SEV-Step

A Single-Stepping Framework for AMD-SEV

Luca Wilke, Jan Wichelmann, Anja Rabich and Thomas Eisenbarth
Single-Stepping Attacks

Idea, Exploits, Root Cause
Single-Stepping: Idea

normal

frequently interrupted

single-stepped
Single-Stepping: Attack Avenues

1. Interrupt latency
   [CCS'18, USENIX'21]

2. Interrupt counting
   [CCS'19, CHES'20-21, USENIX'20]

3. Zero-step replaying
   [USENIX'18, CCS'19, ISCA'19, S&P'21]

4. Amplification
   [ATC'17, CCS'19/21, CHES'17-19, S&P'20-21, USENIX'17/18/22]

CC-BY Jo van Bulck
Can SEV VMs be single-stepped?
Single-Stepping: Mechanism

When do we exit?
- Intercepted Instruction
- Page Fault
- External Interrupt
Single-Stepping: Mechanism

Host / Hypervisor

Setup

VMRUN

Handle exit

VM

Instr 1

Instr 2

Instr 3

APIC Timer Interrupt
Single-Stepping: Root Cause
Single-Stepping: Naive Implementation

Zero Step  Single Step  Multi Step

Setup Timer  VMRUN  Instr 1  Instr 2  Instr 3
Single-Stepping: APIC Timing on SGX

APIC timer oneshot TSC distribution

Measurements from AEX-Notify paper
Single-Stepping: Enlarge the Window

Flush Instr 1 | Setup Timer | VMRUN | Instr 1 | Instr 2 | Instr 3

Fetch -> Decode -> Issue -> Retire

Requires Instruction Page
Single-Stepping: Enlarge the Window

Flush Instr 1 | Setup Timer | VMRUN
---|---|---
Instr 1 | Instr 2 | Instr 3

Fetch -> Decode -> Issue -> Retire

Slow
SEV-Step

Design Goals, Implementation, Ongoing work
SEV-Step: Design Goals

Reusability

- Separate primitives from attack logic

Interactivity

- Bidirectional Communication
SEV-Step: Reusability

Implement everything here?

Linux / KVM (Hypervisor)
- Setup
  - VMRUN
    - Handle exit
  - VM
    - Instr 1
    - Instr 2
    - Instr 3
SEV-Step: Reusability

SEV-Step Library

- Rich API / Complex Attack Logic

Linux / KVM (Hypervisor)

- Setup
- IOCTL API
- VMRUN
- Handle exit

VM

- Instr 1
- Instr 2
- Instr 3

Host User Space

Host Kernel Space

Data Flow

Control Flow
SEV-Step: Design Goals

Reusability

- Separate primitives from attack logic

Interactivity

- Bidirectional Communication
SEV-Step: Interactivity?

- **SEV-Step Library**
  - Rich API / Complex Attack Logic

- **Linux / KVM (Hypervisor)**
  - Setup
  - VMRUN
  - Handle exit

- **IOCTL API**

- **Host User Space**

- **Host Kernel Space**

- **VM**
  - Instr 1
  - Instr 2
  - Instr 3

**Data Flow**

**Control Flow**
SEV-Step: Interactivity

SEV-Step Library
- Initialize Config
- Wait for Event
- Process Event
- (Adapt Config)
- Ack Event

Linux / KVM (Hypervisor)
- Setup
- IOCTL API
- VMRUN
- Handle exit
  - Event?
  - Send & Wait

VM
- Instr 1
- Instr 2
- Instr 3

Data Flow
Control Flow

Host User Space
Host Kernel Space
VM
SEV-Step: Interactivity

SEV-Step Library
- Initialize Config
- Wait for Event
- Process Event
- (Adapt Config)
- Ack Event

Linux / KVM (Hypervisor)
- IOCTL API
- Setup
- VMRUN
- Handle exit
- Event?
- Send & Wait

Shared Memory

Host User Space

Host Kernel Space

VM
- Instr 1
- Instr 2
- Instr 3

Data Flow
Control Flow
SEV-Step: Design Goals

Reusability ✅
- Separate primitives from attack logic

Interactivity ✅
- Bidirectional Communication
SEV-Step: Ongoing Work

• Improve API
  • Current Design: Track/Untrack, Start Stepping/Stop Stepping
  • Goal: High Level Components: „Track Pagefault Sequence“
  • Model SEV-Step + SGX-Step as „drivers“
Summary

• Single-Stepping enables many attacks
• Popularized for SGX with SGX-Step
• SEV-Step Framework
  • First to show that SEV is vulnerable to single-stepping
  • Ease attack research on SEV
  • GPL V2
• Ongoing work
  • Improve attack prototyping
• Play with it, break it, Improve it