Overview of the Intel SMI Transfer Monitor (STM) on Dasharo Firmware

Brian Delgado

December 7, 2023

My Background

- Red Team Lead for the Programmable Solutions Group at Intel Corp
- Research interests:
  - The intersection of security and performance
  - Supervisor operation modes (Intel System Management Mode, Intel SMI Transfer Monitor)
  - Virtualization
  - Firmware Fuzzing
  - Rootkit detection

Overview

- Short SMM / STM intro recap
- Process of enabling STM on Dasharo firmware
- Example of using STM protections
- Discussion

SMM Overview

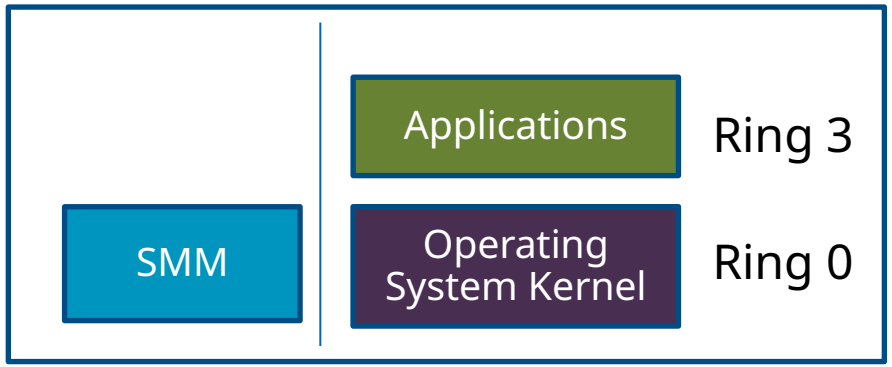
Firmware is widely present on computing devices from PC's, phones, to FPGAs

On x86 platforms, runtime firmware is executed in System Management Mode (SMM)

SMM can be entered transparently to the operating system / hypervisor via an "SMI" (System Management Interrupt)

Upon SMI, CPU threads move from applications/OS/VMM into SMM

SMM is highly privileged, has traditionally had full access to CPU register state, memory, and devices.



Holistic View for SMM Security

Architectural

Principle of least privilege:

- SMM Page table isolation
- Intel Runtime BIOS Resilience
- Intel System Resources Defense
- **STM**

Attack Surface Reduction

Evaluate whether SMM modules need to be in SMM

Secure Coding

Leverage CommBuffer, ASLR, Guard pages, SMM Code Access Check, etc …

Firmware Bill of Materials

Do codebases / images have known vulnerabilities?

Config

Is BIOS setting the appropriate lock bits?

Run CHIPSEC to check.

Fuzzing / Static Analysis

Use testing tools to identify vulnerable code.

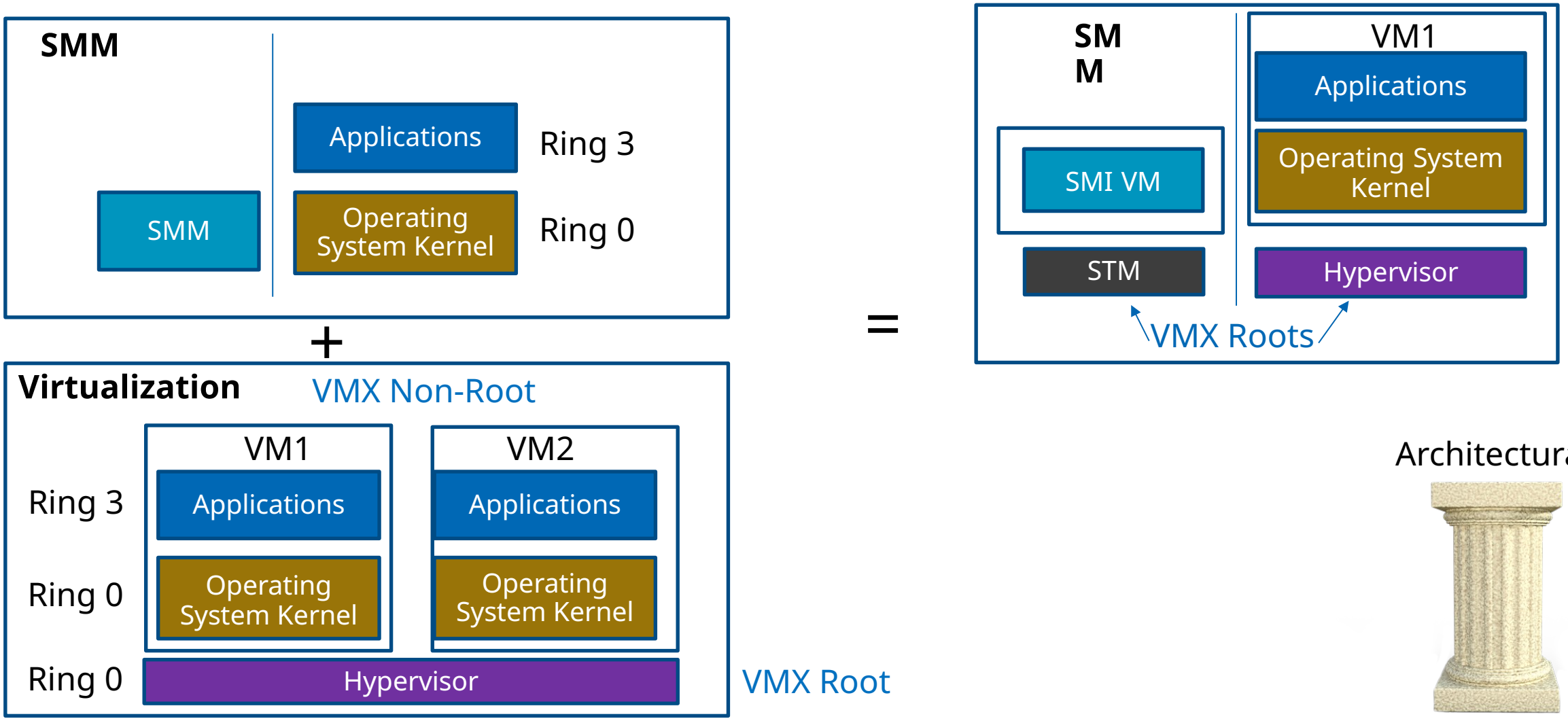
Fuzzing tools from Intel:

- **TSFFS (Coming up in Rowan's talk!)**
- HBFA
- Chipsec

What is the STM?

- Method of applying a protection policy over SMI handler accesses
  - Reduces risk of malicious / buggy / mis-used firmware code
- What protections can the STM provide?
  - Limit SMI handler access to OS/VMM/App:
    - Memory/MMIO, MSRs, IO Ports, PCI devices
- Policy is configurable/dynamic based on requirements
- STM is open-source and enabled in coreboot

SMM + Virtualization = STM



Architectural

Enabling the STM on coreboot

- Much groundwork done in coreboot thanks to STM-PE (Eugene Myers)
- The key areas that needed attention for testing on Dasharo w/ MSI Z690-A:

STM Provisioning

- Building the STM binary for coreboot (Some differences from original Intel version, size is larger)
- Creating appropriate-sized TSEG, MSEG, etc regions
- Ensuring adequate firmware volume space (CBFS) to fit STM binary. Thanks to Dasharo for the resolution!

STM-enabled BIOS Boot

- Resolving SMM memory alignment issue (coreboot 4.22.01 has necessary fix for: get_save_state calculation

STM Launch

- STM resource descriptor adjustment (patch pending)

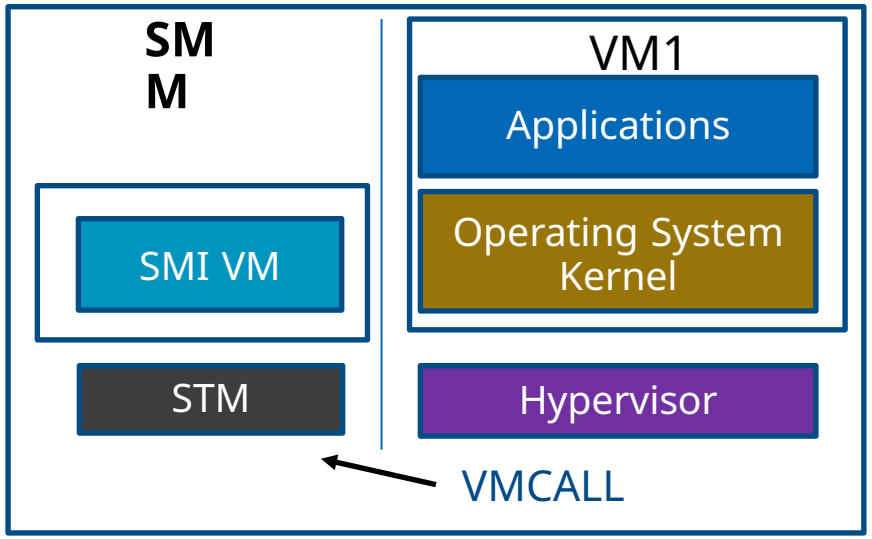
Tools that Helped Enabling

- Serial console capability on MSI board with RTE to debug BIOS boot (Be sure to increase the “Default Console Log Level” in coreboot config to get all messages.)
  - Connect two jumper wires between RTE and MSI board following Dasharo instructions,
  - ssh to RTE
  - Start minicom on RTE (e.g. /dev/TTYS1, Hardware Flow Control On)
- PCI-E serial card on MSI board to get STM serial logs (specify STM serial port address, e.g. 0x2000 in coreboot. Connect serial cable between USB serial on laptop -> PCI-E serial)
- BIOS FlashBack button (very helpful to recover from non-bootable BIOS builds)

STM Launch

```
(STM) TxtProcessorSmmDescriptor: 0x000000004B12AB00
(STM) TxtProcessorSmmDescriptor     - 4B12AB00
(STM)   Signature                   - 4749535350545854
(STM)   Size                        - 0089
(STM)   SmmDescriptorVerMajor       - 01
(STM)   SmmDescriptorVerMinor       - 00
(STM)   LocalApicId                 - 00000000
(STM)   ExecutionDisableOutsideSmrr - 00
(STM)   Intel64Mode                 - 00
(STM)   Cr4Pae                      - 00
(STM)   Cr4Pse                      - 00
(STM)   SmramToVmcsRestoreRequired  - 00
(STM)   ReinitializeVmcsRequired    - 00
(STM)   DomainType                  - 00
(STM)   XStatePolicy                - 00
(STM)   EptEnabled                  - 00
(STM)   SmmCs                       - 0008
(STM)   SmmDs                       - 0010
(STM)   SmmSs                       - 0010
(STM)   SmmOtherSegment             - 0010
(STM)   SmmTr                       - 0020
(STM)   SmmCr3                      - 0000000000000000
(STM)   SmmStmSetupRip              - 0000000000000000
(STM)   SmmStmTeardownRip           - 0000000000000000
(STM)   SmmSmiHandlerRip            - 000000004B12304C
                ...
```

STM Init outputs



Once the STM is configured by the firmware, the host-side hypervisor needs to turn on the STM via a sequence of VMCALLs

Launchers for Linux/Xen are available

STM Demonstration

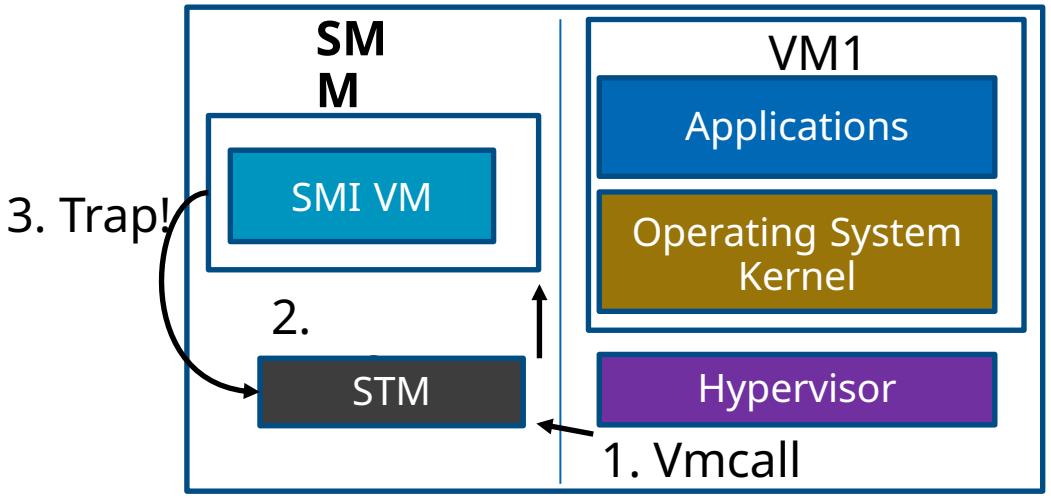
Example: OS/VMM requests protection of MSR_IA32_MISC_ENABLE. Normally, SMM code is free to modify this MSR.

```
MsrDesc.Hdr.RscType = MACHINE_SPECIFIC_REG;
MsrDesc.Hdr.Length = sizeof (STM_RSC_MSR_DESC);
MsrDesc.Hdr.ReturnStatus = 0;
MsrDesc.Hdr.Reserved = 0;
MsrDesc.Hdr.IgnoreResource = 0;
MsrDesc.MsrIndex = MSR_IA32_MISC_ENABLE;      /* 0x1A0 */
MsrDesc.KernelModeProcessing = 0;
MsrDesc.Reserved = 0;
MsrDesc.WriteMask = (uint64_t) - 1;
MsrDesc.ReadMask = 0;
```

Sample protection request

To evaluate: Modified SMI handler to attempt modification of MSR 0x1A0 upon write of specific value to port 0xb2 by kernel module

STM intercepted the write and noted: `(STM) WRMSR (1A0) violation!`



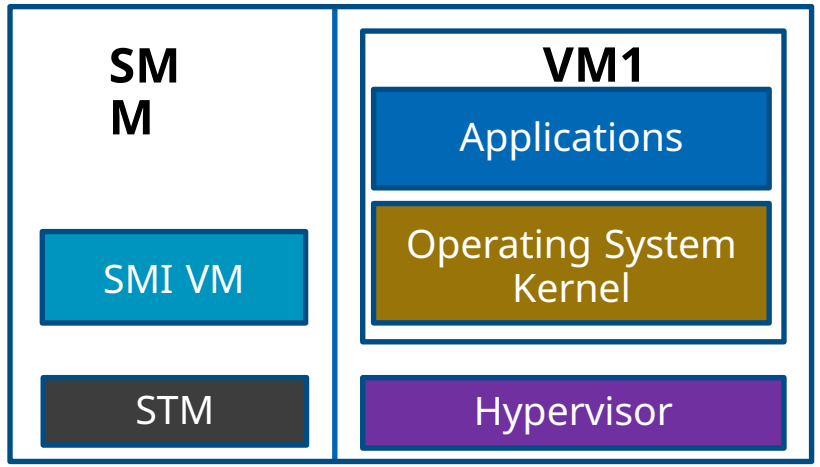
1. Protect Resources Vmcall

2. STM configures MSR Bitmap for SMI VM to preclude write access.

3. SMI handler attempt to write to a protected resources traps to STM.

Conclusion / Questions / Discussion

- STM provides enhanced defenses against buggy or vulnerable SMI handlers
  - Open-source / inspectable SMM monitor
  - Leverages Intel VTx to apply customizable protection policy
    - Helps enable:
      - Principle of least privilege
      - Defense in depth for runtime SMI code
- Dasharo firmware provides a useful open-source testing/demonstration vehicle for this capability



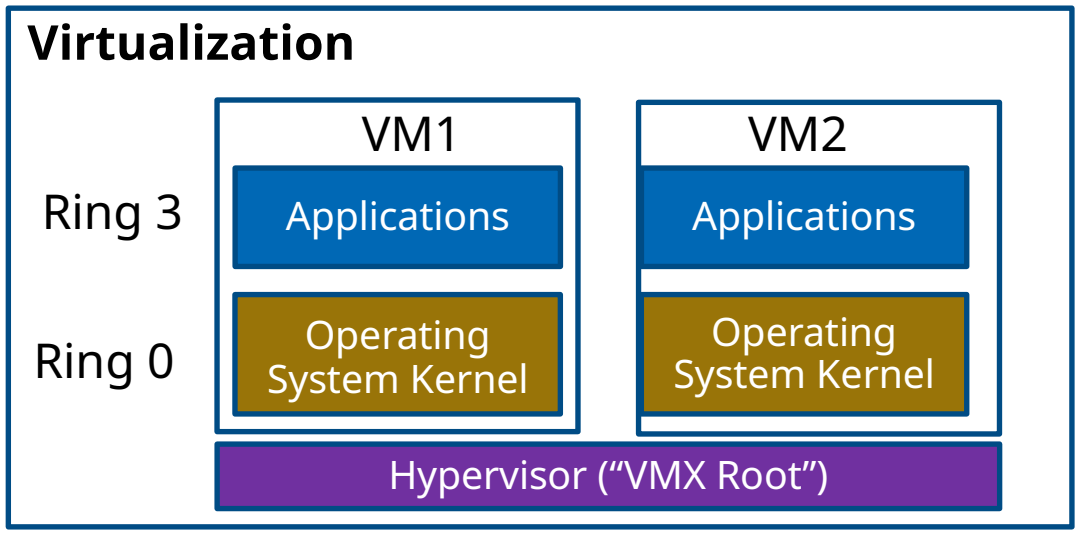
Also check out STM-PE extension for protected-execution in STM VMs

Additional Resources

- SMM Protection in EDK2: Jiewen Yao - SMM Protection in EDKII_Intel (uefi.org)
- STM User Guide: https://www.intel.com/content/dam/develop/external/us/en/documents/stm-user-guide-001-819978.pdf
- STM Code: jyao1/STM (github.com)
- Community-developed STM launcher code: Xen, Linux
- Applying the Principle of Least Privilege to System Management Interrupt Handlers with the Intel SMI Transfer Monitor (HASP 2020)
- Intel Hardware Shield: DRTM-based-computing_whitepaper_FINAL_MAY2021.pdf

Backup

Virtualization Overview (VTx)



Hypervisor maintains Virtual Machine Control Structure (VMCS) to isolate VM CPU contexts and set control info

Enhanced Page Tables (EPT) used to isolate memory ranges of VMs

Can apply permission restrictions via setting bits (e.g. **MSR / IO bitmap**) in VMCS and customizing **EPT** to allow/disallow memory accesses

Notice

“Intel provides these materials as-is, with no express or implied warranties. All products, dates, and figures specified are preliminary, based on current expectations, and are subject to change without notice. The products described might contain design defects or errors known as errata, which might cause the product to deviate from published specifications. Current, characterized errata are available on request. Intel technologies might require enabled hardware, software, or service activation. Some results have been estimated or simulated. Your costs and results might vary. No product or component can be absolutely secure.