UEFI driver rootkit with a bare metal hypervisor

Antoine Jouan <antoine.jouan@lse.epita.fr>
Alex Levigoureux <alex.levigoureux@lse.epita.fr>

Why this project ?

What is UEFI ?

- **U**nited **E**xtensible **F**irmware **I**nterface
- Interface between your **operating system** and your **hardware** during the **boot process**
- Aims to replace the legacy **BIOS**
- Many features like network support, filesystem support, etc…
- Comes with a super cool feature : **UEFI drivers**

UEFI binary types

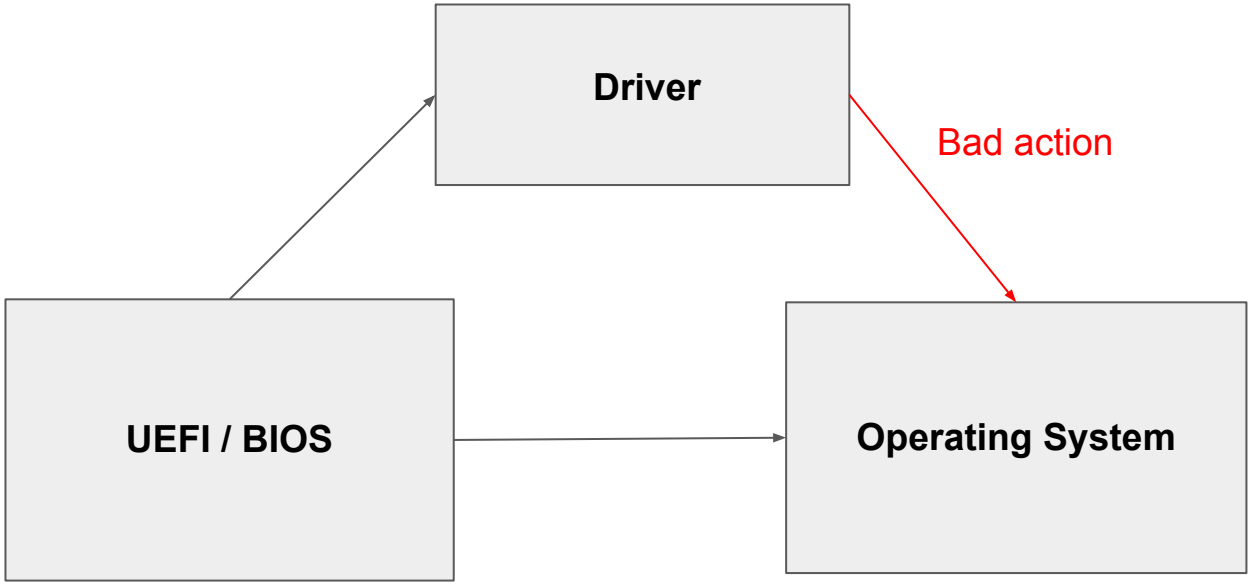
Type	Subsystem code
Application	0xA
Boot service drivers	0xB
Runtime drivers	0xC

Runtime drivers **are not freed** after calling the function **ExitBootServices()**

UEFI drivers

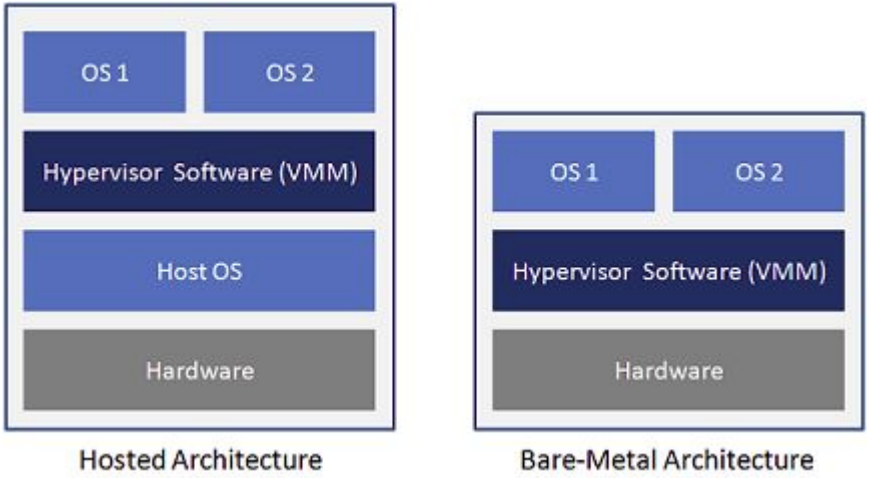
- Ran during the boot process and some can continue to live after
- In most of cases, drivers are placed into EFI partition
- UEFI drivers are basically a PE32+ image without symbol tables
- To develop our driver, we used GNU-EFI

How UEFI bootkits generally works



Hypervisor

- Creates and runs a virtual machine
- Two types of hypervisor : Bare-metal and hosted



Why use a Baremetal Hypervisor ?

As the man intel says

> There is no software-visible bit whose setting indicates whether a logical processor is in VMX non-root operation. This fact may allow a VMM to prevent guest software from determining that it is running in a virtual machine.

Hypervisor can use processor extensions :

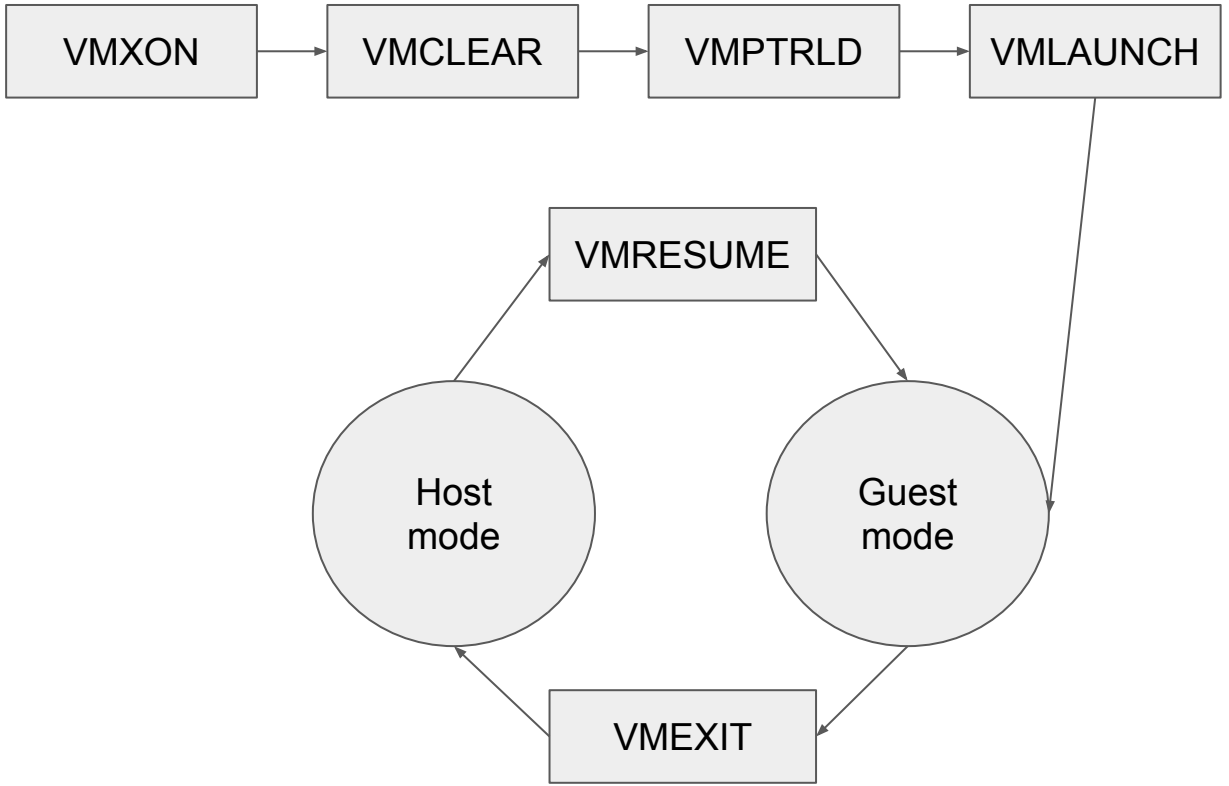
- Intel VT-X
- AMD-V
- ARM7-a and ARM8-a
- RISC-V hypervisor extension

Intel Virtual Machine eXtensions (VMX)

Intel VMX instructions set

VMXON/VMXOFF	Enter/exit VMX operation
VMPTRLD	Load vmcs pointer
VMPTRST	Store vmcs pointer
VMCLEAR	Initialize a vmcs region
VMWRITE / VMREAD	Write/read data in the vmcs region
VMLAUNCH	Launch the current VMCS
VMRESUME	Resume the state of the current VMCS
VMCALL	Call from guest into hypervisor

VMX states


VMXON
VMCLEAR
VMPTRLD
VMLAUNCH
VMRESUME
Host mode
Guest mode
VMEXIT

How to enable VMX

- Check if VMX is supported
- Set bits in CR4 : bit 13 (vmx enable) bit 0 (lock bit)
- Set fixed bits in CR0 and CR4 by reading vmx model specific registers
- Allocate VMXON region (aligned 4KByte page) and write the VMCS revision identifier to the first 30 bits
- Execute VMXON with VMXON region pointer

How to setup VMCS region ?

How to setup VMCS region?

Setup VMCS region

- 4 Kbyte page aligned
- VMCS region store the state of the guest and the host after VMEXIT
- Load using the instruction VMPTRLD

How to setup VMCS region

- Setup GDT/IDT for both
- Setup important registers (RIP, RSP, CR0, CR3, CR4, segment registers, …)
- Setup control fields
- Setup a lot of MSR (IA32_DEBUGCTL, IA32_SYSENTER_EIP, …)
- Possibility to setup EPT and IO bitmap
- Possibility to configure a virtual APIC

How can we use VMX for a Malware ?

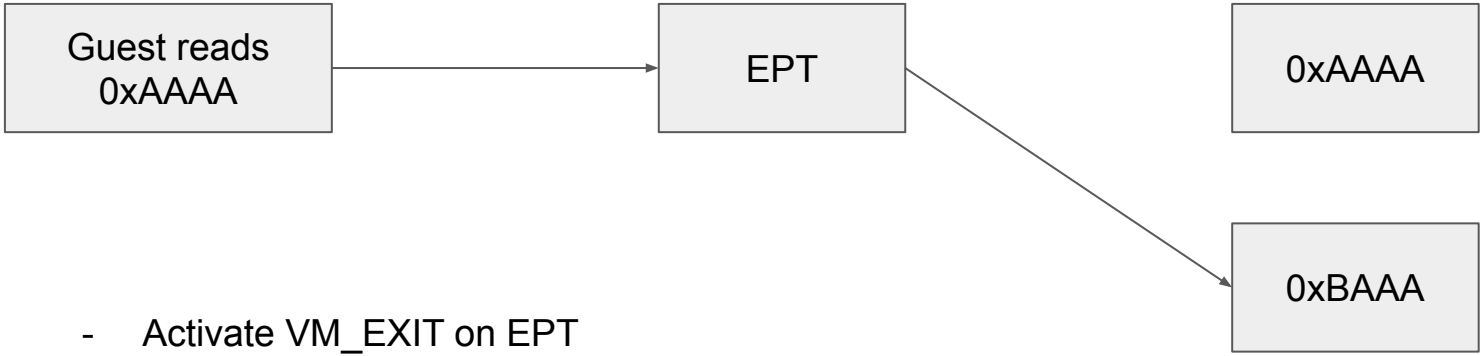
How it works ?

- The host is running with higher privilege and can have access to all the memory
- Possibility to hook function using EPT without triggering OS protection (such as PatchGuard for Windows)
- The bootkit is OS independent

Example of EPT hooking

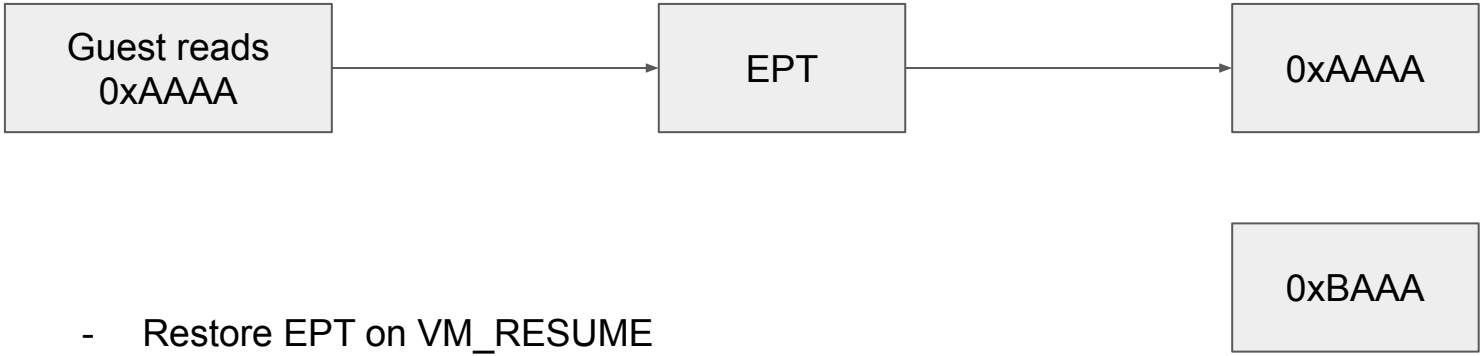
How to intercept memory reading and writing ?

Example of EPT hooking



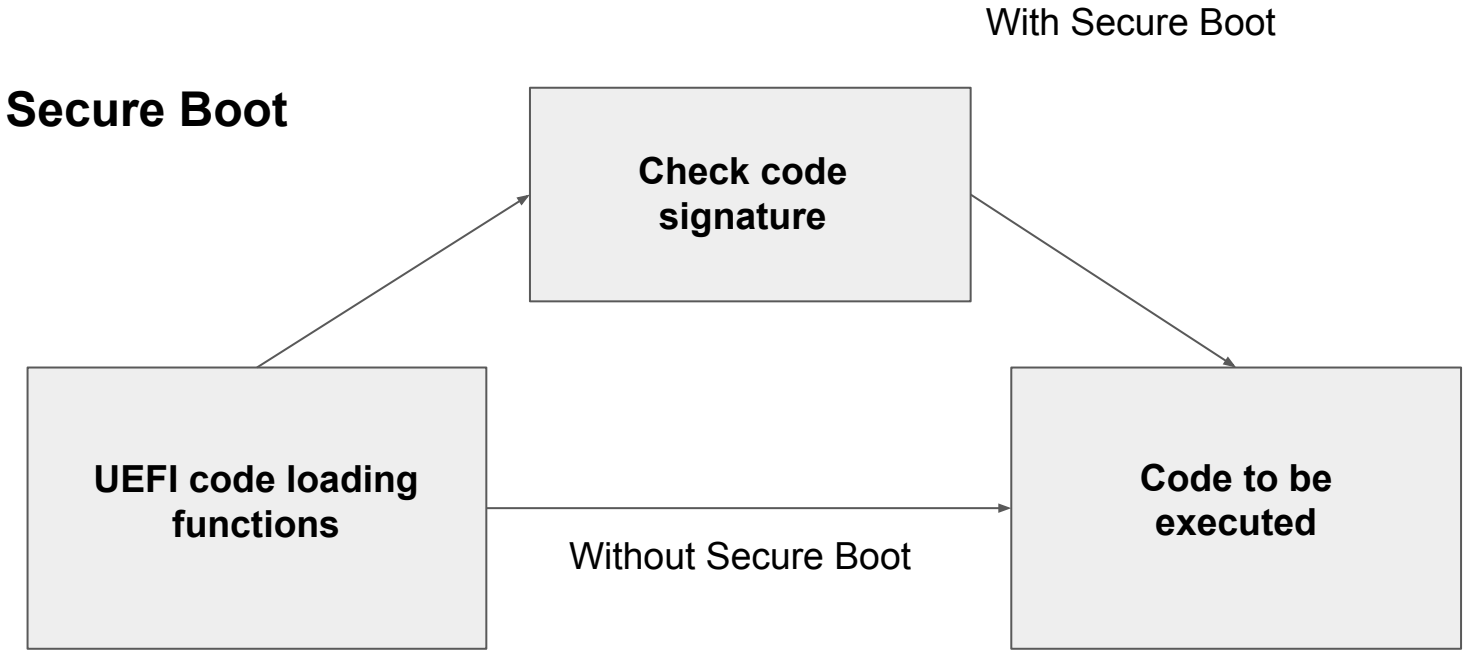
- Activate VM_EXIT on EPT violation
- 0xBAAA is executable only

Example of EPT hooking



- Restore EPT on VM_RESUME

UEFI driver limitations



UEFI driver limitations

- Secure Boot can also write files in whitelist or blacklist
- A function in UEFI will check if the requested file's hash is on a whitelist or a blacklist

How to bypass UEFI protections ?

Find a 0 day :)

- ThinkPwn 2016
- BootHole 2020
- Deactivate the SecureBoot in the bios settings of the target

Any questions ?

Useful links

- https://www.ssi.gouv.fr/uploads/IMG/pdf/uefi-pci-bootkits_sstic_article_fr.pdf
- https://edk2-docs.gitbook.io/edk-ii-uefi-driver-writer-s-guide/
- https://www.blackhat.com/presentations/bh-usa-08/Bailey/BH_US_08_Bailey_Winning_the_Race_to_Bare_Metal_White_Paper.pdf
- https://nixhacker.com/developing-hypervisior-from-scratch-part-1/
- https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sdm.html
- https://www.blackhat.com/docs/asia-17/materials/asia-17-Matrosov-The-UEFI-Firmware-Rootkits-Myths-And-Reality.pdf
- https://github.com/Gbps/gbhv (cool project)