logs request

Log aggregator
API server
kubectl logs
Threats Related to Other Daemons

- Kubelet
  - CVE-2018-1002100: Kubectl copy doesn’t check for paths outside of its destination directory
- Container runtime
  - Spoofed container status, fake container stats
  - Processes not specified by the user that run in sandbox
- Kube-proxy
Attack Surfaces

➢ Kernel
➢ Storage
➢ Network
➢ Daemons
➢ Logging, monitoring, ...
➢ Hardware
➢ ...

Image credit: Jonas Löwgren
Attacks via the Hardware
Summary

➢ Kubernetes is a complex system with many layers of attack surfaces exposed to internal threats

➢ Sandboxes is an upcoming feature to mitigate many of those threats
  i. Leverage hypervisor isolation
  ii. Deeper Kubernetes integration for enhanced protection