

AMD Memory Encryption

João Gabriel Trombeta

João Paulo Taylor Ienczak Zanette

Ranieri Schroeder Althoff

9 de Abril de 2018

Máquinas Virtuais

Máquina Virtual (VM)

Emuladores de computadores lógicos.

Guest: O computador executado pela VM;

Host: Computador que oferece recursos para a VM.

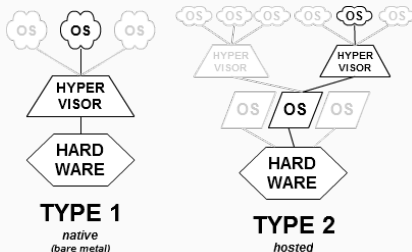
Máquinas Virtuais — Definições

Hypervisor

“Monitor de Máquina Virtual”: cria, executa e gerencia uma ou mais VMs.

Bare Metal: quando executa a VM diretamente no *hardware*.

Hosted: quando executa *software* sobre o SO do *Host*.



Retirado de: <https://en.wikipedia.org/wiki/Hypervisor>

Falhas em Hypervisors

Xen Hypervisor

- Um dos *Hypervisors* mais populares atualmente;
- *Bare-Metal*;
- *Open-Source*.



Retirado de: <http://www-archive.xenproject.org/products/xenhyp.html>

Descrição da falha

- Conforme descrito em [1], é possível fazer leitura/escrita de código arbitrário no espaço de endereçamento do *Guest*;
- Descobrimos a localização da **tabela de hypercall**, é possível executar uma rotina como *Host* (fazendo **escape de máquina virtual**).

Tabela de *Hypercall*

```
ENTRY(hypercalltable)
    .quad do_set_trap_table /* 0 */
    .quad do_mmu_update
    .quad do_set_gdt
    .quad do_stack_switch
    .quad do_set_callbacks
    .quad do_fpu_task_switch /* 5 */
    // skip...
    .quad do_ni_hypercall /* reservedforXenClient */
    .quad do_xenpmu_op /* 40 */
    .rept __HYPERVISOR_arch_0((hypercall_table)/8)
    .quad do_ni_hypercall
    .endr
    .quad do_mca /* 48 */
    .quad paging_domctl_continuation
    .rept NR_hypercalls((hypercalltable)/8)
    .quad do_ni_hypercall
    .endr
```


Procedimento para explorar a falha

1. Criar a situação de acesso irrestrito à memória do *Guest*;
2. Varrer as páginas do *Guest*, fazendo *checksum* do conteúdo de cada uma;
3. Localizar a tabela de *hypercall* através da tabela de argumentos de *hypercall*.
4. Sobrescrever o endereço de um dos *hypercalls* para uma rotina maliciosa;
5. Ativar o evento para o *hypercall* alterado.

AMD Memory Encryption

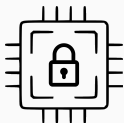
Motivação

Tornar mais seguro o uso de máquinas virtuais através do Secure Memory Encryption (SME) e Secure Encrypted Virtualization (SEV).

AMD Secure Processor

Processador dedicado à criptografia.

- Antigo PSP (Platform Security Processor);
- Independente;
- Executa um *kernel* confiável e de código fechado;
- Isola procedimentos de segurança;
- Memória dedicada;
- Acesso ao CCP (Coprocessador criptográfico).



Security Memory Encryption

- Criptografa e descriptografa dados escritos na RAM;
- Utiliza uma única chave simétrica;
- *Engine* de criptografia *On-chip*;
- Criptografia entre SO/*hypervisor* e RAM;

AMD Memory Encryption — Security Memory Encryption

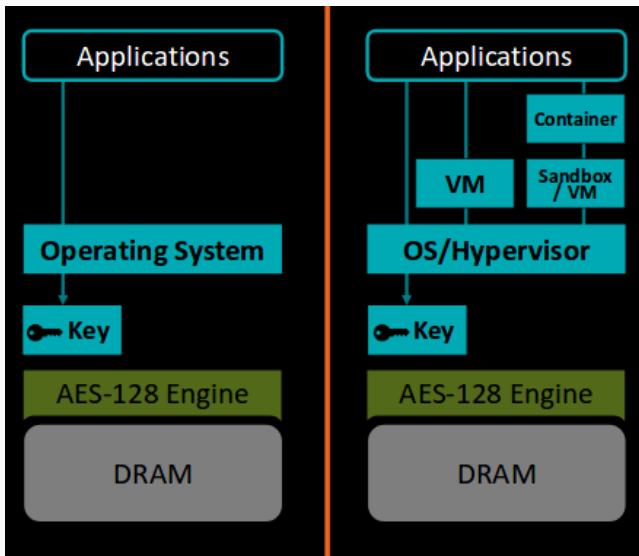


Figura 1: Método de criptografia do SME.

Requisitos

- Suporte em *hardware* por parte do processador (verificável pela chamada de CUID Fn8000_001F);
- Habilitar o recurso deixando o bit 23 do MSR (Model-Specific Register) em 1;
- O último bit mais significativo dos endereços (C-Bit) passa a sinalizar se o dado deve ou não ser criptografado/descriptografado.

AMD Memory Encryption — Security Memory Encryption

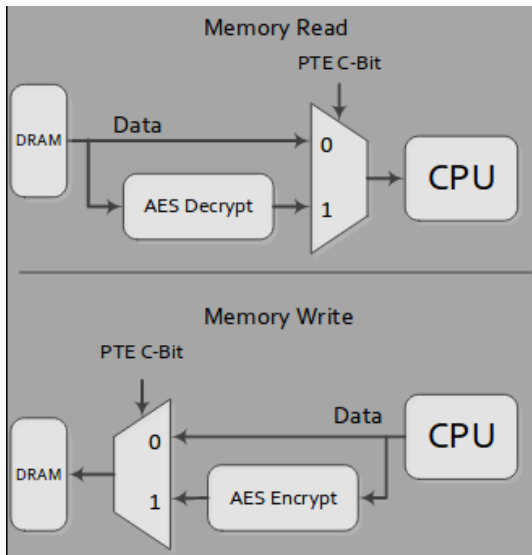


Figura 2: Leitura e escrita no SME.

Transparent SME

Variação do SME em que tudo é criptografado.

- Não necessita suporte pelo SO.

Secure Encrypted Virtualization

Integra o SME com a capacidade de comportar várias VMs criptografadas.

- *Host* possui uma chave;
- Cada VM possui uma chave própria;
- Cada VM é protegida separadamente.

AMD Memory Encryption — Secure Encrypted Virtualization

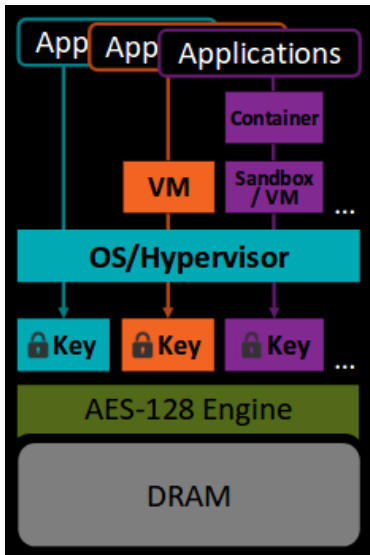


Figura 3: Demonstração do acesso á memória através de chaves.

AMD Memory Encryption — Secure Encrypted Virtualization

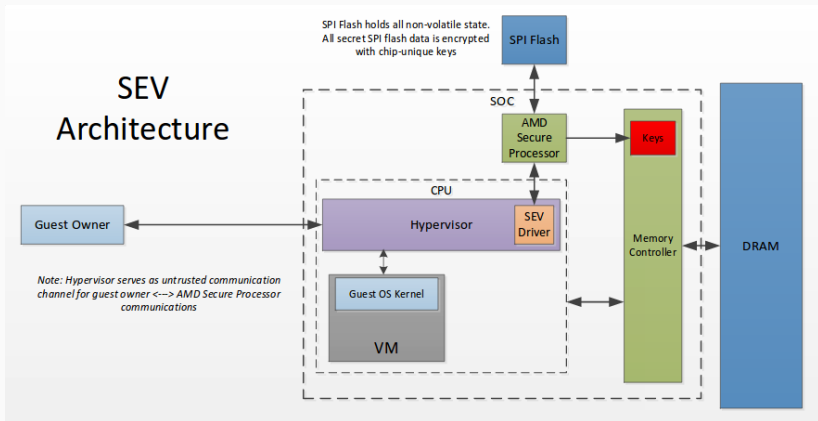


Figura 4: Visão geral da arquitetura do SEV.

Address Space ID

Identifica qual chave pertence a qual VM.

- Utilizado como parte do endereço para *tag* na TLB;
- Auxilia na eficiência permitindo que dados de mais de uma VM permaneça na TLB ao mesmo tempo.

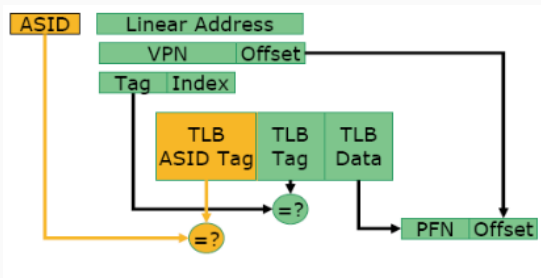


Figura 5: Comportamento da TLB mediante ASID.

AMD Memory Encryption — Secure Encrypted Virtualization

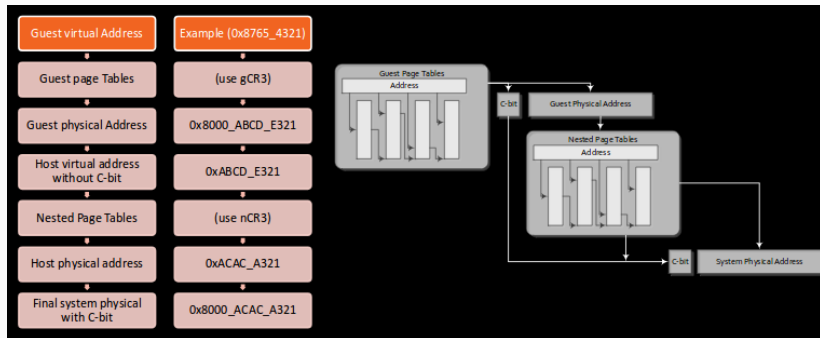


Figura 6: Tradução de endereços do SEV.

Conflito de criptografia

Pode ocorrer de o *Host* e o *Guest* tentarem criptografar a mesma página. Nesse caso, a preferência é utilizar a chave do *Guest*.

Proteção contra falha de segurança

SEV pode oferecer proteção contra a falha apresentada no Xen Hypervisor:

1. Os *Checksums* serão feitos por um *Guest* com acesso irrestrito de leitura/escrita em sua memória;
2. As leituras das páginas serão descriptografadas usando a chave do *Guest*;
3. A região das *hypercalls* está criptografada usando a chave do *Host*;
4. Logo, as leituras das *hypercalls* será feita erroneamente e portanto não serão identificadas pela assinatura esperada.

Utilização do SME e SEV

Set-up

1. Comandos são efetuados pelo *Command buffer*;
2. Endereço do *Command buffer* precisa ser definido nos registradores `CmdBufAddr_{Hi,Lo}`;
3. O *Driver* altera o *Command buffer* habilitando os campos de bits necessários para descrição do comando;
4. O AMD-SP executa o comando;
5. *Command buffer* é atualizado com a resposta do comando.

Emissão de um comando

31	30-26	25-16	15-1	0
0	--	ID	--	IR

ID: ID do comando a ser emitido;

IR: Habilitar interrupções ao finalizar comando.

Resposta da AMD-SP

31	30-16	15-0
1	--	ST

ST: Código de status.

Comandos

- INIT;
- SHUTDOWN;
- PLATFORM_RESET;
- PLATFORM_STATUS;
- PEK_GEN;
- PEK_CSR;
- PEK_CERT_IMPORT;
- PDH_GEN;
- PDH_CERT_EXPORT.

Utilização do SME e SEV — Execução de comandos

Status	Código
SUCCESS	0000h
INVALID_PLATFORM_STATE	0001h
INVALID_GUEST_STATE	0002h
INVALID_CONFIG	0003h
INVALID_LENGTH	0004h
ALREADY_OWNED	0005h
INVALID_CERTIFICATE	0006h
POLICY_FAILURE	0007h
INACTIVE	0008h
INVALID_ADDRESS	0009h
BAD_SIGNATURE	000Ah
BAD_MEASUREMENT	000Bh
ASID_OWNED	000Ch
INVALID_ASID	000Dh
WBINVD_REQUIRED	000Eh
DFFLUSH_REQUIRED	0009h
INVALID_GUEST	0010h
INVALID_COMMAND	0011h
ACTIVE	0012h
HWERROR_PLATFORM	0013h
HWERROR_UNSAFE	0014h
UNSUPPORTED	0015h
INVALID_PARAM	0016h

Tabela 1: Relação dos códigos de status dados pelo AMD-SP. Mais detalhes sobre a tabela estão em [2]



Comentários finais

Possíveis falhas

Apesar dos esforços, a implementação atual do SEV ainda possui falhas de segurança.

- A mesma chave fica associada a uma VM durante toda a existência dela;
- A implementação de *Nested Pages* pode ser explorada para trocar informações entre *Guests* aproveitando que o *Host* consegue descriptografá-los.

Referências

-  S. Luan, “Exploit two Xen Hypervisor vulnerabilities.”
-  I. Advanced Micro Devices, “Secure Encrypted Virtualization API version 0.14,” tech. rep.
-  D. Kaplan, “AMD x86 Memory Encryption technologies.”
-  “AMD Memory Encryption tutorial,” tech. rep.
-  Z.-H. Du, Z. Ying, Z. Ma, Y. Mai, P. Wang, J. Liu, and J. Fang, “Secure Encrypted Virtualization is unsecure!,” tech. rep.
-  T. W. David Kaplan, Jeremy Powell, “AMD memory encryption,” tech. rep.
-  I. Advanced Micro Devices, “AMD-V nested paging,” tech. rep.