Towards Improving Container Security by Preventing Runtime Escapes

Michael Reeves, Dave Tian, Antonio Bianchi, and Z. Berkay Celik
What is a container?

- **Container**: Process
  - Kernel Isolation Mechanisms
  - Kernel Security Mechanisms
- **Isolation**: limit container’s access
  - Cgroups
  - Namespaces
- **Security**: limit container abilities
  - Linux Security Modules
  - Seccomp
  - Capabilities

---

Docker

Linux Kernel

Storage
- Device Mapper
- Btrfs
- Aufs

Namespaces
- PID
- NET
- MNT
- User

Networking
- veth
- bridge
- iptables

Cgroups
- CPU
- device
- memory

Security
- Capability
- SELinux
- seccomp
Why do adversaries want to control container hosts?

1. CryptoJacking
2. Corporate Espionage
3. Exfiltrate Customer Data
How do adversaries attack container hosts?

- Misconfiguration (1)
- Kernel Bug (2)
- Runtime Vulnerability (3)

Container != VM
Vulnerabilities => Escape
Escape => Host Code Execution
Related Work: Container Escapes

- Lin et al. (ACSAC):
  - measure kernel exploit container escapes
- Anton (Thesis):
  - measures container escape exploits, only 4 CVEs
- Martin et al. (Computer Communications):
  - measure container configuration escapes
- Flauzac et al. (Procedia Computer Science):
  - limited comparison between technology features
- Allodi et al. (ACM):
  - conducted vulnerability risk assessment
Contributions:

No work has analyzed container runtime vulnerabilities

1. Container security study:
   • 11 runtimes and 59 corresponding container runtime CVEs
   • 28 container runtime CVEs have PoC exploits

2. Container CVE taxonomy:
   • Container runtime PoC: 7-class taxonomy
   • 46% of all container runtime PoC CVEs => 9 different container escape exploits
   • Main cause of escape: host component leaked into the container

3. User namespace defense:
   • prevent exploitation of root privileges on host
   • 7 of 9 exploits prevented
Security Study
Threat Model:

- **System:**
  - Linux Server Container Host

- **Adversary:**
  - Container Code Execution
  - Goal: control host

- **TCB:**
  - Container Configuration
  - Linux Kernel
  - Server Hardware

- **Threat:**
  - Runtime Vulnerability
Runtime Vulnerability: Container Escape Exploit

CVE-2019-5736: runc vulnerability

1. Override Entrypoint
2. At initialization:
   a. EVIL.so loaded by entrypoint
   b. Overwrite container runtime with evil backdoor
3. New container executes evil

Adversary Escapes and executes Code on the host!
Data Collection

1. Build List:
   a. 11 Runtimes

2. Query CVEs:
   a. NVDTOOLS 59 CVEs
3. Filter CVEs:
   a. remove 9 DoS CVEs
   b. focus on CVEs w/ PoC

4. Search PoC: Github/Google
   a. missing PoC (not in NVD or public internet)
Exploit Analyzer Framework:

- binary analysis (low level)
- MITRE ATT&CK (high level)
- steps (low + high level)

Categories:
consider cause and impact

runc: CVE-2019-5736:

1. symlink container entry point to /proc/self/exe
2. save malicious.so with new function to overwrite host runtime engine onto container
3. entry point executes in the container on startup
4. /proc/self/exe executes container_runtime ON CONTAINER
5. host runtime loads the malicious.so
6. .so overwrites container_runtime -> evil
7. new containers will execute evil, code execution achieved
CVE Taxonomy

Key:
C2H: container to host
LHA: limited-host-access
H: host
C: container

Container Runtime PoC Exploit Categories

- C2H-escape: 46%
- H-priv-esc: 21%
- C2H-LHA: 14%
- unpatched-system: 7%
- C-priv-esc: 4%
- C2H-net: 4%
- MAC-bypass: 4%
Escape Exploits

Main Issue:
Unintended host components exposed in container

3 Causes:
1. File Descriptor Mishandling
2. Component Missing Access Control
3. Host Execution in Container Context
Namespace Defense
User Namespaces

Design:
Container User mapped to non-privileged host user

Prevents:
Exploiting root permissions on the host
Prevented: Container Escape Exploit

CVE-2019-5736: runc vulnerability

1. Override Entrypoint
2. At initialization:
   a. EVIL.so loaded by entypoint
   b. FAILS TO OVERRIDE ENTRY

Why?:
Container root user != host root user
No permissions on host to overwrite!
## Defense Effectiveness

<table>
<thead>
<tr>
<th>Exploits Prevented (Failed)</th>
<th>Exploits still Succeed</th>
<th>Effectiveness</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>2</td>
<td>78%</td>
</tr>
</tbody>
</table>
Defense Limitation

Main Limitation:
host runtime utility executes as root, but depends on adversary-controlled container objects

Why?:
CVE- 2019-19921: TOCTOU in runC’s volume mount operation. Occurs before container fully initialized

Solution:
Fix runC software to open volume mount directories as file descriptor rather than a string
Conclusions

• **Security Study:**
  • 11 runtimes and their 59 CVEs.

• **Taxonomy:**
  • 7-classes over the 28 CVEs with publicly available PoC exploits
  • Main cause of container escapes is a host component leaked into the container

• **Defense:**
  • Main idea: prevent exploitation of root privileges on host
  • User namespace prevents 7 of 9 exploits
Questions?
mreeve@sandia.gov
https://mastermjr.com
BACKUP:
12 Container Escape CVES -> 9 PoC Exploits

Container Escape CVEs by Runtime

Container Escape Exploits by Runtime
Namespaces Explained

**PID Namespace**

**Design:**
Isolate the processes of separate containers

**Prevents:**
Observing processes across the whole operating system