Protecting SGX Enclaves From Practical Side-Channel Attacks
Key issue of the cloud: We cannot trust it
We cannot trust the cloud

- Thousands of employees
- Legal obligations
- Infrastructure vulnerabilities

Patch alert! Easy-to-exploit flaw in Linux kernel rated 'high risk'
Urgent security triage needed

Cloud Data Leak Exposes Information on 123 Million Americans

By: Sean Michael Kerner | December 20, 2017
We cannot trust the cloud

- Thousands of employees
- Legal obligations
- Infrastructure vulnerabilities

Cloud Data Leak Exposes Information on 123 Million Americans

By: Sean Michael Kerner | December 20, 2017

Patch alert! Easy-to-exploit flaw in Linux kernel rated 'high risk'

Urgent security triage needed
We cannot trust the cloud

- Thousands of employees
- Legal obligations
- Infrastructure vulnerabilities

Cloud Data Leak Exposes Information on 123 Million Americans

By: Sean Michael Kerner | December 20, 2017
Privileged attack vectors

I'm Eve. I control the OS and Hypervisor

Hosted process
Privileged attack vectors: Network

Sending: SECRET
Privileged attack vectors: Network

Sending: SECRET

I see: SECRET

Network traffic

Hosted process
Privileged attack vectors: Network

I see: $%sdf%#$

Sending: SECRET

Network traffic

TLS

Hosted process
Privileged attack vectors: Memory

Network traffic

I write SECRET to memory

Hosted process

Memory

Process's data:

..... SECRET....
Privileged attack vectors: Memory

I read: SECRET

I write SECRET to memory

Network traffic

Hosted process

Memory

Process's data:
.... SECRET....
Privileged attack vectors: Memory

I read: $%sdf%# 

I write SECRET to memory

Network traffic

Hosted process

Memory

SGX Enclave
Privileged attack vectors: System Calls

Network traffic → Hosted process

I request: `mmap()`

SGX Enclave

Memory

System calls

OS
Privileged attack vectors: System Calls

I request: `mmap()`

I return: exploit
Privileged attack vectors: System Calls

- **Hosted process**
  - I request: `mmap()`
  - I return: exploit

- **SGX Enclave**
- **Memory**

- **Network traffic**

- **Error!**
  - System calls
  - I return: exploit

- **OS**
Privileged attack vectors: Shared Resources

I write to address: 0x123

Network traffic

Hosted process

SCONE

OS

Memory

SGX Enclave
Privileged attack vectors: Shared Resources

I write to address: 0x123

I see an access to: 0x123

Network traffic

Hosted process

SCONE

OS

System calls

Memory

SGX Enclave

Caches

Page Tables
Privileged attack vectors: Shared Resources

This talk
Existing solutions

Low overhead

Cloak [1]
Düppel [2]

Low effort

Dr.SGX [3]
Déjà Vu [4]
T-SGX [5]

(no code changes required)

Existing solutions

- **Low overhead**
  - Cloak [1]
  - Düppel [2]

- **Low effort**
  - Dr.SGX [3]
  - Déjà Vu [4]
  - T-SGX [5]

**Varys**

- 15% average slowdown
- No code changes required
Approach

Rely but verify
Approach

Rely but verify

Request isolation from the untrusted OS
Approach

Rely but verify

Request isolation from the untrusted OS

Check within the enclave
Varys implements a low-cost protection for Intel SGX enclaves against side-channel attacks by creating an isolated environment and verifying it at runtime.
Varys implements a low-cost protection for Intel SGX enclaves against side-channel attacks by creating an isolated environment and verifying it at runtime.
Varys implements a low-cost protection for Intel SGX enclaves against side-channel attacks by creating an isolated environment and verifying it at runtime.
Side-channel attacks

Hosted process
Side-channel attacks

```c
if (secret == 0)
    read(addr1)
else
    read(addr2)
```
Side-channel attacks

if (secret == 0)
    read(addr1)
else
    read(addr2)
Side-channel attacks

```python
if (secret == 0)
    read(addr1)
else
    read(addr2)
```
Side-channel attacks

```c
if (secret == 0)
    read(addr1)
else
    read(addr2)
```

Running...

Hosted process

Shared resource

addr1

addr2
if (secret == 0)
  read(addr1)
else
  read(addr2)
Side-channel attacks

```c
if (secret == 0)  
    read(addr1)
else
    read(addr2)
```
Side-channel attacks

if (secret == 0)
    read(addr1)
else
    read(addr2)

Hosted process

Shared resource

addr1

addr2
Side-channel attacks

if (secret == 0)
    read(addr1)
else
    read(addr2)

Shared resource

addr1

addr2

addr1 was accessed!
Side-channel attacks

if (secret == 0)
    read(addr1)
else
    read(addr2)

Shared resource

addr1

addr2 was not accessed!
Side-channel attacks

if (secret == 0)
    read(addr1)
else
    read(addr2)
Side-channel attacks

if (secret == 0) read(addr1)
else read(addr2)

The secret is 0
Vulnerable shared resources

- CPU caches
- Page tables
- FPU
- Memory bus
- ...

"slaps modern cpu" You won’t believe how many side channels this thing can hold
Vulnerable shared resources

- CPU caches (L1, L2)
- Page tables
- FPU
- Memory bus
- ...

Varys
Varys implements a low-cost protection for Intel SGX enclaves against side-channel attacks by creating an **isolated environment** and verifying it at runtime.
Attack requirements

- High interrupt rate
- Predefined cache state
- Shared core
Attack requirements

- High interrupt rate
- Predefined cache state
- Shared core

Isolated environment
Varys implements a low-cost protection for Intel SGX enclaves against side-channel attacks by creating an isolated environment and verifying it at runtime.
Design

- High preemption rate
- Predefined cache state
- Shared core

Restrict and terminate
Cache eviction
Trusted reservation
Design

- High preemption rate
- Predefined cache state
- Shared core
- Restrict and terminate
- Cache eviction
- Trusted reservation
Restricting preemption rate

- Attack exit rate: ~ 5000 exits/s.
Restricting preemption rate

- Attack exit rate: ~ 5000 exits/s.
- Normal exit rate: ~ 30 exits/s.
Restricting preemption rate

- Attack exit rate: ~ 5000 exits/s.
- Normal exit rate: ~ 30 exits/s.
Asynchronous Enclave Exit (AEX)

CPU state
- RIP = 0x123
- RAX = 0x111
- RBX = 0x222

Interrupt

Hosted process

OS

SGX

SGX Enclave

Memory
Asynchronous Enclave Exit (AEX)

Hosted process

Memory

CPU state
- RIP = 0x123
- RAX = 0x111
- RBX = 0x222
Asynchronous Enclave Exit (AEX)

CPU state:
- RIP = 0x100
- RAX = 0x000
- RBX = 0x000

Interrupt

OS

Hosted process

SGX

SGX Enclave
- RIP = 0x123
- RAX = 0x111
- RBX = 0x222

Memory

SSA
Detecting interrupts

Hosted process

CPU state
- RIP = 0x123
- RAX = 0x111
- RBX = 0x222

OS

SGX Enclave

Memory

SSA

SGX
Detecting interrupts

**CPU state**
- RIP = 0x123
- RAX = 0x111
- RBX = 0x222

**Diagram**
- OS
  - Hosted process
    - SGX
      - SGX Enclave
        - Memory
          - SSA
          - RIP = 0x000
Detecting interrupts

- Hosted process
- Memory
- SGX Enclave

CPU state:
- RIP = 0x123
- RAX = 0x111
- RBX = 0x222

Read SSA

OS

SGX

Memory

RIP = 0x000

SSA
Detecting interrupts

Hosted process

Memory

SGX Enclave

SGX

OS

CPU state

RIP = 0x123
RAX = 0x111
RBX = 0x222

Still 0x000 Continue..

SSA
Detecting interrupts

CPU state
- RIP = 0x123
- RAX = 0x111
- RBX = 0x222

Interrupt

Hosted process

OS

SGX

SGX Enclave
- RIP = 0x123
- RAX = 0x111
- RBX = 0x222

Memory

SSA
Detecting interrupts

CPU state
- RIP = 0x123
- RAX = 0x111
- RBX = 0x222

Hosted process

SGX

Memory
- RIP = 0x123
- RAX = 0x111
- RBX = 0x222

OS

Read SSA
Detecting interrupts

There was an interrupt!

CPU state:
- RIP = 0x123
- RAX = 0x111
- RBX = 0x222

OS

Not 0x000
Design

- High preemption rate
- Predefined cache state
- Shared core

Restrict and terminate
Cache eviction
Trusted reservation
Hiding cache traces

Hosted process

Cache

addr1

addr2
Hiding cache traces
Hiding cache traces

There was an interrupt!

Hosted process

Cache

addr1

addr2
Hiding cache traces
Hiding cache traces

Hosted process

Cache

addr1

addr2
Hiding cache traces
Hiding cache traces

Access addr1

Hosted process

Cache

addr1

addr2
Hiding cache traces

Hosted process

Cache

addr1

addr2

???
Design

- High preemption rate
- Predefined cache state
- Shared core
- Restrict and terminate
- Cache eviction
- Trusted reservation
Preventing core sharing

- Occupy both hyperthreads
Preventing core sharing

- Occupy both hyperthreads
  - Use process affinity

![Diagram showing Core 1 with two processes](image-url)
How do we ensure reservation?
How do we ensure reservation?
Handshake

- Use shared access timing
Handshake

- Use shared access timing

Write to 0x123
Handshake

- Use shared access timing
Handshake

- Use shared access timing

It was fast!
Handshake

- Use shared access timing
Handshake

- Use shared access timing

[Diagram showing a handshake with Core 1 and Core 2, with a process, an attacker, and a process, and a note indicating reading from 0x123.]
Handshake

- Use shared access timing

It was slow! Something is wrong...
Design

- High preemption rate
- Predefined cache state
- Shared core
- Restrict and terminate
- Cache eviction
- Trusted reservation
Varys **implements** a low-cost protection for Intel SGX enclaves against side-channel attacks by creating an isolated environment and verifying it at runtime.
Implementation

Source code → LLVM pass (Exit detection) → Compiler (SCONE) → Hardened binary

Runtime library (Handshake & cache eviction)
Varys implements a **low-cost** protection for Intel SGX enclaves against side-channel attacks by creating an isolated environment and verifying it at runtime.
Evaluation: performance

Normalized runtime (w.r.t. native)

Lower is better
Evaluation: performance

Lower is better
Evaluation: performance

Lower is better
Evaluation: performance

Lower is better
Evaluation: performance

Lower is better
Handshake and eviction only at enclave exits

- 20-30 times per second
Evaluation: multithreading
Evaluation: multithreading

Lower is better
Evaluation: multithreading

EPC paging ⇒ higher exit rate
Evaluation: multithreading

Lower is better

EPC paging ⇒ higher exit rate

False positives
Varys implements a low-cost protection for Intel SGX enclaves against side-channel attacks by creating an isolated environment and verifying it at runtime.
Evaluation: security

- Privileged cache SCA
  - Target: L1 cache
- No eviction
Evaluation: security

- Privileged cache SCA
  - Target: L2 cache
- No eviction
Evaluation: security

- Privileged cache SCA
  - Target: L2 cache
- No eviction
Evaluation: security

- Privileged cache SCA
  - Target: L2 cache
- Varys protection
Evaluation: security

- Privileged cache SCA
  - Target: L2 cache
- Varys protection
Summary

- Varys: side-channel protection for SGX enclaves
- "Rely but verify" approach
  - Ask OS for
    - Lower interrupt rate
    - Paired thread allocation
  - Verify the request
- Evict caches on enclave exits
Summary

- **Varys**: side-channel protection for SGX enclaves
- "Rely but verify" approach
  - Ask OS for
    - Lower interrupt rate
    - Paired thread allocation
  - Verify the request
- Evict caches on enclave exits

Thanks!

oleksii.oleksenko@tu-dresden.de
@oleksii_o