Agenda

- Firecracker Design
- Correction of Errors
- Two inglorious bugs
What is Firecracker?
What is Firecracker?

- Lightweight VMM written in Rust
- Multi-tenant cloud workloads (containers/functions)
- Used in production by AWS Lambda
- Open Source

https://github.com/firecracker-microvm/firecracker/
Firecracker Design

Host (x86_64/aarch64)

Host OS (Linux Kernel)

KVM
Firecracker Design - Threads

HTTP Client

UDS

API Server

VMM

Guest vCPUs

KVM

Host OS (Linux Kernel)
Firecracker Design - Devices

- Host OS (Linux Kernel)
- API Server
- HTTP Client
- UDS
- KVM
- File
- Tap
- Guest vCPUs
- MMIO
- Virtio Block
- Virtio Net
- PIO
- Serial
- i8042
- VMM
Firecracker Design - Security

- **Host OS (Linux Kernel)**
  - API Server
    - HTTP Client
      - UDS
    - File
    - Tap
  - KVM
- **VMM**
  - Virtio Block
  - Virtio Net
  - Serial
  - i8042
  - PIO
  - MMIO
- **Guest vCPUs**
- **Security Boundaries**
  - Virtualization
  - Jailer + Seccomp

**Components**
- VMM
- MMIO
- PIO
- VIRTIO
- PIO
- Serial
- i8042
- Virtio Block
- Virtio Net
- File
- Tap
Firecracker Properties

- Boot Time ~125ms*
- Low memory overhead ~3 MiB*
- Oversubscription CPU & Memory

* workload & configuration dependent; check out https://github.com/firecracker-microvm/firecracker/blob/master/SPECIFICATION.md
“You are destined to fail.”

L. David Marquet, “Turn the Ship Around”
- Correction of Errors
- Understand the root cause
- 5 whys
- Take corrective actions & prevent same kind of mistakes

Inglorious ... Release
MADvise?

---

Problem:

- Firecracker intermittently exits with error code 128
- SYS_MADVISE is not whitelisted

Impact: Customers are unable to update Firecracker

Fix: Whitelist SYS_MADVISE

Affected Versions: v0.15.0, v0.15.1
Firecracker - Seccomp

---

- Seccomp Filters:
  - None
  - Basic
  - Advanced (Default)
- Whitelist Approach
- ~30 whitelisted syscalls
- Seccomp Action:
  - Trap -> update metrics, log errors, exit with error code
Firecracker - Development

GitHub

Firecracker-v0.14.0

PR #42

Reviewer 1

Reviewer 2

Firecracker CI

~ 20 mins
Firecracker - Development

GitHub

PR #42

Reviewer 1

Reviewer 2

Firecracker-v0.14.0

master

Firecracker CI

~ 20 mins
Firecracker - Development

GitHub

Firecracker-v0.14.0

PR #42
merge ❤

master
Firecracker - Development

GitHub

master

Firecracker-v0.14.0  Firecracker-v0.15.0
Firecracker - Development

GitHub

Firecracker-v0.14.0

master

Firecracker-v0.15.0

AWS Lambda
Firecracker - Development

GitHub

Firecracker-v0.14.0

Firecracker-v0.15.0

master

Lambda CI

AWS Lambda
Firecracker - Development

GitHub

Firecracker-v0.14.0
Firecracker-v0.15.0

master

AWS Lambda

Lambda CI

Production
Firecracker - Development
The Whys

- Why wasn’t SYS_madvise whitelisted?
  - V0.15.0 included an update to Rust 1.32
  - Changed out of memory handling in Rust runtime
- Why didn’t we catch it in Firecracker CI?
  - Compromise between CI time & coverage
  - Syscall triggered by specific workloads
All good, right?
Wrong....

Andreea Florescu @SgAnd... · 09 Mar
Doing a patch release to fix a patch release to fix a broken release.
syscalls: actually whitelist madvise for musl

Signed-off-by: Alexandru Agache <aagch@amazon.com>

Commit: d29c8d7a94ee9421a14c3e976112761ce4f6ab3e

alexandruag authored and andreaflorescu committed on Mar 9
Inglorious Releases
Corrective Actions

---

- Add long running tests
- Improve seccomp
  - Whitelist vs Blacklist
  - Auto-generate seccomp whitelist? -> 60 syscalls
  - Still discussing: [https://github.com/firecracker-microvm/firecracker/issues/1177](https://github.com/firecracker-microvm/firecracker/issues/1177)
Lessons Learned

- Testing, testing, testing!
- Use workloads as close as possible to production
- Logs and metrics saved the day (and engineering time)
“Math is hard.”
Pesky Plus Sign
“+” vs “overflowing_add()”

---

**Problem:** Unchecked arithmetic in memory model code

**Potential Impact:** Abrupt termination of guest OS

**Fix:** Checked arithmetic

**Affected Versions:** < v0.12.0
Anatomy of a read()

The process of reading data from a file in a virtual environment involves several components and interactions. This diagram illustrates the flow from the guest user space to the guest kernel and then to the Firecracker virtio blk device driver.

1. **Guest**:
   - **prog**: The user space program initiates the read() function.
   - **kernel**: This is where the kernel processes the read() call.

2. **Firecracker**:
   - **event**: An event is triggered, possibly indicating the need to read from a file.
   - **virtio blk device**: This component is responsible for handling virtio blk device operations.
   - **File (emulated blk device)**: The file system manages the emulated blk device.
   - **memory model**: This model manages the data flow internally.
   - **read_to_memory(offset, count)**: The function used to read data from memory.
   - **Guest memory**: The guest's memory where the read data is stored.

3. **Data Flow**:
   - The read() function in the guest program invokes the kernel's read() function.
   - The kernel then calls the virtio blk device driver.
   - The virtio blk device driver communicates with the Firecracker's file system and memory model.
   - Data is read from memory to the guest's memory, with `offset+count` specifying the start and end of the data read.

This diagram provides a clear visual representation of how data is read in a virtualized environment, highlighting the key components and their interactions.
Anatomy of a read() - Silent Failure

Guest
-prog

userspace | kernel

Firecracker

virtio blk device

virtio blk driver

event

File (emulated blk device)

memory model

read_to_memory(offset, count)

Guest memory

data

read()
Anatomy of a read() - Panic

Guest

prog  read()  virtio blk driver

userspace  kernel

Firecracker

virtio blk device

File (emulated blk device)

memory model

read_to_memory(offset, count)

Guest memory
Rust Panic

- Expected problems: Result type, error propagation
- Unexpected problems: panic
  - Unwind: affects panicking thread, recoverable
  - Abort: SIGABRT, affects all threads, unrecoverable
Rust Panic

- Expected problems: Result type, error propagation
- Unexpected problems: panic
  - Unwind: affects panicking thread, recoverable
  - Abort: SIGABRT, affects all threads, irrecoverable
- Panic hook
  - Flush metrics
  - Say goodbye
The Problem

- **Caller**: virtio device code
- **Faulty driver**...
The Problem

- Caller: virtio device code
- Faulty driver...

Debug build:

thread 'fc_vmm' panicked at 'attempt to add with overflow'

Very, very bad!
The Problem

- Caller: virtio device code
- Faulty driver...

Release build:
- No symptoms until malfunction
  Worse!
The Solution

- Checked arithmetic: Rust standard
- Turns a hidden panic condition into a gracefully handled Result
- Faulty driver...
The Solution

- Checked arithmetic: Rust standard
- Turns a hidden panic condition into a gracefully handled Result
- Faulty driver:
  - Logged error message
  - Incremented error metric
The Whys

- Why were there no overflow checks in place?
  - Hidden error condition
  - Code unchanged since #1
- Why didn’t we catch it in Firecracker CI?
  - Community report, community fix
  - Drivers in CI images didn’t trigger it
  - CI didn’t lint Rust code
rust-clippy

---

- Rust code linter, available as cargo subcommand
- `clippy::integer_arithmetic`
- >200 warnings at the time this issue was fixed
  - Correctness, Restriction, Style and more
Corrective Actions

- cargo clippy test in Firecracker CI
  - Warnings treated as errors
  - Find and fix obscure error conditions
  - Improve overall code quality
- Replace panic conditions with error propagation
  - unwrap, expect
- Roadmap: virtio device input fuzzing
Lessons Learned

- Testing, testing, testing!
- Linting, linting, linting!
- The Rust compiler is strict, but doesn’t protect from everything
Conclusions

---

- Seccomp is hard
- Math is hard

The problem is not the problem, but your attitude about the problem.

Capt. Jack Sparrow
Thank you!

Andreea Florescu
fandree@amazon.com

Alexandra Iordache
aghecen@amazon.com