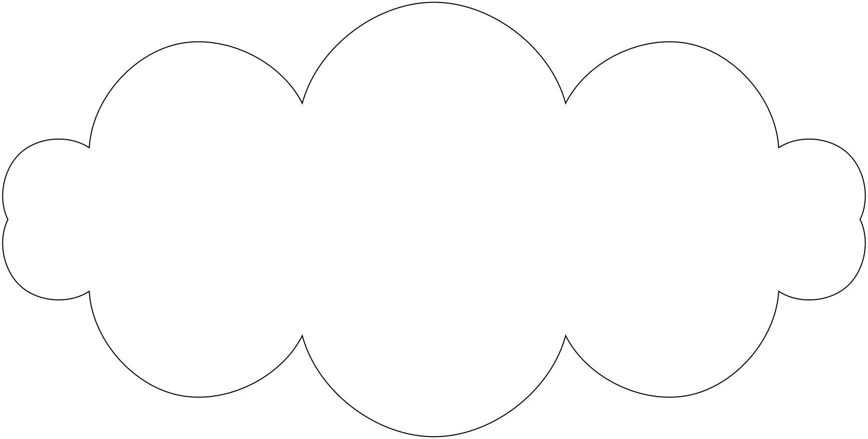
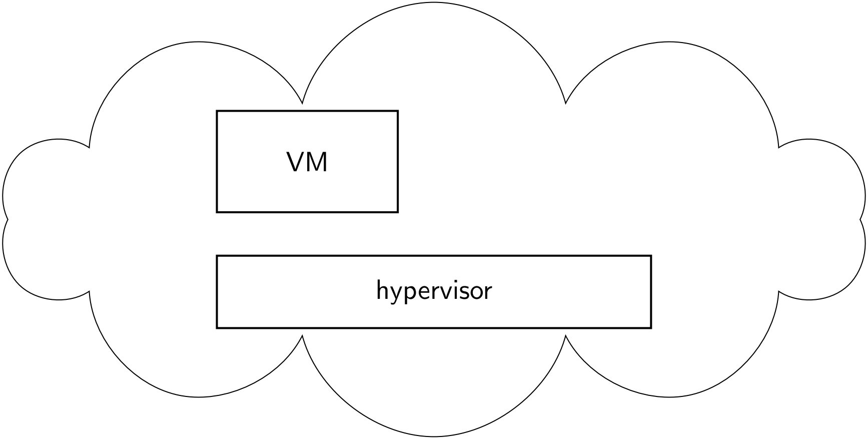
SEVerity: Code Injection Attacks against Encrypted Virtual Machines

Mathias Morbitzer, Sergej Proskurin, Martin Radev, Marko Dorfhuber and Erick Quintanar Salas, 27th May 2021

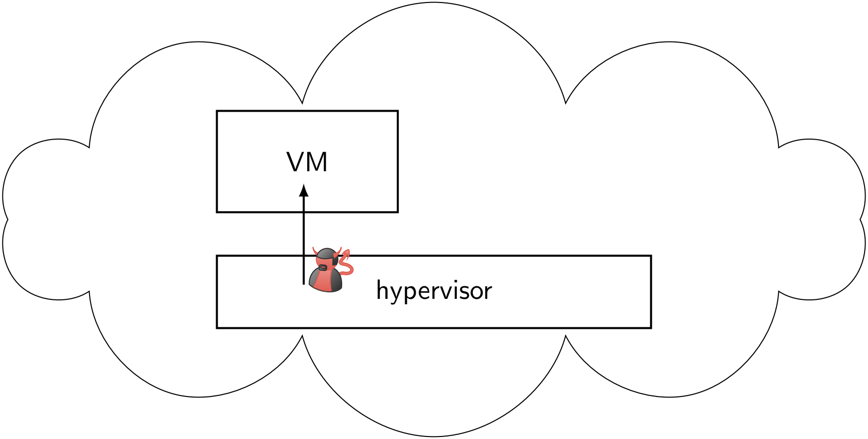
AMD SEV



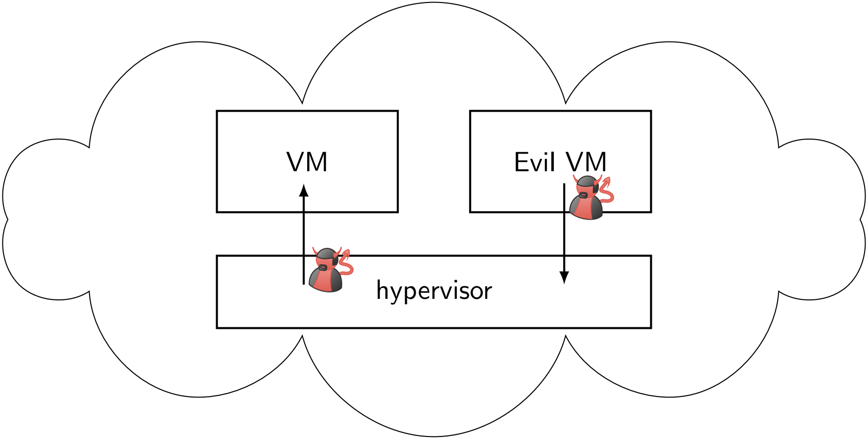
AMD SEV



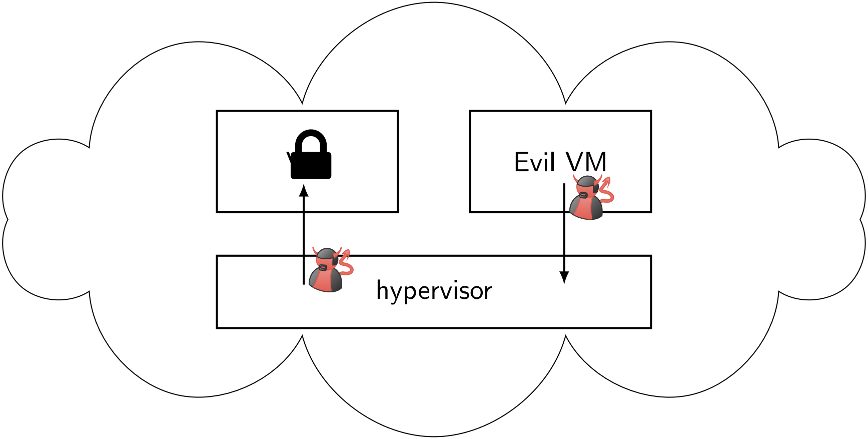
AMD SEV



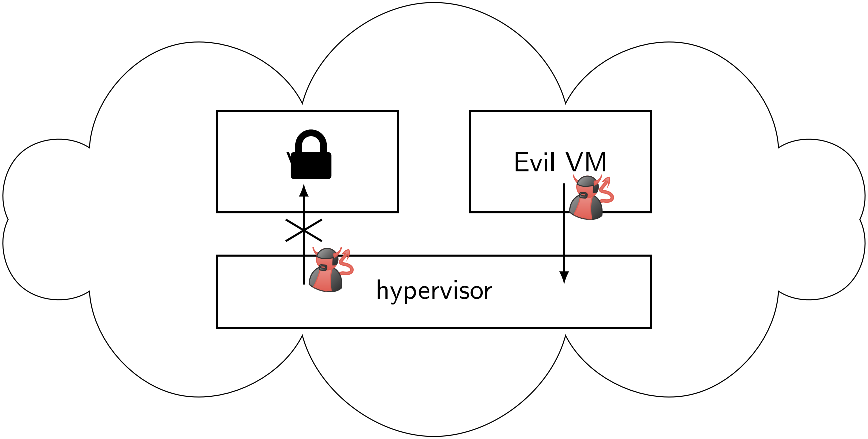
AMD SEV



AMD SEV



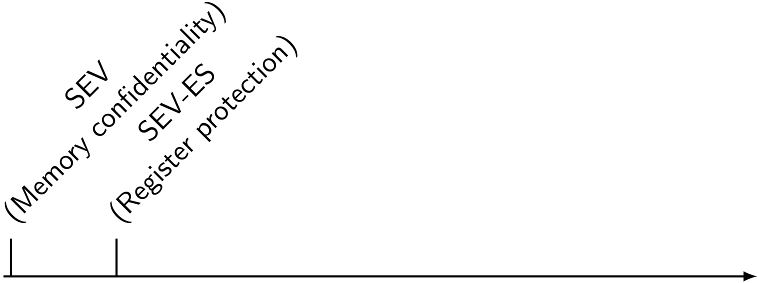
AMD SEV



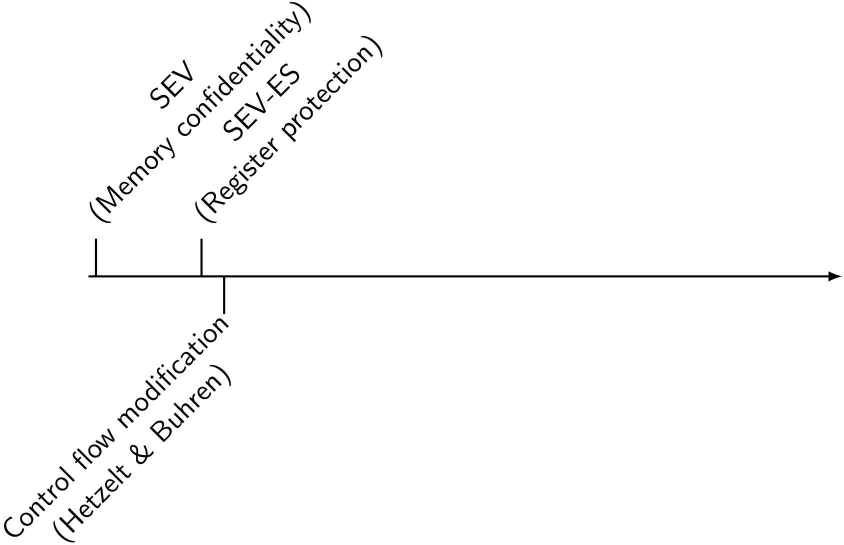
An (incomplete) timeline on AMD SEV

SEV (Memory confidentiality)

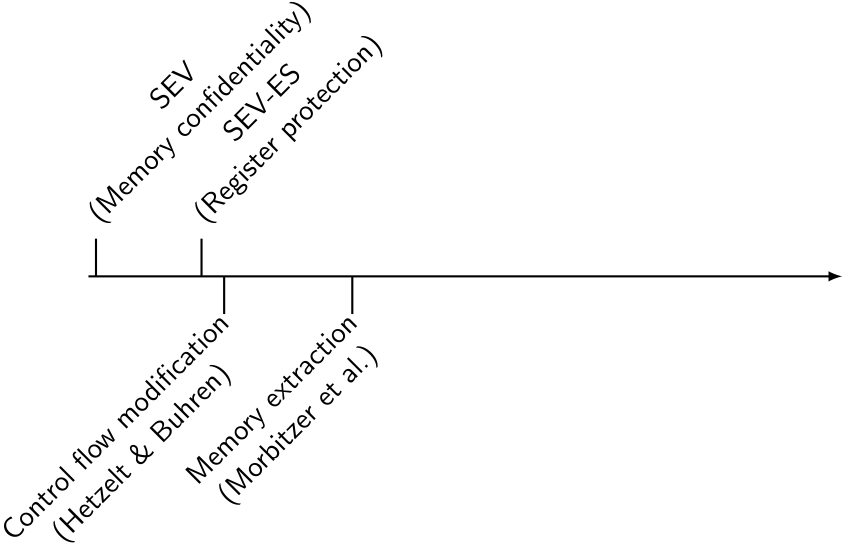
An (incomplete) timeline on AMD SEV



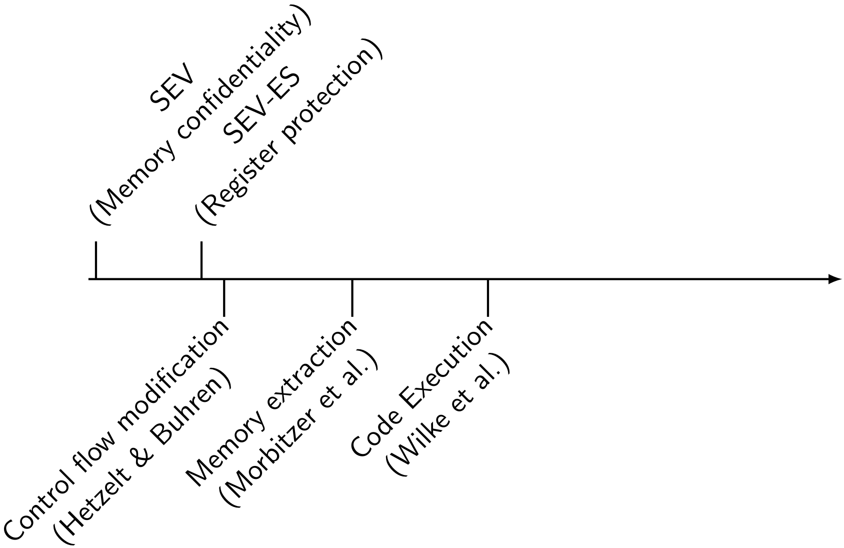
An (incomplete) timeline on AMD SEV



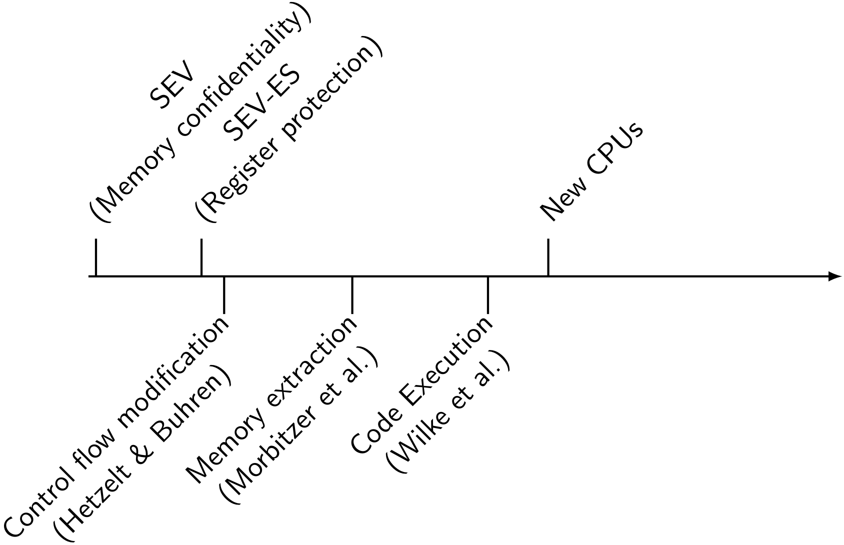
An (incomplete) timeline on AMD SEV



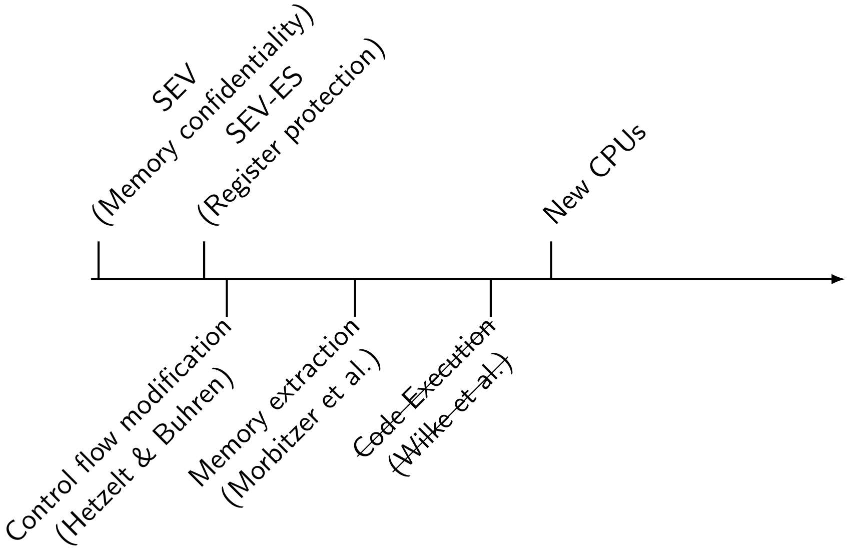
An (incomplete) timeline on AMD SEV



An (incomplete) timeline on AMD SEV



An (incomplete) timeline on AMD SEV



SEV
(Memory confidentiality)

SEV-ES
(Register protection)

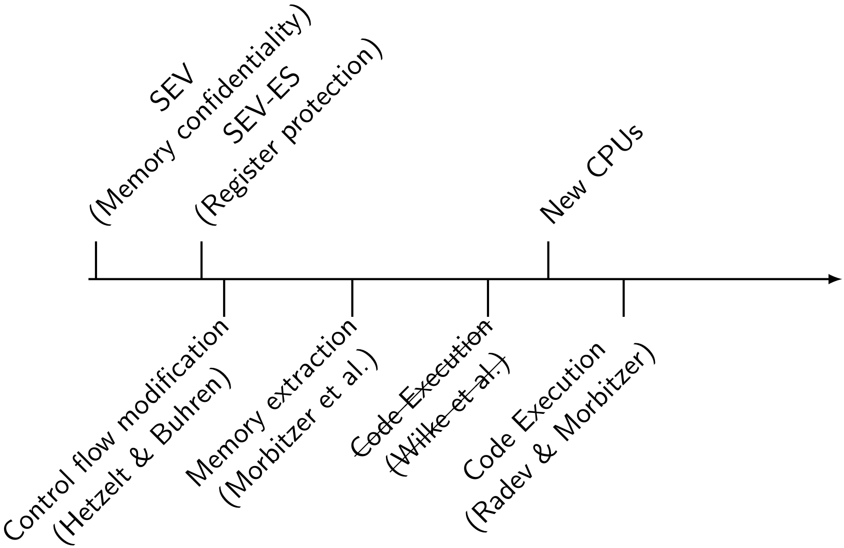
New CPUs

Control flow modification
(Hetzelt & Buhren)

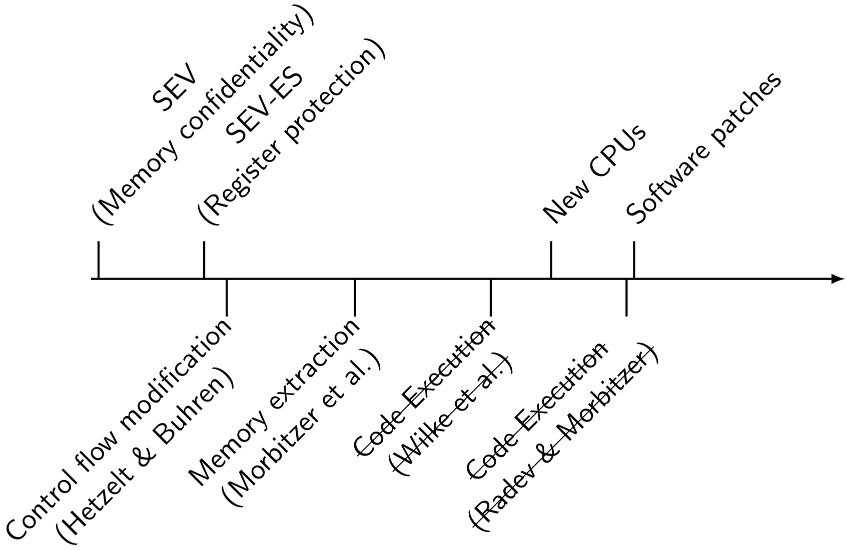
Memory extraction
(Morbitzer et al.)

~~Code Execution~~
~~(Wilke et al.)~~

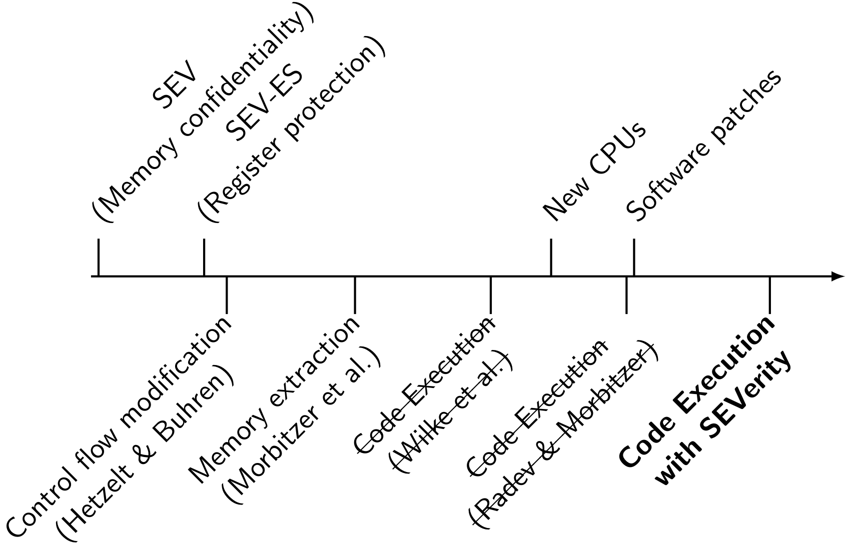
An (incomplete) timeline on AMD SEV



An (incomplete) timeline on AMD SEV



An (incomplete) timeline on AMD SEV



SEV (Memory confidentiality)

SEV-ES (Register protection)

New CPUs

Software patches

Control flow modification (Hetzelt & Buhren)

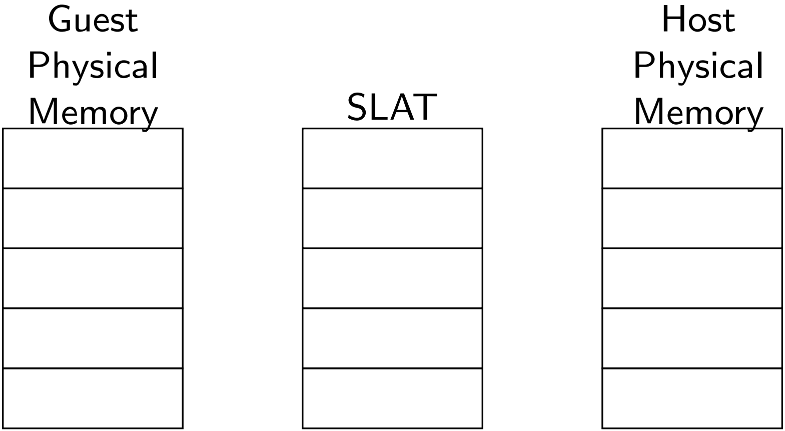
Memory extraction (Morbitzer et al.)

~~Code Execution (Wilke et al.)~~

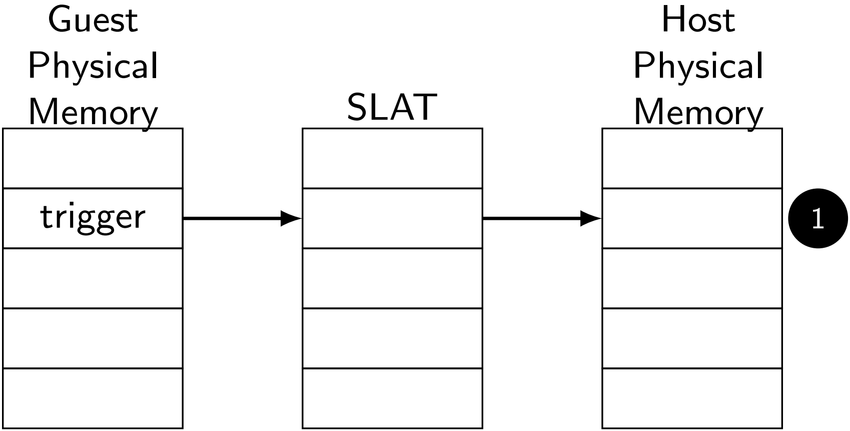
~~Code Execution (Radev & Morbitzer)~~

Code Execution with SEVerity

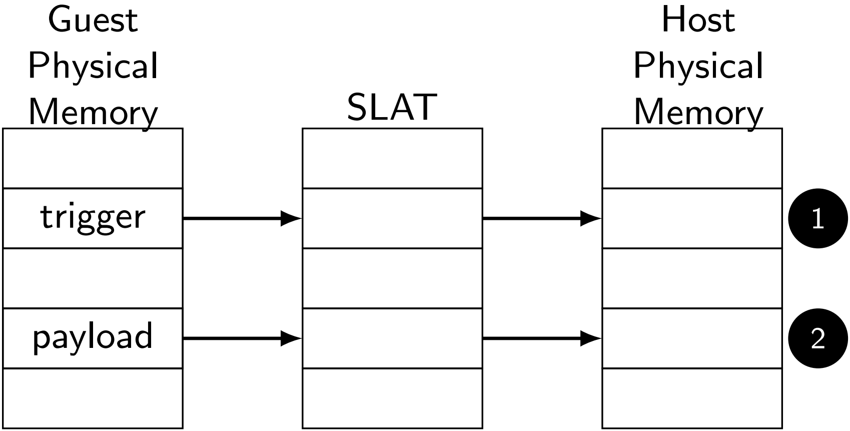
Attack Overview



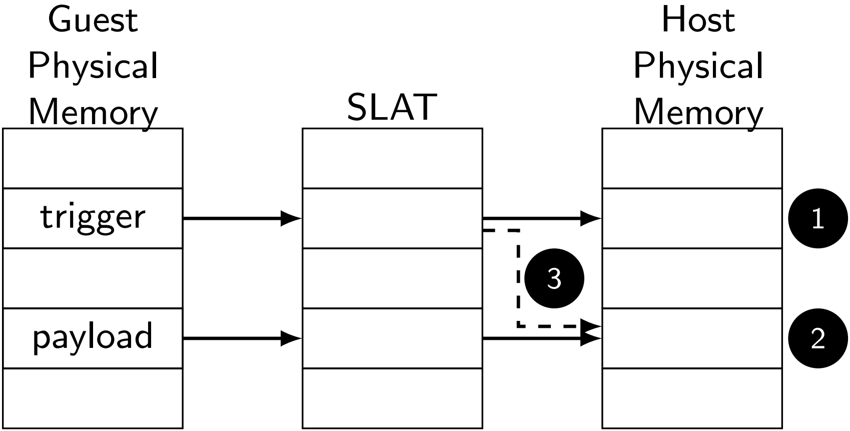
Attack Overview



Attack Overview



Attack Overview



❶ Identifying the trigger

- Use of Non Maskable Interrupts (NMIs)

❶ Identifying the trigger

- Use of Non Maskable Interrupts (NMIs)
- VM executes NMI handler immediately

➊ Identifying the trigger

- Use of Non Maskable Interrupts (NMIs)
- VM executes NMI handler immediately
  → perfect trigger

➊ Identifying the trigger

- Use of Non Maskable Interrupts (NMIs)
- VM executes NMI handler immediately
  → perfect trigger
- Analyze kernel binary to determine location of NMI handler

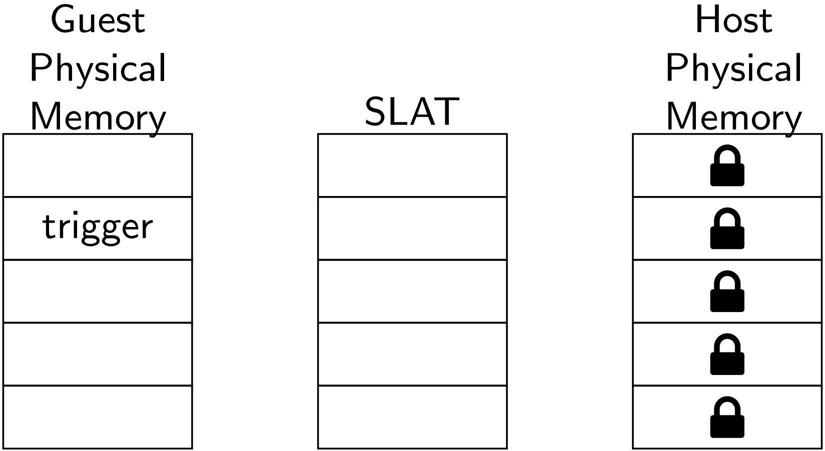
❶ Identifying the trigger

- Use of Non Maskable Interrupts (NMIs)
- VM executes NMI handler immediately
  → perfect trigger
- Analyze kernel binary to determine location of NMI handler
- KASLR randomizes the kernel's offset in the VM's virtual and physical memory

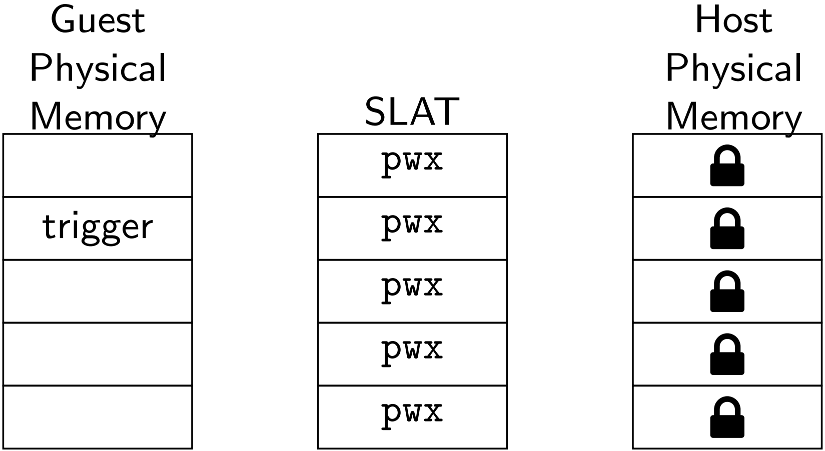
❶ Identifying the trigger

- Use of Non Maskable Interrupts (NMIs)
- VM executes NMI handler immediately
  → perfect trigger
- Analyze kernel binary to determine location of NMI handler
- KASLR randomizes the kernel's offset in the VM's virtual and physical memory
- Three methods to determine KASLR offset

❶ Identifying the trigger: probing NMI handler

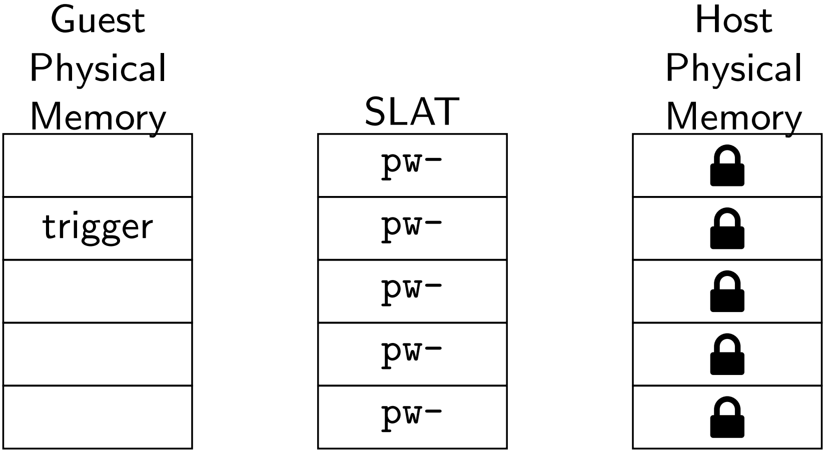


❶ Identifying the trigger: probing NMI handler



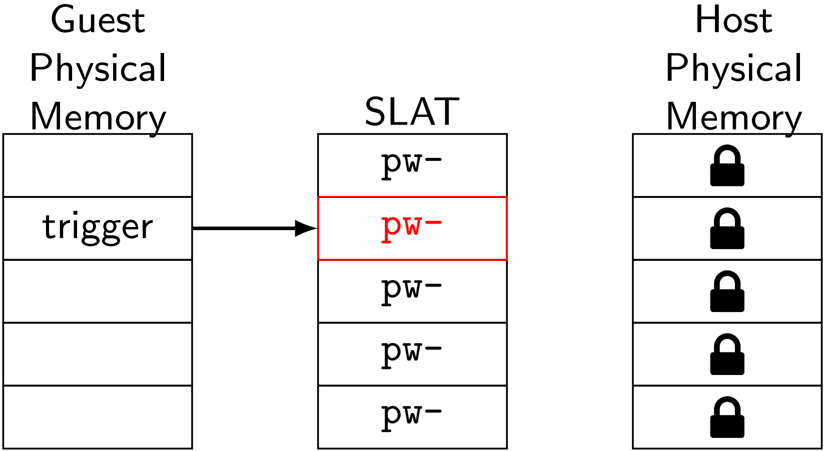
Guest Physical Memory	SLAT	Host Physical Memory
	`pwx`	🔒
trigger	`pwx`	🔒
	`pwx`	🔒
	`pwx`	🔒
	`pwx`	🔒

❶ Identifying the trigger: probing NMI handler

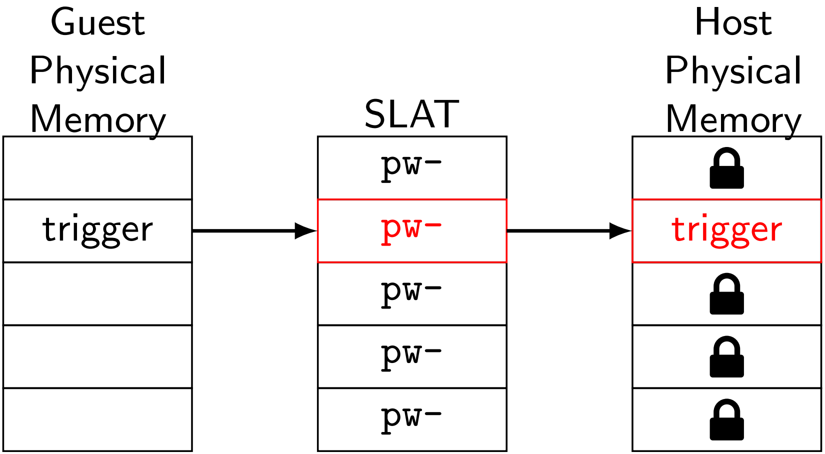


Guest Physical Memory	SLAT	Host Physical Memory
	`pw-`	🔒
trigger	`pw-`	🔒
	`pw-`	🔒
	`pw-`	🔒
	`pw-`	🔒

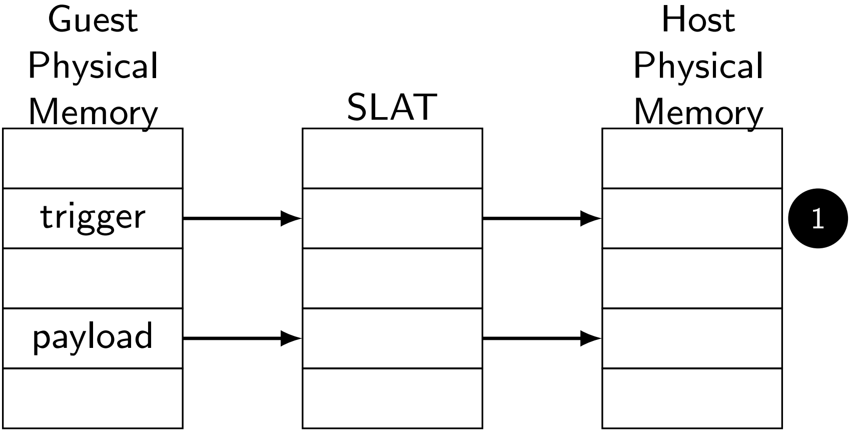
❶ Identifying the trigger: probing NMI handler



❶ Identifying the trigger: probing NMI handler



Attack Overview



Guest Physical Memory

SLAT

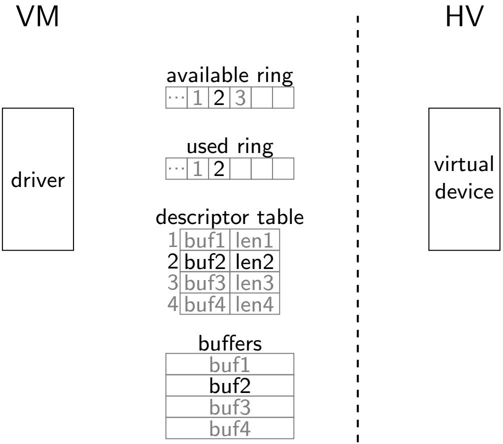
Host Physical Memory

trigger

payload

1

❷ Identifying the payload: virtio without SEV



VM

HV

driver

virtual device

available ring

⋯ | 1 | 2 | 3 | |

used ring

⋯ | 1 | 2 | | |

descriptor table

1	buf1	len1
2	buf2	len2
3	buf3	len3
4	buf4	len4

buffers

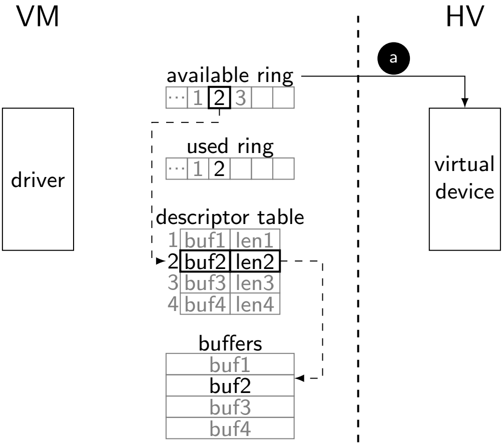
buf1

buf2

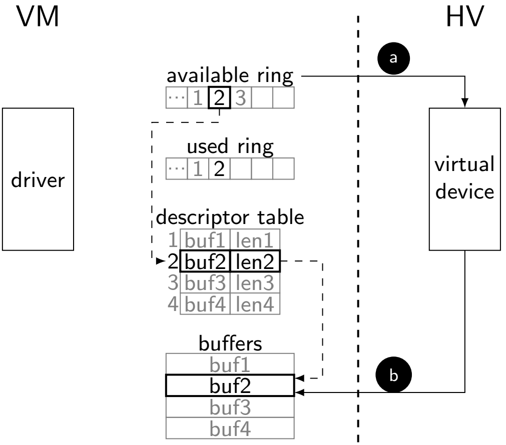
buf3

buf4

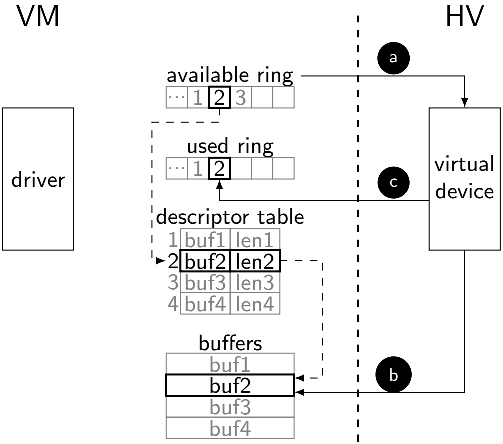
❷ Identifying the payload: virtio without SEV



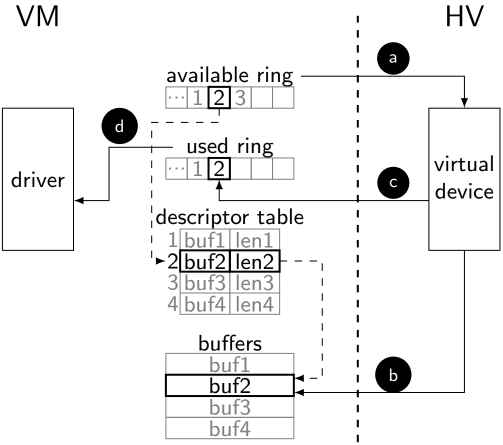
❷ Identifying the payload: virtio without SEV



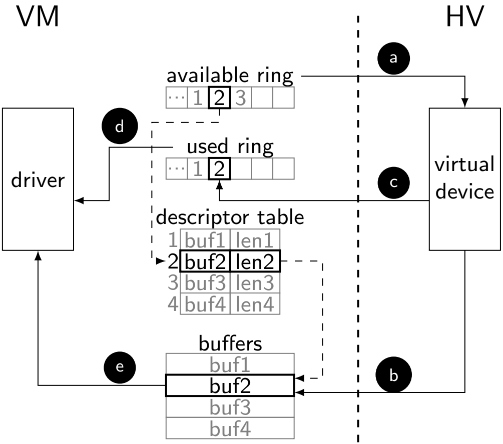
❷ Identifying the payload: virtio without SEV



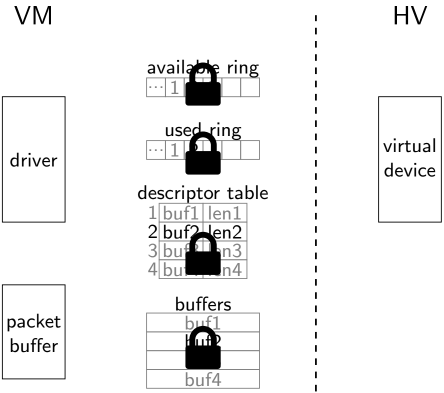
❷ Identifying the payload: virtio without SEV



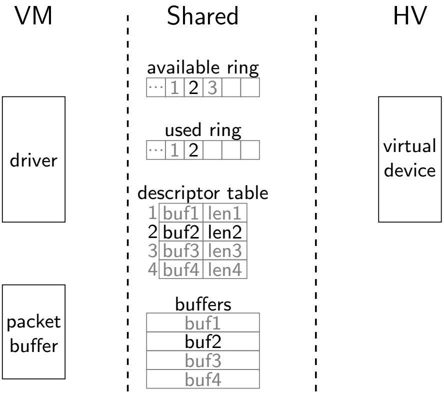
❷ Identifying the payload: virtio without SEV



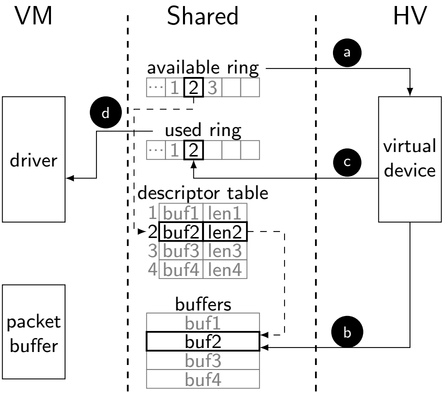
❷ Identifying the payload: virtio with SEV



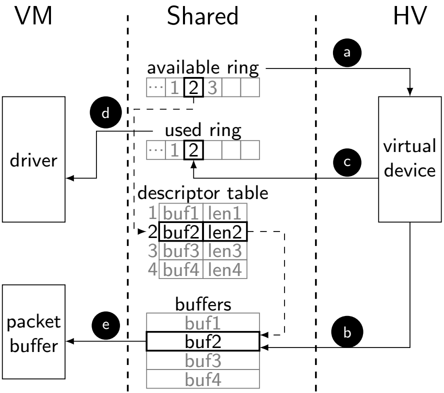
❷ Identifying the payload: virtio with SEV



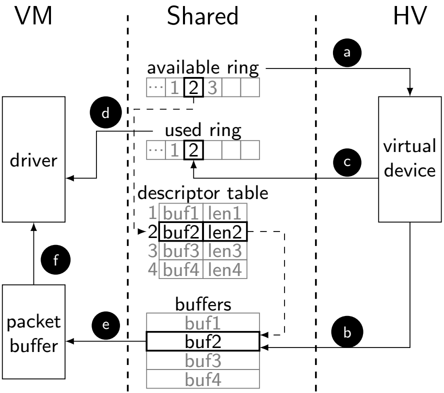
❷ Identifying the payload: virtio with SEV



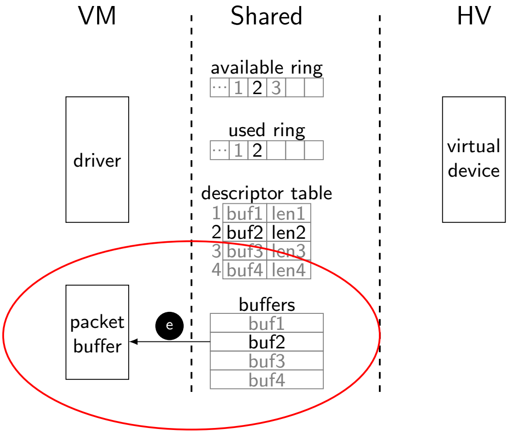
❷ Identifying the payload: virtio with SEV



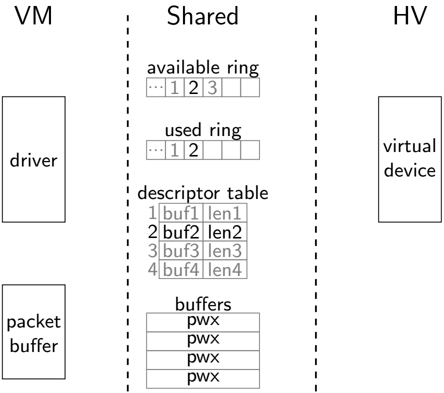
❷ Identifying the payload: virtio with SEV



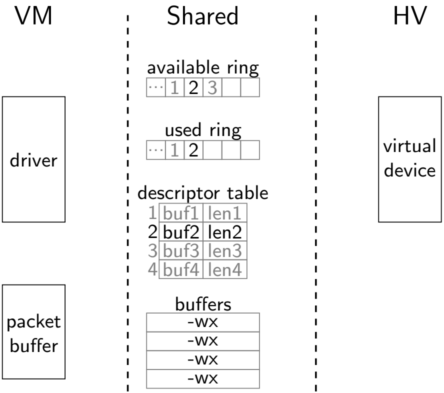
❷ Identifying the payload: virtio with SEV



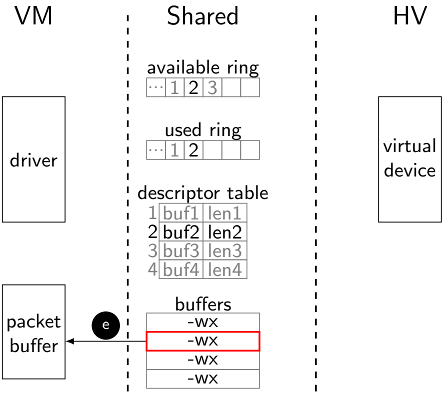
❷ Identifying the payload: virtio with SEV



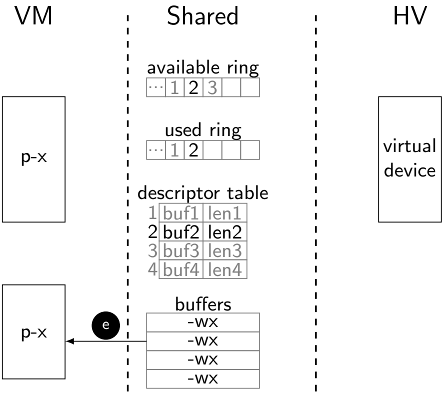
❷ Identifying the payload: virtio with SEV



❷ Identifying the payload: virtio with SEV



❷ Identifying the payload: virtio with SEV



VM

Shared

HV

available ring

···	1	2	3		

used ring

···	1	2			

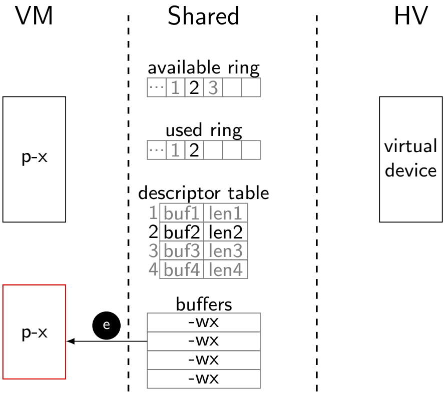
descriptor table

1	buf1	len1
2	buf2	len2
3	buf3	len3
4	buf4	len4

buffers

-wx
-wx
-wx
-wx

❷ Identifying the payload: virtio with SEV



VM

Shared

HV

available ring

···	1	2	3		

used ring

···	1	2			

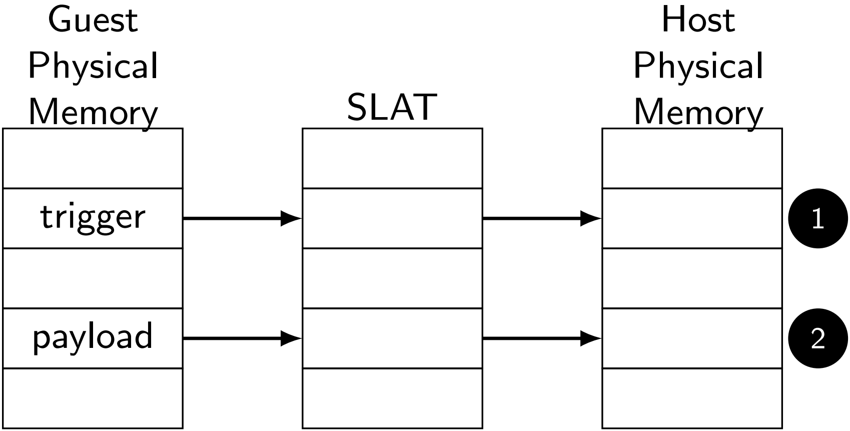
descriptor table

1	buf1	len1
2	buf2	len2
3	buf3	len3
4	buf4	len4

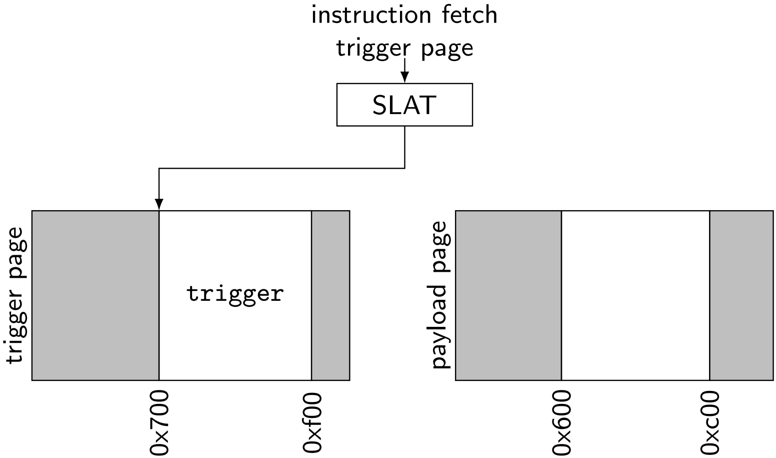
buffers

-wx
-wx
-wx
-wx

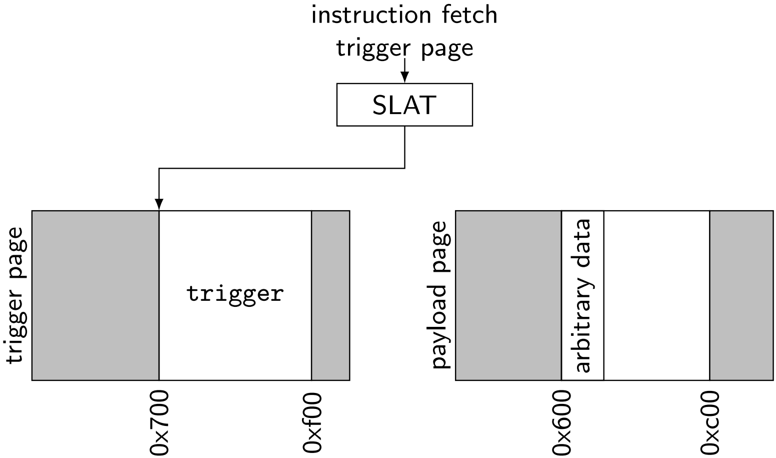
Attack Overview



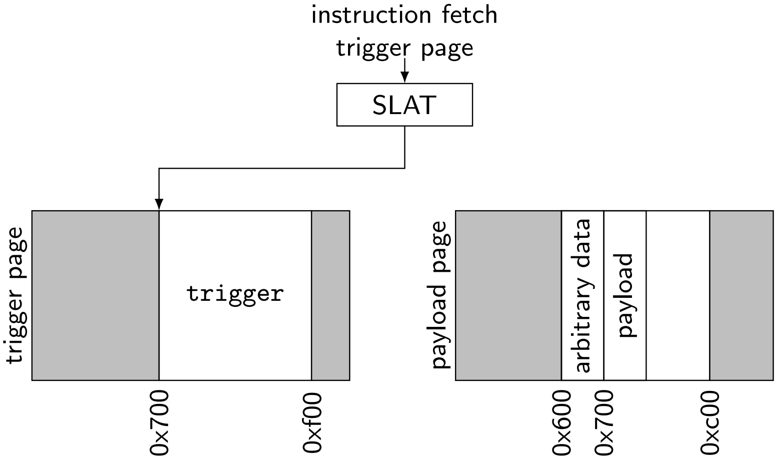
❸ Remapping


instruction fetch
trigger page
SLAT
trigger page
trigger
0x700
0xf00
payload page
0x600
0xc00

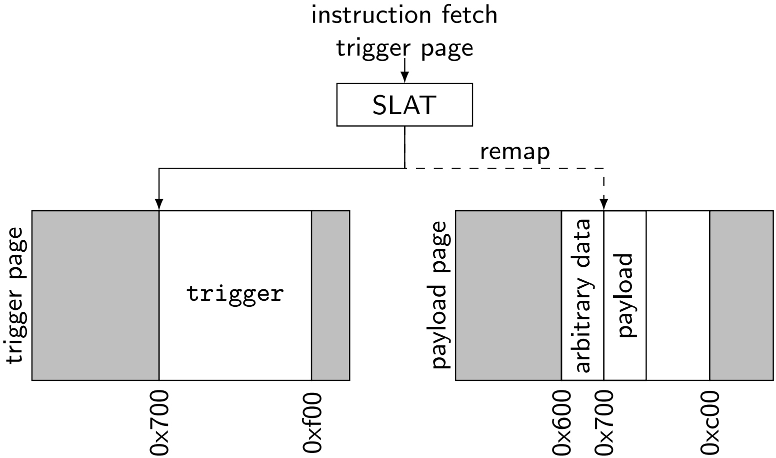
❸ Remapping


instruction fetch
trigger page
SLAT
trigger page
trigger
0x700
0xf00
payload page
arbitrary data
0x600
0xc00

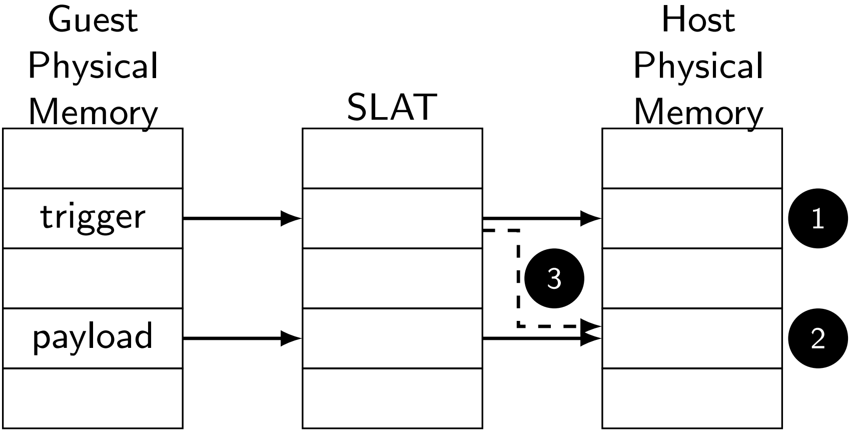
❸ Remapping


instruction fetch
trigger page
SLAT
trigger page
trigger
0x700
0xf00
payload page
arbitrary data
payload
0x600
0x700
0xc00

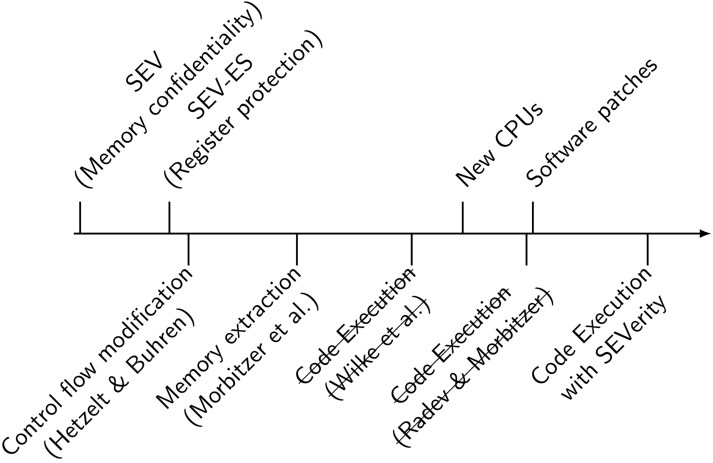
❸ Remapping


instruction fetch
trigger page
SLAT
remap
trigger page
trigger
0x700
0xf00
payload page
arbitrary data
payload
0x600
0x700
0xc00

Attack Overview



A (still incomplete) timeline on AMD SEV



SEV (Memory confidentiality)

SEV-ES (Register protection)

New CPUs

Software patches

Control flow modification (Hetzelt & Buhren)

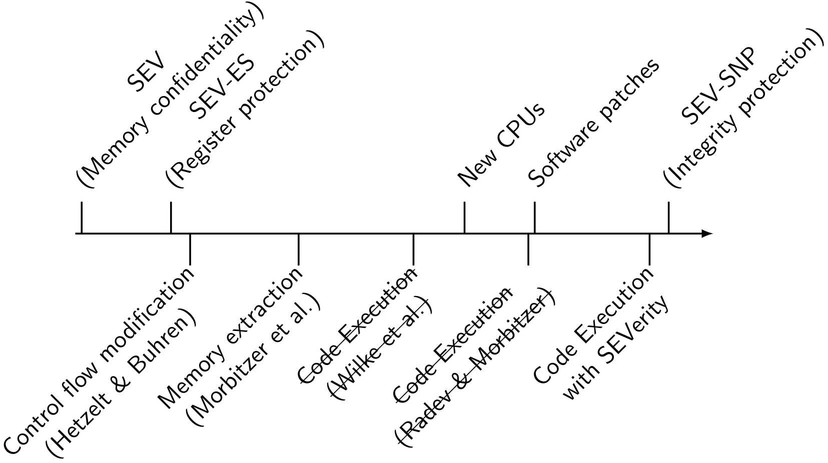
Memory extraction (Morbitzer et al.)

~~Code Execution (Wilke et al.)~~

~~Code Execution (Radev & Morbitzer)~~

Code Execution with SEVerity

A (still incomplete) timeline on AMD SEV



SEV (Memory confidentiality)

SEV-ES (Register protection)

New CPUs

Software patches

SEV-SNP (Integrity protection)

Control flow modification (Hetzelt & Buhren)

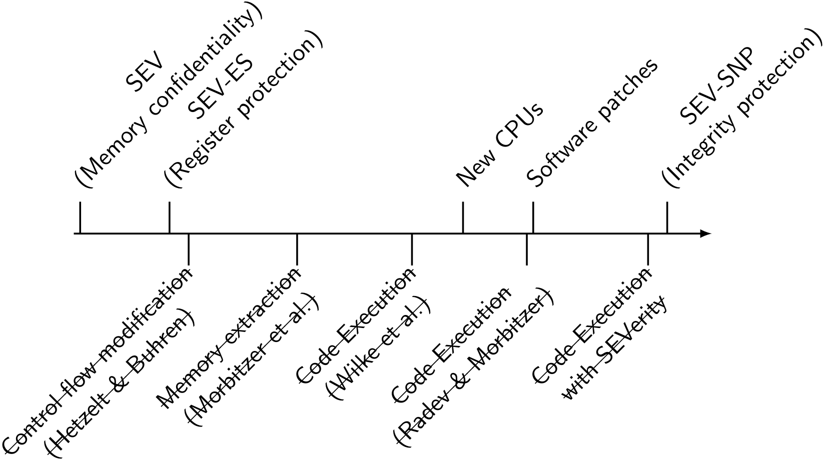
Memory extraction (Morbitzer et al.)

~~Code Execution (Wilke et al.)~~

~~Code Execution (Radev & Morbitzer)~~

Code Execution with SEVerity

A (still incomplete) timeline on AMD SEV



SEV (Memory confidentiality)

SEV-ES (Register protection)

~~Control flow modification (Hetzelt & Buhren)~~

~~Memory extraction (Morbitzer et al.)~~

~~Code Execution (Wilke et al.)~~

New CPUs

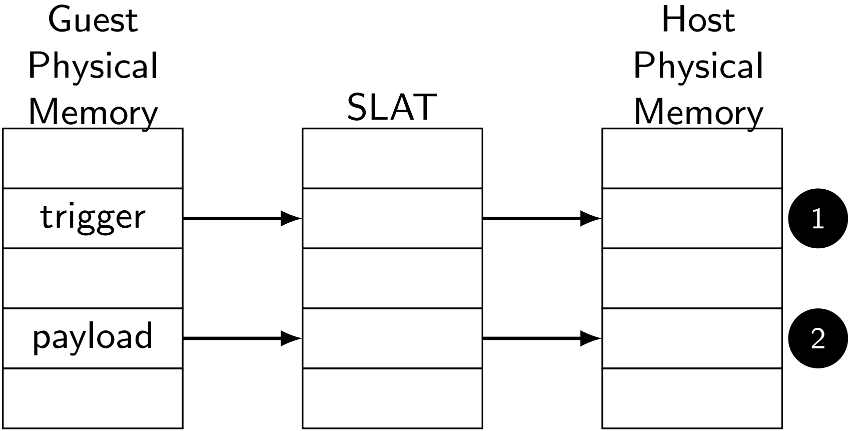
~~Code Execution (Radev & Morbitzer)~~

Software patches

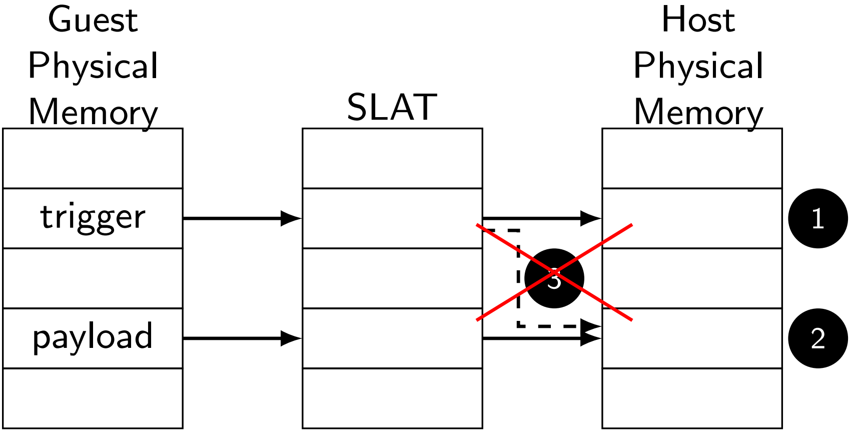
~~Code Execution with SEVerity~~

SEV-SNP (Integrity protection)

Attack overview with SEV-SNP



Attack overview with SEV-SNP



Conclusion

- SEVerity allows to execute arbitrary code in SEV-protected VMs

Conclusion

- SEVerity allows to execute arbitrary code in SEV-protected VMs
  → Using page tracking and SLAT remapping

Conclusion

- SEVerity allows to execute arbitrary code in SEV-protected VMs
  → Using page tracking and SLAT remapping
- PoC uses Linux & virtio
  → but general concept applies to all guest OS

Conclusion

- SEVerity allows to execute arbitrary code in SEV-protected VMs
  → Using page tracking and SLAT remapping
- PoC uses Linux & virtio
  → but general concept applies to all guest OS
- SEV and SEV-ES are vulnerable to various attacks

Conclusion

- SEVerity allows to execute arbitrary code in SEV-protected VMs
  → Using page tracking and SLAT remapping
- PoC uses Linux & virtio
  → but general concept applies to all guest OS
- SEV and SEV-ES are vulnerable to various attacks
- SEV-SNP adds integrity protection

Conclusion

- SEVerity allows to execute arbitrary code in SEV-protected VMs
  → Using page tracking and SLAT remapping
- PoC uses Linux & virtio
  → but general concept applies to all guest OS
- SEV and SEV-ES are vulnerable to various attacks
- SEV-SNP adds integrity protection
  - SEV-SNP capable CPUs available since Q1 2021

Conclusion

- SEVerity allows to execute arbitrary code in SEV-protected VMs
  → Using page tracking and SLAT remapping
- PoC uses Linux & virtio
  → but general concept applies to all guest OS
- SEV and SEV-ES are vulnerable to various attacks
- SEV-SNP adds integrity protection
  - SEV-SNP capable CPUs available since Q1 2021
  - First software patches also available

Questions?

Photo by Nicole Lawton from FreeImages