Confining (Un)Trusted Execution Environments

Michael Schwarz

November 20, 2019 - SILM

Graz University of Technology

- Sandboxes assume trusted system and untrusted application

- Sandboxes assume trusted system and untrusted application

→ Protects the system from harm

- Sandboxes assume trusted system and untrusted application

→ Protects the system from harm

- Protect the application from the system?

- Sandboxes assume trusted system and untrusted application

→ Protects the system from harm

- Protect the application from the system?
- Assumption: untrusted system, trusted application

- Sandboxes assume trusted system and untrusted application

→ Protects the system from harm

- Protect the application from the system?
- Assumption: untrusted system, trusted application

→ Isolation of application

- Applications for isolation:

- Applications for isolation:
  - Working with sensitive data (e.g., passwords, money)

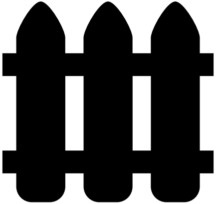
- Applications for isolation:
  - Working with sensitive data (e.g., passwords, money)
  - Distrusting the cloud provider

Motivation

- Applications for isolation:
  - Working with sensitive data (e.g., passwords, money)
  - Distrusting the cloud provider
  - Intellectual property (e.g., algorithms)

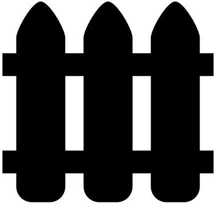
- Applications for isolation:
  - Working with sensitive data (e.g., passwords, money)
  - Distrusting the cloud provider
  - Intellectual property (e.g., algorithms)
  - Rights management (DRM)

- Applications for isolation:
  - Working with sensitive data (e.g., passwords, money)
  - Distrusting the cloud provider
  - Intellectual property (e.g., algorithms)
  - Rights management (DRM)
- Ensures security even against active attacks

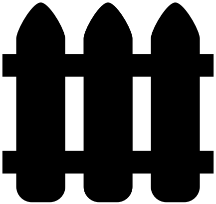


- Requires some form of hardware support

Isolation

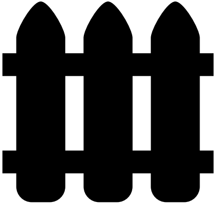


- Requires some form of hardware support
- Well-known isolation: user space - kernel space



- Requires some form of hardware support
- Well-known isolation: user space - kernel space

→ Protects OS against malicious applications

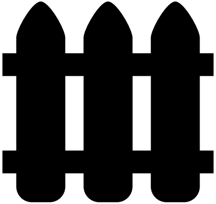


- Requires some form of hardware support
- Well-known isolation: user space - kernel space

→ Protects OS against malicious applications

→ Applications also mutually isolated

Isolation

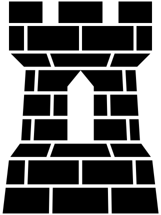


- Requires some form of hardware support
- Well-known isolation: user space - kernel space

$\rightarrow$ Protects OS against malicious applications

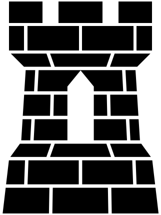
$\rightarrow$ Applications also mutually isolated

- Enforced by the hardware ($\rightarrow$ page table)

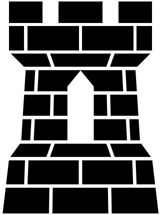


- Secure area of a CPU

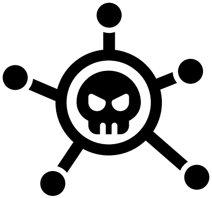
- Secure area of a CPU
- Integrity and confidentiality guarantees for code and data



- Secure area of a CPU
- Integrity and confidentiality guarantees for code and data
- Hardware still shared with other applications

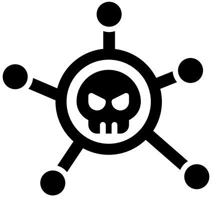


- Secure area of a CPU
- Integrity and confidentiality guarantees for code and data
- Hardware still shared with other applications
- (Nearly) no performance impacts

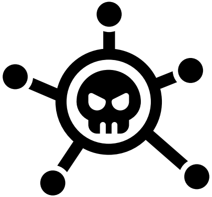


- Assumptions in TEEs:

Trusted-Execution Environments Threat Model

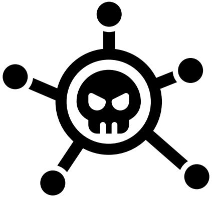


- Assumptions in TEEs:
  - Attacker controls the OS



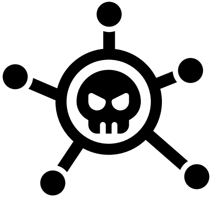
- Assumptions in TEEs:
  - Attacker controls the OS
  - Only the CPU is trusted

Trusted-Execution Environments Threat Model

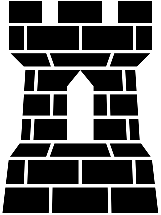


- Assumptions in TEEs:
  - Attacker controls the OS
  - Only the CPU is trusted
- TEE memory is encrypted and inaccessible to OS

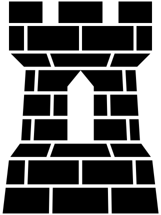
Trusted-Execution Environments Threat Model



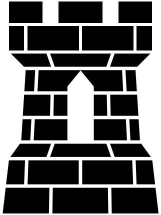
- Assumptions in TEEs:
  - Attacker controls the OS
  - Only the CPU is trusted
- TEE memory is encrypted and inaccessible to OS
- TEE has access to OS



- Implementations for various CPUs

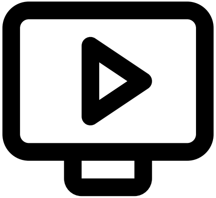


- Implementations for various CPUs
  - Intel: Software Guard Extension (SGX) and Management Engine (ME)
  - ARM and AMD: TrustZone

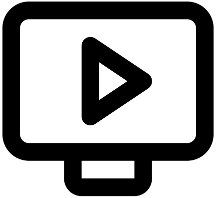


- Implementations for various CPUs
  - Intel: Software Guard Extension (SGX) and Management Engine (ME)
  - ARM and AMD: TrustZone
- Widely used in mobile phones

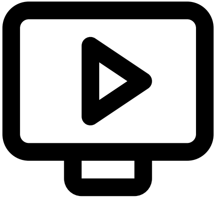
Real-World TEE Example



- Netflix uses Widevine DRM

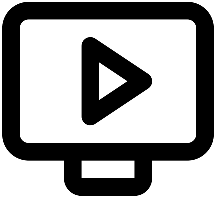


- Netflix uses Widevine DRM
- DRM in TrustZone



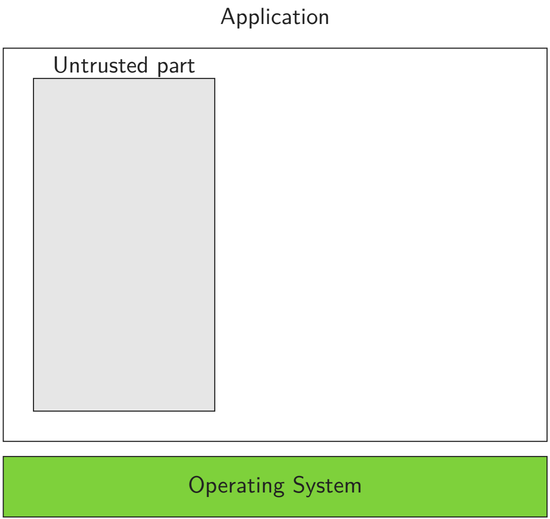
- Netflix uses Widevine DRM
- DRM in TrustZone
- Video is directly drawn on screen

Real-World TEE Example

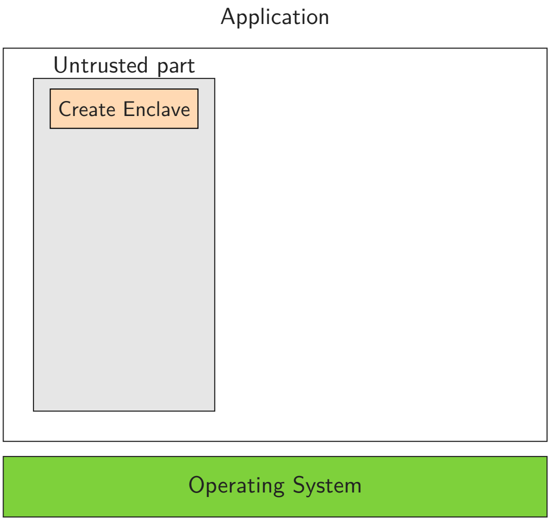


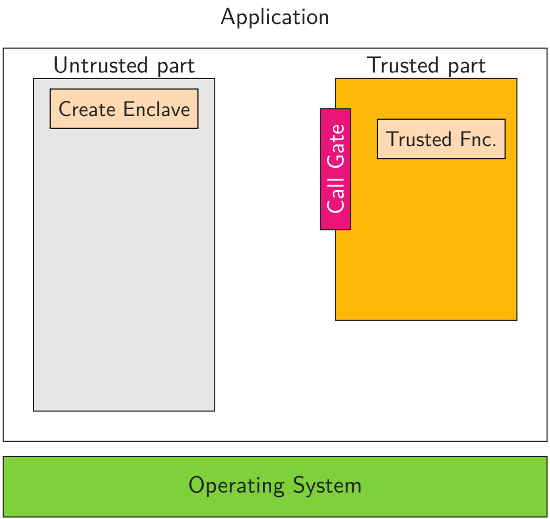
- Netflix uses Widevine DRM
- DRM in TrustZone
- Video is directly drawn on screen
- No app (not even root) can access video data

Execution Flow (SGX)

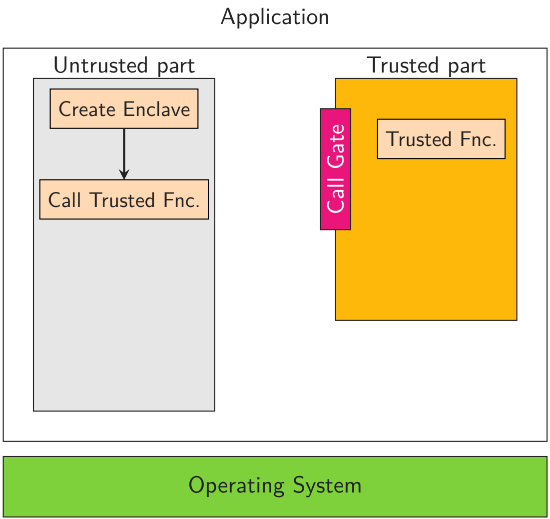


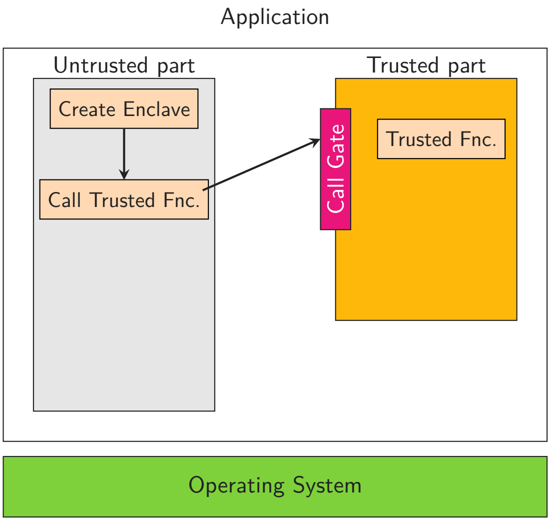
Execution Flow (SGX)




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

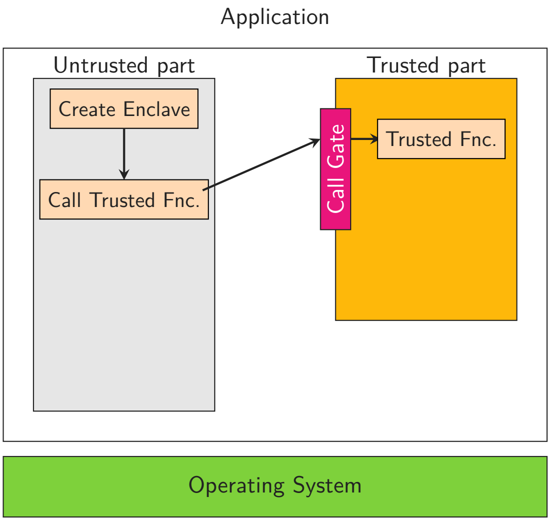
Execution Flow (SGX)



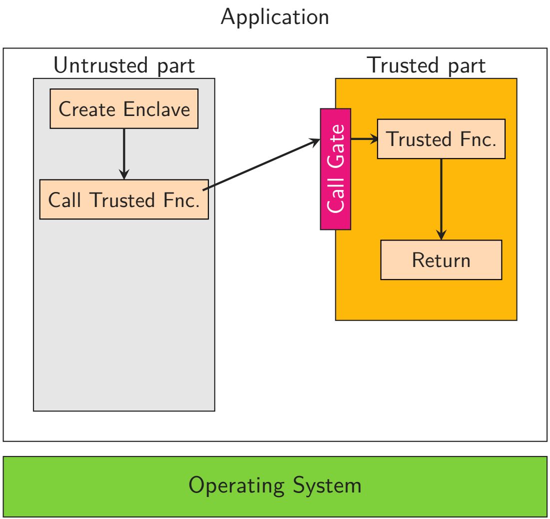


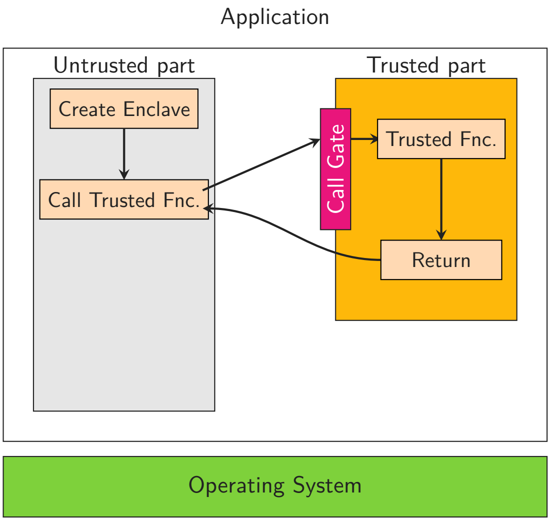
Application
Untrusted part
Create Enclave
Call Trusted Fnc.
Trusted part
Call Gate
Trusted Fnc.
Operating System

Execution Flow (SGX)

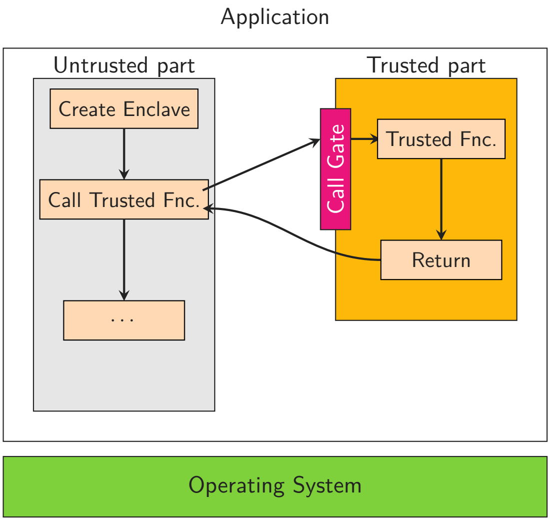


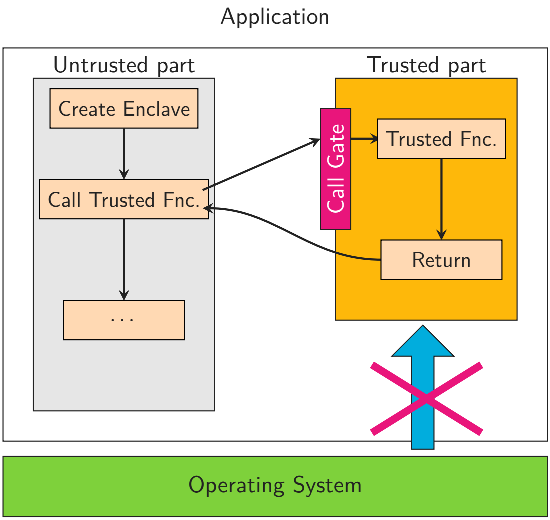
Execution Flow (SGX)

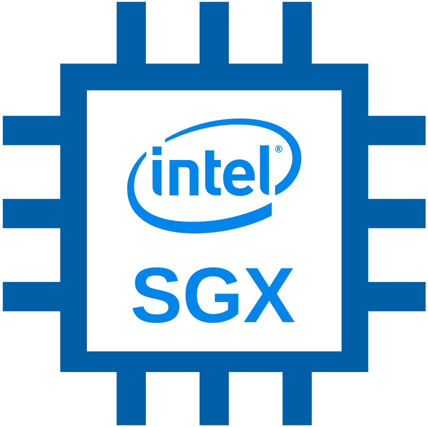


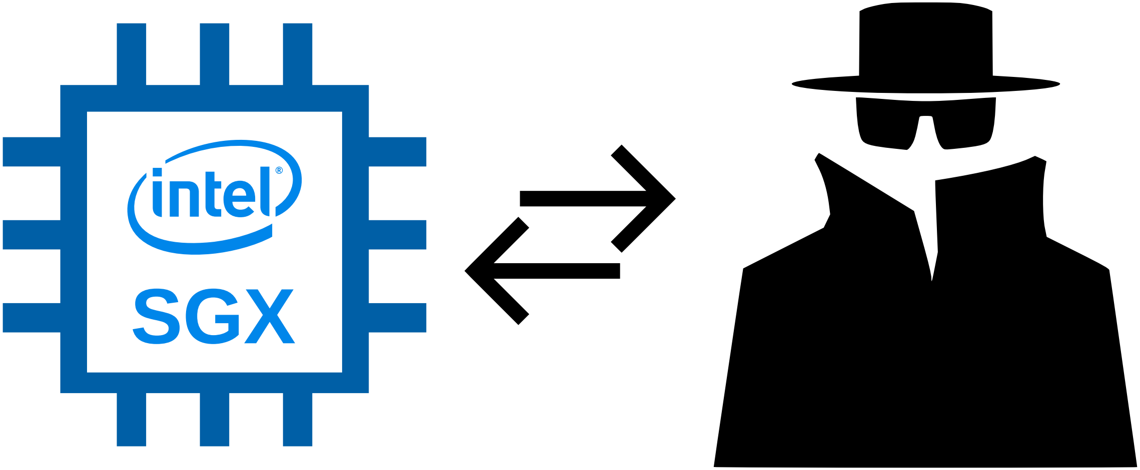

Application
Untrusted part
Trusted part
Create Enclave
Call Trusted Fnc.
Call Gate
Trusted Fnc.
Return
Operating System

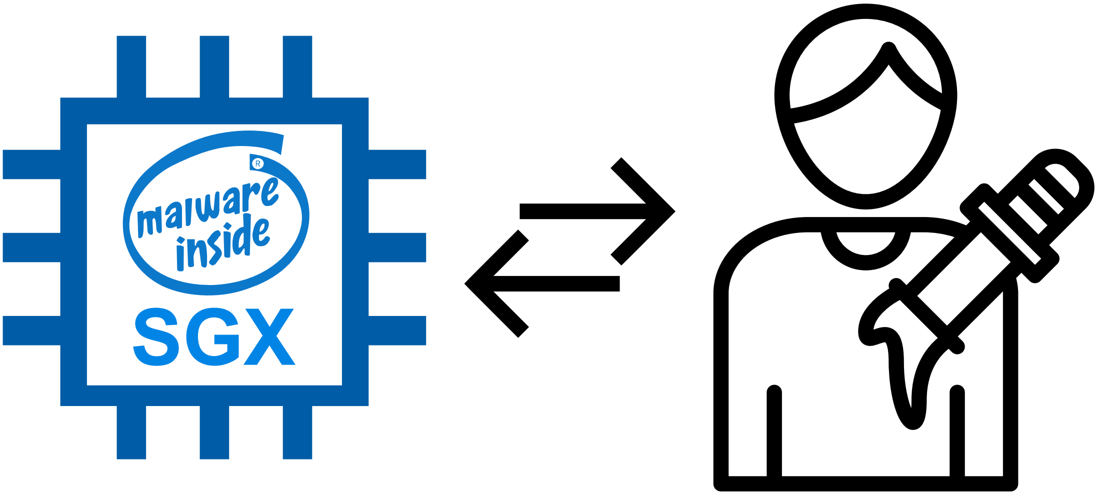
Execution Flow (SGX)

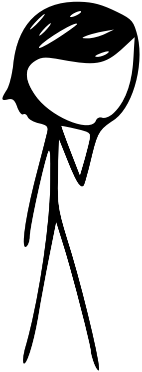



Application
Untrusted part
Trusted part
Create Enclave
Call Trusted Fnc.
...
Call Gate
Trusted Fnc.
Return
Operating System

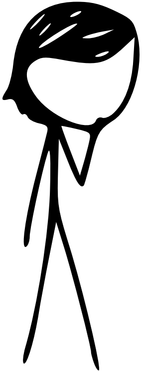

intel
SGX


intel
SGX

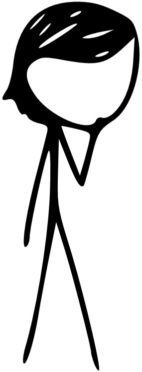

malware inside
SGX



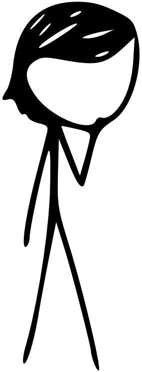
- Enclaves are black boxes



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



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



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

2013 Rutkowska: Irreversible Malware

http://theinvisiblethings.blogspot.com/2013/09/thoughts-on-intels-upcoming-software.html

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:

SGX Limitations

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

- 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

- Side-channel attacks from SGX possible

Side-Channel Attacks from SGX

- Side-channel attacks from SGX possible
- Allow attacker to spy on meta data

- Side-channel attacks from SGX possible
- Allow attacker to spy on meta data
- Completely hide an attack

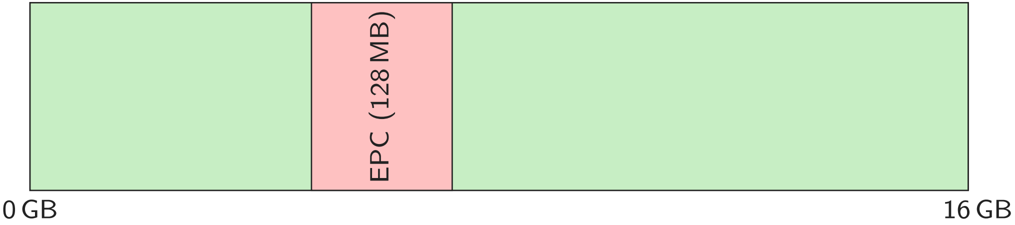
- Cache attacks preventable on source level

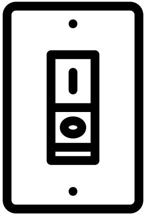
- Cache attacks preventable on source level
- Side-channel resistant crypto

Countermeasures

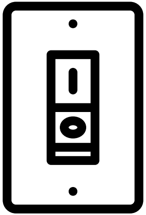
- Cache attacks preventable on source level
- Side-channel resistant crypto
- Default in most crypto libraries


0 GB
16 GB


EPC (128 MB)
0 GB
16 GB

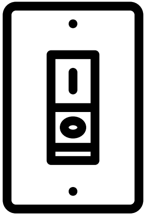


- What happens if a bit flips in the EPC?



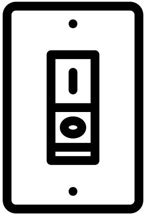
- What happens if a bit flips in the EPC?
- Integrity check will fail!

Bit Flips in the EPC



- What happens if a bit flips in the EPC?
- Integrity check will fail!

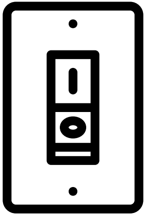
→ Locks up the memory controller



- What happens if a bit flips in the EPC?
- Integrity check will fail!

→ Locks up the memory controller

→ Not a single further memory access!

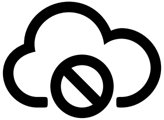


- What happens if a bit flips in the EPC?
- Integrity check will fail!

→ Locks up the memory controller

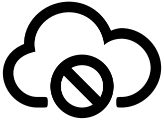
→ Not a single further memory access!

→ System halts immediately



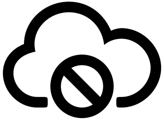
- If a malicious enclave induces a bit flip, ...

Bit Flips in the EPC

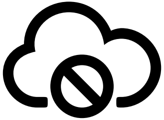


- If a malicious enclave induces a bit flip, ...
- ...the entire machine halts

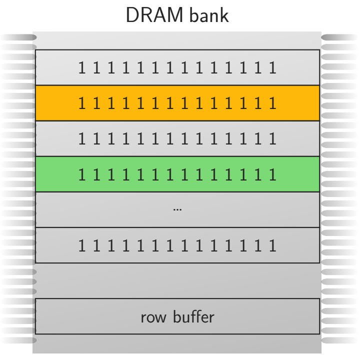
Bit Flips in the EPC

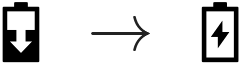


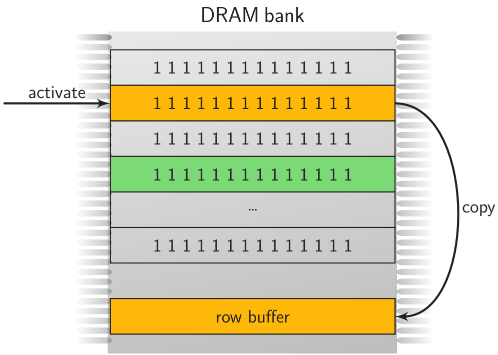
- If a malicious enclave induces a bit flip, ...
- ...the entire machine halts
- ...including co-located tenants

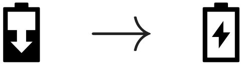


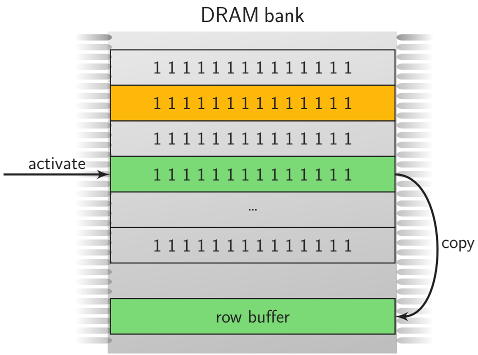
- If a malicious enclave induces a bit flip, ...
- ...the entire machine halts
- ...including co-located tenants
- Denial-of-Service Attacks in the Cloud [Gru+18; Jan+17]

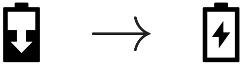

DRAM bank
1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1
...
1 1 1 1 1 1 1 1 1 1 1 1 1 1
row buffer



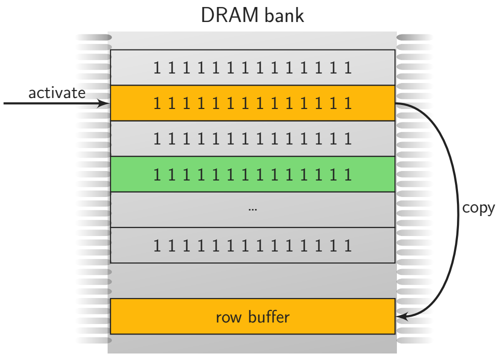

DRAM bank
activate
1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1
...
1 1 1 1 1 1 1 1 1 1 1 1 1 1
copy
row buffer

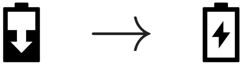




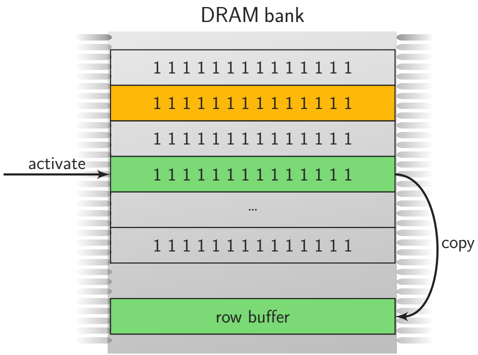


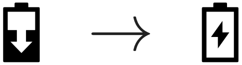
Cells leak faster upon proximate accesses → Rowhammer



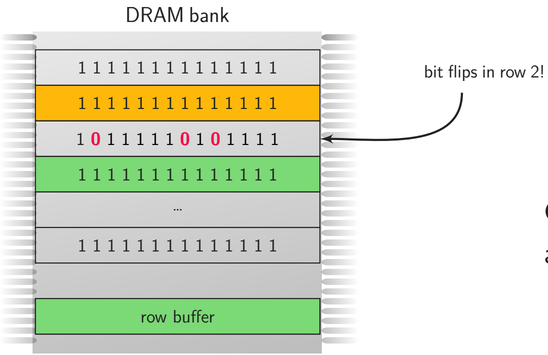


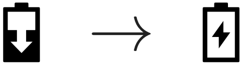
Cells leak faster upon proximate accesses → Rowhammer



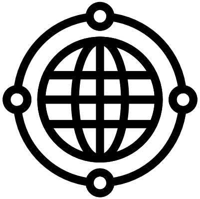


Cells leak faster upon proximate accesses → Rowhammer



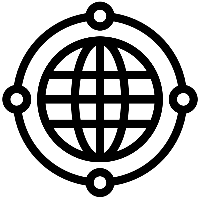


Cells leak faster upon proximate accesses → Rowhammer



- 85% affected (estimation 2014)
- 52% affected (estimation 2015)

How widespread is the issue?



- 85% affected (estimation 2014)
- 52% affected (estimation 2015)

- First believed to be safe
- We showed bit flips in 2016
- 67% affected (estimation 2016)

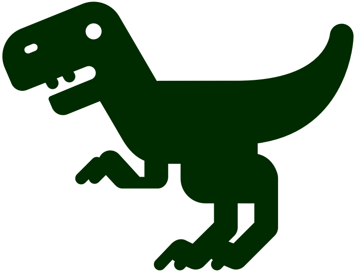
- Dangerous attacks but difficult in practice

Zero Days?

- Dangerous attacks but difficult in practice
- More relevant: zero days in enclaves

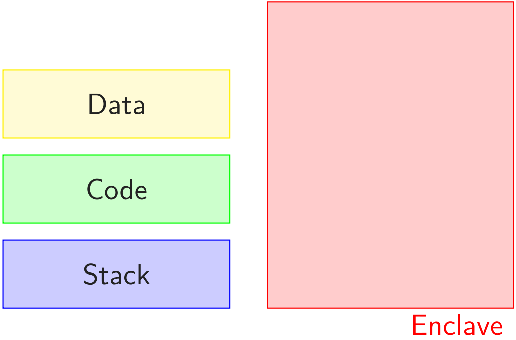
- Dangerous attacks but difficult in practice
- More relevant: zero days in enclaves

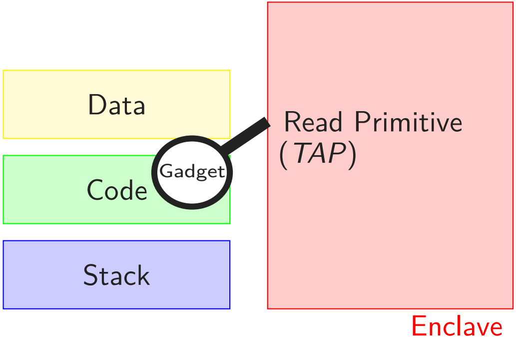
→ Super malware

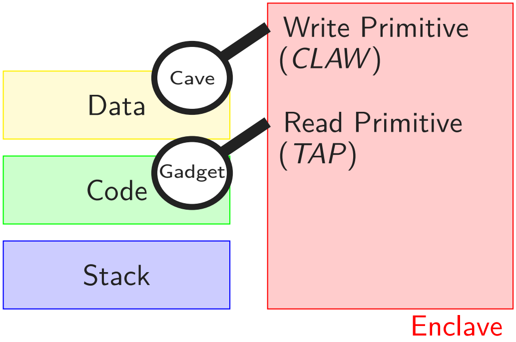


TEE-REX

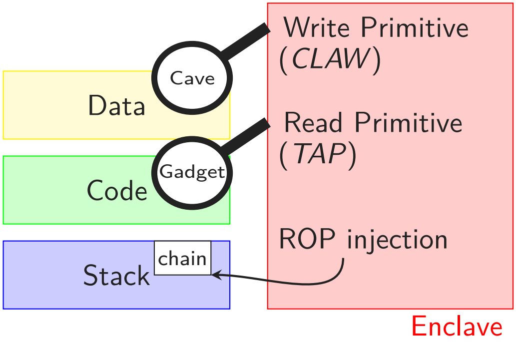
Trusted **E**xecution **E**nvironment **R**eturn-oriented-programming **EX**ploit

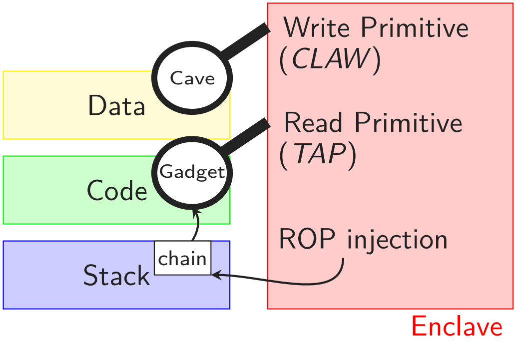

Data
Code
Stack
Enclave

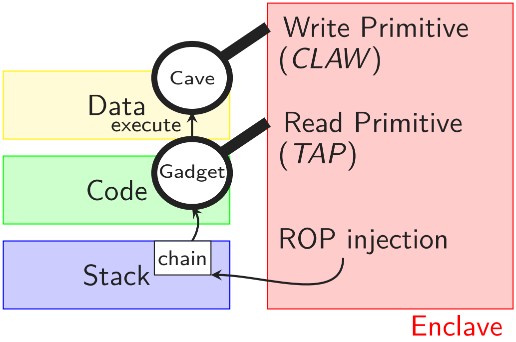

Data
Code
Gadget
Stack
Read Primitive
(*TAP*)
Enclave

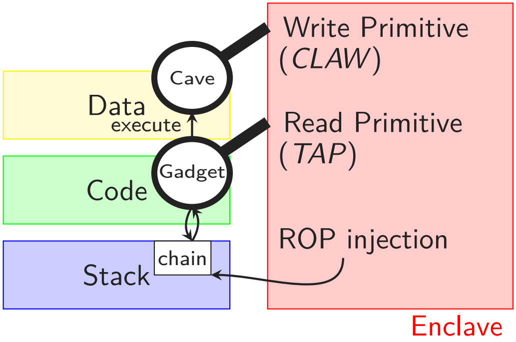

Write Primitive
(*CLAW*)
Read Primitive
(*TAP*)
Cave
Data
Gadget
Code
Stack
Enclave

Attack Overview




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


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

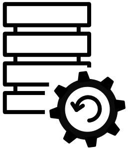

Write Primitive
(*CLAW*)
Cave
Data
execute
Read Primitive
(*TAP*)
Gadget
Code
ROP injection
chain
Stack
Enclave

- Enclave can access host memory...

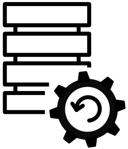
- Enclave can access host memory...
- ...but crashes on invalid access

Problems

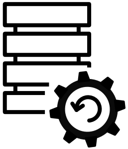
- Enclave can access host memory...
- ...but crashes on invalid access
- No syscall or exception handler available



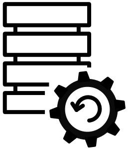
- Intel TSX: hardware transactional memory



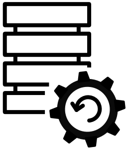
- Intel TSX: hardware transactional memory
- Multiple reads and writes are atomic



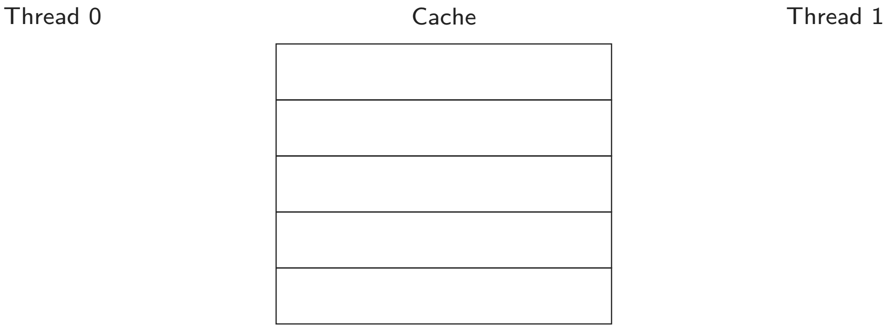
- Intel TSX: hardware transactional memory
- Multiple reads and writes are atomic
- Operations in a transaction

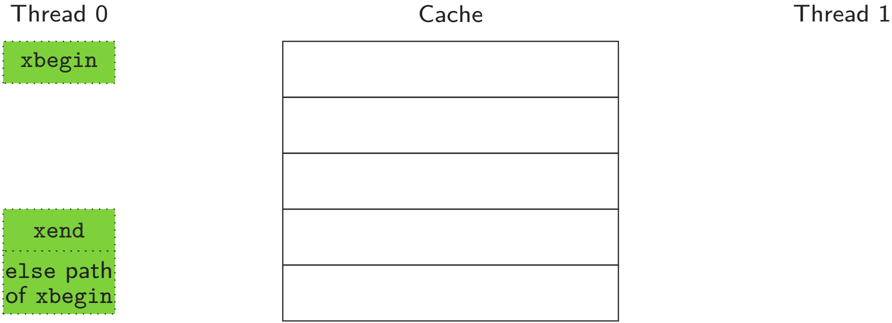


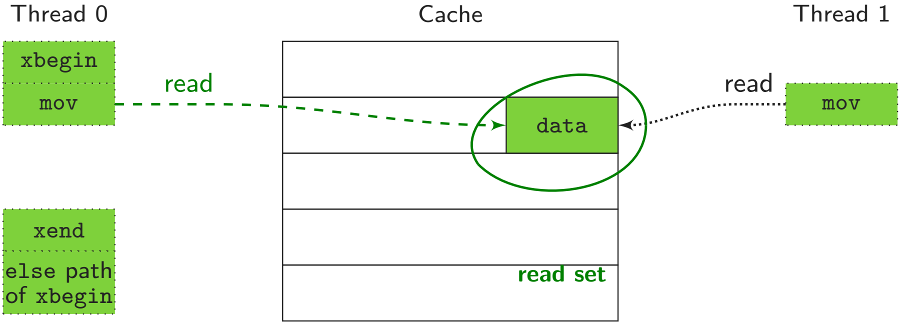
- Intel TSX: hardware transactional memory
- Multiple reads and writes are atomic
- Operations in a transaction
- Conflict $\rightarrow$ abort and roll back

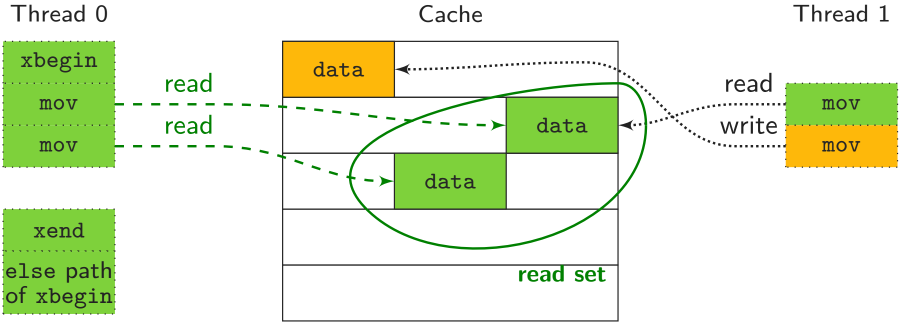


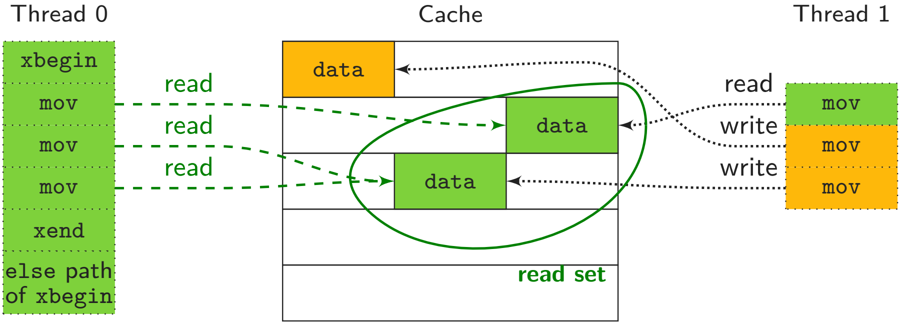
- Intel TSX: hardware transactional memory
- Multiple reads and writes are atomic
- Operations in a transaction
- Conflict → abort and roll back
- Faults are suppressed

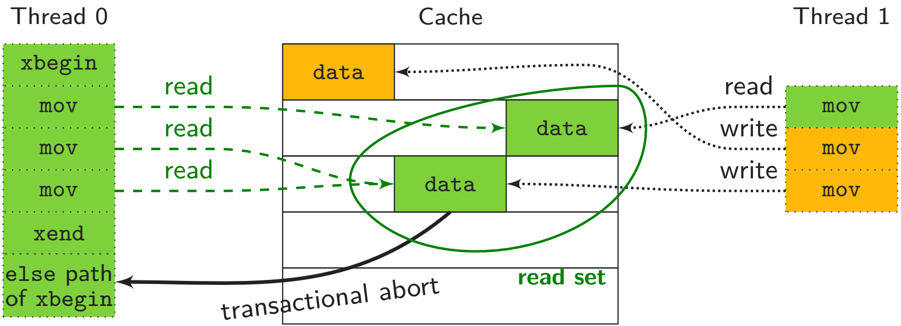

Thread 0
Cache
Thread 1


Thread 0
Cache
Thread 1
xbegin
xend
else path
of xbegin


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


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


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

- Segmentation fault is a fault

TAP

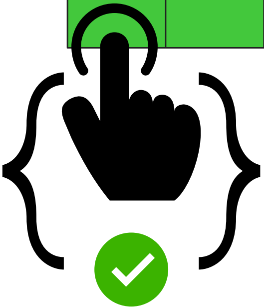
- Segmentation fault is a fault
- Suppressed in TSX transaction

- 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

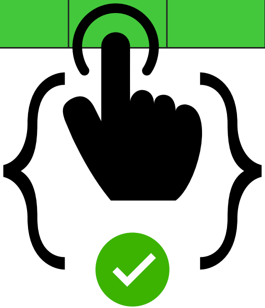
Valid	Valid	Valid	Invalid	Invalid	Valid	Invalid	Invalid



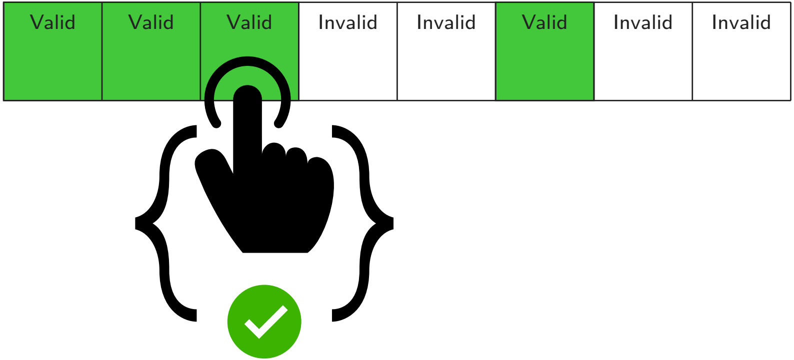
TAP

Host Memory

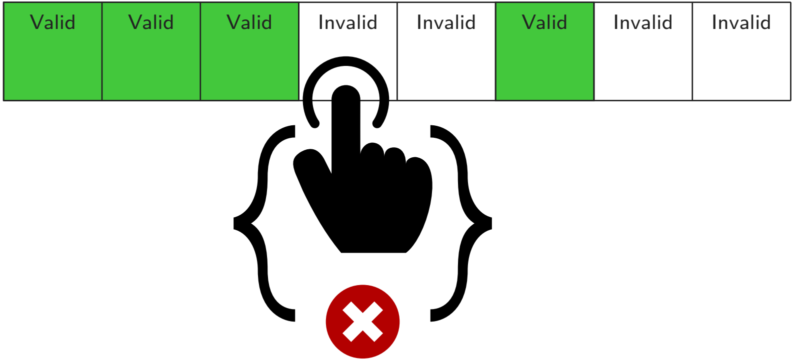
Valid	Valid	Valid	Invalid	Invalid	Valid	Invalid	Invalid



Host Memory

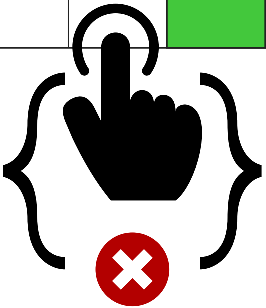


Host Memory

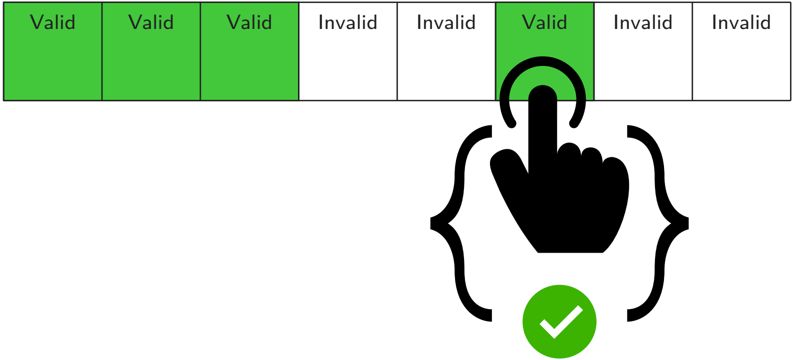


Host Memory

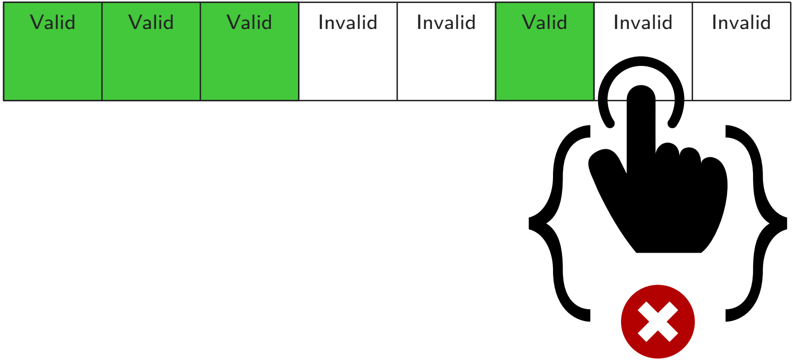
Valid	Valid	Valid	Invalid	Invalid	Valid	Invalid	Invalid



Host Memory

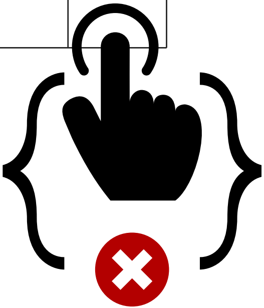


Host Memory



Host Memory

Valid	Valid	Valid	Invalid	Invalid	Valid	Invalid	Invalid



- 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

→ No architectural write

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

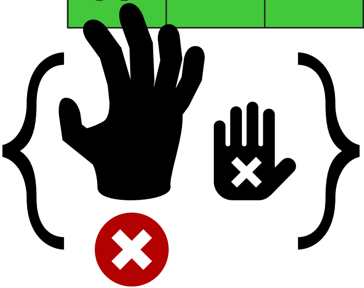
$\rightarrow$ No architectural write

- Abort code $\rightarrow$ explicit or implicit

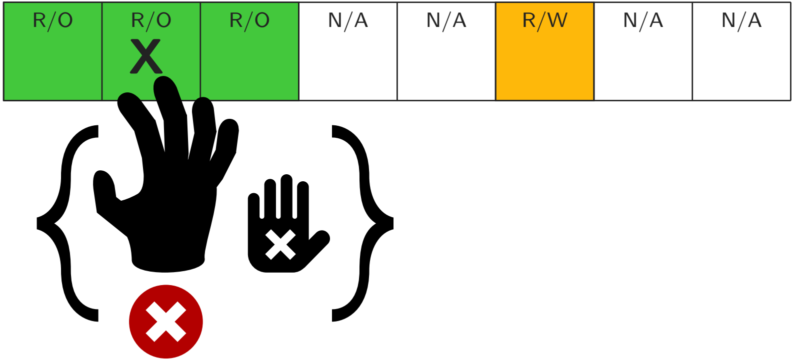
Host Memory

R/O	R/O	R/O	N/A	N/A	R/W	N/A	N/A

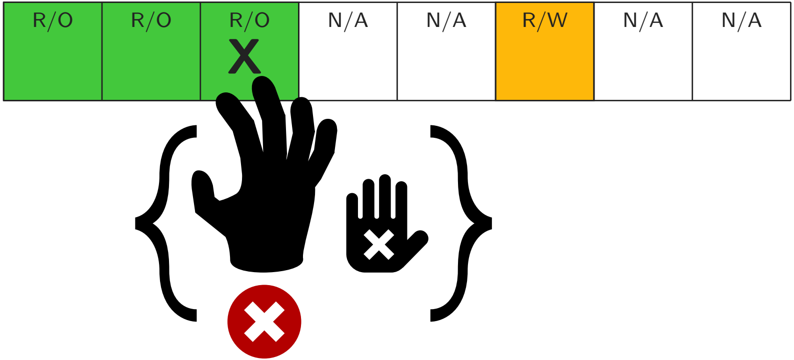
X



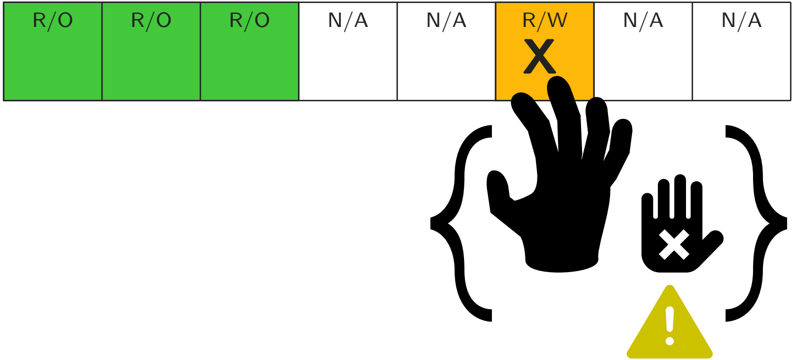
Host Memory



Host Memory



Host Memory



- TAP+CLAW → find writable memory

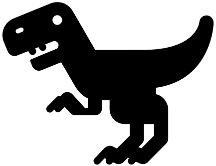
- TAP+CLAW → find writable memory

→ Robust write-anything-anywhere primitive

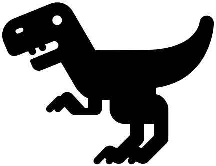
- TAP+CLAW → find writable memory

→ Robust write-anything-anywhere primitive

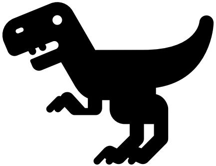
→ Store malicious payload



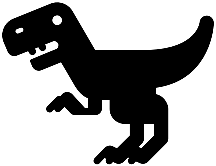
1. TAP: find ROP gadgets



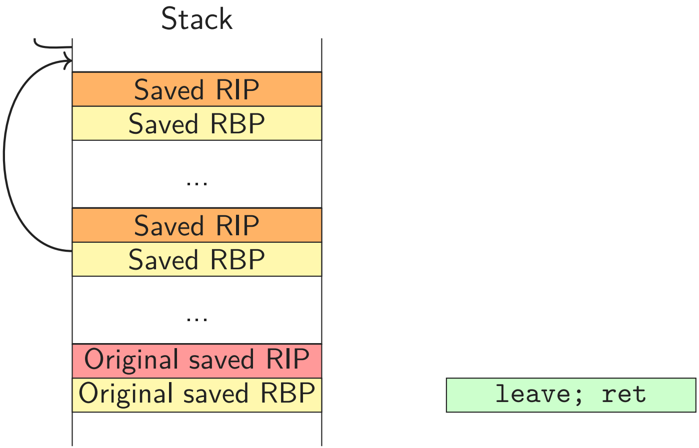
1. TAP: find ROP gadgets
2. CLAW: find writable memory (data cave)

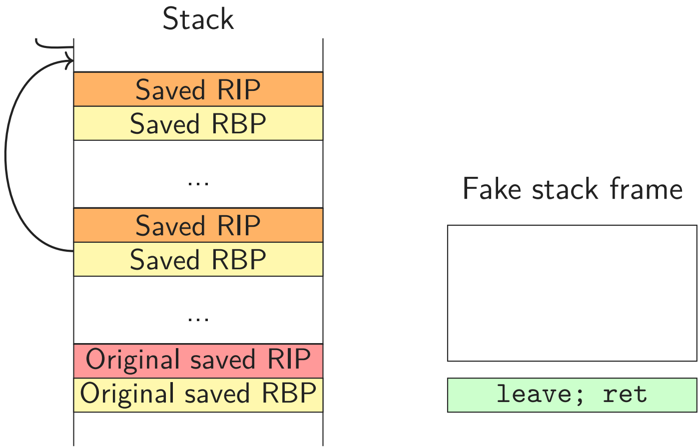


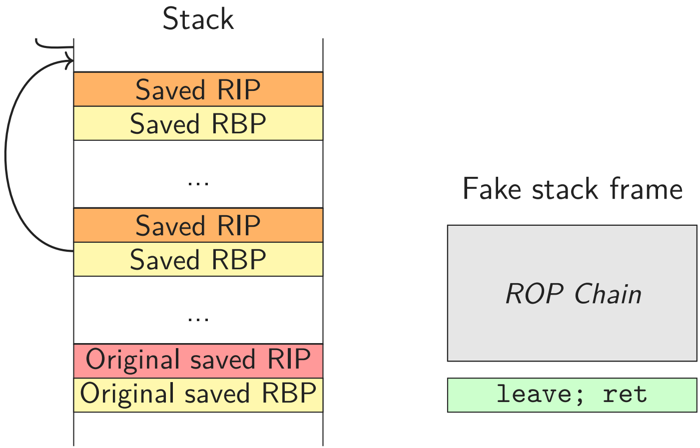
1. TAP: find ROP gadgets
2. CLAW: find writable memory (data cave)
3. Inject ROP gadgets into host stack

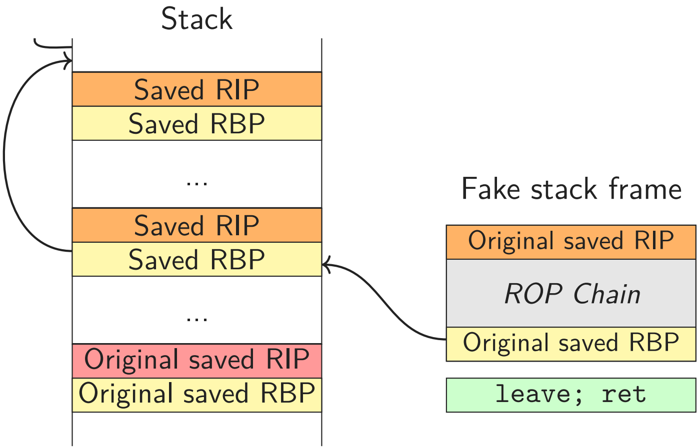


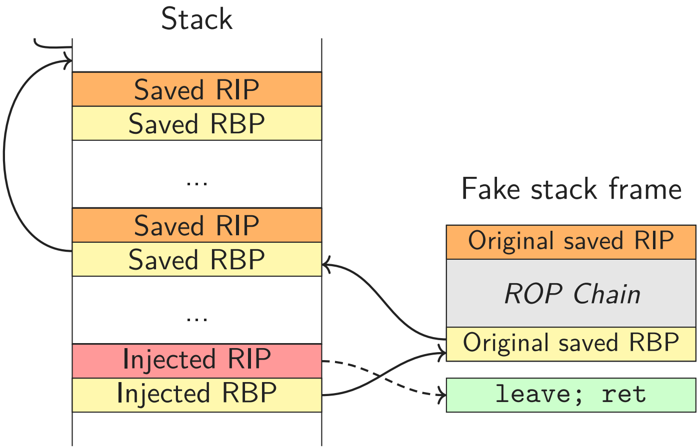
1. TAP: find ROP gadgets
2. CLAW: find writable memory (data cave)
3. Inject ROP gadgets into host stack
4. Profit!


Stack
Saved RIP
Saved RBP
...
Saved RIP
Saved RBP
...
Original saved RIP
Original saved RBP
leave; ret

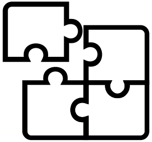

Stack
Saved RIP
Saved RBP
...
Saved RIP
Saved RBP
...
Original saved RIP
Original saved RBP
Fake stack frame
leave; ret


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

Gadgets

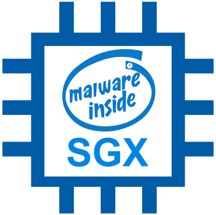


64.8 MB writable data

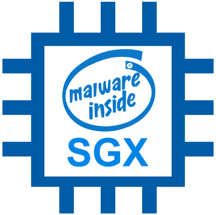
`mprotect` ROP gadgets

Several pages writable data

`mprotect` ROP gadgets

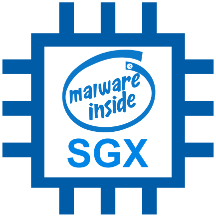


- Remote attestation + dynamic loading → no emulation, no binary

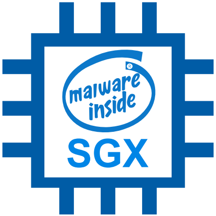


- Remote attestation + dynamic loading → no emulation, no binary
- Host continues normally → (nearly) no traces

Full Exploit



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



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

→ Securely and stealthily deploying zero days sgxrop : zsh — Konsole

File Edit View Bookmarks Settings Help

```
mschwarz@t480sms2 /tmp/sgxrop %
```

sgxrop : zsh

sgxrop : zsh — Konsole

File Edit View Bookmarks Settings Help

```
mschwarz@t480sms2 /tmp/sgxrop % ./a
```

sgxrop : zsh

sgxrop : app — Konsole

File Edit View Bookmarks Settings Help

```
mschwarz@t480sms2 /tmp/sgxrop % ./app
Call trace:
|
+--- foo enter
        |
        +--- bar enter
                |
                +--- enclave enter
                           |
[ENCLAVE] <Start @ 0x7ffffffff000>
[ENCLAVE] <Saved RSP: 7fff082d4320>
[ENCLAVE] <Saved RBP: 7fff082d47e0>
[ENCLAVE] <Searching for stack frame...>
[ENCLAVE] <Stack frame @ 296: 556108dbe4c0 / 7fff082d4920 (3d8d4800000008be / 7fff082d4940)>
[ENCLAVE] <Stack frame @ 328: 556108dbe541 / 7fff082d4a28 (4800200c48058d48 / 7fff082d61ee)>
[ENCLAVE] <RIP @ 0x7fff082d4928>
[ENCLAVE] <RBP @ 0x7fff082d4920>
[ENCLAVE] <Searching for gadgets...>
[ENCLAVE] <Found gadget [SYSCALL] @ 0x7fff083f47ec>
```

sgxrop : app

sgxrop : app — Konsole

File Edit View Bookmarks Settings Help

```
|
+--- foo enter
        |
        +--- bar enter
                |
                +--- enclave enter
                            |
[ENCLAVE] <Start @ 0x7ffffffff000>
[ENCLAVE] <Saved RSP: 7fff082d4320>
[ENCLAVE] <Saved RBP: 7fff082d47e0>
[ENCLAVE] <Searching for stack frame...>
[ENCLAVE] <Stack frame @ 296: 556108dbe4c0 / 7fff082d4920 (3d8d4800000008be / 7fff082d4940)>
[ENCLAVE] <Stack frame @ 328: 556108dbe541 / 7fff082d4a28 (4800200c48058d48 / 7fff082d61ee)>
[ENCLAVE] <RIP @ 0x7fff082d4928>
[ENCLAVE] <RBP @ 0x7fff082d4920>
[ENCLAVE] <Searching for gadgets...>
[ENCLAVE] <Found gadget [SYSCALL] @ 0x7fff083f47ec>
[ENCLAVE] <Found gadget [POP_RDI] @ 0x7fe81a457287>
[ENCLAVE] <Found gadget [POP_RSI] @ 0x7fe81a456167>
[ENCLAVE] <Found gadget [LEAVE] @ 0x7fe81a4409ea>
[ENCLAVE] <Found gadget [POP_RDX] @ 0x7fe81a228f7a>
[ENCLAVE] <Found gadget [POP_RAX] @ 0x7fe819fe6af4>
[ENCLAVE] <Found gadget [XCHG_RAX_RDI] @ 0x7fe819f3e8e5>
[ENCLAVE] <Searching for data cave...>
```

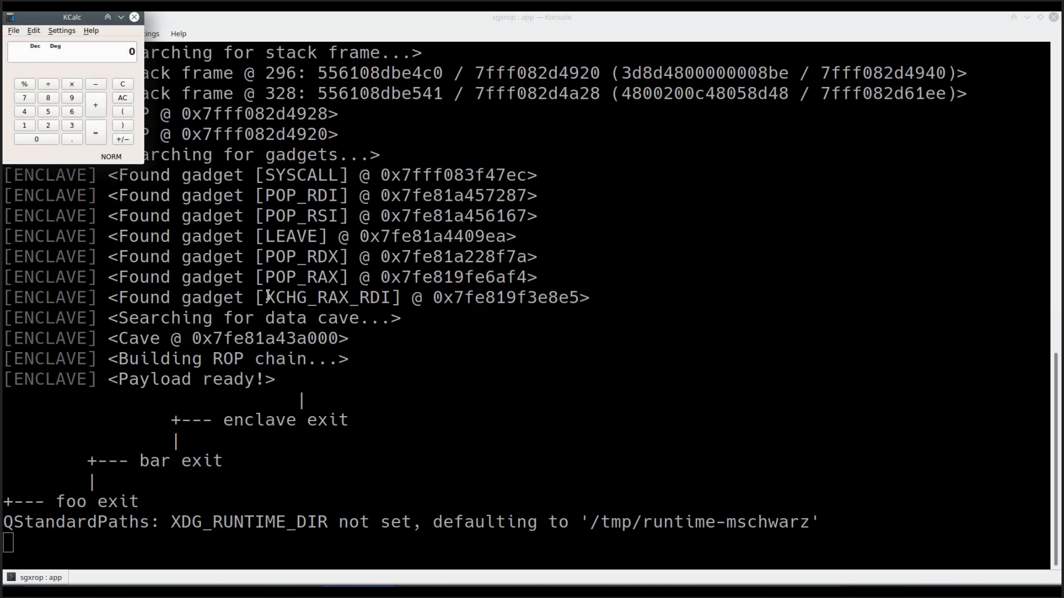
sgxrop : app

sgxrop : app — Konsole

File Edit View Bookmarks Settings Help

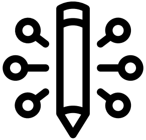
```
[ENCLAVE] <Searching for stack frame...>
[ENCLAVE] <Stack frame @ 296: 556108dbe4c0 / 7fff082d4920 (3d8d4800000008be / 7fff082d4940)>
[ENCLAVE] <Stack frame @ 328: 556108dbe541 / 7fff082d4a28 (4800200c48058d48 / 7fff082d61ee)>
[ENCLAVE] <RIP @ 0x7fff082d4928>
[ENCLAVE] <RBP @ 0x7fff082d4920>
[ENCLAVE] <Searching for gadgets...>
[ENCLAVE] <Found gadget [SYSCALL] @ 0x7fff083f47ec>
[ENCLAVE] <Found gadget [POP_RDI] @ 0x7fe81a457287>
[ENCLAVE] <Found gadget [POP_RSI] @ 0x7fe81a456167>
[ENCLAVE] <Found gadget [LEAVE] @ 0x7fe81a4409ea>
[ENCLAVE] <Found gadget [POP_RDX] @ 0x7fe81a228f7a>
[ENCLAVE] <Found gadget [POP_RAX] @ 0x7fe819fe6af4>
[ENCLAVE] <Found gadget [XCHG_RAX_RDI] @ 0x7fe819f3e8e5>
[ENCLAVE] <Searching for data cave...>
[ENCLAVE] <Cave @ 0x7fe81a43a000>
[ENCLAVE] <Building ROP chain...>
[ENCLAVE] <Payload ready!>
                          |
                +--- enclave exit
                |
        +--- bar exit
        |
+--- foo exit
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-mschwarz'
```

sgxrop : app

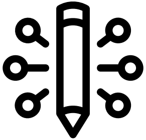

KCalc
File Edit Settings Help
Dec Deg
0
% ÷ × − C
7 8 9 + AC
4 5 6 (
1 2 3 =)
0 . +/−
NORM
sgxrop : app — Konsole
ings Help
arching for stack frame...>
ack frame @ 296: 556108dbe4c0 / 7fff082d4920 (3d8d4800000008be / 7fff082d4940)>
ack frame @ 328: 556108dbe541 / 7fff082d4a28 (4800200c48058d48 / 7fff082d61ee)>
P @ 0x7fff082d4928>
P @ 0x7fff082d4920>
arching for gadgets...>
[ENCLAVE] <Found gadget [SYSCALL] @ 0x7fff083f47ec>
[ENCLAVE] <Found gadget [POP_RDI] @ 0x7fe81a457287>
[ENCLAVE] <Found gadget [POP_RSI] @ 0x7fe81a456167>
[ENCLAVE] <Found gadget [LEAVE] @ 0x7fe81a4409ea>
[ENCLAVE] <Found gadget [POP_RDX] @ 0x7fe81a228f7a>
[ENCLAVE] <Found gadget [POP_RAX] @ 0x7fe819fe6af4>
[ENCLAVE] <Found gadget [XCHG_RAX_RDI] @ 0x7fe819f3e8e5>
[ENCLAVE] <Searching for data cave...>
[ENCLAVE] <Cave @ 0x7fe81a43a000>
[ENCLAVE] <Building ROP chain...>
[ENCLAVE] <Payload ready!>
+--- enclave exit
+--- bar exit
+--- foo exit
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-mschwarz'
sgxrop : app

Try It!

https://github.com/IAIK/SGXROP

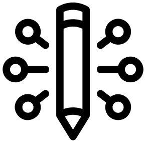


- Asymmetric threat model



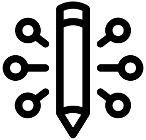
- Asymmetric threat model
- Enclaves assumed always benign

Design Problems



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

Design Problems



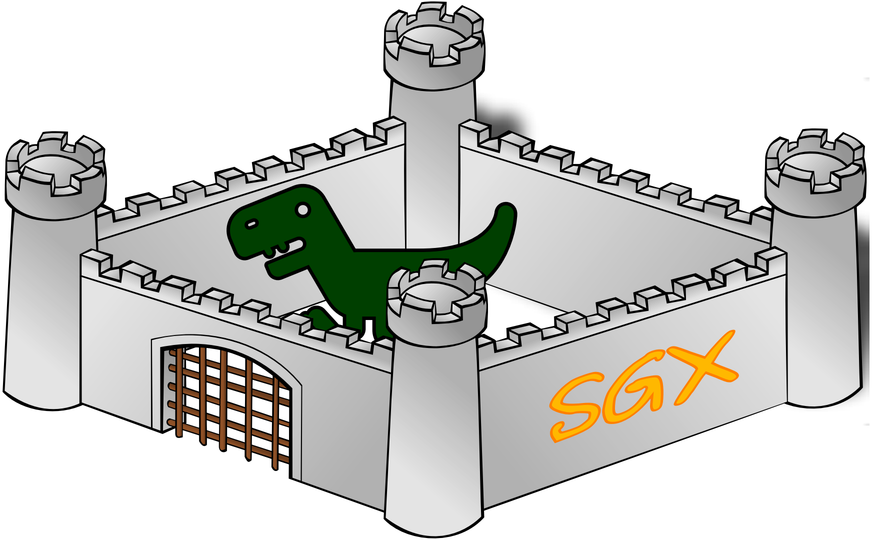
- Asymmetric threat model
- Enclaves assumed always benign
- Not realistic in most scenarios
- Full memory access avoidable → reduce attack surface

{...}

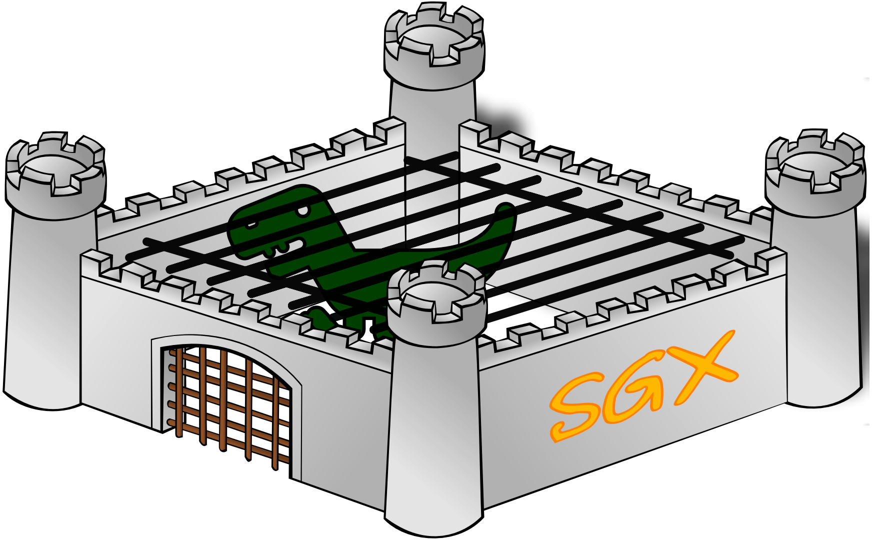
Takeaways

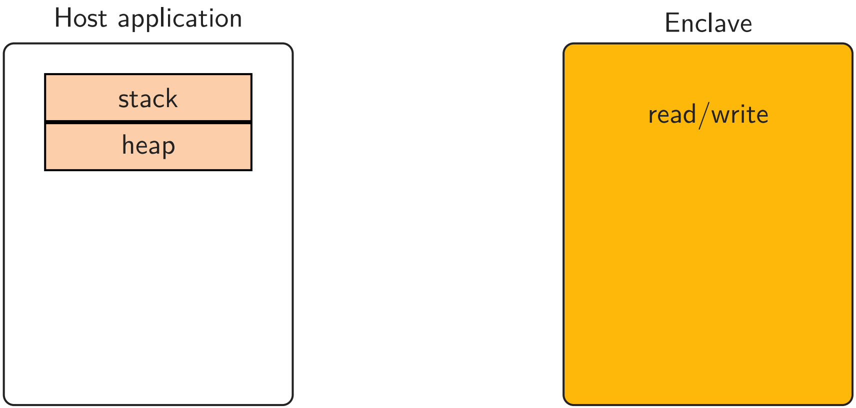
- Asymmetric threat model in SGX fosters malware
- SGX hides and protects malware
- Easy to port existing malware to SGX ROP

SGXJail [Wei+19]

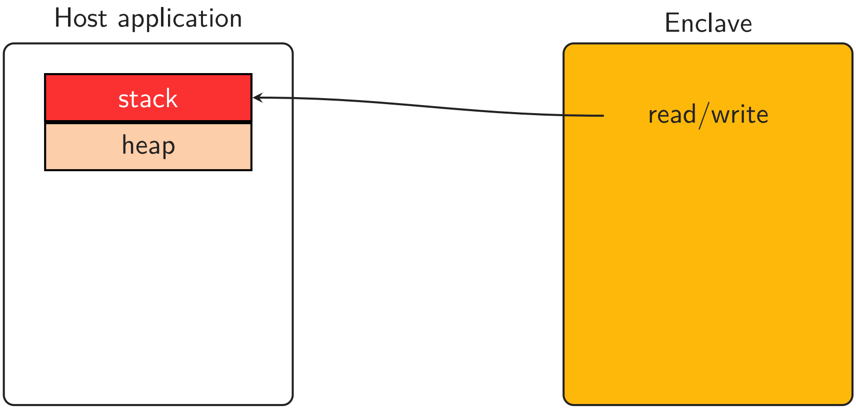


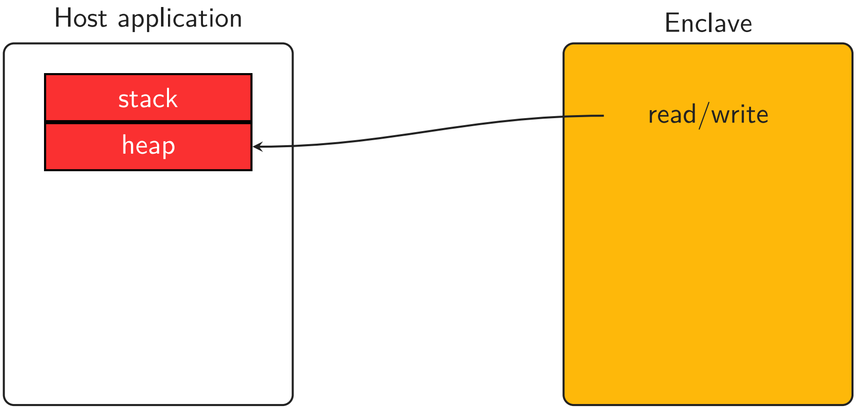
SGXJail [Wei+19]

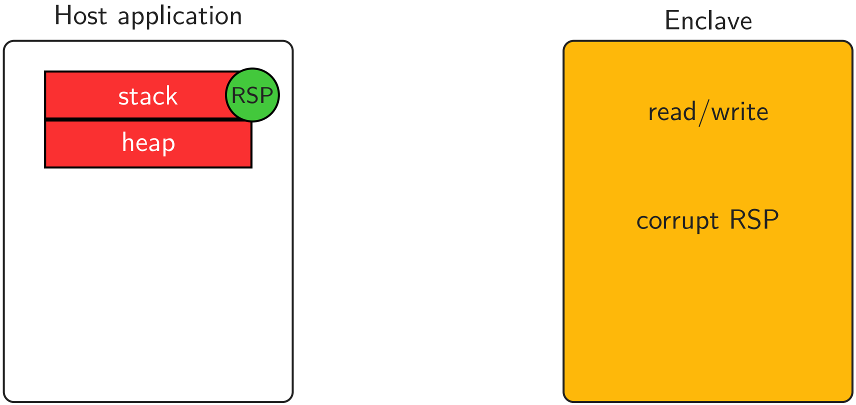


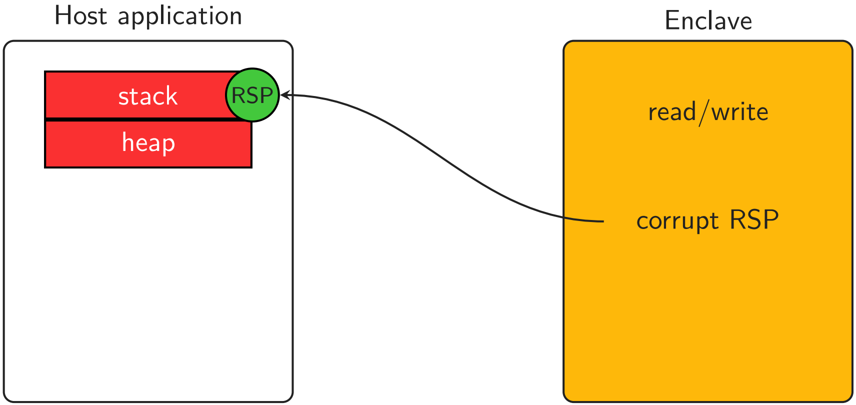

Host application
stack
heap
Enclave
read/write

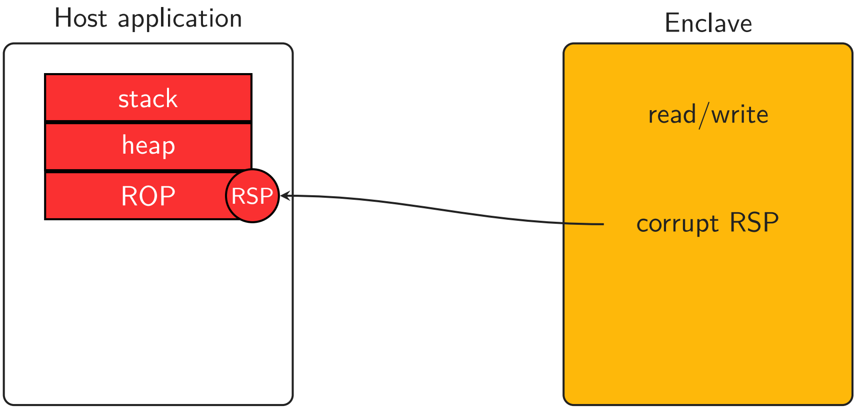
Defense

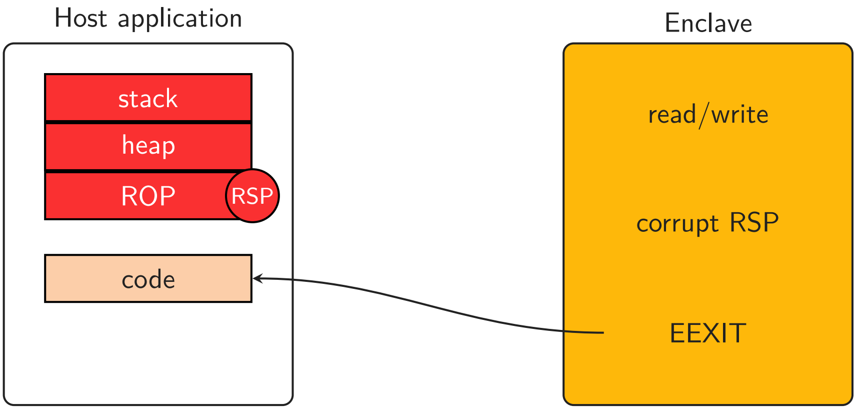


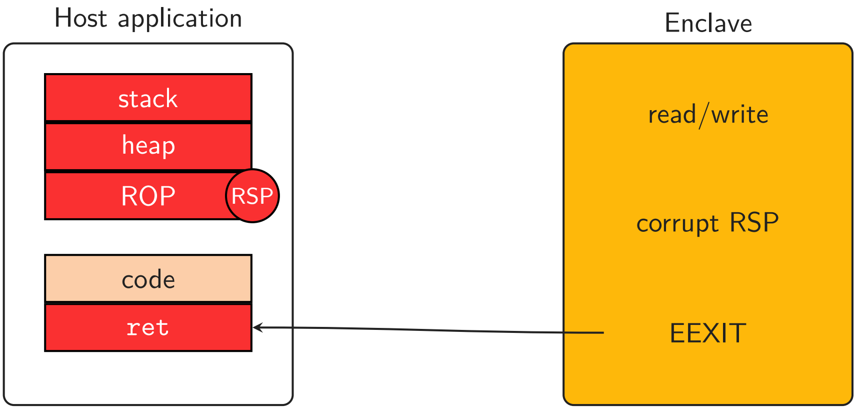

Host application
stack
heap
Enclave
read/write


Host application
stack
RSP
heap
Enclave
read/write
corrupt RSP


Host application
stack
RSP
heap
Enclave
read/write
corrupt RSP


Host application
stack
heap
ROP
RSP
Enclave
read/write
corrupt RSP


Host application
stack
heap
ROP
RSP
code
Enclave
read/write
corrupt RSP
EEXIT


Host application
stack
heap
ROP
RSP
code
ret
Enclave
read/write
corrupt RSP
EEXIT

- Arbitrary read

- Arbitrary read
  - Bypass randomization-based defenses (ASLR)

- Arbitrary read
  - Bypass randomization-based defenses (ASLR)
  - Discover ROP gadgets

- Arbitrary read
  - Bypass randomization-based defenses (ASLR)
  - Discover ROP gadgets
- Arbitrary write

- Arbitrary read
  - Bypass randomization-based defenses (ASLR)
  - Discover ROP gadgets
- Arbitrary write
  - Memory corruption

- Arbitrary read
  - Bypass randomization-based defenses (ASLR)
  - Discover ROP gadgets
- Arbitrary write
  - Memory corruption
- Arbitrary EEXIT

- Arbitrary read
  - Bypass randomization-based defenses (ASLR)
  - Discover ROP gadgets
- Arbitrary write
  - Memory corruption
- Arbitrary EEXIT
  - Direct code-reuse attacks

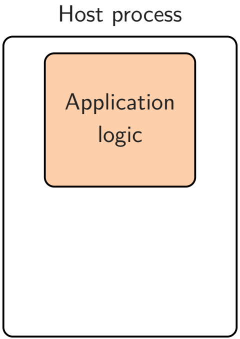
- Root problem: asymmetric trust

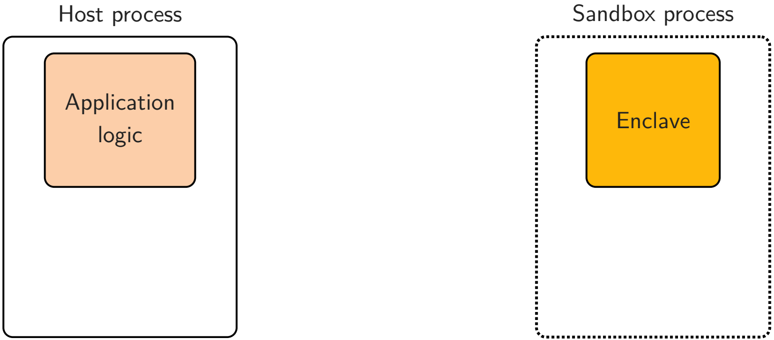
Asymmetric Trust

- Root problem: asymmetric trust
- Assumption: Enclave is fully trusted

- Root problem: asymmetric trust
- Assumption: Enclave is fully trusted
- Goal: mutual distrust

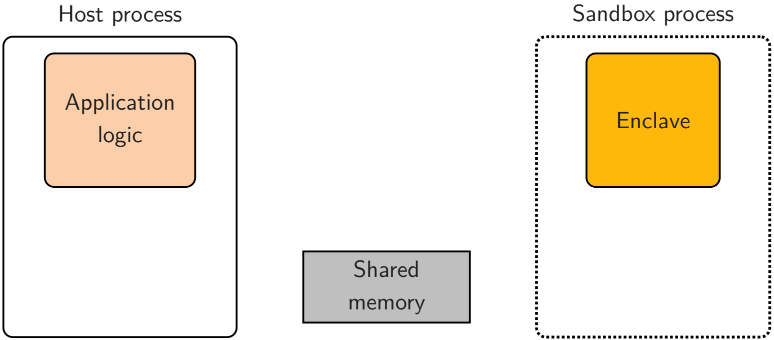
Sandboxing


Host process
Application logic

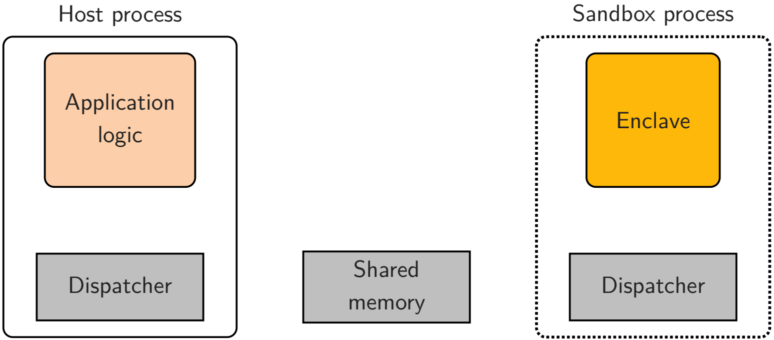


- Process isolation breaks arbitrary read/write

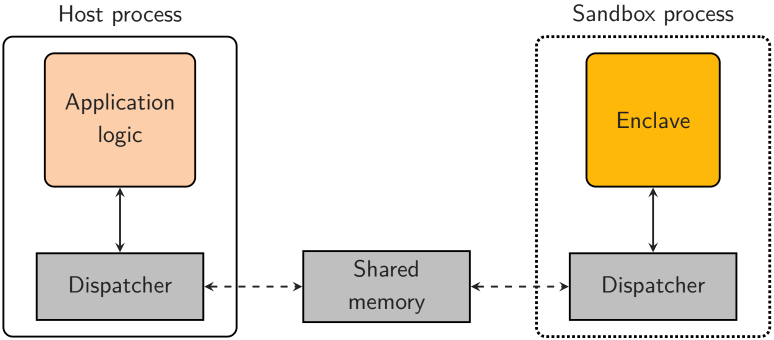
SGXJail Sandboxing



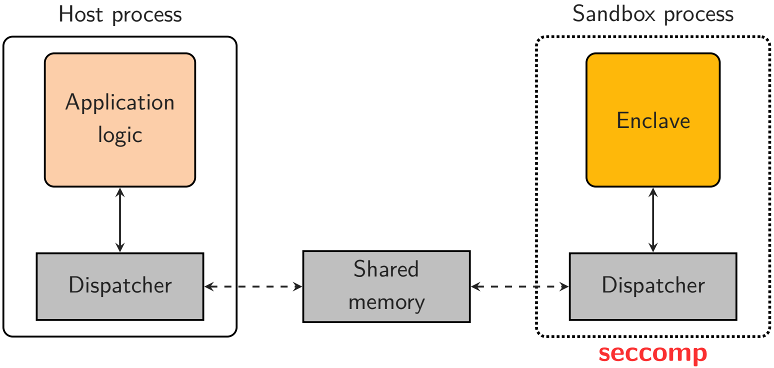
- Process isolation breaks arbitrary read/write



- Process isolation breaks arbitrary read/write
- ECALLs and OCALLs via shared memory

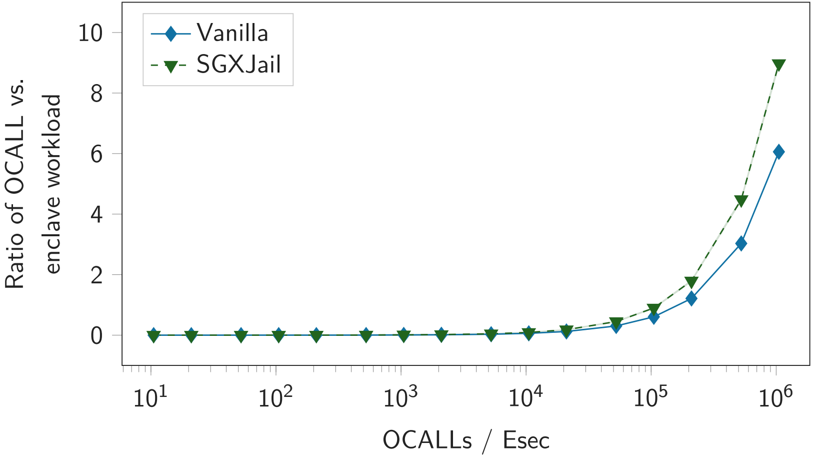


- Process isolation breaks arbitrary read/write
- ECALLs and OCALLs via shared memory

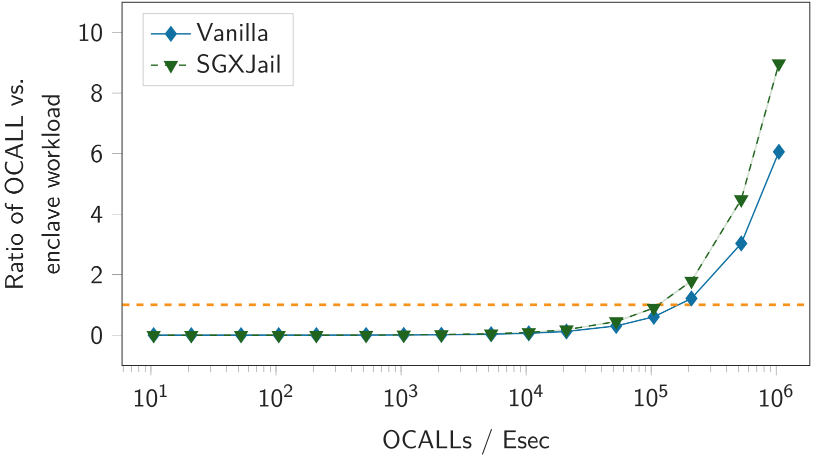


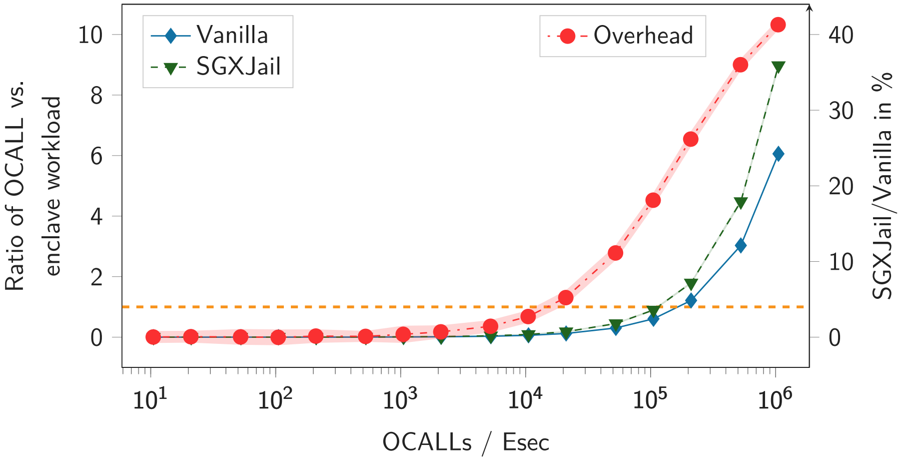
- Process isolation breaks arbitrary read/write
- ECALLs and OCALLs via shared memory
- seccomp syscall filter breaks arbitrary EEXIT

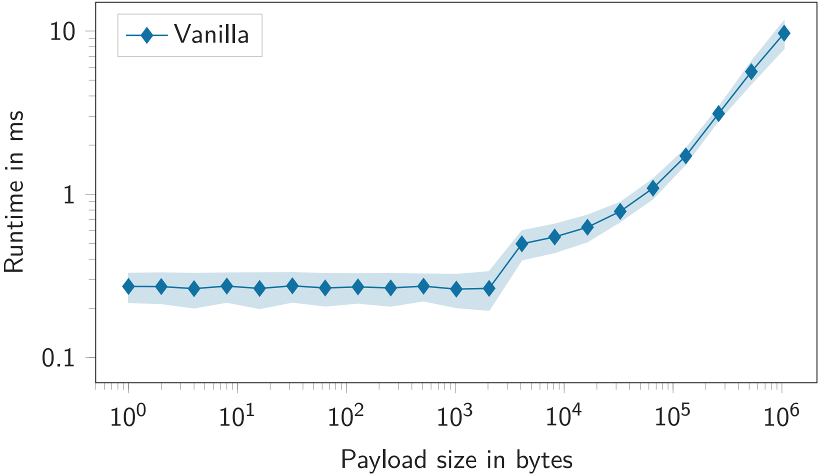
Evaluation: OCALL

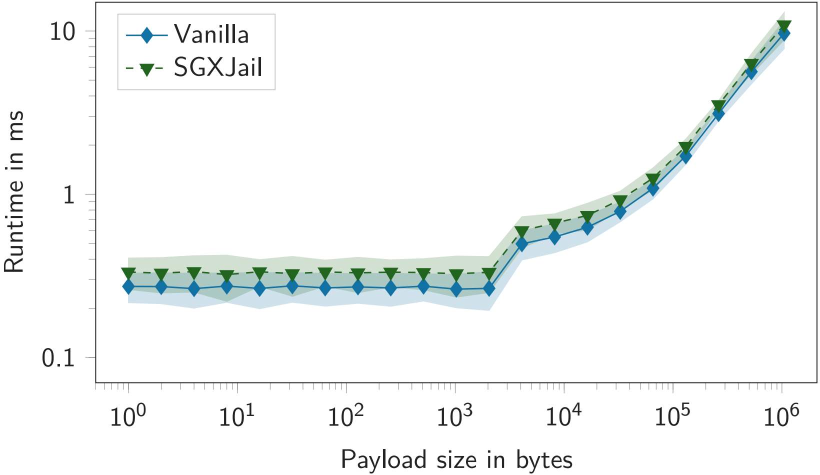


Evaluation: OCALL



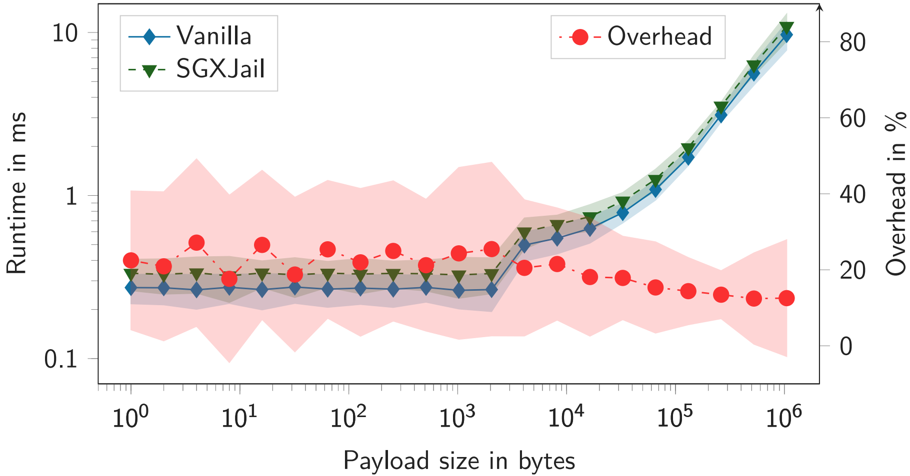

Vanilla
SGXJail
Overhead
Ratio of OCALL vs. enclave workload
SGXJail/Vanilla in %
OCALLs / Esec
0
2
4
6
8
10
0
10
20
30
40
10^1
10^2
10^3
10^4
10^5
10^6

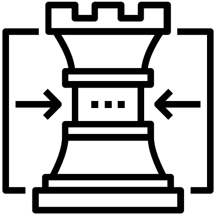

Vanilla
10
1
0.1
Runtime in ms
10^0
10^1
10^2
10^3
10^4
10^5
10^6
Payload size in bytes



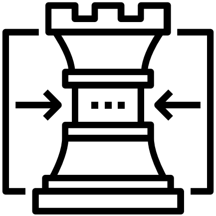
Vanilla
SGXJail
10
1
0.1
Runtime in ms
10^0
10^1
10^2
10^3
10^4
10^5
10^6
Payload size in bytes

Evaluation: Protected File System





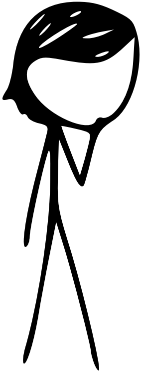
- Compatible with unmodified enclaves



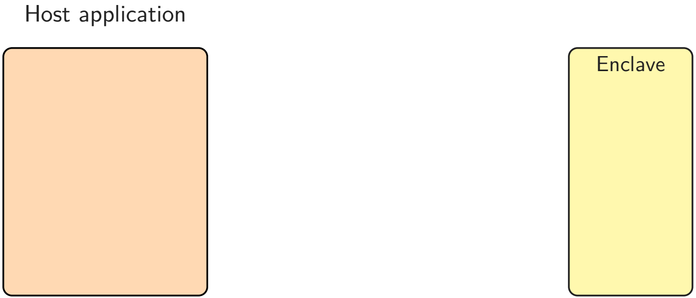
- Compatible with unmodified enclaves
- Fully integrated in SGX SDK

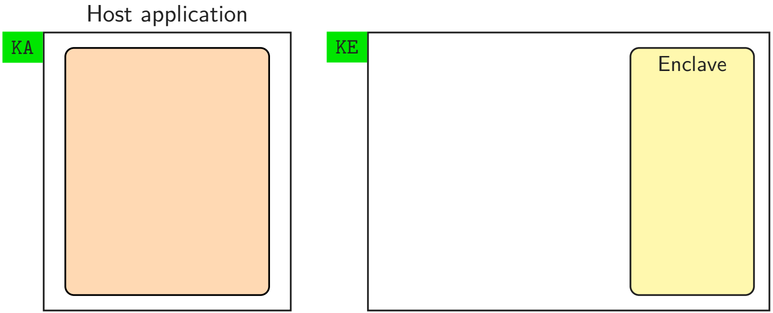
  https://github.com/IAIK/SGXJail

- Small overhead only due to ECALLs/OCALLs

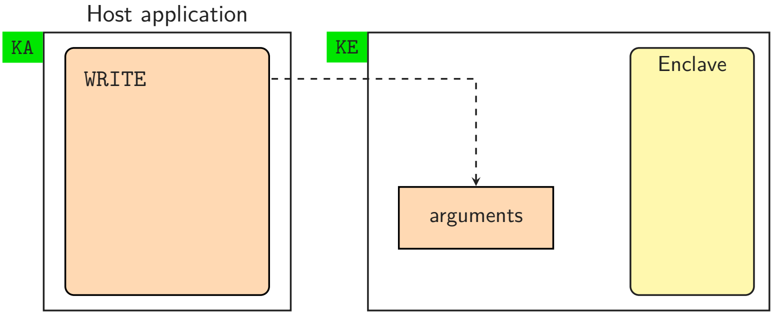


Can we implement it in hardware?

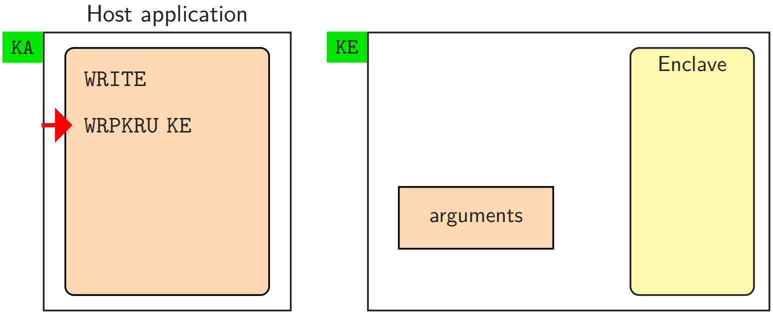

Host application
Enclave



- Intel Memory Protection Keys (MPK)

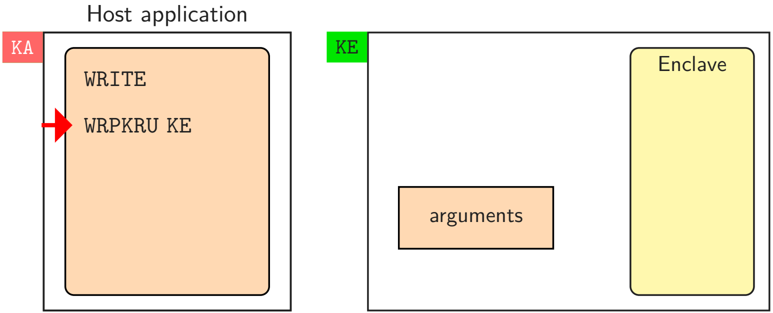


- Intel Memory Protection Keys (MPK)

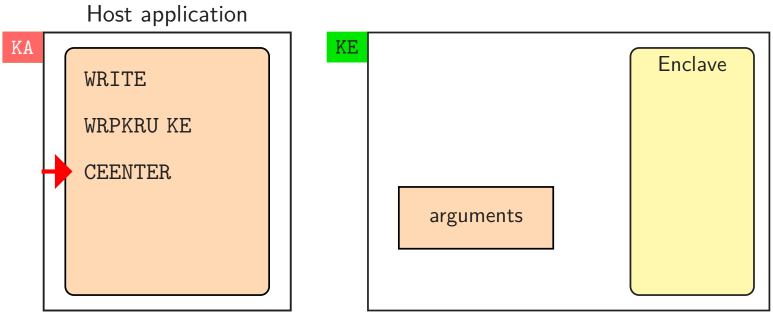


- Intel Memory Protection Keys (MPK)
- MPK disables host memory

Hardware SGXJail

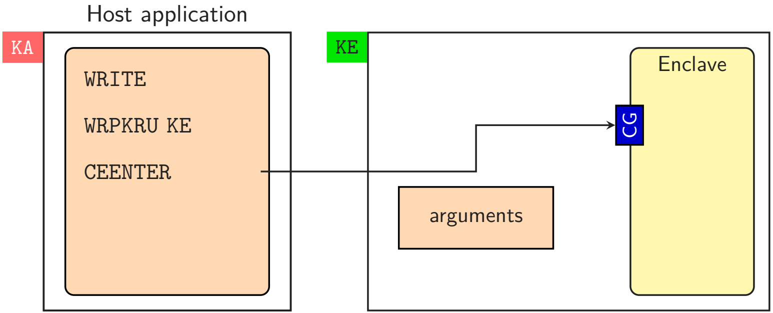


- Intel Memory Protection Keys (MPK)
- MPK disables host memory

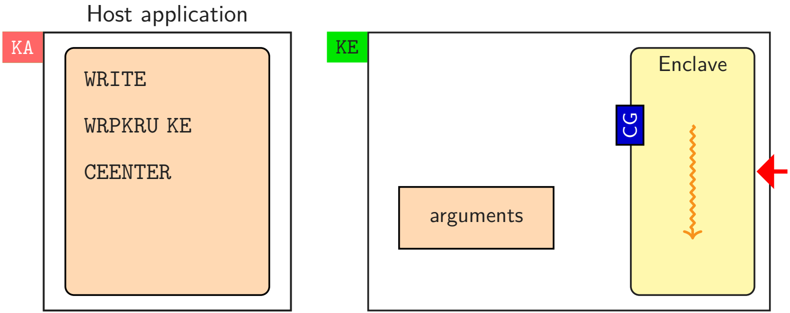


- Intel Memory Protection Keys (MPK)
- MPK disables host memory
- Confined EENTER instruction

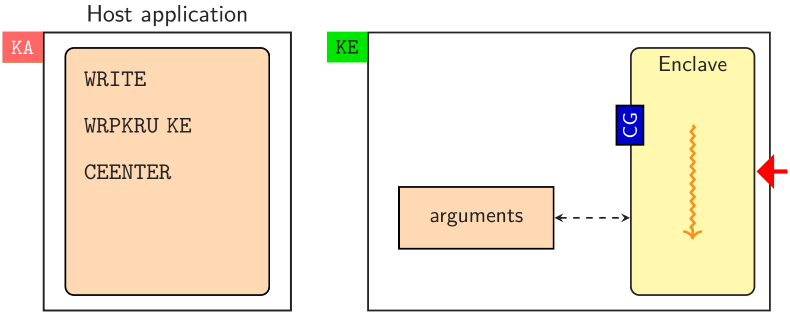
Hardware SGXJail



- Intel Memory Protection Keys (MPK)
- MPK disables host memory
- Confined EENTER instruction

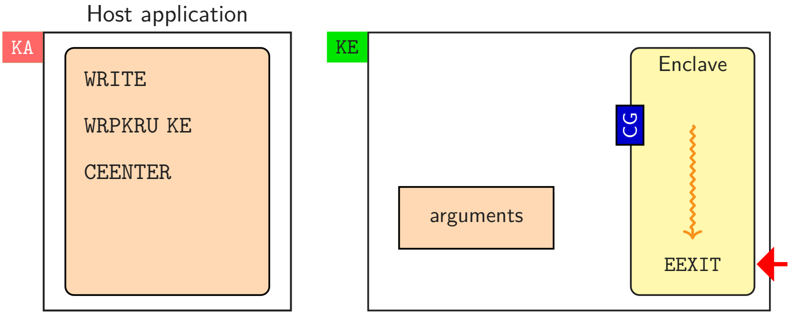


- Intel Memory Protection Keys (MPK)
- MPK disables host memory
- Confined EENTER instruction



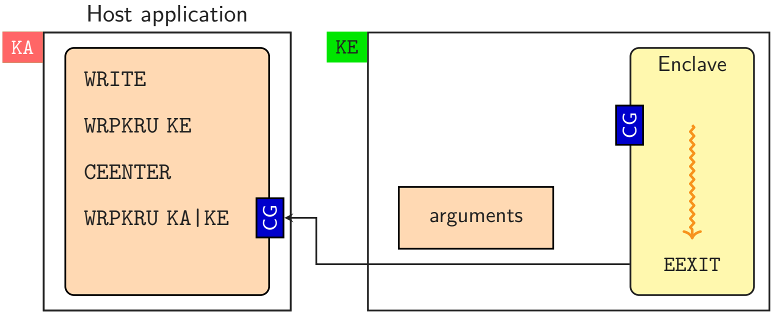
- Intel Memory Protection Keys (MPK)
- MPK disables host memory
- Confined EENTER instruction

- Enclave accesses arguments



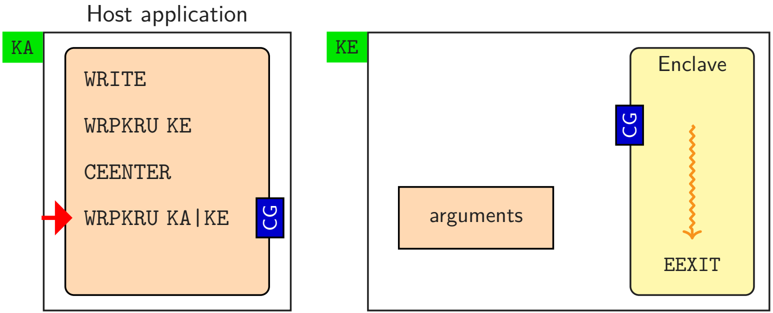
- Intel Memory Protection Keys (MPK)
- MPK disables host memory
- Confined EENTER instruction

- Enclave accesses arguments
- EEXIT only to call gate after CEENTER



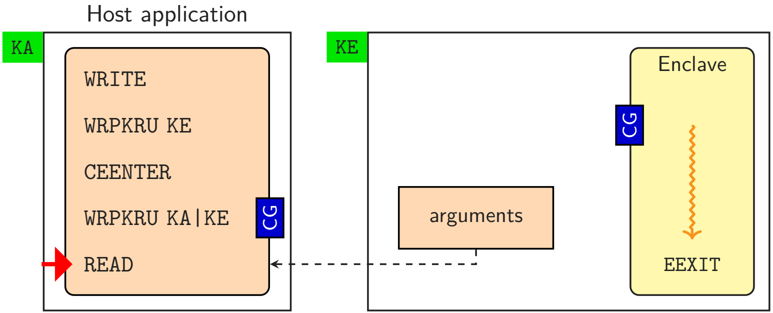
- Intel Memory Protection Keys (MPK)
- MPK disables host memory
- Confined EENTER instruction

- Enclave accesses arguments
- EEXIT only to call gate after CEENTER



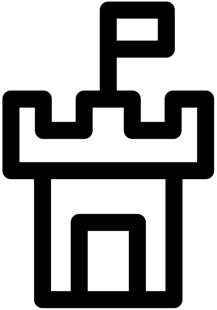
- Intel Memory Protection Keys (MPK)
- MPK disables host memory
- Confined EENTER instruction
- MPK enables host memory

- Enclave accesses arguments
- EEXIT only to call gate after CEENTER

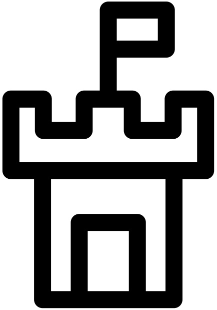


- Intel Memory Protection Keys (MPK)
- MPK disables host memory
- Confined EENTER instruction
- MPK enables host memory

- Enclave accesses arguments
- EEXIT only to call gate after `CEENTER`

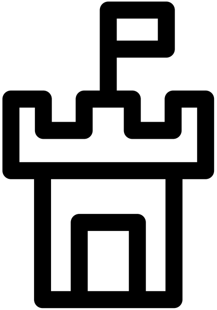


- Almost zero runtime overhead

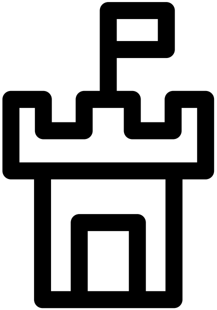


- Almost zero runtime overhead
- Highly compatible (opt-in)

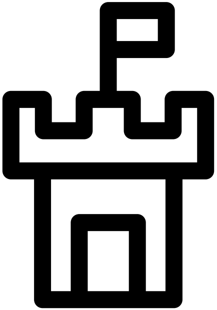
Hardware SGXJail



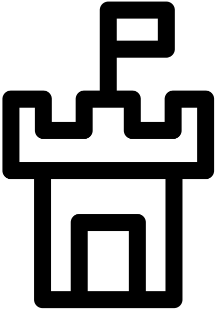
- Almost zero runtime overhead
- Highly compatible (opt-in)
- New `CEENTER` instruction



- Almost zero runtime overhead
- Highly compatible (opt-in)
- New `CEENTER` instruction
  - Make MPK immutable inside enclave



- Almost zero runtime overhead
- Highly compatible (opt-in)
- New `CEENTER` instruction
  - Make MPK immutable inside enclave
  - Enforce exit call gate



- Almost zero runtime overhead
- Highly compatible (opt-in)
- New `CEENTER` instruction
  - Make MPK immutable inside enclave
  - Enforce exit call gate
  - Can be implemented in microcode

Key Insights

- Secure execution → enclave malware
- Better threat models
  - SGX: asymmetric trust
  - SGXJail: mutual distrust
- Protection almost for free
- Future: reason about security of enclave API

Thank you!

Confining (Un)Trusted Execution Environments

Michael Schwarz (@misc0110)

November 20, 2019 - SILM

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.
- M. Schwarz, S. Weiser, and D. Gruss. Practical Enclave Malware with Intel SGX. In: DIMVA. 2019.
- S. Weiser, L. Mayr, M. Schwarz, and D. Gruss. SGXJail: Defeating Enclave Malware via Confinement. In: RAID. 2019.