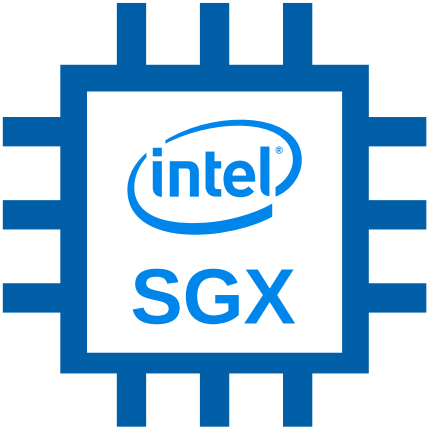
Practical Enclave Malware with Intel SGX

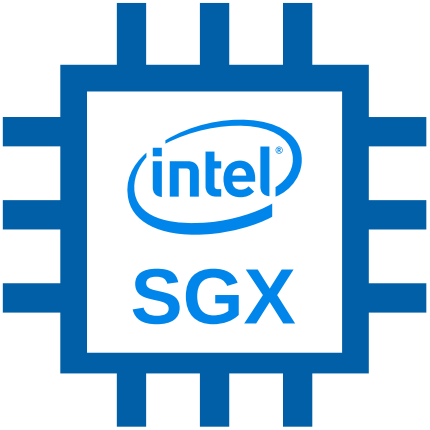
Michael Schwarz, Samuel Weiser, Daniel Gruss

June 20, 2019 - DIMVA'19

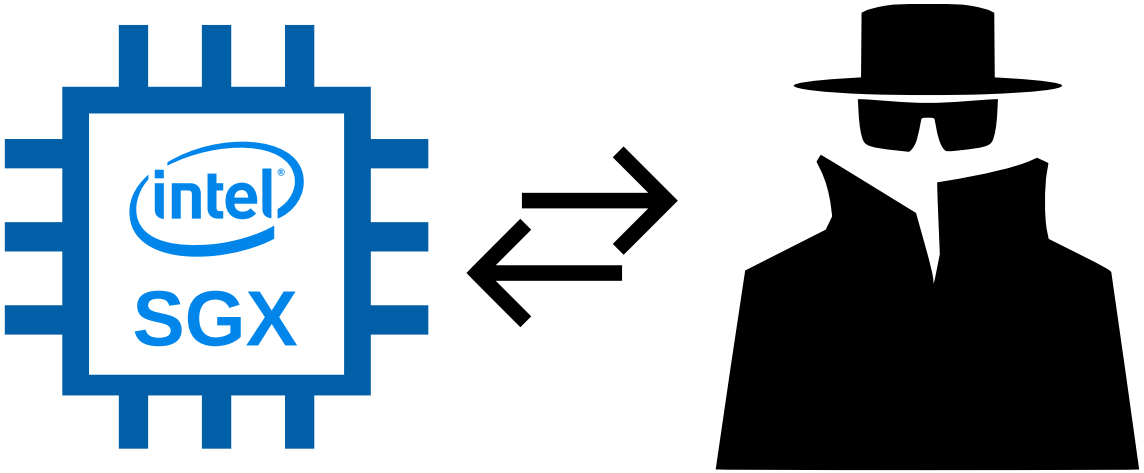
Graz University of Technology

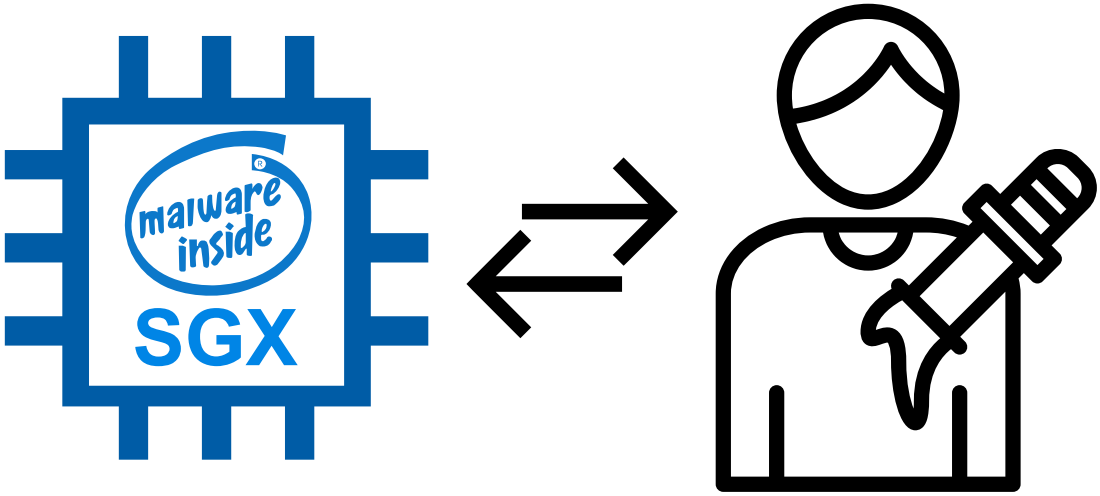
Outline

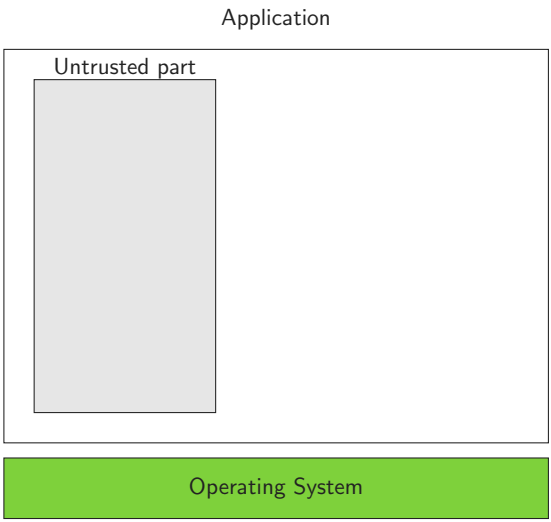


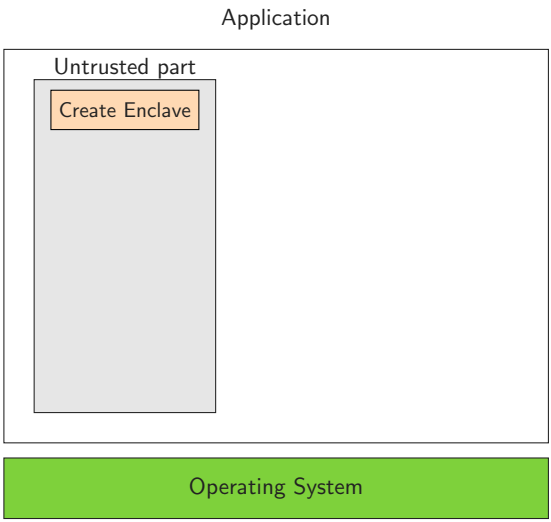

intel
SGX

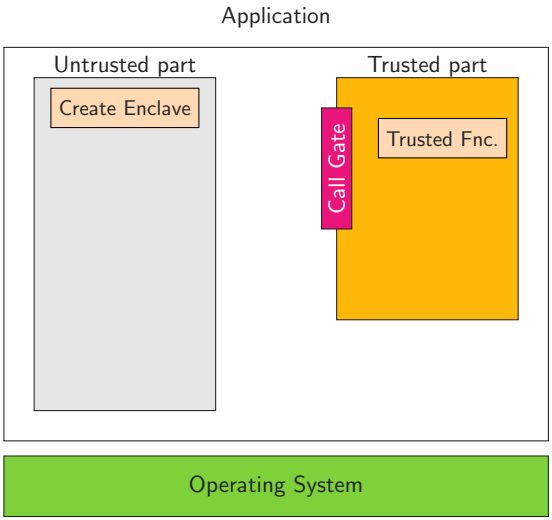
Outline

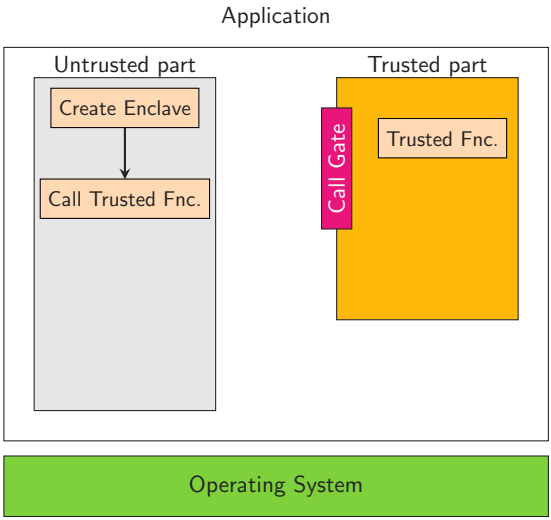


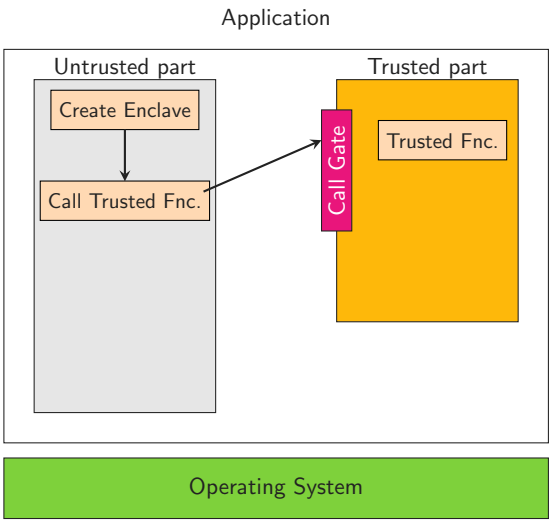

malware
inside
SGX

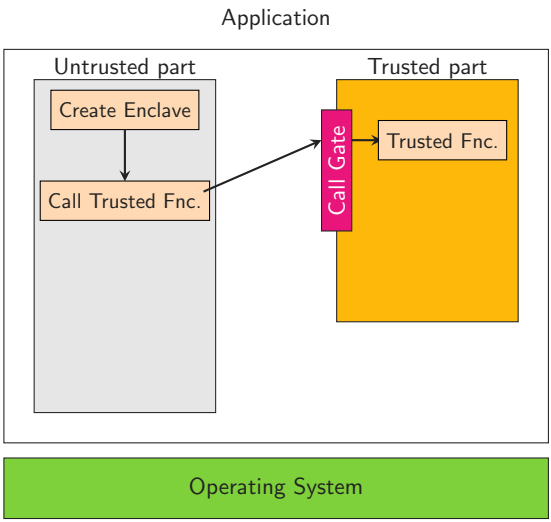

Application
Untrusted part
Operating System

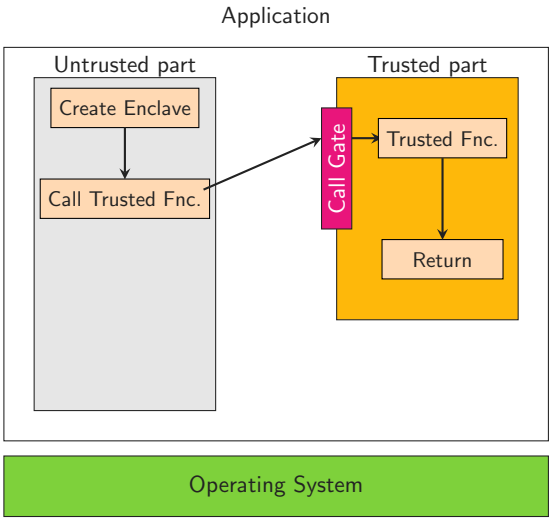

Application
Untrusted part
Create Enclave
Operating System

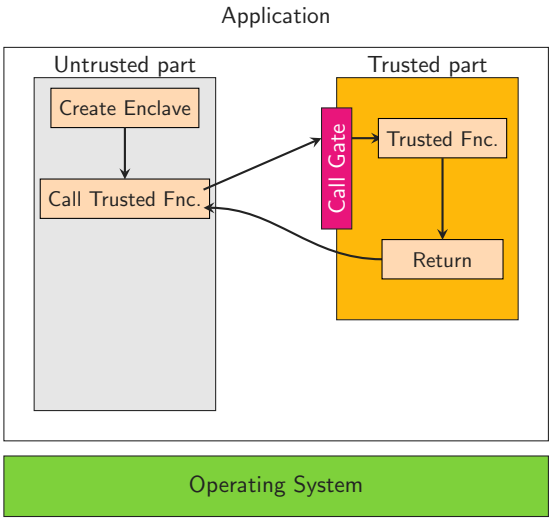

Application
Untrusted part
Create Enclave
Trusted part
Call Gate
Trusted Fnc.
Operating System

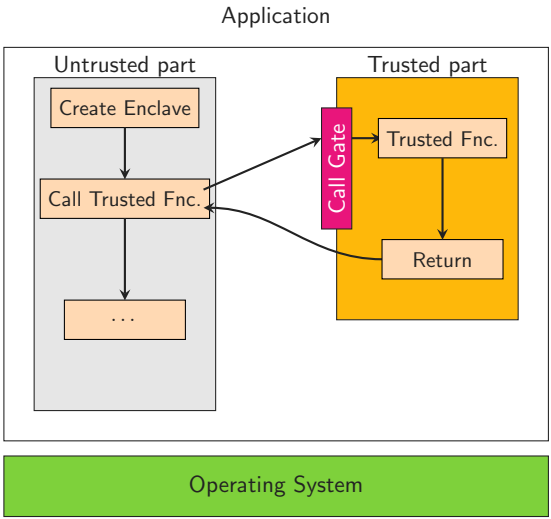

Application
Untrusted part
Create Enclave
Call Trusted Fnc.
Trusted part
Call Gate
Trusted Fnc.
Operating System

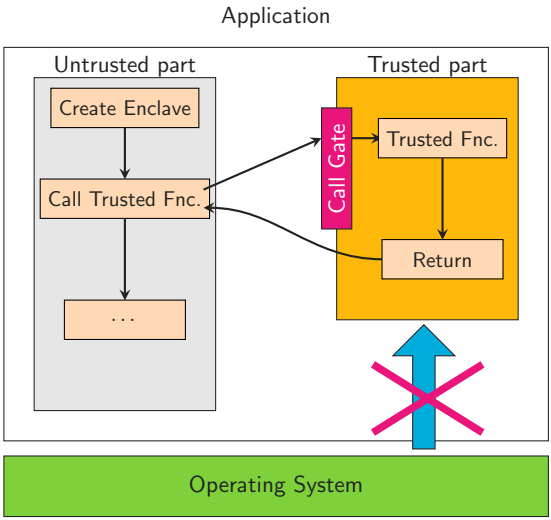

Application
Untrusted part
Create Enclave
Call Trusted Fnc.
Trusted part
Call Gate
Trusted Fnc.
Operating System


Application
Untrusted part
Trusted part
Create Enclave
Call Trusted Fnc.
Call Gate
Trusted Fnc.
Operating System


Application
Untrusted part
Trusted part
Create Enclave
Call Trusted Fnc.
Call Gate
Trusted Fnc.
Return
Operating System


Application
Untrusted part
Trusted part
Create Enclave
Call Trusted Fnc.
Call Gate
Trusted Fnc.
Return
Operating System


Application
Untrusted part
Trusted part
Create Enclave
Call Trusted Fnc.
...
Call Gate
Trusted Fnc.
Return
Operating System


Application
Untrusted part
Trusted part
Create Enclave
Call Trusted Fnc.
...
Call Gate
Trusted Fnc.
Return
Operating System

What if?

- Enclaves are black boxes

What if?

- Enclaves are black boxes
- Protected from all applications and OS

What if?

- Enclaves are black boxes
- Protected from all applications and OS
- What if they contain malicious code?

What if?

- Enclaves are black boxes
- Protected from all applications and OS
- What if they contain malicious code?
- Can we hide zero days?

Intel's Statement

[...] Intel is aware of this research which is based upon assumptions that are outside the threat model for Intel SGX. The value of Intel SGX is to execute code in a protected enclave; however, Intel SGX does not guarantee that the code executed in the enclave is from a trusted source [...]

Classical exploits cannot be mounted within SGX:

Classical exploits cannot be mounted within SGX:

- No syscalls

Classical exploits cannot be mounted within SGX:

- No syscalls
- No shared memory/libraries

Classical exploits cannot be mounted within SGX:

- No syscalls
- No shared memory/libraries
- No interprocess communication

Classical exploits cannot be mounted within SGX:

- No syscalls
- No shared memory/libraries
- No interprocess communication
- Blocked instructions

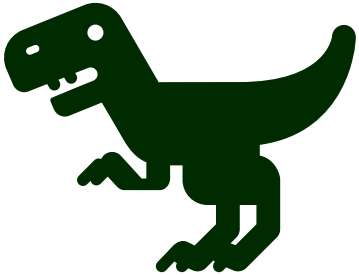
State-of-the-art Malicious Enclaves

- Side-channel attacks from SGX [Sch+17]

- Side-channel attacks from SGX [Sch+17]
- Fault attacks from SGX [Gru+18; Jan+17]

State-of-the-art Malicious Enclaves

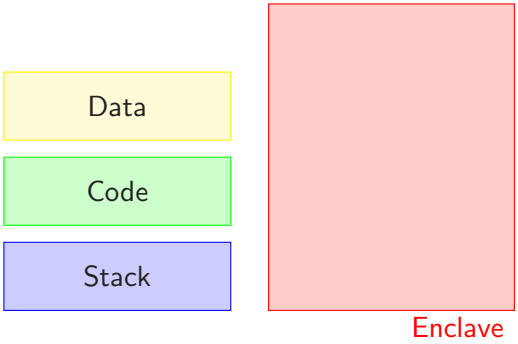
- Side-channel attacks from SGX [Sch+17]
- Fault attacks from SGX [Gru+18; Jan+17]
- No real exploits from SGX so far



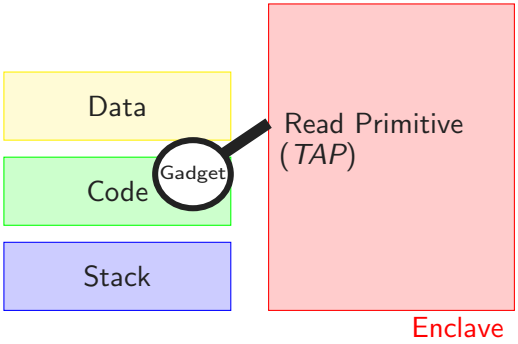
TEE-REX

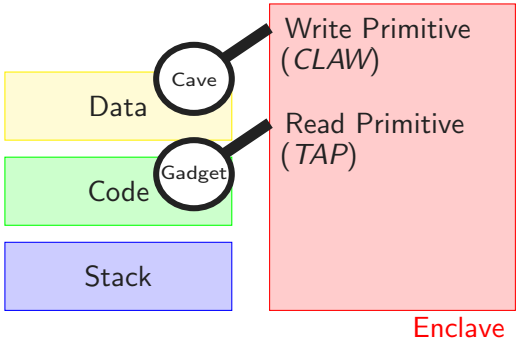
Trusted **E**xecution **E**nvironment **R**eturn-oriented-programming **EX**ploit

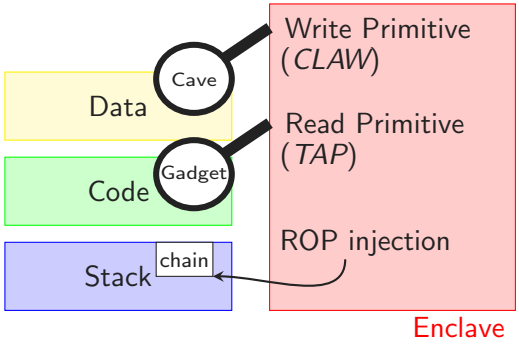
Attack Overview

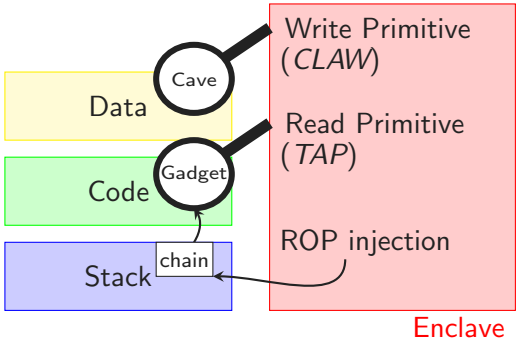


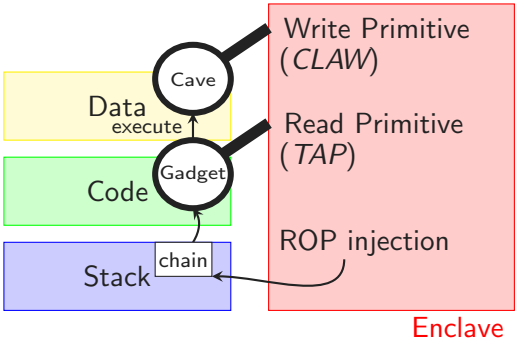
Attack Overview

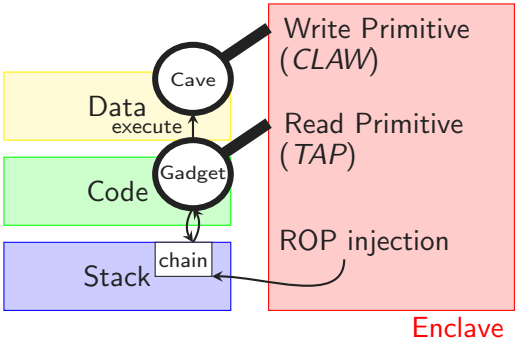



Write Primitive
(*CLAW*)
Cave
Data
Read Primitive
(*TAP*)
Gadget
Code
Stack
Enclave


Write Primitive
(*CLAW*)
Cave
Data
Read Primitive
(*TAP*)
Gadget
Code
ROP injection
chain
Stack
Enclave


Write Primitive
(*CLAW*)
Cave
Data
Read Primitive
(*TAP*)
Gadget
Code
ROP injection
chain
Stack
Enclave

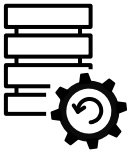

Write Primitive
(CLAW)
Cave
Data
execute
Read Primitive
(TAP)
Gadget
Code
ROP injection
chain
Stack
Enclave


Write Primitive
(CLAW)
Cave
Data
execute
Read Primitive
(TAP)
Gadget
Code
ROP injection
chain
Stack
Enclave

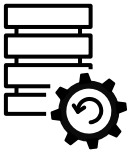
- Enclave can access host memory...

- Enclave can access host memory...
- ...but crashes on invalid access

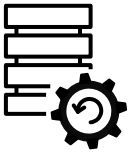
- Enclave can access host memory...
- ...but crashes on invalid access
- No syscall or exception handler available



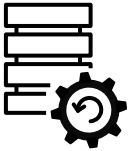
- Intel TSX: hardware transactional memory



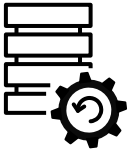
- Intel TSX: hardware transactional memory
- Multiple reads and writes are atomic



- Intel TSX: hardware transactional memory
- Multiple reads and writes are atomic
- Operations in a transaction



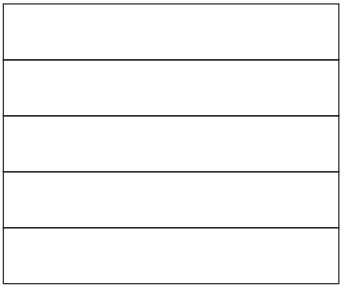
- Intel TSX: hardware transactional memory
- Multiple reads and writes are atomic
- Operations in a transaction
- Conflict → abort and roll back



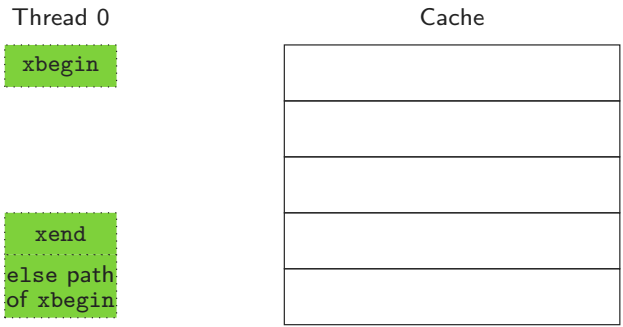
- Intel TSX: hardware transactional memory
- Multiple reads and writes are atomic
- Operations in a transaction
- Conflict → abort and roll back
- Faults are suppressed

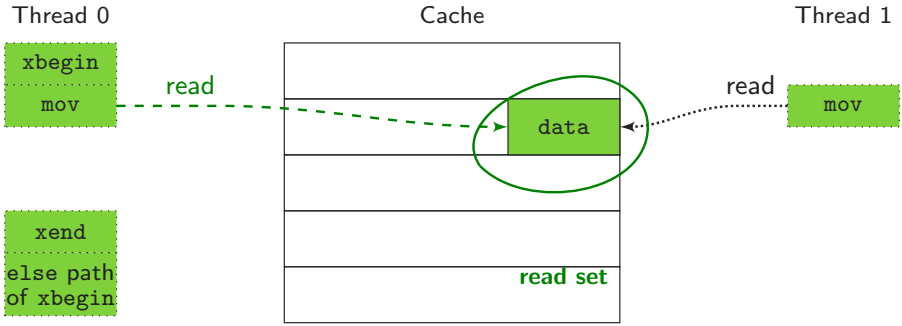
Thread 0

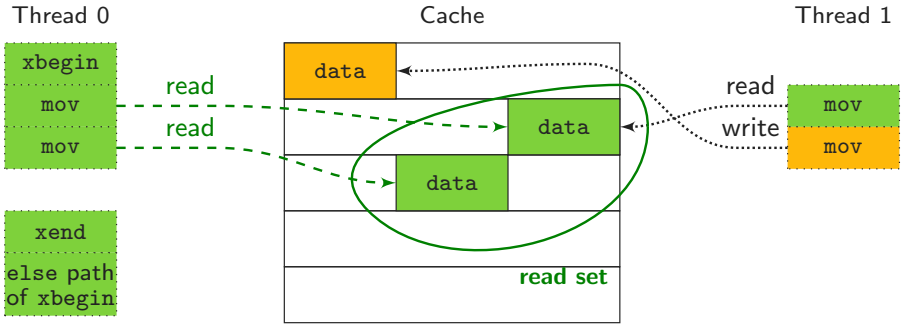
Cache

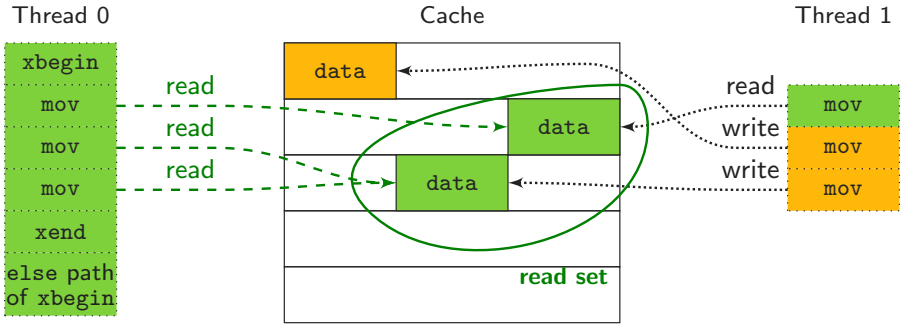


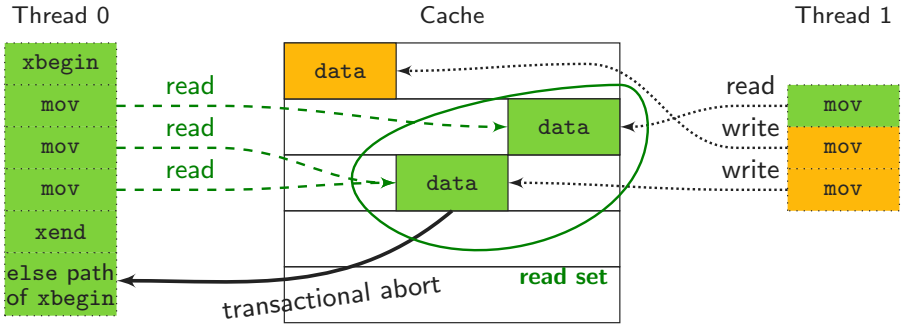
Thread 1


Thread 0
Cache
Thread 1
xbegin
xend
else path
of xbegin


Thread 0
Cache
Thread 1
xbegin
mov
read
read
mov
data
xend
else path
of xbegin
read set


Thread 0
Cache
Thread 1
xbegin
mov
mov
read
read
data
data
data
read
write
mov
mov
xend
else path
of xbegin
read set


Thread 0
Cache
Thread 1
xbegin
mov
mov
mov
xend
else path of xbegin
read
read
read
data
data
data
read set
read
write
write
mov
mov
mov

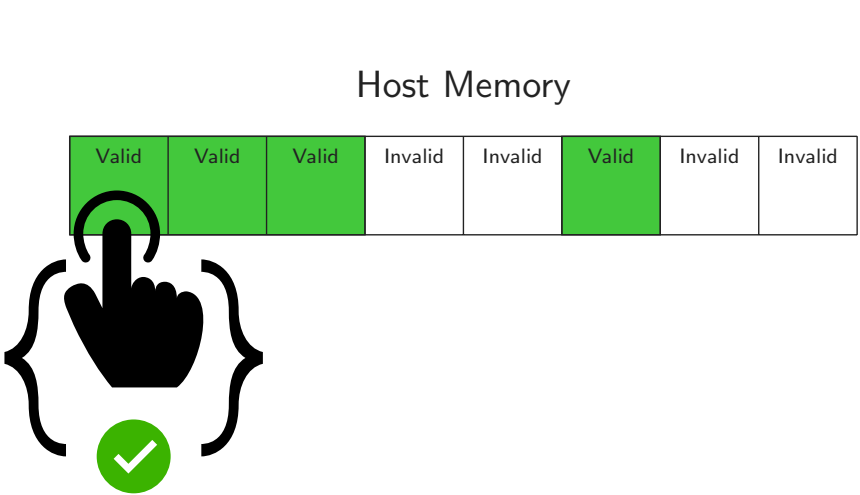

Thread 0
Cache
Thread 1
xbegin
mov
mov
mov
xend
else path of xbegin
read
read
read
data
data
data
read
write
write
mov
mov
mov
transactional abort
read set

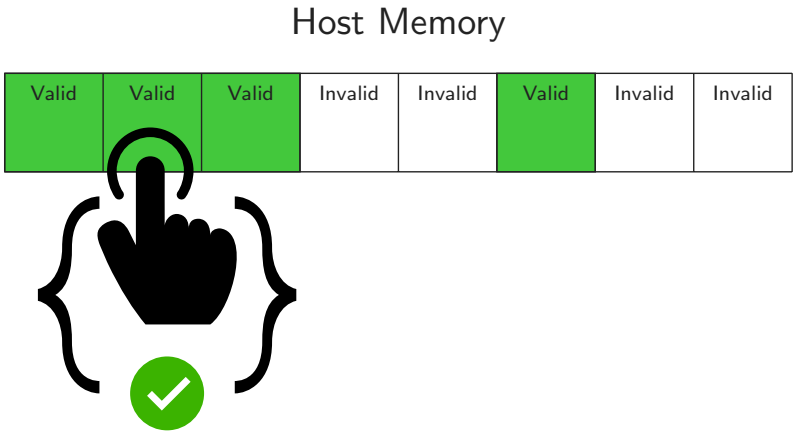
- Segmentation fault is a fault

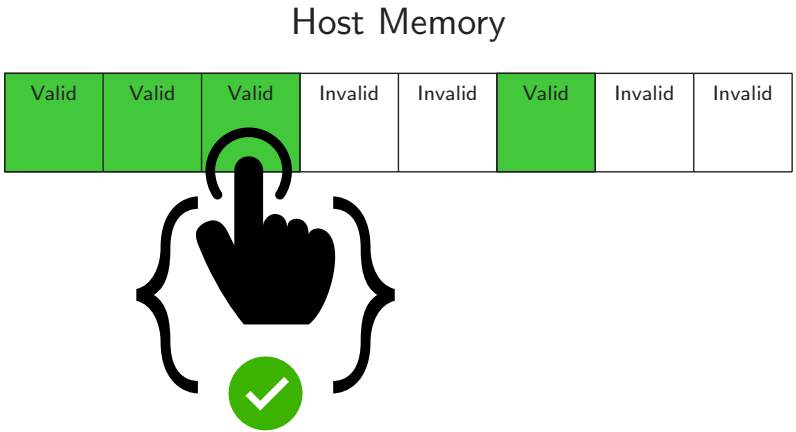
- Segmentation fault is a fault
- Suppressed in TSX transaction

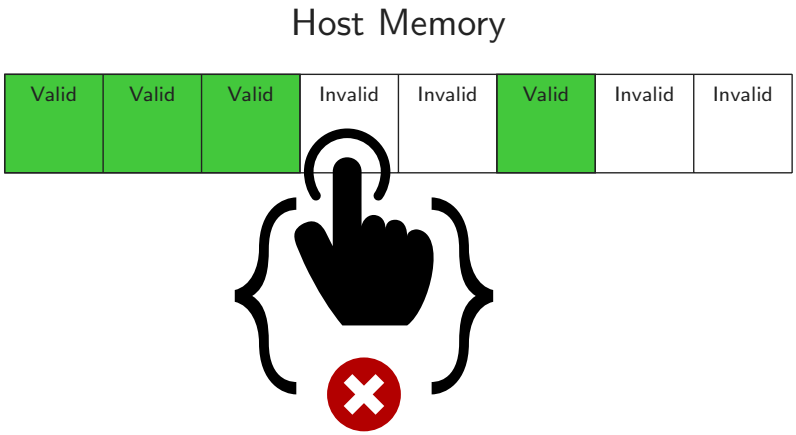
- Segmentation fault is a fault
- Suppressed in TSX transaction
- Abort code → "don't try again"

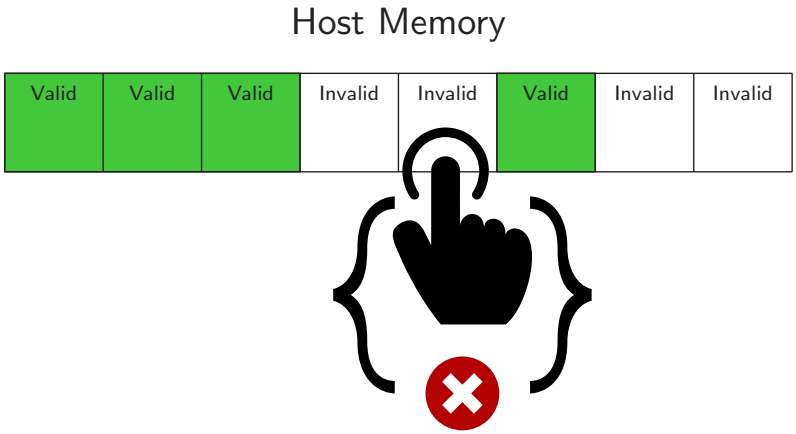
- Segmentation fault is a fault
- Suppressed in TSX transaction
- Abort code → "don't try again"
- Valid page → transaction succeeds

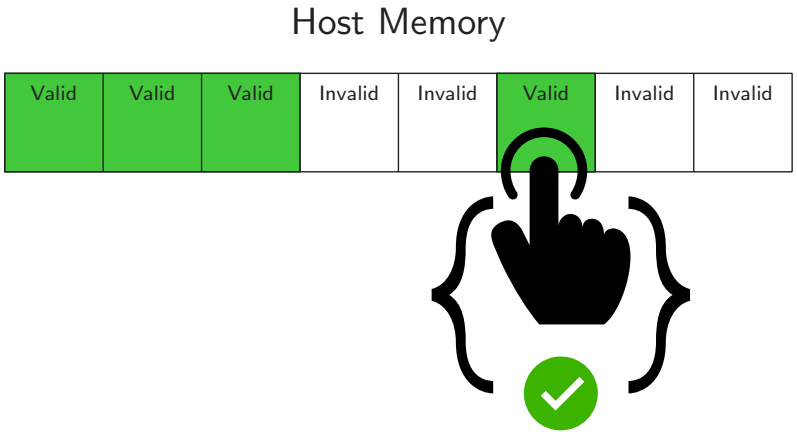

Host Memory
Valid
Valid
Valid
Invalid
Invalid
Valid
Invalid
Invalid

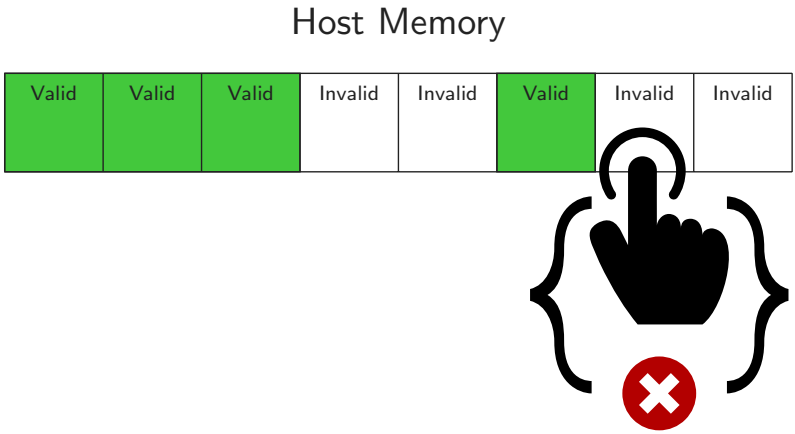

Host Memory
Valid
Valid
Valid
Invalid
Invalid
Valid
Invalid
Invalid


Host Memory
Valid
Valid
Valid
Invalid
Invalid
Valid
Invalid
Invalid

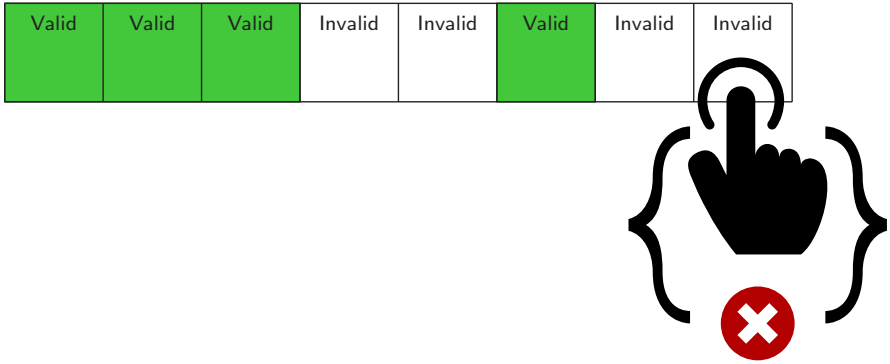

Host Memory
Valid
Valid
Valid
Invalid
Invalid
Valid
Invalid
Invalid


Host Memory
Valid
Valid
Valid
Invalid
Invalid
Valid
Invalid
Invalid


Host Memory
Valid
Valid
Valid
Invalid
Invalid
Valid
Invalid
Invalid


Host Memory
Valid
Valid
Valid
Invalid
Invalid
Valid
Invalid
Invalid

Host Memory



- Entire memory: 45 min

- Entire memory: 45 min
- Start from saved RIP/RSP: few seconds

- Entire memory: 45 min
- Start from saved RIP/RSP: few seconds
- Undetectable by OS

- Entire memory: 45 min
- Start from saved RIP/RSP: few seconds
- Undetectable by OS
- Used to find ROP gadgets

- Write to mapped page...

- Write to mapped page...
- ...abort immediately

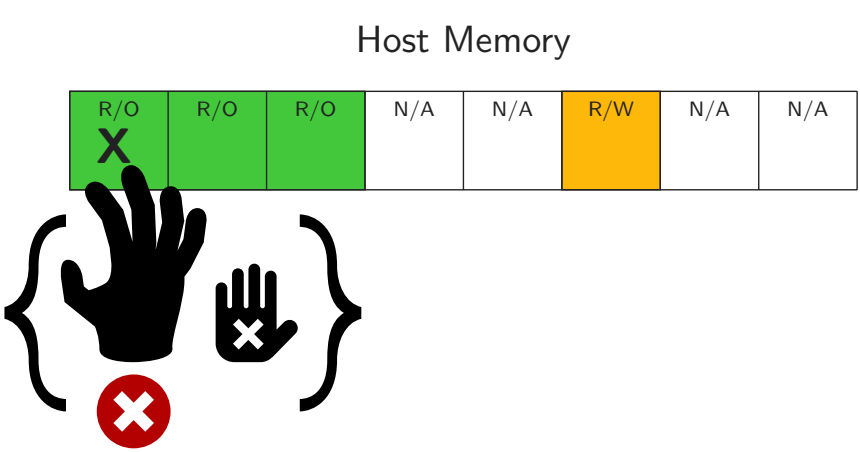
- Write to mapped page...
- ...abort immediately

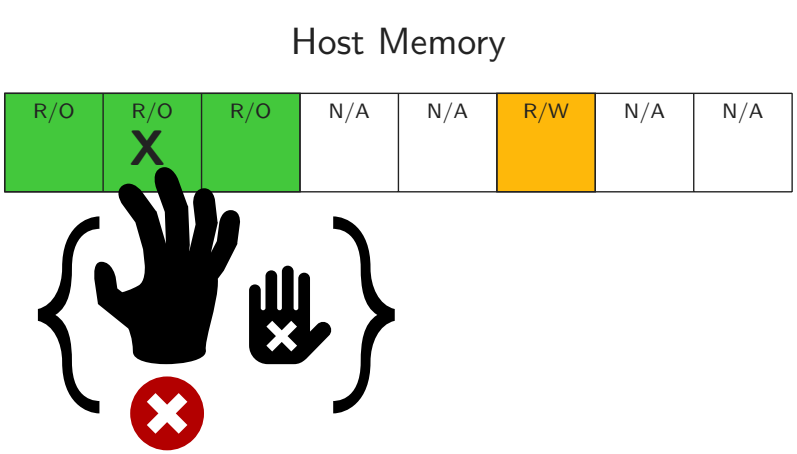
$\rightarrow$ No architectural write

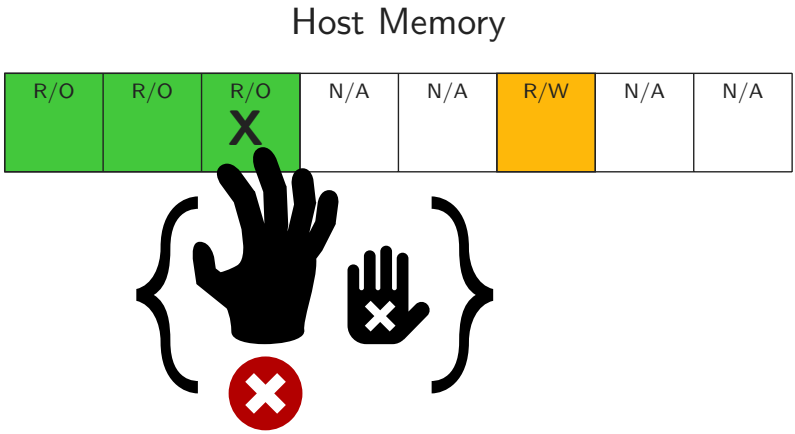
- Write to mapped page...
- ...abort immediately

→ No architectural write

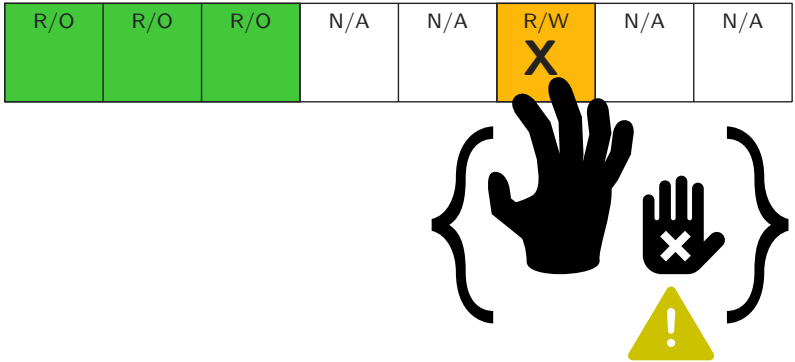
- Abort code → explicit or implicit


Host Memory
R/O
R/O
R/O
N/A
N/A
R/W
N/A
N/A


Host Memory
R/O
R/O
X
R/O
N/A
N/A
R/W
N/A
N/A


Host Memory
R/O
R/O
R/O
N/A
N/A
R/W
N/A
N/A

Host Memory



- TAP+CLAW → find writable memory

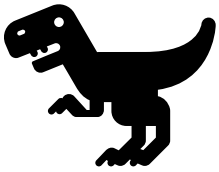
- TAP+CLAW → find writable memory

→ Robust write-anything-anywhere primitive

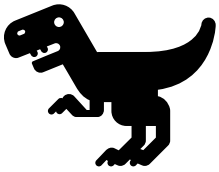
- TAP+CLAW → find writable memory

→ Robust write-anything-anywhere primitive

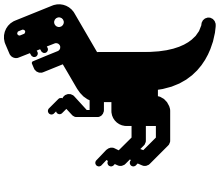
→ Store malicious payload



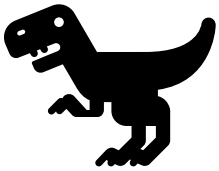
1. TAP: find ROP gadgets



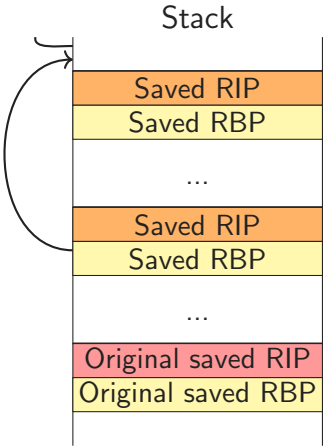
1. TAP: find ROP gadgets
2. CLAW: find writable memory (data cave)



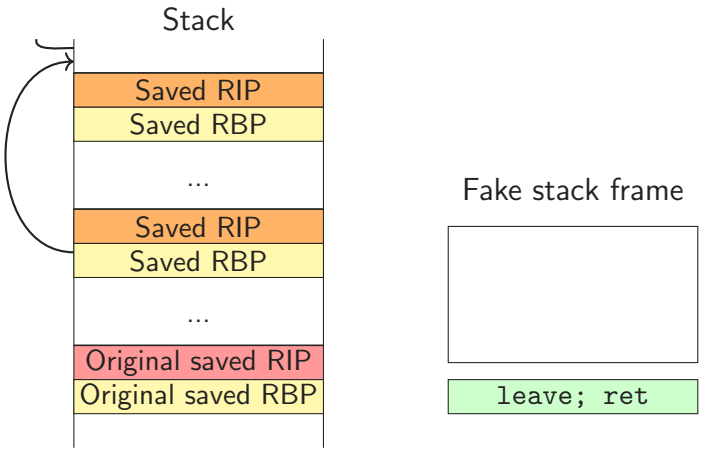
1. TAP: find ROP gadgets
2. CLAW: find writable memory (data cave)
3. Inject ROP gadgets into host stack

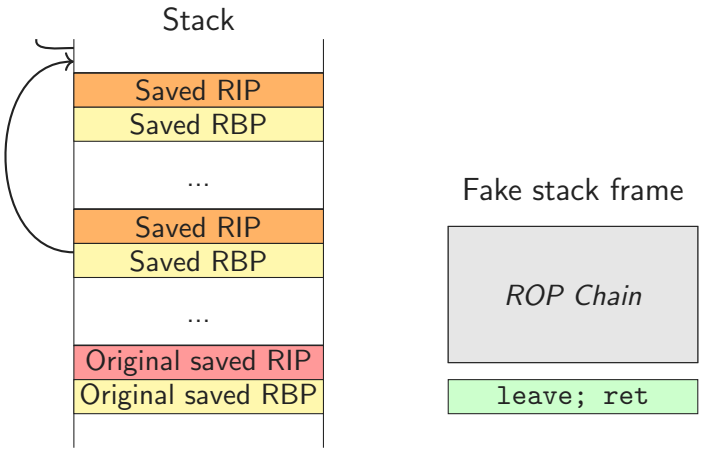


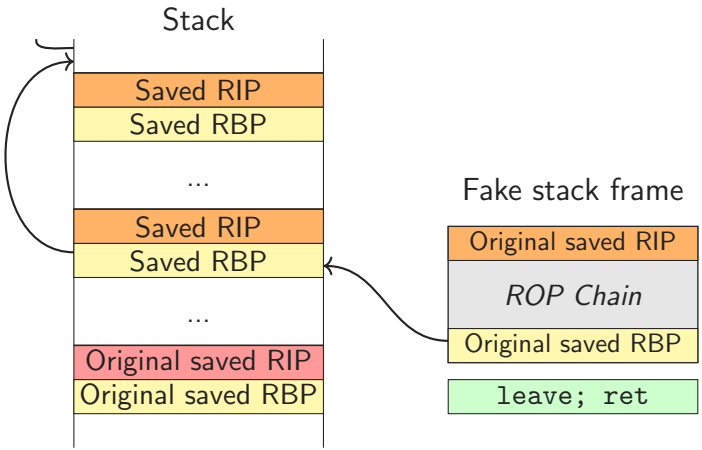
1. TAP: find ROP gadgets
2. CLAW: find writable memory (data cave)
3. Inject ROP gadgets into host stack
4. Profit!

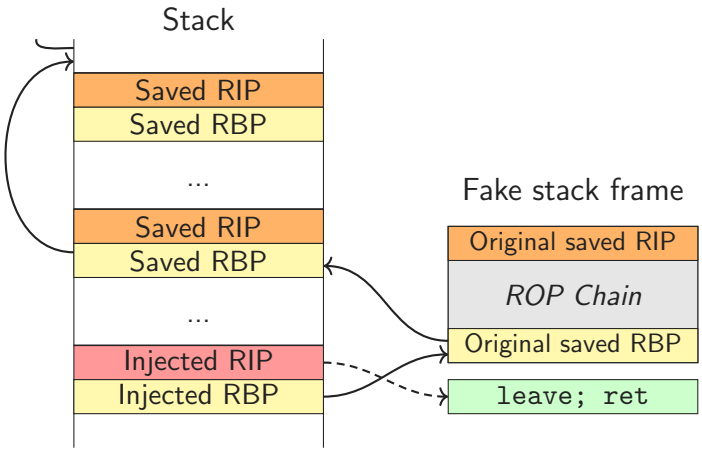

Stack
Saved RIP
Saved RBP
...
Saved RIP
Saved RBP
...
Original saved RIP
Original saved RBP

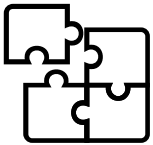
```
leave; ret
```


Stack
Saved RIP
Saved RBP
...
Saved RIP
Saved RBP
...
Original saved RIP
Original saved RBP
Fake stack frame
leave; ret


Stack
Saved RIP
Saved RBP
...
Saved RIP
Saved RBP
...
Original saved RIP
Original saved RBP
Fake stack frame
ROP Chain
leave; ret


Stack
Saved RIP
Saved RBP
...
Saved RIP
Saved RBP
...
Original saved RIP
Original saved RBP
Fake stack frame
Original saved RIP
ROP Chain
Original saved RBP
leave; ret

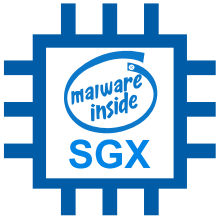

Stack
Saved RIP
Saved RBP
...
Saved RIP
Saved RBP
...
Injected RIP
Injected RBP
Fake stack frame
Original saved RIP
ROP Chain
Original saved RBP
leave; ret



64.8 MB writable data
`mprotect` ROP gadgets

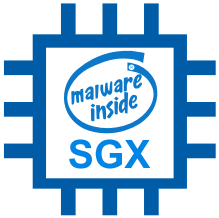
Several pages writable data
`mprotect` ROP gadgets

Full Exploit



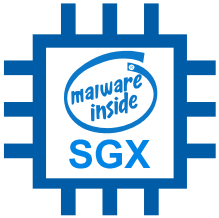
- Remote attestation + dynamic loading → no emulation, no binary

Full Exploit



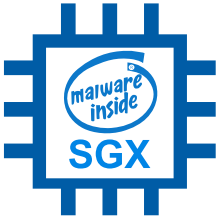
- Remote attestation + dynamic loading → no emulation, no binary
- Host continues normally → (nearly) no traces

Full Exploit



- Remote attestation + dynamic loading → no emulation, no binary
- Host continues normally → (nearly) no traces
- Trigger-based → plausible deniability

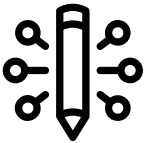
Full Exploit



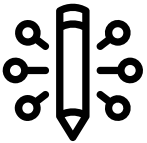
- Remote attestation + dynamic loading → no emulation, no binary
- Host continues normally → (nearly) no traces
- Trigger-based → plausible deniability

→ Securely and stealthily deploying zero days

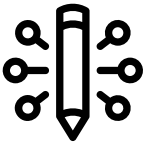
https://github.com/IAIK/SGXROP



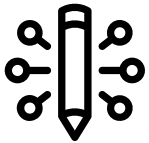
- Asymmetric threat model



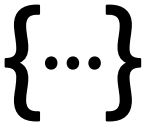
- Asymmetric threat model
- Enclaves assumed always benign



- Asymmetric threat model
- Enclaves assumed always benign
- Not realistic in most scenarios



- Asymmetric threat model
- Enclaves assumed always benign
- Not realistic in most scenarios
- Full memory access avoidable → reduce attack surface



Takeaways

- Asymmetric threat model in SGX fosters malware
- SGX hides and protects malware
- Easy to port existing malware to SGX ROP

Thank you!

Practical Enclave Malware with Intel SGX

Michael Schwarz (@misc0110), Samuel Weiser, Daniel Gruss

June 20, 2019 - DIMVA'19

Graz University of Technology

D. Gruss, M. Lipp, M. Schwarz, D. Genkin, J. Juffinger, S. O'Connell, W. Schoechl, and Y. Yarom. Another Flip in the Wall of Rowhammer Defenses. In: S&P. 2018.

Y. Jang, J. Lee, S. Lee, and T. Kim. SGX-Bomb: Locking Down the Processor via Rowhammer Attack. In: SysTEX. 2017.

M. Schwarz, D. Gruss, S. Weiser, C. Maurice, and S. Mangard. Malware Guard Extension: Using SGX to Conceal Cache Attacks. In: DIMVA. 2017.