PROTECTING VM REGISTER STATE WITH AMD SEV-ES
DAVID KAPLAN
LSS 2017
BACKGROUND-- HARDWARE MEMORY ENCRYPTION

**AMD Secure Memory Encryption (SME) / AMD Secure Encrypted Virtualization (SEV)**

- Hardware AES engine located in the memory controller performs inline encryption/decryption of DRAM
- Minimal performance impact
  - Extra latency only taken for encrypted pages
- No application changes required
- Encryption keys are managed by the AMD Secure Processor and are hardware isolated
  - not known to any software on the CPU

*Defense against unauthorized access to memory*
BACKGROUND - SECURE ENCRYPTED VIRTUALIZATION (SEV)

- Protects VMs/Containers from each other, administrator tampering, and untrusted Hypervisor
- One key for Hypervisor and one key per VM, groups of VMs, or VM/Sandbox with multiple containers
- Cryptographically isolates the hypervisor from the guest VMs
- Integrates with existing AMD-V technology
- System can also run unsecure VMs

Enhances isolation of VMs

Traditional Virtualization

Hypervisor

Guest

AMD Secure Encrypted Virtualization

Hypervisor

Guest

Guest

Guest

Applications

Container

Sandbox/VM

AES-128 Engine

DRAM
UPDATES SINCE LSS 2016

⚠️ Hardware is available!
- Ryzen/ThreadRipper support TSME/SME only
- EPYC supports SEV as well
- Demo of SEV in action: https://youtu.be/qgiUuTmXyGs (just search for “amd security”)

⚠️ Linux support underway
- OVMF (BIOS) patches accepted 7-10-2017 (https://github.com/tianocore/edk2/commits?author=codomania)
- SME Linux kernel patches accepted 7-18-2017 (likely to be included in 4.14)
- SEV Linux kernel patches under RFC

⚠️ Please help review patches!
SOME PERFORMANCE DATA (SME)

Estimate based on data collected on config:
- OS Ubuntu 16.04 running stock kernel: 4.10
- BIOS WDL7628N, release Date: 06/26/2017
- EPYC 2.2 GHz fixed frequency, SMT on
- Host memory 512GB @ 2667MHz, 64GB per socket for host.
- Compiled with GCC 6.1

More information about SPEC CPU ® 2006 can be found at [http://www.spec.org](http://www.spec.org)
Estimate based on data collected on config:
• OS Ubuntu 16.04 running stock kernel: 4.10
• BIOS WDL7628N, release Date: 06/26/2017
• EPYC 2.2 GHz fixed frequency, SMT on
• Host memory 512GB @ 2667MHz, 64GB per socket for host.
• Compiled with GCC 6.1
• KVM/QEMU

SEV kernel: https://github.com/AMDESE/AMDSEV

More information about SPEC CPU ® 2006 can be found at http://www.spec.org
STREAM PERFORMANCE

Host system setup
OS Ubuntu 16.04 running kernel: 4.13.0-rc1-sev-rfc-3-2
BIOS WDL7628N, release Date: 06/26/2017
EPYC Silicon at 2.2 GHz fixed frequency, SMT on
Host memory 512GB @ 2667MHz, 64GB per socket for host.

Guest config
SW config same as host
HW: 4 vcpus, 95% of 20GB memory
   qemu process bound to node 0 and die 0 cpus
   each vcpu thread taskset to a unique core of die 0

<table>
<thead>
<tr>
<th></th>
<th>Host SME</th>
<th>Guest NoSEV</th>
<th>Guest SEV</th>
</tr>
</thead>
<tbody>
<tr>
<td>COPY</td>
<td>-0.17</td>
<td>-1.45</td>
<td>-1.37</td>
</tr>
<tr>
<td>SCALE</td>
<td>-0.15</td>
<td>-1.45</td>
<td>-1.44</td>
</tr>
<tr>
<td>ADD</td>
<td>-0.48</td>
<td>-2.01</td>
<td>-1.90</td>
</tr>
<tr>
<td>TRIADD</td>
<td>-0.45</td>
<td>-1.98</td>
<td>-1.92</td>
</tr>
</tbody>
</table>

STREAM 4T (% OF THEORETICAL MAX)
SEV-ES (Encrypted State) provides additional VM security on top of AMD SEV memory encryption.

SEV protects guest memory using memory encryption.

SEV-ES protects guest register state:
- Register state is encrypted using guest memory encryption key.
- Only guest is allowed to modify its register state.
- Register state is integrity protected to prevent rollback attacks.

New architectural features allow guests to selectively allow HV access to state when needed for VM emulation purposes.
THREAT MODEL

⚠️ Like SEV, SEV-ES requires trust in the HW, AMD Secure Processor, and guest VM

⚠️ In SEV-ES, the HV is less trusted than SEV as it is only able to do the following
  - Run guest VMs (SEV/SEV-ES do not protect against DOS)
  - Manage memory allocation (maintain nested tables)
  - Inject interrupts/exceptions into guest
  - Emulate devices/services as requested by the guest

⚠️ In particular, SEV-ES protects against attacks such as
  - Exfiltration (HV observing guest register state during exits)
  - Control flow (modifying guest register state to change control flow)
World switches now swap ALL register state
- Includes all segment registers, GPRs, FPU state (see Table B-4 in APM Vol2)
- All register state is encrypted with the guest encryption key
- Integrity value is calculated and stored in a protected page

The guest is notified by a new exception (#VC) when certain events occur
- The guest decides what state (if any) to share with the HV
- The guest invokes the HV to perform the required tasks
- The guest updates its state based on the output from the HV

The guest and HV use a special structure to communicate
- Guest-Hypervisor Communication Block (GHCB)
- Location set by guest, mapped as unencrypted memory page
**TYPES OF EXITS**

- **Automatic Exits (AE)**
  - Events that occur asynchronously to the guest (e.g. interrupts)
  - Events that do not require exposing guest state (e.g. HLT)
  - Nested page faults not due to MMIO emulation
  - AE events save all state and exit to HV
  - Only action HV can do is just resume the guest w/o modifications

- **Non-Automatic Exits (NAE)**
  - All other exit events
  - NAE events cause a #VC instead of a VMEXIT
  - Guest handler may invoke the HV via VMGEXIT instruction

---

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Notes</th>
<th>HW Advances</th>
<th>RIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>52h</td>
<td>VMEXIT_MC</td>
<td>Machine check exception</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>60h</td>
<td>VMEXIT_INTR</td>
<td>Physical INTR</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>61h</td>
<td>VMEXIT_NMI</td>
<td>Physical NMI</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>63h</td>
<td>VMEXIT_INIT</td>
<td>Physical INIT</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>64h</td>
<td>VMEXIT_VINTR</td>
<td>Virtual INTR</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>77h</td>
<td>VMEXIT_PAUSE</td>
<td>PAUSE instruction</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>78h</td>
<td>VMEXIT_HLT</td>
<td>HLT instruction</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>7fh</td>
<td>VMEXIT_SHUTDOWN</td>
<td>Shutdown</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>8fh</td>
<td>VMEXIT_EFER_WRITE_TRAP</td>
<td>Write to EFER</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>90h-9fh</td>
<td>VMEXIT_CR[0-15]_WRITE_TRAP</td>
<td>Write to Crx</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>400h</td>
<td>VMEXIT_NPF</td>
<td>Only if PFCODE[3]=0 (no reserved bit error)</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>403h</td>
<td>VMEXIT_VMEXIT</td>
<td>VMEXIT instruction</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>VMEXIT_INVALID</td>
<td>Invalid guest state</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>
NAE FLOW EXAMPLE

**CPUID**

**Guest**
- Guest triggers VMEXIT condition
- #VC handler modifies state
- #VC handler copies state to GHCB

**CPU HW**
- Send #VC to guest
- Save/encrypt guest state
- Load HV state
- HV handles exit
- VMRUN

**Hypervisor**
- Read GHCB => see CPUID_EXIT
- Read RAX, emulate CPUID
- Write RAX/RBX/RCX/RDX to GHCB

**Copy**
- Copy RAX to GHCB
- Copy RAX/RBX/RCX/RDX to register state

Error Code => CPUID

Write CPUID_EXIT to GHCB

Guest triggers VMEXIT condition

Read RAX, emulate CPUID

Load/decrypt guest state

#VC handler modifies state

IRET

Send #VC to guest

Save/encrypt guest state

Load HV state
To facilitate HV/OS interoperability, AMD is working on defining a GHCB format/contract
- GHCB layout will mirror the VMCB layout
- Guest OS is expected to supply certain values on certain exceptions (e.g. RDMSR requires RCX)

GHCB specification is in development, will be open for comments shortly

A new MSR defines the location of the GHCB, value is per-guest
- On boot, this MSR will contain information about the SEV configuration
- Once ready, the guest will write the MSR with the guest physical address of the GHCB
MMIO

- SEV-ES assumes that MMIO pages are marked with a reserved bit set in the nested tables
  - This is what KVM does today
  - Other page faults (e.g. not present) are handled as AEs

- A guest d-side access that encounters a reserved page fault throws a #VC
  - Guest #VC handler must read RIP and determine what access is required
  - Guest #VC handler calls HV to read/write MMIO bytes as required
  - Hypervisor does not crack/emulate instruction since guest #VC handler does this
#VC HANDLER OPTIMIZATIONS

- The #VC handler may be used to reduce total world switches needed

- Example: Avoid VMEXIT for static values
  - After first CPUID, remember results and use them in the future
  - Avoids user programs from taking CPUID VMEXITs

- Example: Fine-grained MMIO traps
  - #VC handler checks page offset and decides if it merits a VMEXIT
  - Could allow for write coalescing (group many MMIO updates into one VMEXIT)
  - Optimize MMIO reads with static results
First priority is to finalize GHCB software format/conventions

KVM
- Support for atomic world switch
- Read/write register values from GHCB instead of VMCB
- Support for new exits (e.g. read/write MMIO)
- Call to AMD Secure Processor to initialize/measure initial VMCB state

(Guest) Kernel
- New #VC exception handler
- Instruction cracking for #VC handler
Whitepapers
- SEV-ES: http://support.amd.com/TechDocs/Protecting%20VM%20Register%20State%20with%20SEV-ES.pdf
- SEV: http://developer.amd.com/wordpress/media/2013/12/AMD_Memory_Encryption_Whitepaper_v7-Public.pdf

Technical Documentation
  - SEV: Section 15.34
  - SEV-ES: Section 15.35

Code
- GitHub: https://github.com/AMDESE/AMDSEV
EPYC PLATFORM ECOSYSTEM
DISCLAIMER & ATTRIBUTION

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

ATTRIBUTION

© 2015 Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo and combinations thereof are trademarks of Advanced Micro Devices, Inc. in the United States and/or other jurisdictions. Other names are for informational purposes only and may be trademarks of their respective owners.