Confidential Computing with OpenBSD

Hans-Jörg Höxer

Confidential Virtual Machines with OpenBSD

Werkstattbericht

Hans-Jörg Höxer

Confidential Virtual Machines with OpenBSD

Agenda

- **Introduction**
- Memory encryption for Confidential VMs — SEV
- vCPU state encryption — SEV-ES
- Conclusion

About

Hans-Jörg Höxer

- Mid-2000s:
  - hshoexer@openbsd.org
- genua GmbH (www.genua.de):
  - hshoexer@genua.de
  - OpenBSD based products
  - Firewalls and VNP-Appliances
  - Confidential Computing…

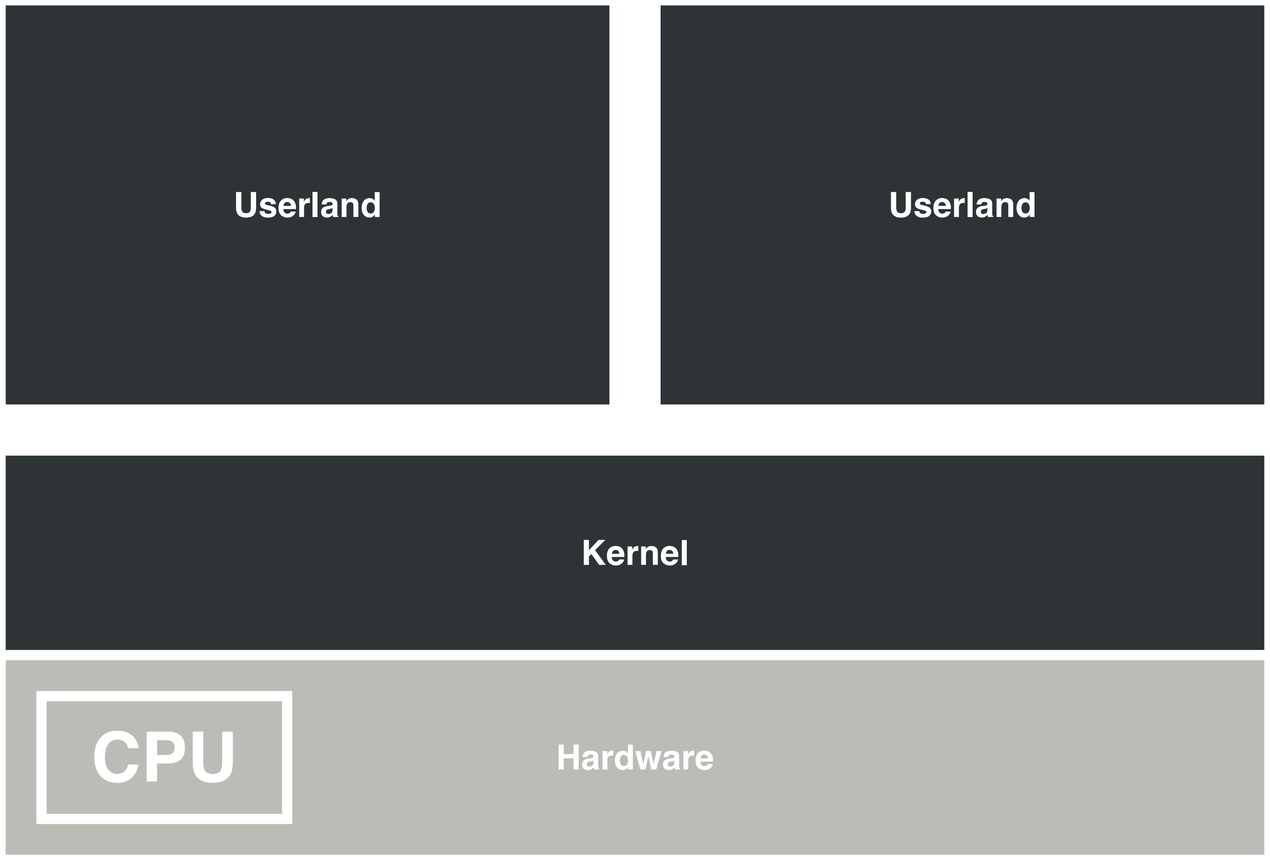
Confidential Computing

What is this all about?

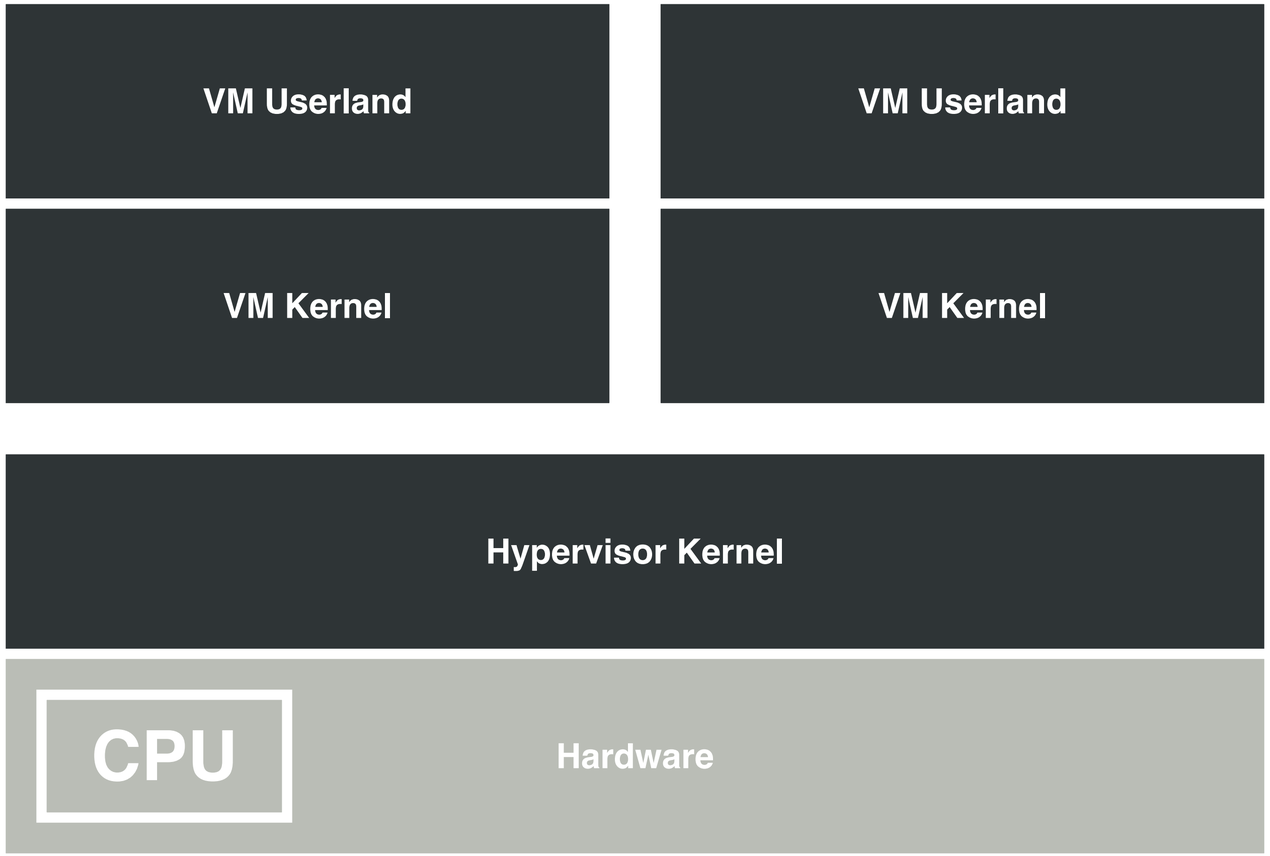
- Problem:
  - Sensitive data in an untrusted environment
  - Context: Virtualisation, VMs, cloud
- Supposed solution:
  - Confidential VM on untrusted hypervisor
  - “Turn public cloud into private cloud”
  - Bold claims…

➡Learn by implementing for OpenBSD (vmm(4)/vmd(8))

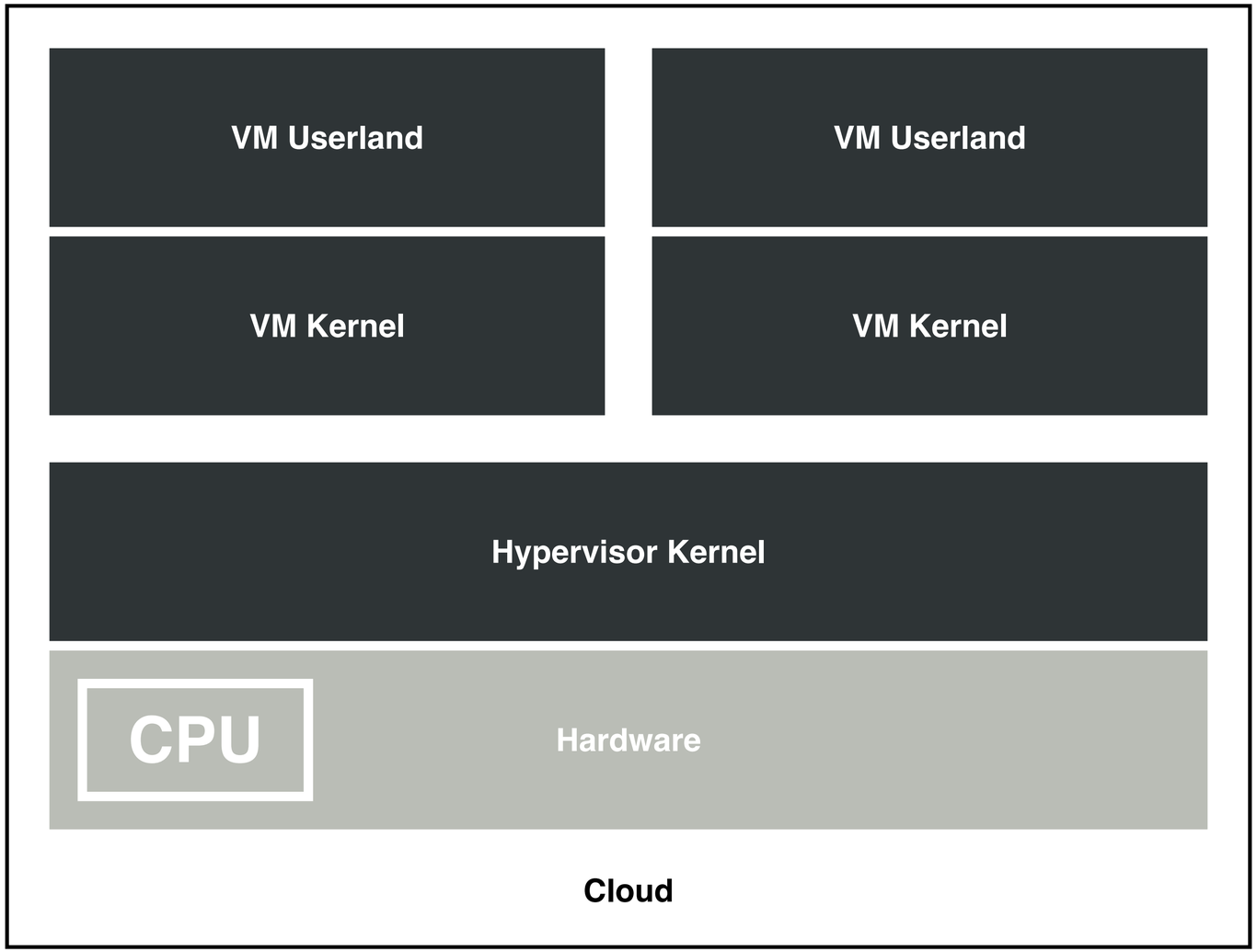
Trusted Environment



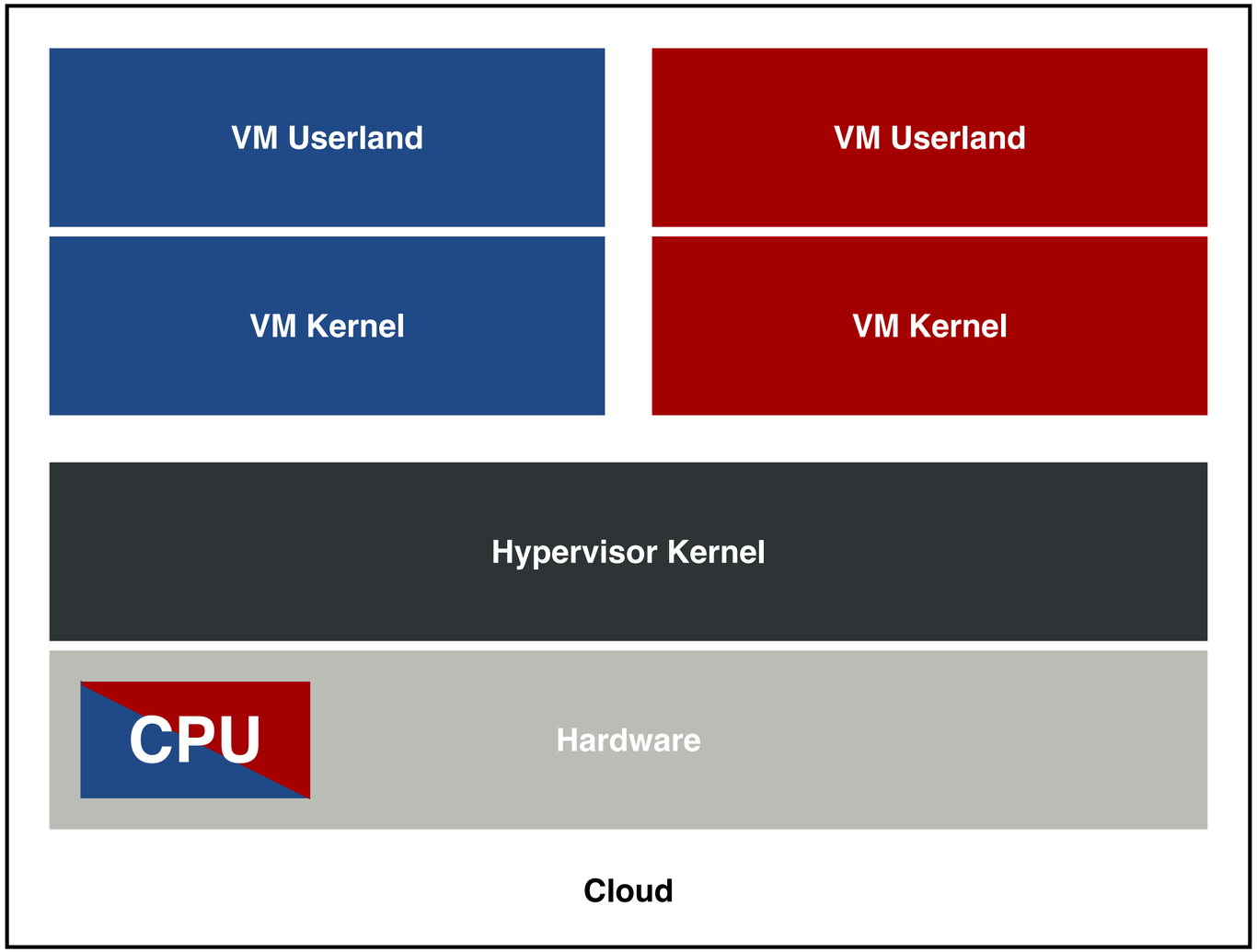
Trusted Environment



(Un)trusted Environment?



Confidential Virtual Machine



Confidential Computing

*

Claims

- Techniques to protect computing workload from its untrusted environment
  - Data confidentiality
  - Data integrity
  - Code integrity
- Isolation levels
  - Function or library isolation
  - Application isolation
  - ⭐Virtual machine isolation (Confidential Virtual Machine)

Confidential Virtual Machines

CPU Support

- In the CPU we trust:
  - ⭐Runtime encryption
  - Attestation
  - More techniques (eg. restricted injection)
- Examples:
  - **AMD SEV, SEV-ES, SEV-SNP**
  - Intel TDX, Arm CCA

Confidential Virtual Machines

CPU Support

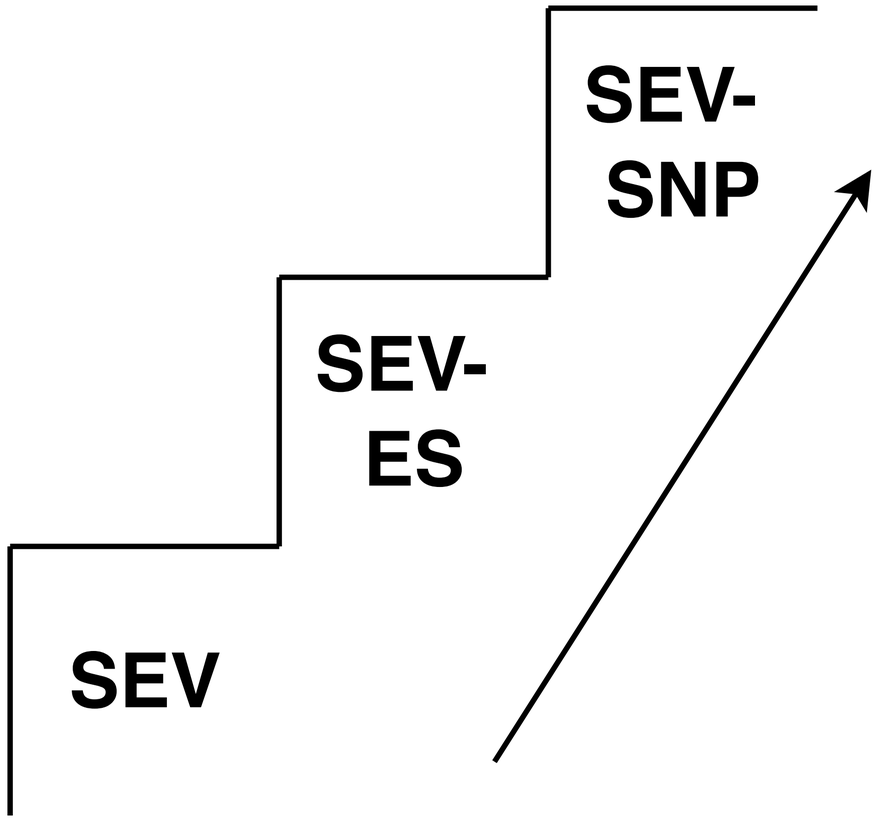
In the CPU we trust!

In AMD we trust…

AMD SEV

Building Blocks

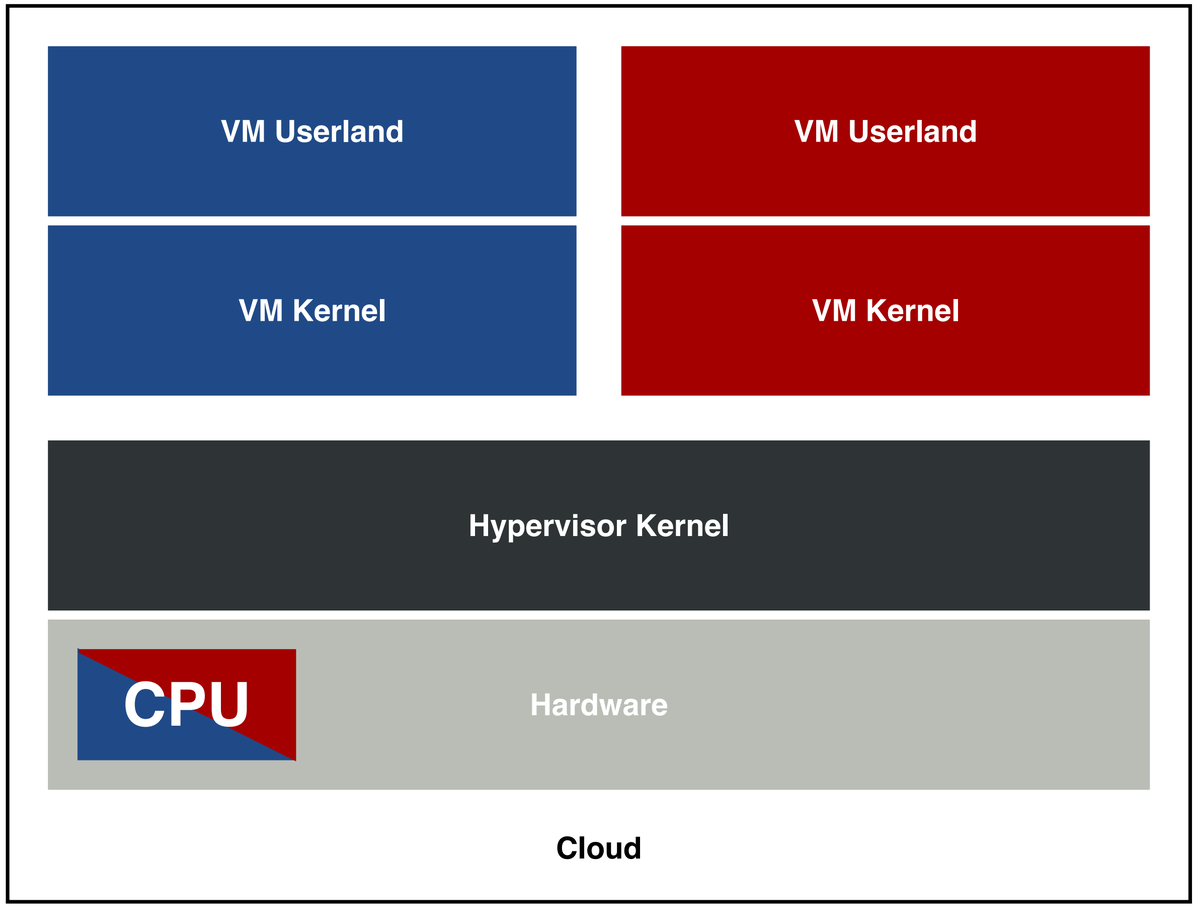
- Secure Encrypted Virtualisation (SEV)
  - Runtime Encryption
- SEV Encrypted State (SEV-ES)
  - vCPU State Encryption
- SEV Secure Nested Paging (SEV-SNP)
  - Integrity Protection



AMD SEV-* Building Blocks

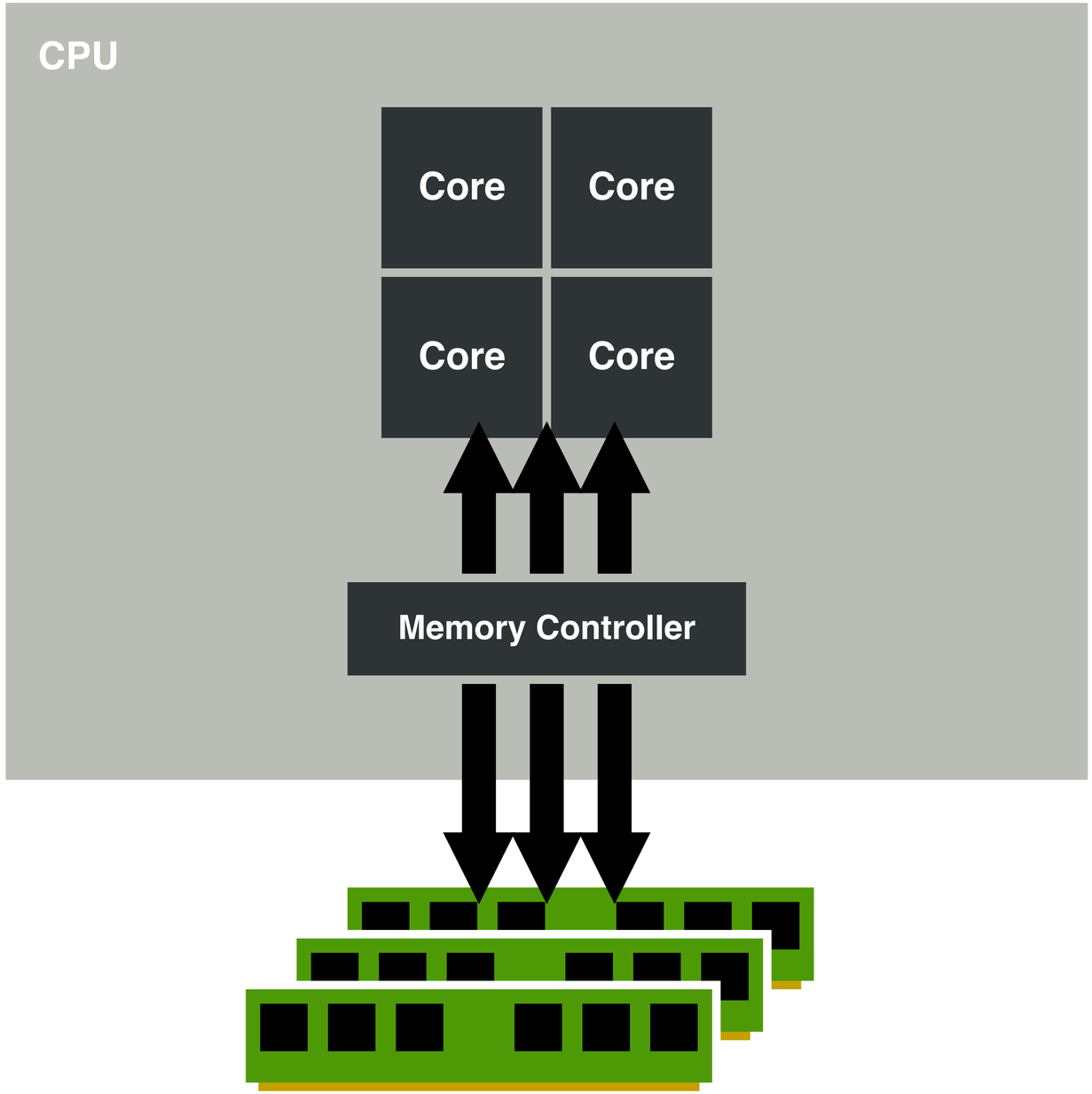
AMD Secure Encrypted Virtualisation

Confidential VM



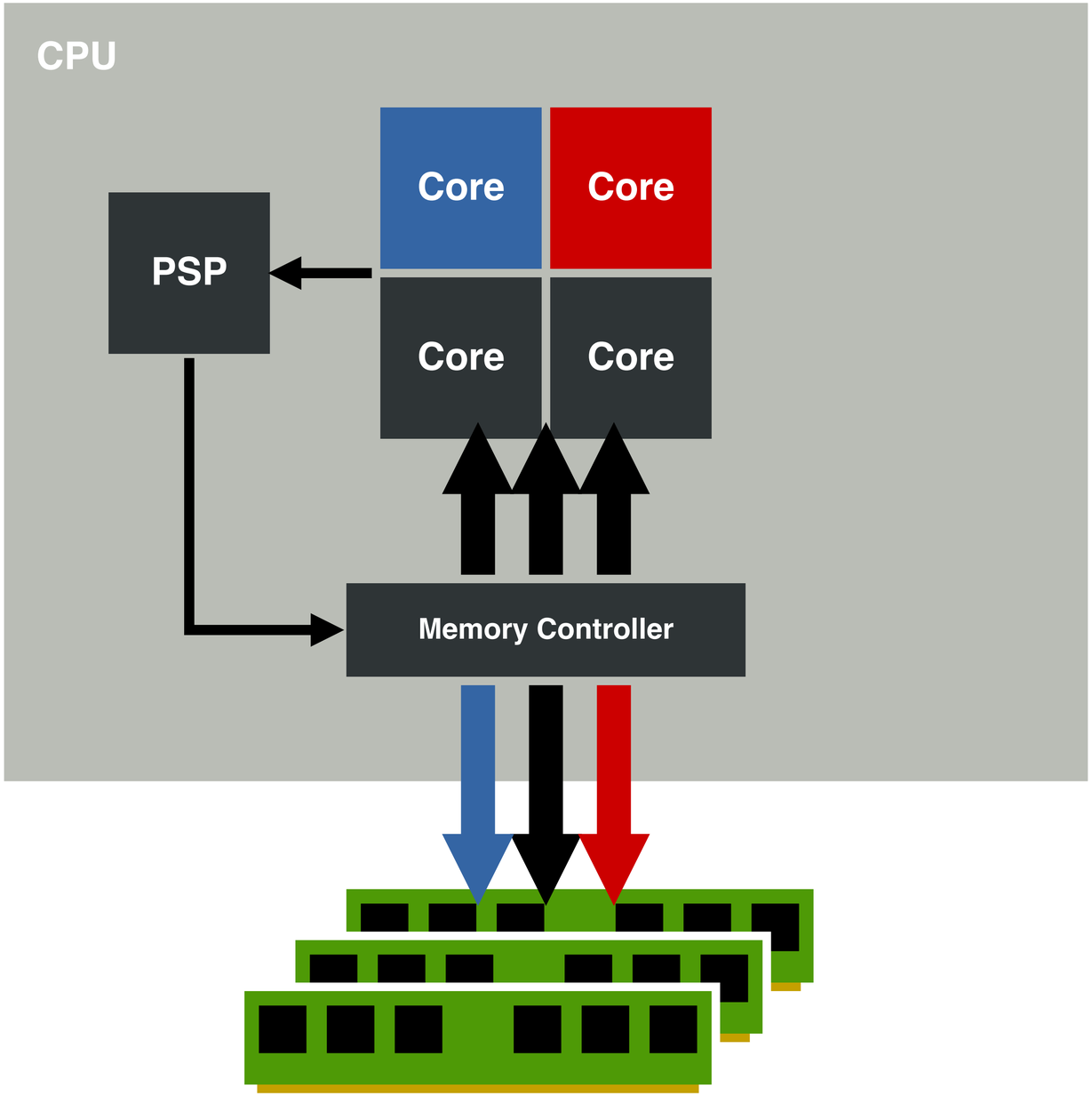
AMD SEV

Architecture



AMD SEV

Architecture



Confidential Computing with OpenBSD

Learning by doing

- Goals (November 2023):
  - Support for AMD SEV-*
  - OpenBSD as both hypervisor and guest
  - psp(4), vmd(8), vmm(4), GENERIC
- Compatibility with Linux/KVM:
  - OpenBSD guest on confidential Linux/KVM

Confidential Virtual Machines with OpenBSD

Agenda

- Introduction
- **Memory encryption for Confidential VMs — SEV**
- vCPU state encryption — SEV-ES
- Conclusion

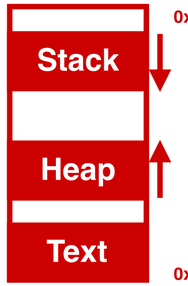
AMD SEV

Secure Encrypted Virtualisation

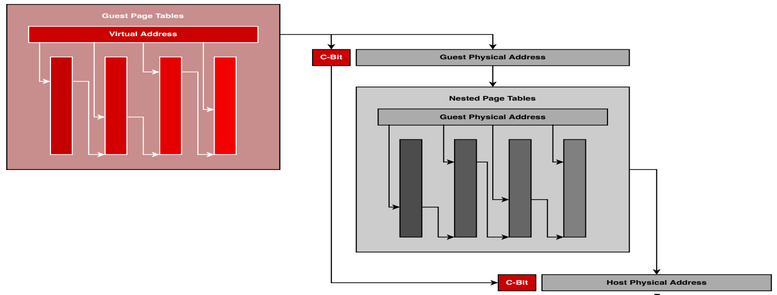
- Guest VM controls encryption!
  - Page tables:
    - “Crypt bit” (C-bit)
    - Private data
    - Public data — shareable:
      - DMA bounce buffers used by virtio(4)
      - Implemented in bus_dma(9)
- Guest and host support in OpenBSD 7.6

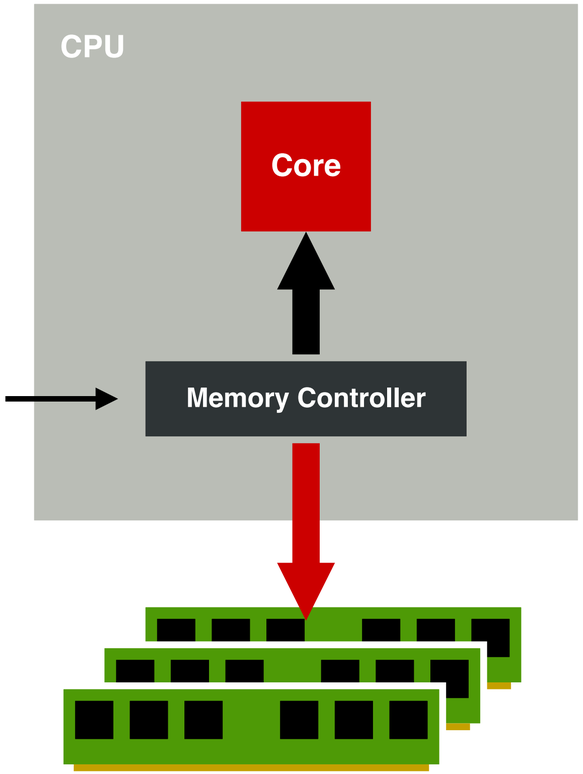
AMD SEV

Memory Access



0x0000 097D E374 FAB0





ASID	Key
1	5da0...
2	c23b...
3	561a...

0x0008 0001 BF7E 1AB0

0x0008 2001 BF7E 1AB0

0x0000 0001 BF7E 1AB0

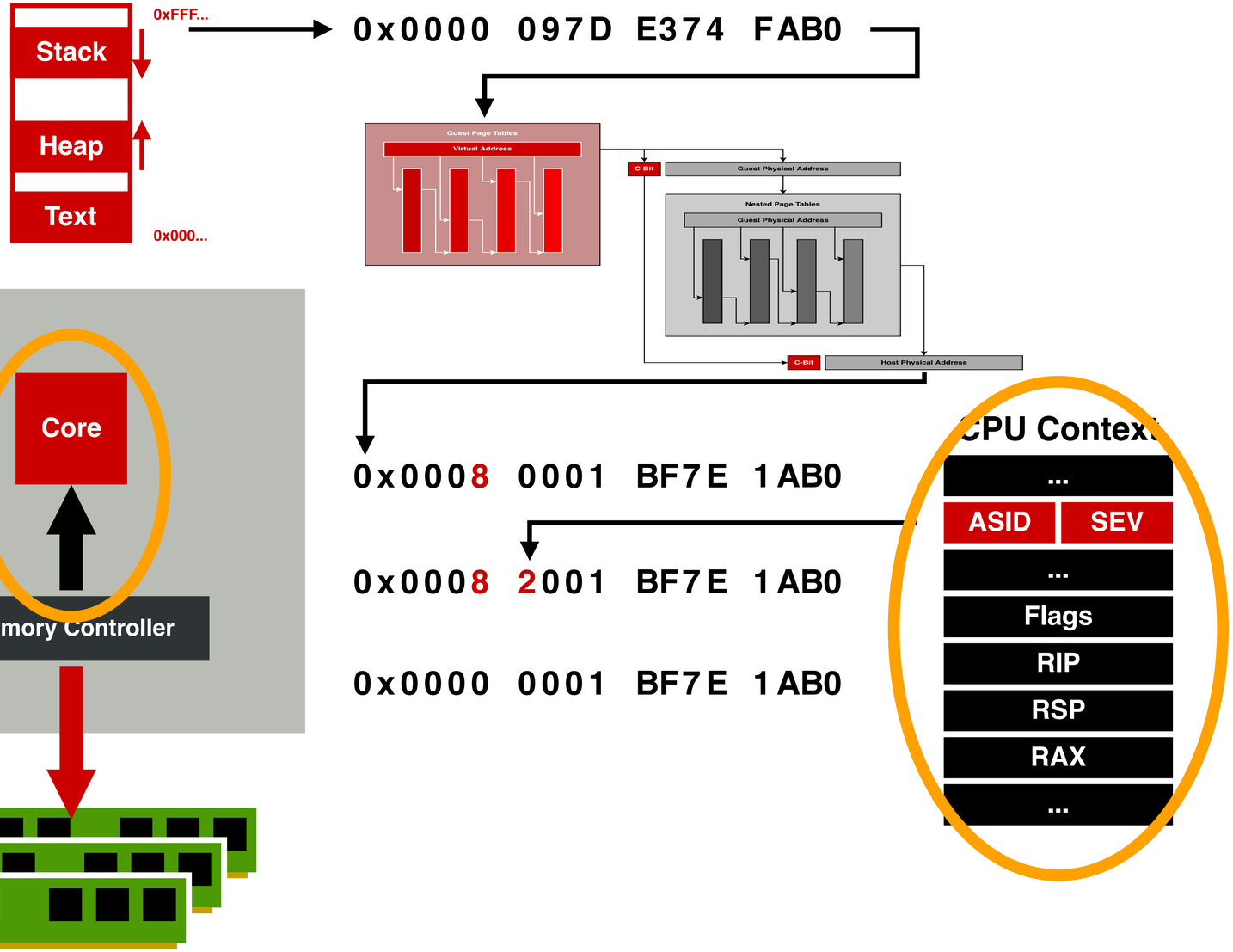
CPU Context

...
ASID / SEV
...
Flags
RIP
RSP
RAX
...

*

AMD SEV

Memory Access



*

Does SEV actually work?

*

The heat is on…

- Dump memory (RAM)
- Measure compressibility per page
- Plot heat map

hexdump

Warm boot marker

*

- Without SEV

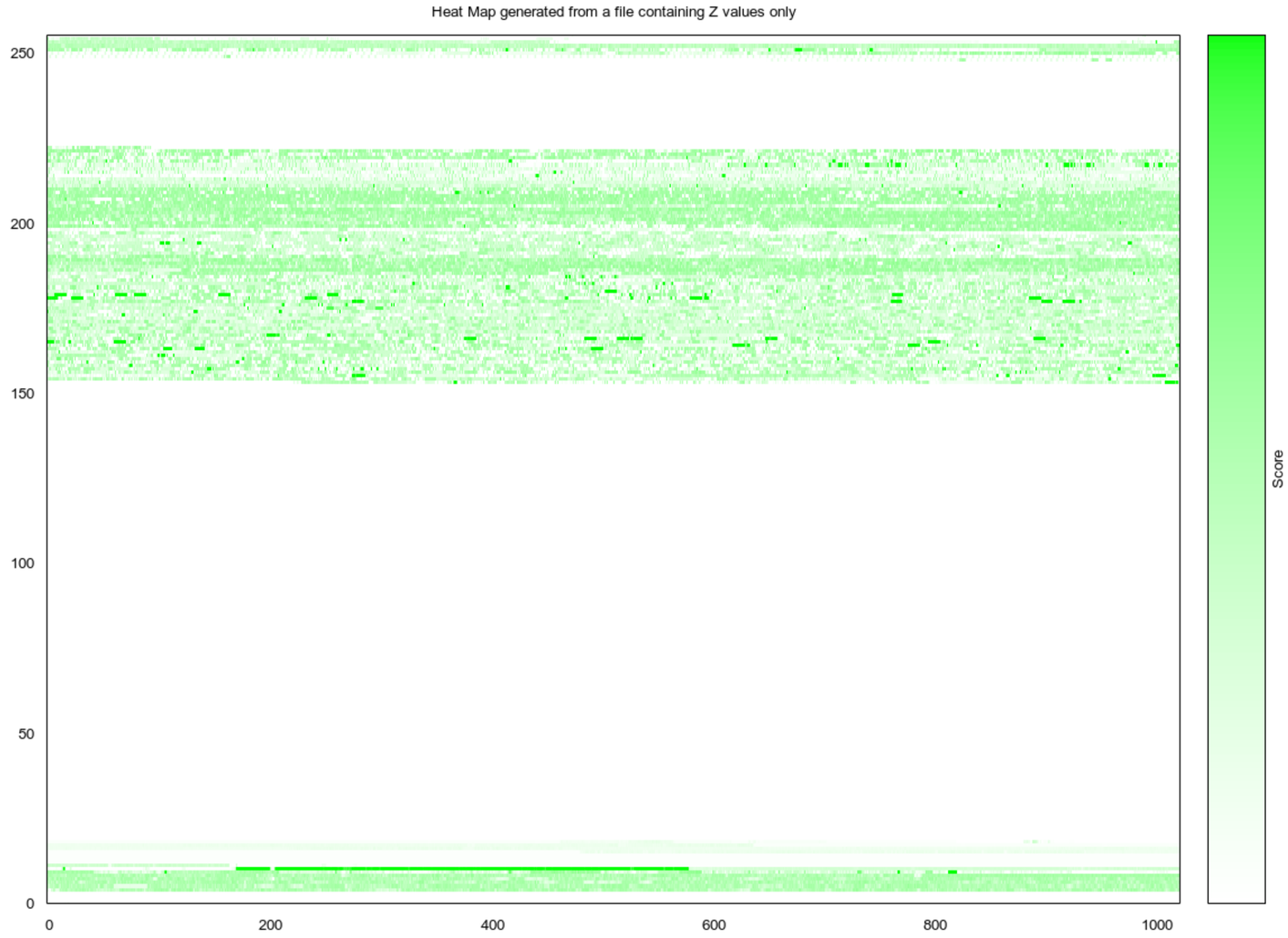
```
00000000  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000470  00 00 34 12 00 00 00 00  00 00 00 00 00 00 00 00  |..4.............|
00000480  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
```

- With SEV

```
00000000  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000470  28 30 32 e7 6e d3 4f 45  08 7e 3d 6f dc 71 71 22  |(02.n.OE.~=o.qq"|
00000480  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
```

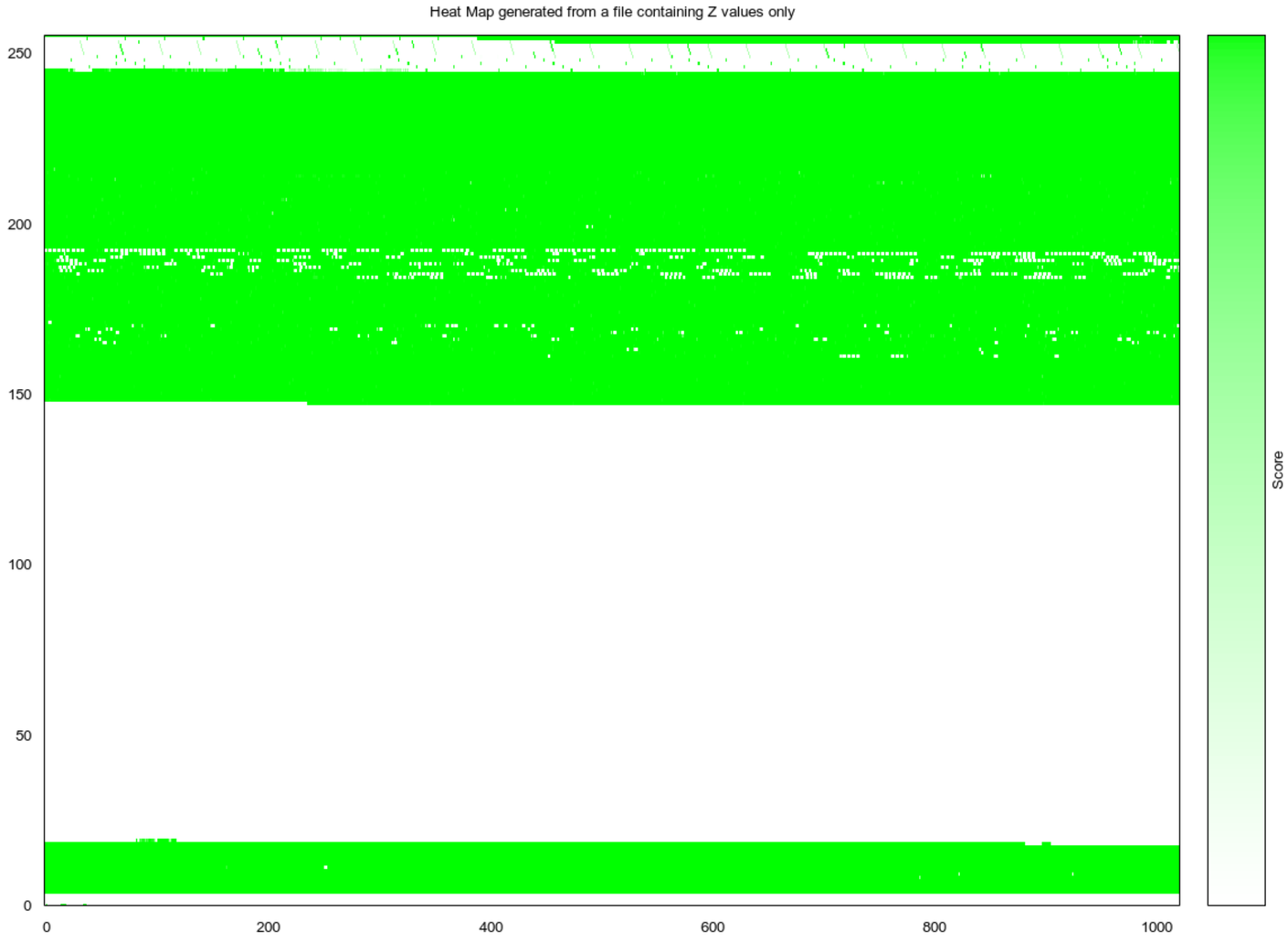
No encryption

*



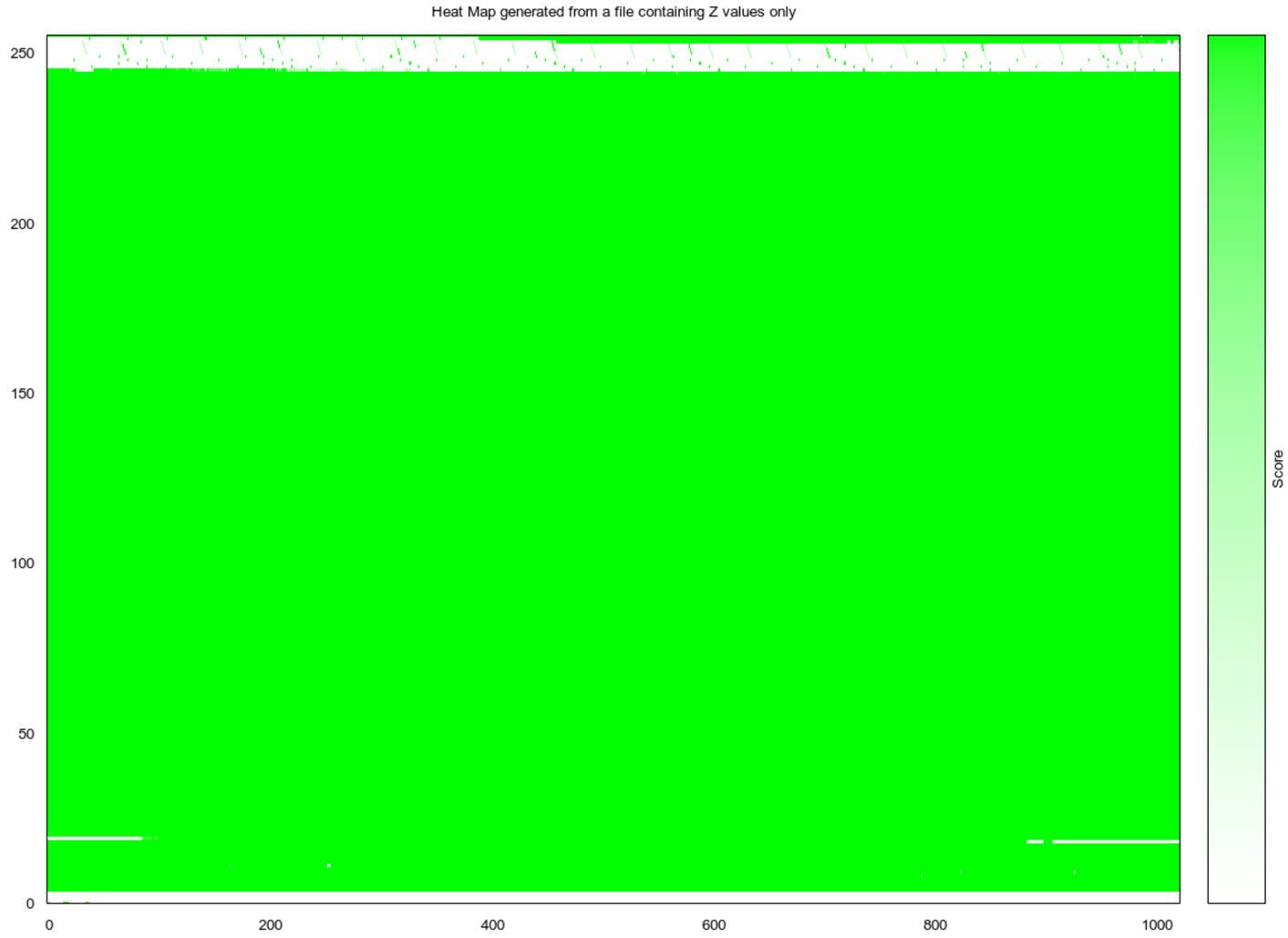
SEV enabled

*


Heat Map generated from a file containing Z values only
250
200
150
100
50
0
0
200
400
600
800
1000
Score

“Page zero hack”

*



AMD SEV

Limitations

- Problem:
  - vCPU state visible to (untrusted) hypervisor
  - Including extended FPU state (AES-NI)
- Solution:
  - SEV-ES
  - Encrypting vCPU state

Confidential Virtual Machines with OpenBSD

Agenda

- Introduction
- Memory encryption for Confidential VMs — SEV
- **vCPU state encryption — SEV-ES**
- Conclusion

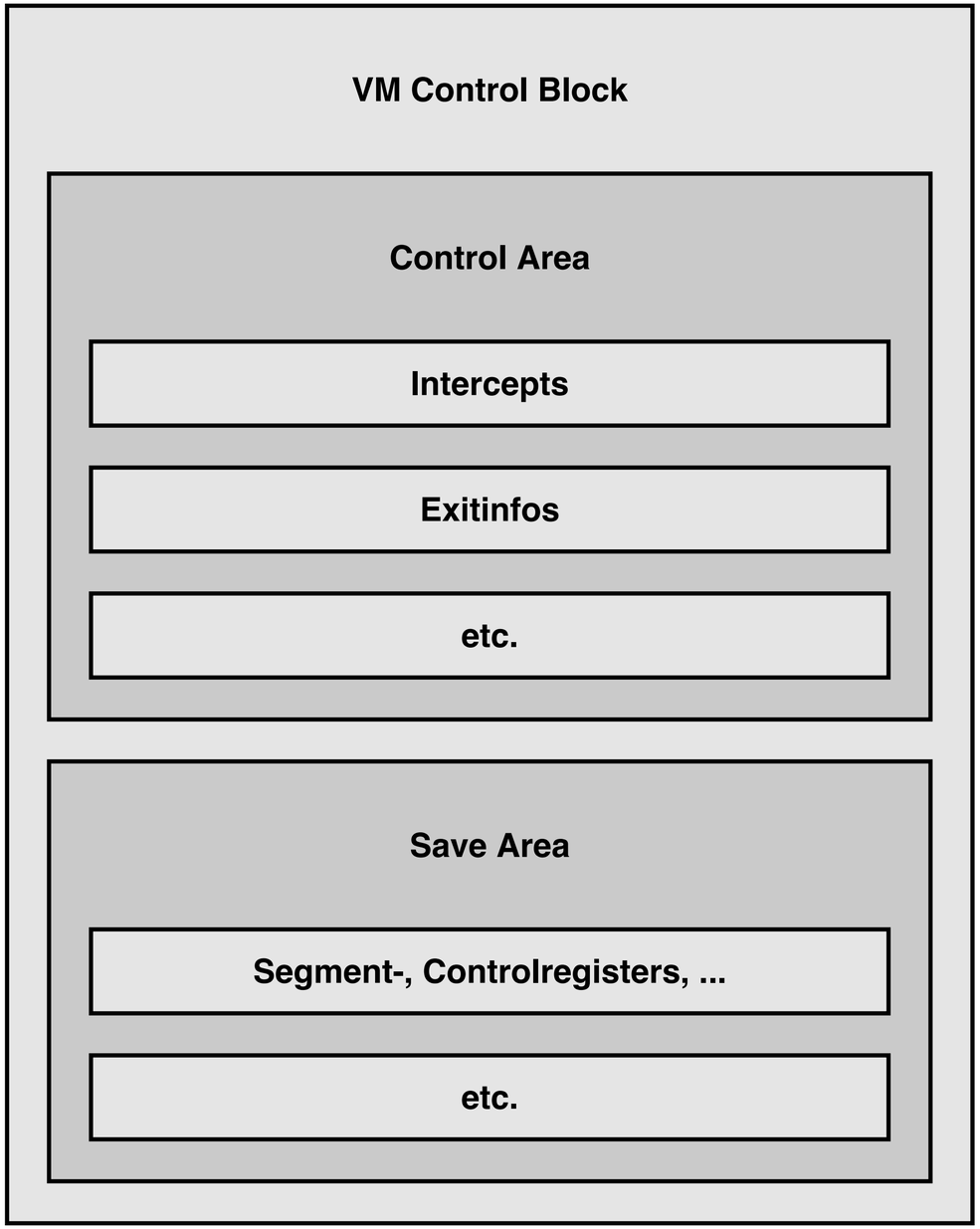
Saving and restoring state — SVM/SEV

VM Exit and Entry

- Regular SVM or SEV enabled VM:
  - Minimal state saved in VMCB
  - vmm(4) saves all remaining state in vCPU data structure
    - See exception/interrupt handling and stack frame

AMD SVM/SEV
VMCB

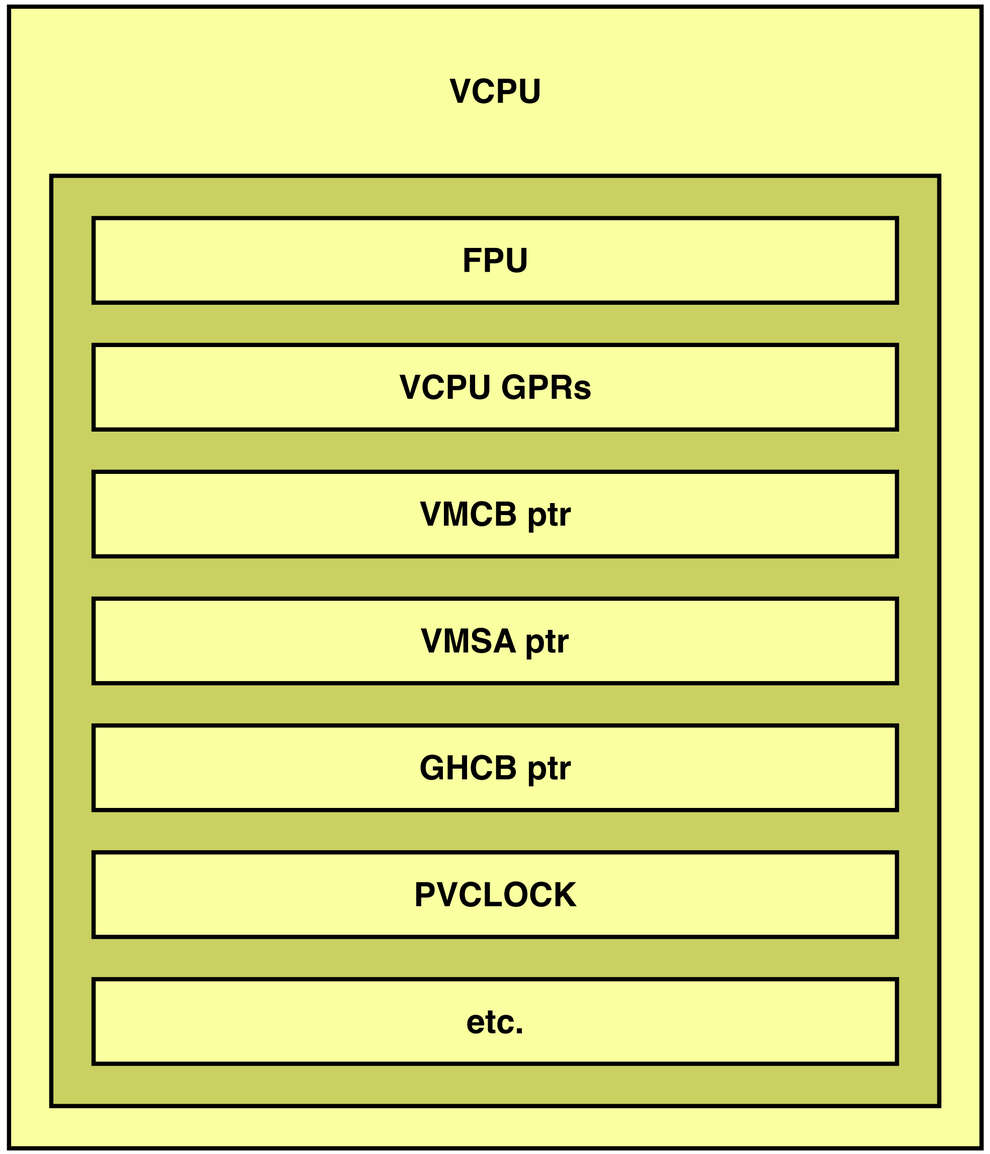
- Virtual Machine Control Block (VMCB)
  - Hardware defined
  - Control Area
  - Save Area
    - Minimal vCPU state
  - VMSAVE and VMLOAD



VMCB

AMD SVM/SEV
vCPU

- vCPU data structure
  - Software defined
  - Maintained by vmm(4)
  - vCPU state
  - Auxiliary data



SVM and SEV

VM Exit and Entry

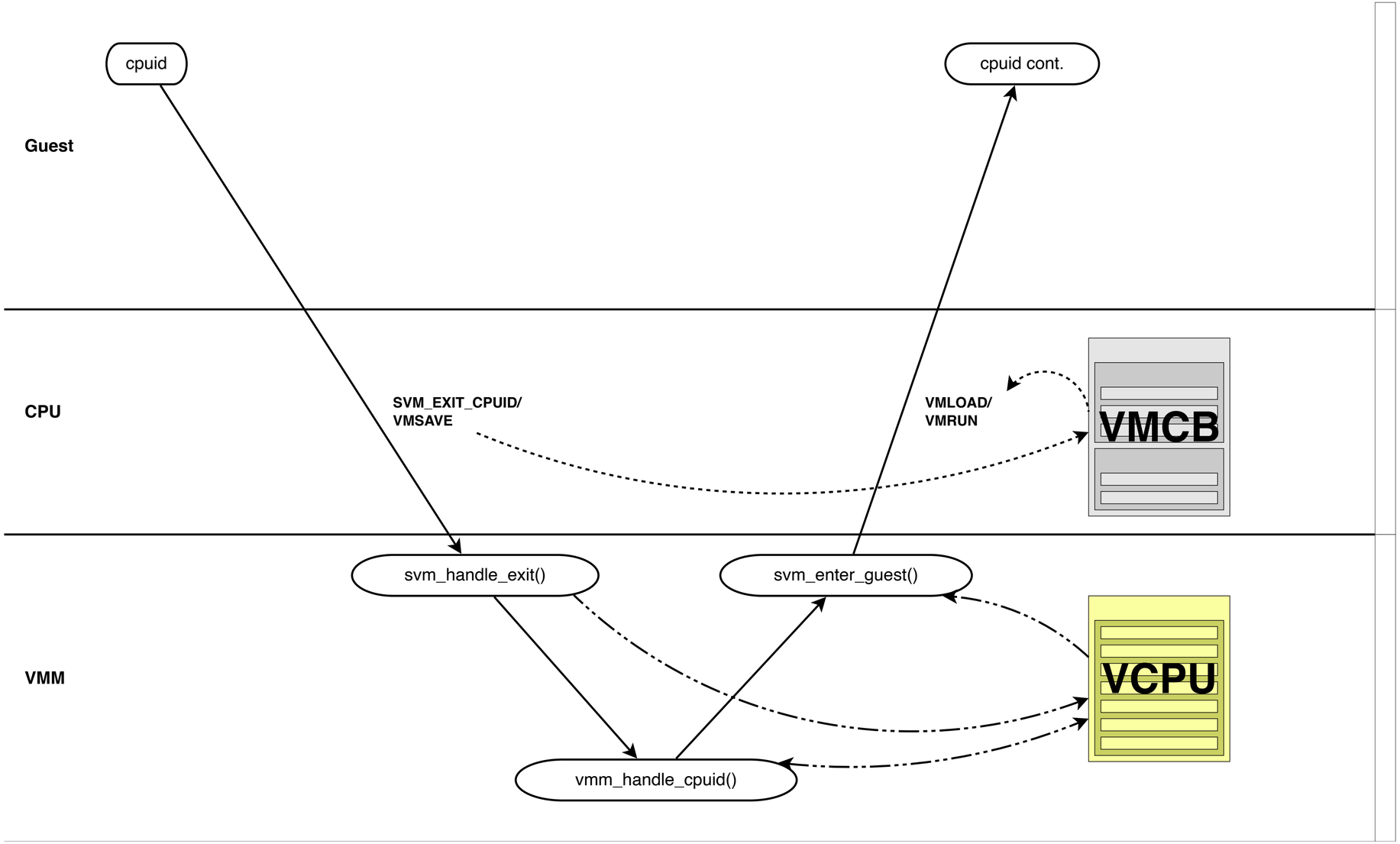
sys/arch/amd64/include/specialreg.h:

```
#define CPUID(code, eax, ebx, ecx, edx)                                 \
        __asm volatile("cpuid"                                          \
            : "=a" (eax), "=b" (ebx), "=c" (ecx), "=d" (edx)            \
            : "a" (code))
```

Code (aka function) 0:

- eax: Largest standard function
- ebx, ecx, edx: “AuthenticAMD”, “GenuineIntel”, …

VM Exit


cpuid
cpuid cont.
Guest
CPU
SVM_EXIT_CPUID/
VMSAVE
VMLOAD/
VMRUN
VMCB
svm_handle_exit()
svm_enter_guest()
VMM
VCPU
vmm_handle_cpuid()

Saving and restoring state — SEV-ES

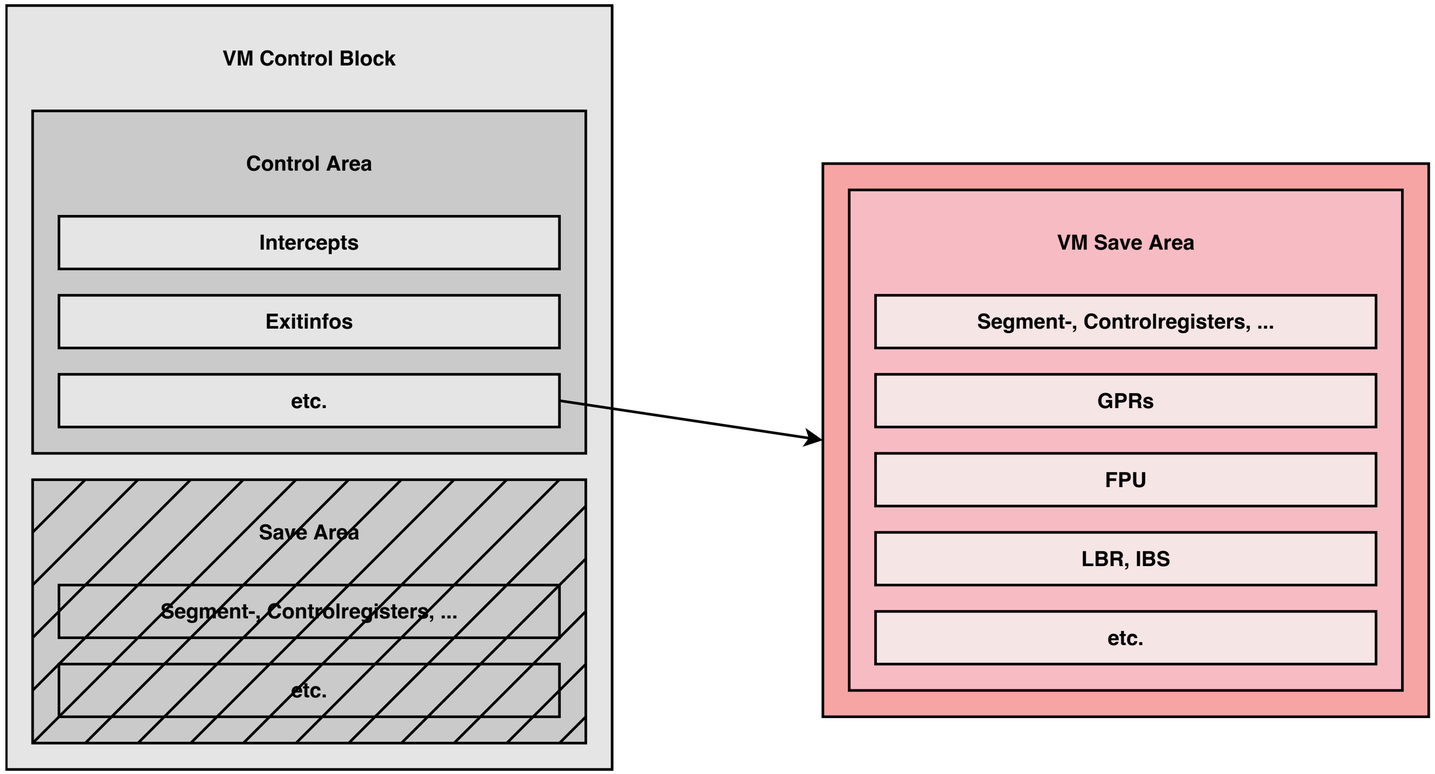
VM Exit and Entry

- SEV-ES enabled VM:
  - Full vCPU state saved automatically to encrypted VMSA
  - vCPU state invisible (encrypted) for vmm(4)
  - New #VC trap on exit events

AMD SEV-ES

VMSA

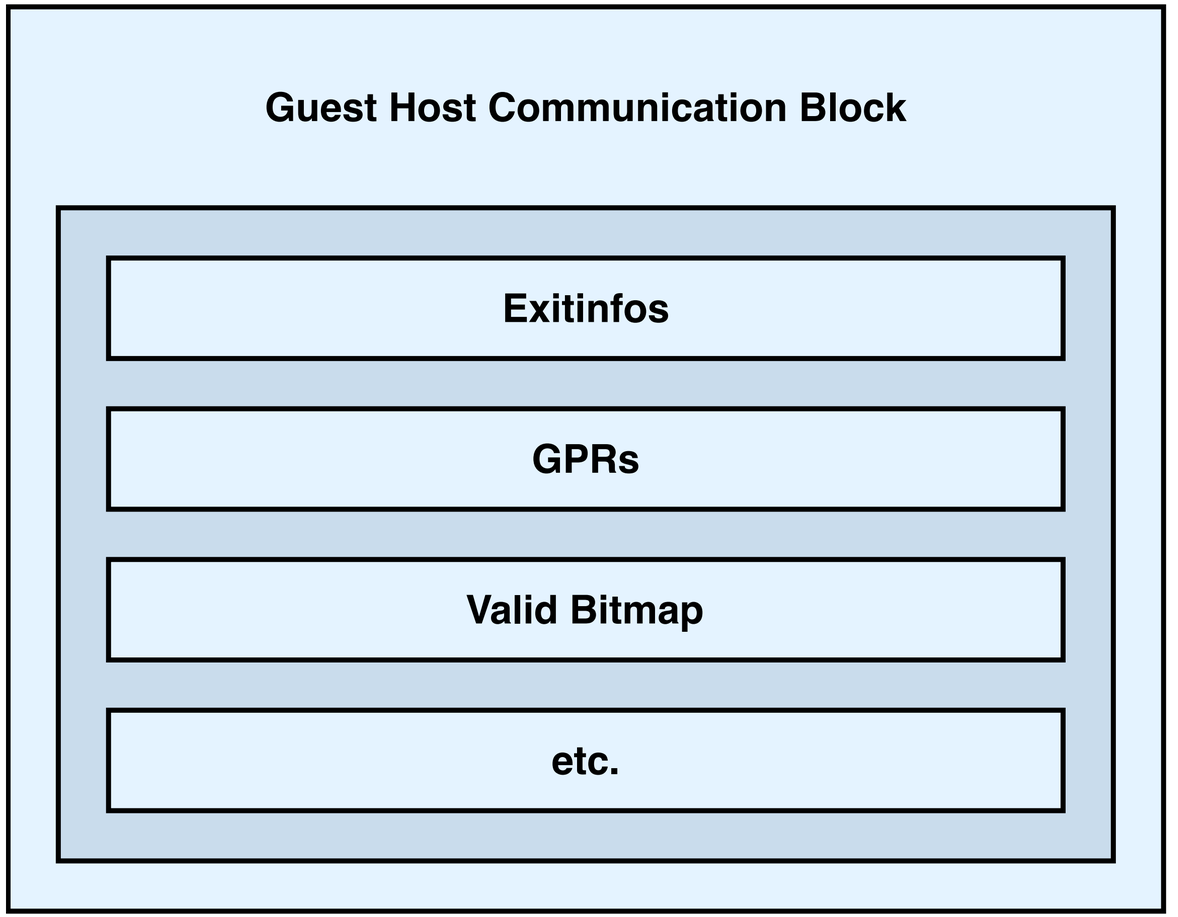
- Virtual Machine Save Area
  - Maintained by CPU
  - Full vCPU state
  - Encrypted
- “Swapped” with host state



#VC Trap

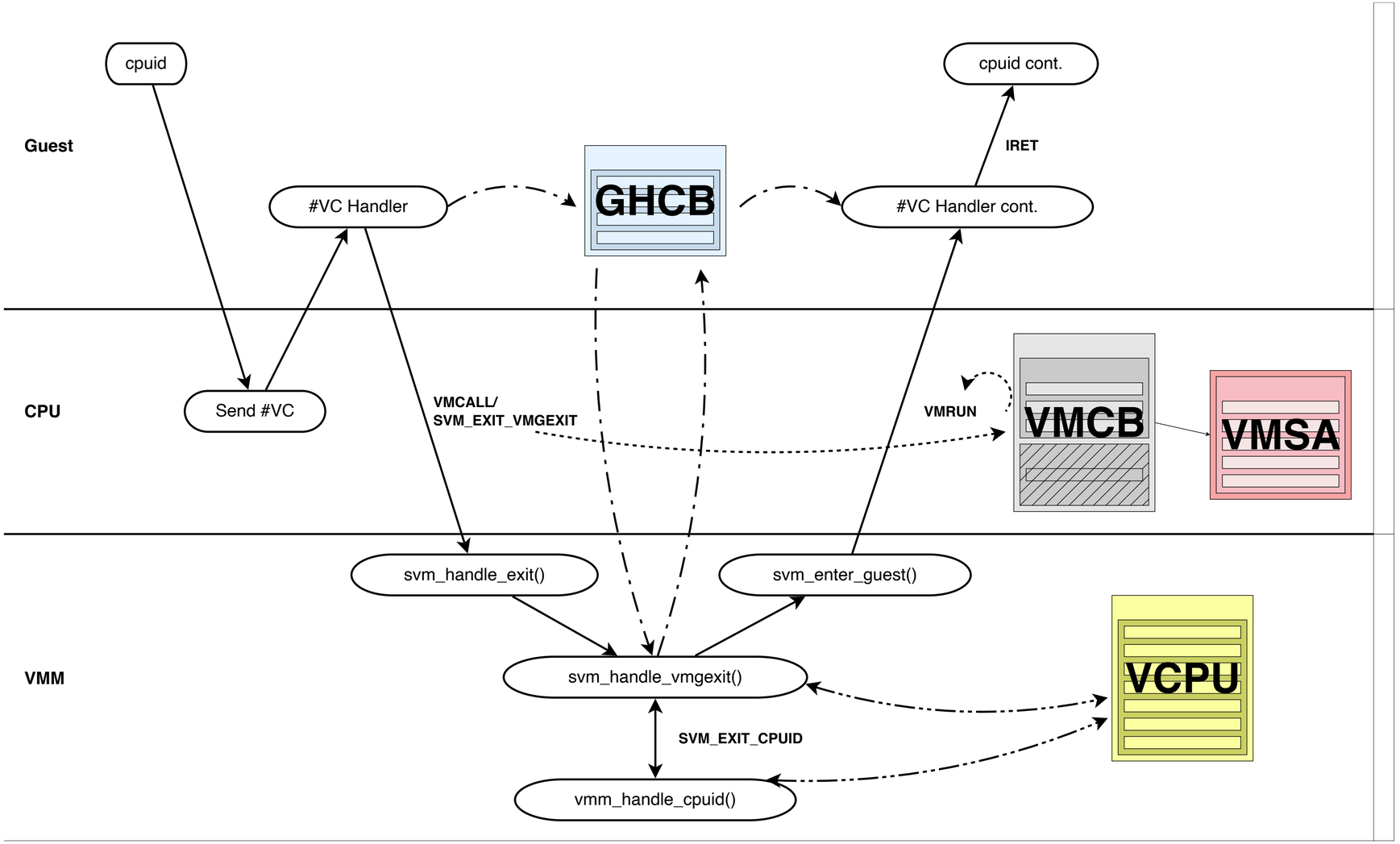
Non-Automatic VM Exits

- VM Exit redirected to #VC trap handler
- Guest decides on what vCPU state to be shared with vmm(4)
- Software defined Guest Host Communication Block (GHCB)
- Unencrypted memory shared by guest with vmm(4)
- GHCB MSR points to GPA of GHCB
- GHCB MSR visible in VMCB @0xA0



GHCB

NAE VM Exit


cpuid
cpuid cont.
Guest
IRET
#VC Handler
GHCB
#VC Handler cont.
CPU
Send #VC
VMCALL/
SVM_EXIT_VMGEXIT
VMRUN
VMCB
VMSA
svm_handle_exit()
svm_enter_guest()
VMM
svm_handle_vmgexit()
VCPU
SVM_EXIT_CPUID
vmm_handle_cpuid()

AMD SEV-ES

Limitations

- Problem:
  - No integrity protection for guest memory
  - Malicious event injection
- Solution:
  - SEV-SNP
  - IOMMU with SNP
  - Limited event injection

Confidential Virtual Machines with OpenBSD

Agenda

- Introduction
- Memory encryption for Confidential VMs — SEV
- vCPU state encryption — SEV-ES
- **Conclusion**

Confidential Computing with OpenBSD

Conclusion

- Goals (November 2023):
  - Support for AMD SEV-*
  - psp(4), vmd(8), vmm(4), GENERIC
  - OpenBSD as both hypervisor and guest
  - Compatibility with Linux/KVM
- Status:
  - ☑ SEV — OpenBSD 7.6 (October 2024)
  - ☑ SEV-ES — OpenBSD 7.8 (October 2025)
  - ☐ SEV-SNP — work in progress

Conclusion

Next steps and beyond

- Next goals
  - IOMMU support for SEV-SNP
  - vmm(4)/vmd(8) support for SEV-SNP
  - OpenBSD SEV-SNP guest already running on Linux/KVM host
  - Performance?
  - Attestation?
  - Security?
  - ...

Thank you!

Questions?

Don’t forget to remember!