SEVurity: No Security Without Integrity

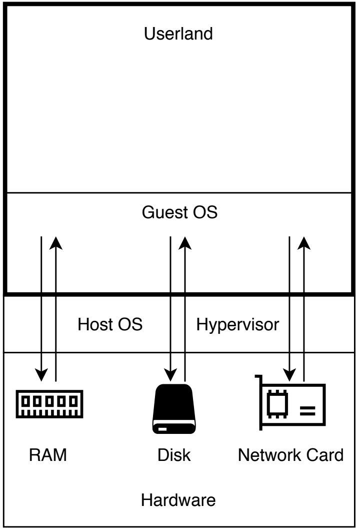
Luca Wilke [1] Jan Wichelmann [1] Mathias Morbitzer [2]
Thomas Eisenbarth [1]

[1] University of Lübeck

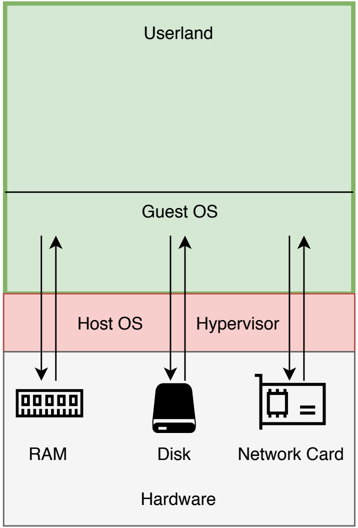
[2] Fraunhofer AISEC München

IEEE S&P 20.05.2020

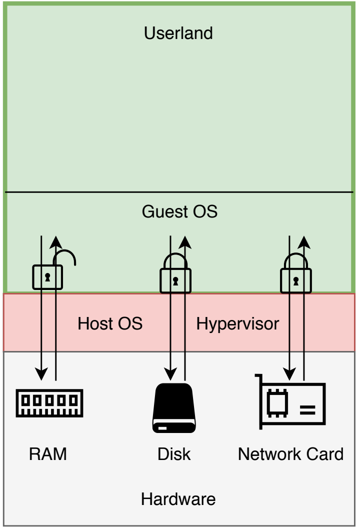
Plain VM setup



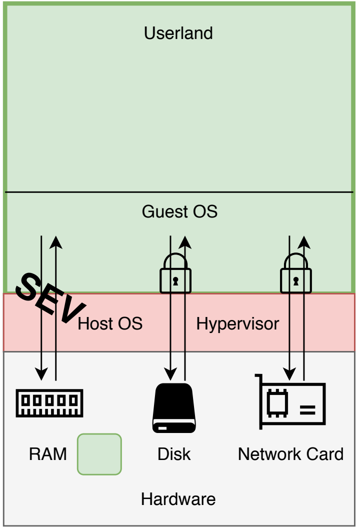
... has trust issues.



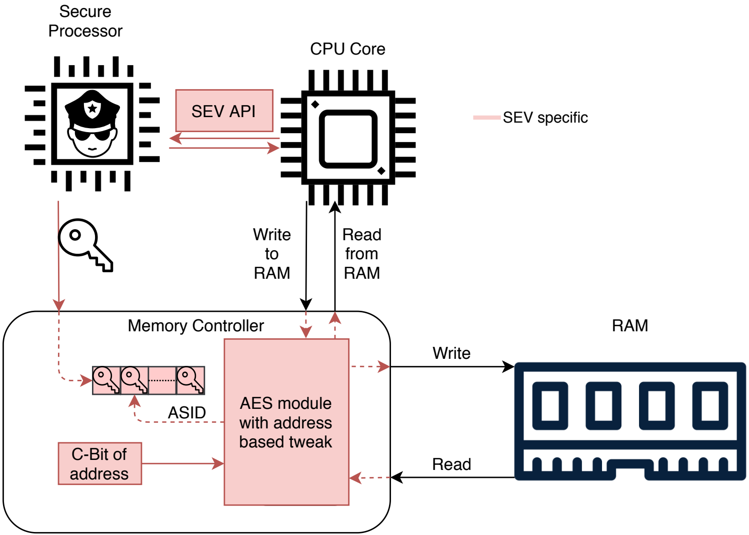
… has trust issues.



SEV to the rescue?



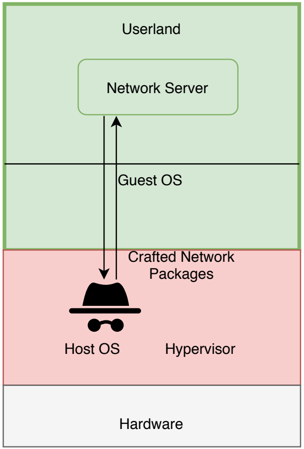
SEV Architecture



Roadmap

- Encryption mode analysis:
  - Contribution: Reverse engineered updated encryption mode
- Injection attack
  - Goal: Build encryption oracle for SEV-ES
  - Contribution: No control over I/O required ; minimal assumptions on the VM

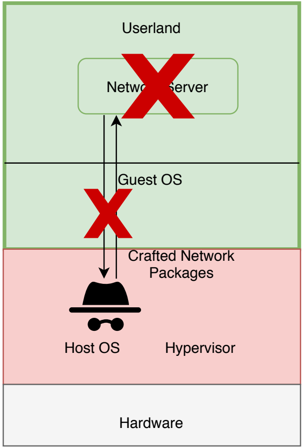
Prior attacks[1]



- Attacker needs to send (crafted) network packages
  ⇒ Increased risk of detection

[1] Zhao-Hui Du et al. “Secure encrypted virtualization is unsecure”. In: *arXiv:1712.05090* (2017); Mengyuan Li, Yinqian Zhang, and Zhiqiang Lin. “Exploiting Unprotected I/O Operations in AMD’s Secure Encrypted Virtualization”. In: *28th USENIX Security Symposium*. 2019.

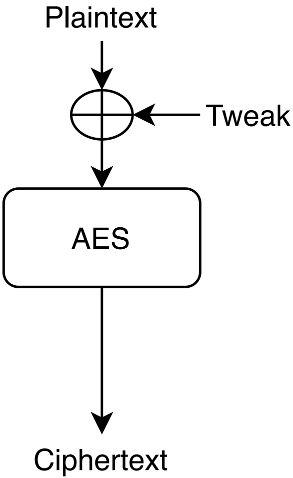
Our attack

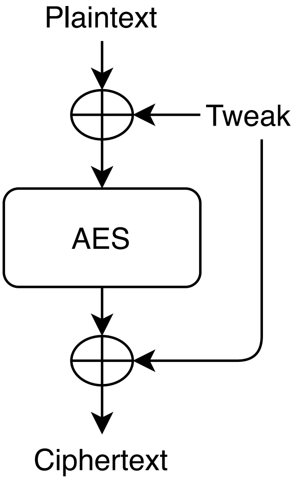


- No dependencies on services inside the VM
- No control over I/O operations required
  $\Rightarrow$ stealthy

Encryption Mode

Encryption modes





Tweak function

Tweak constant	Value (16 Byte)
t_4	`82 25 38 38 ...`
t_5	`ec 09 9c ec ...`
$\vdots$	$\vdots$
t_{12}	`b0 92 30 c2 ...`
$\vdots$	$\vdots$

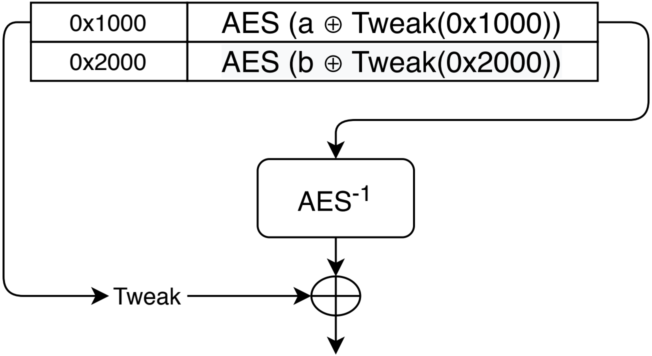
$$\text{Tweak}(0x1000) = t_{12}$$

$$\text{Tweak}(0x1010) = t_{12} \oplus t_4$$

Injection Attack

Injecting values into the VM

Goal: Manipulate data read by the VM

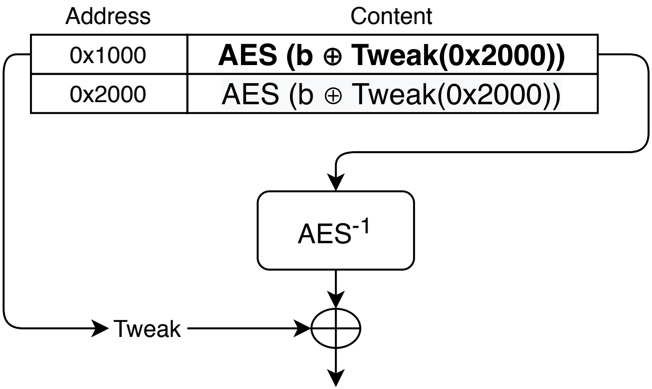


Address	Content
0x1000	AES (a ⊕ Tweak(0x1000))
0x2000	AES (b ⊕ Tweak(0x2000))

$a \oplus \text{Tweak}(0x1000) \oplus \text{Tweak}(0x1000)$

Injecting values into the VM

Goal: Manipulate data read by the VM



Address	Content
0x1000	**AES (b ⊕ Tweak(0x2000))**
0x2000	AES (b ⊕ Tweak(0x2000))

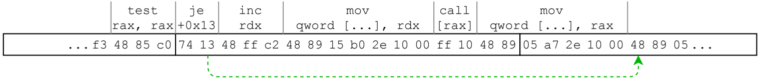
b ⊕ **Tweak(0x2000)** ⊕ **Tweak(0x1000)**

Injecting values into the VM

- Using the guest kernel as a known plaintext source gives us control over 2 bytes
- Upper limit is 4 bytes, due to tweak periodicity

Two bytes can bite

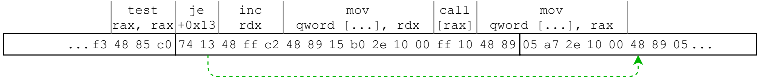
- Skip code with relative jumps:
  . . . *if(suppliedPw != correctPw)* { . . . *abort();* . . . } . . .



before injection

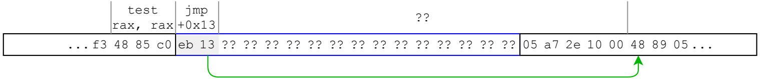
Two bytes can bite

- Skip code with relative jumps:
  . . . *if(suppliedPw != correctPw)* { . . . *abort();* . . . } . . .



```
 test      je     inc          mov              call          mov
 rax, rax +0x13   rdx    qword [...], rdx      [rax]  qword [...], rax
...f3 48 85 c0|74 13 48 ff c2 48 89 15 b0 2e 10 00 ff 10 48 89|05 a7 2e 10 00 48 89 05...
```

before injection

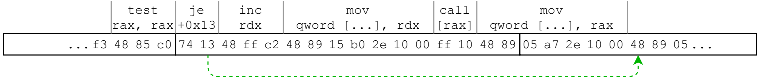


```
 test      jmp
 rax, rax +0x13                         ??
...f3 48 85 c0|eb 13 ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ??|05 a7 2e 10 00 48 89 05...
```

after injection

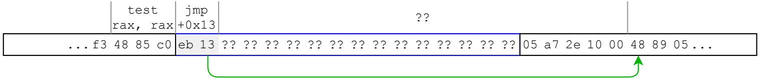
Two bytes can bite

- Skip code with relative jumps:
  . . . *if(suppliedPw != correctPw)* { . . . *abort();* . . . } . . .



```
              test   |  je  | inc  |          mov          | call |          mov
           rax, rax|+0x13 | rdx  | qword [...], rdx      |[rax]| qword [...], rax
...f3 48 85 c0|74 13 48 ff c2 48 89 15 b0 2e 10 00 ff 10 48 89|05 a7 2e 10 00 48 89 05...
```

before injection

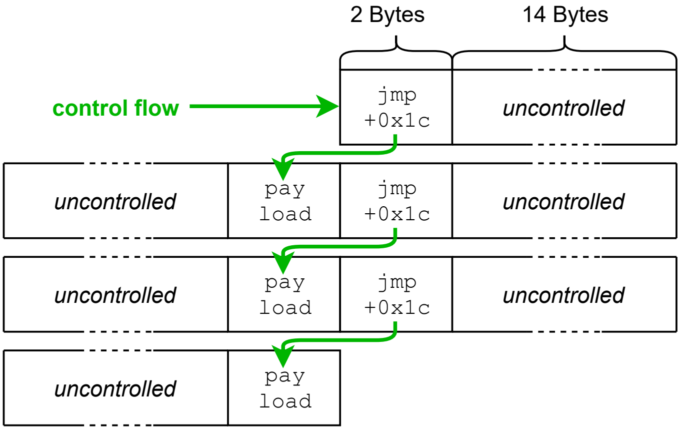


```
              test   | jmp  |
           rax, rax|+0x13 |                 ??
...f3 48 85 c0|eb 13 ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ??|05 a7 2e 10 00 48 89 05...
```

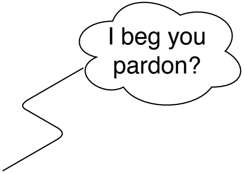
after injection

- Abort functions early by inserting a *ret* instruction:
  . . . *SampleRandomness();* . . . *doCrypto();* . . .

Complex injections



Complex injections


I beg you pardon?

```
movabs rax,0xffff873084739495
```

```
48 b8 95 94 73 84 30 87 ff ff
```

From 2 bytes to 16 bytes

- Big Idea: Inject simple program that ”calculates” complex values
  1. Get data into register:

From 2 bytes to 16 bytes

- Big Idea: Inject simple program that ”calculates” complex values
    1. Get data into register:
       while(rax != 0x9a842f) { inc rax }

From 2 bytes to 16 bytes

- Big Idea: Inject simple program that ”calculates” complex values
  1. Get data into register:
     ~~*while(rax != 0x9a842f) { inc rax }*~~
     while(true) { inc rax ; notify HV }
  2. Get data into RAM:
     push rax

From 2 bytes to 16 bytes

- Big Idea: Inject simple program that ”calculates” complex values
  1. Get data into register:
     ~~*while(rax != 0x9a842f) { inc rax }*~~
     while(true) { inc rax ; notify HV }
  2. Get data into RAM:
     push rax

⇒ 16 byte encryption oracle ⇒ arbitrary code execution

Countermeasures

- XEX mode with stronger tweak function
  - Seems to be the case for Zen2
- Integrity protection
  - Does not seem to be planned. Future extension SEV-SNP will instead prohibit the HV from writing to VM memory

Summary

- Scenario: Malicious hypervisor
- Encryption mode analysis
  - AES with static, low entropy tweak
  - No integrity protection or freshness
  - Discovered updated XEX mode
- Injection attack: Encryption oracle for SEV-ES
  1. Use guest kernel as known plaintext source
  2. Move ciphertext blocks to get control of 2 bytes
  3. Bootstrap 16 byte encryption oracle

     ⇒ Execute arbitrary code

Thanks for your attention!
Contact: l.wilke@uni-luebeck.de

UNIVERSITÄT ZU LÜBECK

Fraunhofer AISEC

UzL-ITS/SEVurity

@lucawilkeUzL