

A gente descobre depois

João Gabriel Trombeta
João Paulo Taylor Ienczak Zanette
Ranieri Schroeder Althoff

8 de Abril de 2018

Conteúdo

1	Máquinas Virtuais	2
1.1	Sobre Máquinas Virtuais	2
1.2	Hypervisor	2
1.3	Falhas em Hypervisors	2
1.3.1	Xen Hypervisor	2
2	AMD Memory Encryption	4
2.1	Security Memory Encryption	4
2.2	Secure Encrypted Virtualization	4
2.3	Aplicação do SME e SEV	4
2.4	4

Capítulo 1

Máquinas Virtuais

1.1 Sobre Máquinas Virtuais

De maneira sucinta, máquinas virtuais (VM — *Virtual Machines*) são computadores sendo executados por outros computadores. Chama-se de **Guest** a máquina virtual em si, e de **Host** o *hardware* que oferece recursos para executar a VM. No host uma camada de software chamada hypervisor permite a execução de múltiplas máquinas virtuais em uma única máquina física, independentes e cada uma executando seu próprio SO.

1.2 Hypervisor

Também chamado de Monitor de Máquina Virtual (VMM — *Virtual Machine Monitor*), um **Hypervisor** é um componente (seja *hardware*, *software* ou *firmware*) responsável por criar e executar uma VM, sendo o *Host* o computador em que o Hypervisor é executado.

O Hypervisor é responsável pela camada de abstração entre o host e os guests, realizando o gerenciamento de recursos, uma vez que cada guest trabalha na ilusão de que todos os recursos de hardware são seus. Cada VM deve ser isolada para evitar que uma possa comprometer o funcionamento de outra, por isso toda interação com o meio físico é intermediada pelo Hypervisor, que é fortemente protegido das VM.

1.3 Falhas em Hypervisors

1.3.1 Xen Hypervisor

Uma falha de segurança detectada com relação a Hypervisors foi explorada no Xen Hypervisor (criado pelo Xen Project, composto por membros da The Linux Foundation), em que é possível chamar uma função arbitrária alterando a tabela de *Hypercalls* (semelhante a uma *vtable*). Uma *Hypercall* é *software trap*

do Hypervisor para executar operações privilegiadas (como atualizar tabelas de página).

Para explorar a falha, primeiramente é necessário descobrir a localização da tabela de *Hypercalls*. Para isso, deve-se procurar pela assinatura da página. Porém, como a página não possui um formato tão previsível, é difícil de localizá-la (o que é feito pelo *checksum* do conteúdo da página). Em compensação, a tabela de argumentos dos *Hypercalls* possui um formato previsível, já que seu conteúdo — que é o número de argumentos de cada *Hypercall* — é fixo, e portanto seu *checksum* também é previsível. Além disso, a tabela de argumentos sempre se encontra na página seguinte à tabela de *Hypercalls*, e portanto, ao encontrar uma, se tem a localização da outra. Aliado à possibilidade de leitura e escrita de código arbitrário, feito através de falhas nas regras de verificação de segurança de escrita em páginas do *Hypervision*, é possível então efetuar escape de máquina virtual (i.e. acessar recursos do *Host* que não pertencem à máquina virtual).

1.3.2 VMware ESXi e KVM

Capítulo 2

AMD Memory Encryption

2.1 AMD Secure Processor

Grande parte das funções de criptografia executadas em um AMD utilizam um processador dedicado e independente, o AMD Secure Processor (AMD-SP, antigamente chamado de Platform Security Processor), que garante que componentes sensíveis à segurança não recebam interferência do software dos processadores principais.

O AMD-SP roda um kernel seguro de código fechado, que pode executar tarefas do sistema assim como tarefas de terceiros confiáveis, tendo o administrador controle sobre quais tarefas de terceiros são designadas ao processador. Além disso, o Secure Processor possui uma SRAM dedicada e acesso direto ao CCP, que é composto por um gerador de números aleatórios, várias engines para o processamento de algoritmos de criptografia, e um bloco para o armazenamento de chaves.

2.2 Security Memory Encryption

O AMD Secure Memory Encryption (SME) é um mecanismo que pode ser utilizado para criptografar os dados que vão para a DRAM, com a finalidade de evitar ataques físicos. Durante o boot, o AMD Secure Processor gera uma chave que será usada para criptografar e descriptografar os dados que transitam pela DRAM. Como a engine de criptografia está dentro do chip o impacto das operações é pequeno.

A figura ?? mostra como ocorre a criptografia. A engine é posicionada entre o OS/Hypervisor, onde dada a chave o dado é criptografado antes de ser salvo na DRAM. Ao passar da DRAM para o SO novamente, os dados são descriptografados.

Esse tipo de segurança não previne ataques advindos de um Hypervisor comprometido, uma vez que ele possui acesso aos dados de maneira direta, esse tipo de segurança tem com o objetivo evitar ataques como **probe attack na**

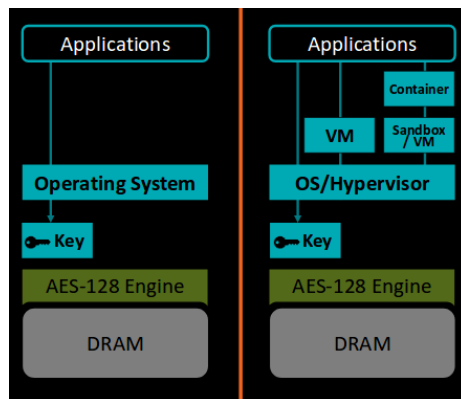


Figura 2.1: Método de criptografia do SME

DRAM, instalação de hardware que possa acessar a memória do guest, ataques que possam capturar dados de DIMM e NVDIMM.

Para utilizar SME, é necessário verificar se o processador possui suporte para esse recurso, o que pode ser verificado através da chamada de CPUID Fn8000_001F, e que durante o boot o bit 23 de SYSCFG MSR esteja definido como 1 para sinalizar que esse recurso está habilitado. Após isso, ao fazer acesso à DRAM, é visto o último bit mais significativo do endereço, chamado de C-bit, que define se o dado deve ou não ser criptografado.

Figura 2.2: Leitura e escrita no SME

Como visto em ??, para a leitura, antes do dado passar para CPU, duas versões do dado são inseridas como entrada de um mux, um com o dado como estava na DRAM e outra com o dado após passar pelo circuito responsável pela criptografia. O controle do mux recebe o bit mais significativo do endereço, o C-bit, caso seja 1 significa que o dado está criptografado e a CPU precisa da informação descriptografada, caso seja 0 significa que o dado pode ser passado direto para a CPU.

Para a escrita a lógica é a mesma, caso o C-bit seja 1 o dado deve ser criptografado antes de ser inserido na DRAM, caso seja 0 o dado pode ser salvo diretamente.

Ainda existe uma variação chamada Transparent SME, onde tudo é criptografado. Nesse caso não é necessário suporte do SO, tendo em vista que não é preciso fazer o controle de quais endereços serão criptografados e quais não. O processo de acesso à memória ocorre da mesma forma que em SME.

2.3 Secure Encrypted Virtualization

2.4 Aplicação do SME e SEV

2.5

Bibliografia

- [1] H. Chen, X. Jia, and H. Li, “A brief introduction to iot gateway,” in *IET International Conference on Communication Technology and Application (ICCTA 2011)*, pp. 610–613, Oct 2011.
- [2] T. N. Stack, “Prototyping.” <https://thenewstack.io/tutorial-prototyping-a-sensor-node-and-iot-gateway-with-arduino-and-raspberry-pi-part-1/>. [Online; acessado em 7 de abril de 2018].
- [3] A. Botta, W. de Donato, V. Persico, and A. Pescapé, “Integration of cloud computing and internet of things: A survey,” *Future Generation Computer Systems*, vol. 56, pp. 684 – 700, 2016.